Devoir2-Exo1

May 3, 2023

[271]:
```
# Pour visualiser les données j'utilise la bibliothèque Pandas étudiée en␣
↪DataScience -1
# Importation des librairies nécessaires
import pandas as pd
```

1 1) Ouverture du fichier

[218]:
```
# Importation des données
vente_pneus = pd.read_csv('vente_pneus.csv', sep = ';')
```

[219]:
```
vente_pneus.head()
```

[219]:

	Unnamed: 0	2017	2018	2019	2020	2021	2022
0	1	8.0	9.0	10.0	12.0	16.0	9.0
1	2	10.0	11.0	12.0	14.0	18.0	11.0
2	3	12.0	13.0	14.0	16.0	20.0	13.0
3	4	19.0	20.0	20.0	23.0	27.0	19.0
4	5	23.0	22.0	22.0	27.0	31.0	21.0

[220]:
```
print(vente_pneus.dtypes)
```

```
Unnamed: 0     object
2017          float64
2018          float64
2019          float64
2020          float64
2021          float64
2022          float64
dtype: object
```

[221]:
```
vente_pneus.shape
```

[221]:
```
(54, 7)
```

[222]:
```
# Renommer la colonne 'colonne1' en 'nouvelle_colonne'
vente_pneus = vente_pneus.rename(columns={'Unnamed: 0': 'numero semaine'})
```

[223]:
```
vente_pneus.head()
```

[223]:

	numero semaine	2017	2018	2019	2020	2021	2022
0	1	8.0	9.0	10.0	12.0	16.0	9.0
1	2	10.0	11.0	12.0	14.0	18.0	11.0
2	3	12.0	13.0	14.0	16.0	20.0	13.0
3	4	19.0	20.0	20.0	23.0	27.0	19.0
4	5	23.0	22.0	22.0	27.0	31.0	21.0

Au vu des données il s'agit de vente de pneu par année et par semaine dans une année (il y en a bien 52). A quoi corresponds Somme?.

[224]:
```
vente_pneus['2021'].sum()
```

[224]:
```
2893.0
```

Il s'agit donc d'inférer la valeur des ventes la 52ème semaine

Somme semble correspondre approximativement à la moitié de la somme des éléments dans une colonne (sans prendre en compte les valeurs manquantes représentée dans pandas en NaN.

Et si je représentai les courbes pneus vendu par semaine?

1.1 3) Représentation des données

[225]:
```
# Suppression des NaN
vente_pneus = vente_pneus.dropna()
```

[226]:
```
vente_pneus.head()
```

[226]:

	numero semaine	2017	2018	2019	2020	2021	2022
0	1	8.0	9.0	10.0	12.0	16.0	9.0
1	2	10.0	11.0	12.0	14.0	18.0	11.0
2	3	12.0	13.0	14.0	16.0	20.0	13.0
3	4	19.0	20.0	20.0	23.0	27.0	19.0
4	5	23.0	22.0	22.0	27.0	31.0	21.0

[227]:
```
# Suppression de l'index 53 (somme)
vente_pneus.drop(53, axis=0, inplace=True)
```

[228]:
```
vente_pneus['numero semaine']=vente_pneus['numero semaine'].astype(float,␣
↪errors = 'raise')
```

[229]:
```
vente_pneus.head()
```

[229]:

	numero semaine	2017	2018	2019	2020	2021	2022
0	1.0	8.0	9.0	10.0	12.0	16.0	9.0
1	2.0	10.0	11.0	12.0	14.0	18.0	11.0

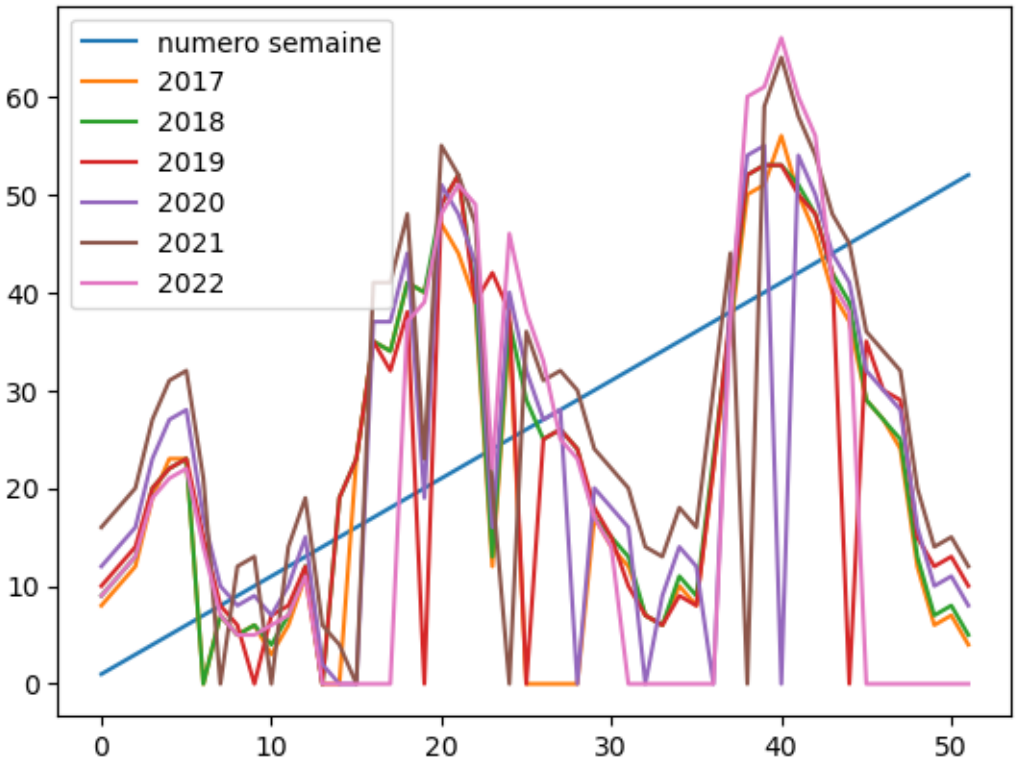
```
2              3.0   12.0   13.0   14.0   16.0   20.0   13.0
3              4.0   19.0   20.0   20.0   23.0   27.0   19.0
4              5.0   23.0   22.0   22.0   27.0   31.0   21.0
```

[230]:
```
import matplotlib.pyplot as plt
```

Essayons de représenter les données

[231]:
```
vente_pneus.plot()
```

[231]: <AxesSubplot:>



On observe une forme récurrente,cette représentation n'est pas trés pratique, une autre forme de representation consiste à représenter les données comme un nuage de points.

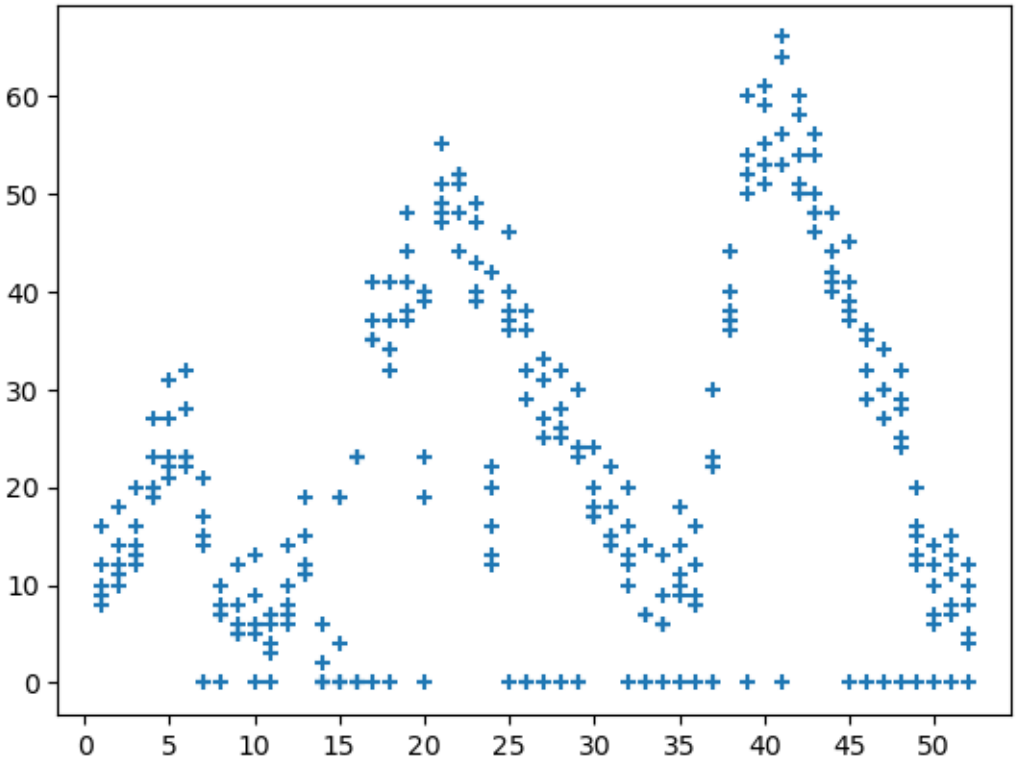
[232]:
```
y_raw=pd.
  ↪concat([vente_pneus['2017'],vente_pneus['2018'],vente_pneus['2019'],vente_pneus['2020'],ven
```

[233]:
```
x_raw=pd.concat([vente_pneus['numero semaine'],vente_pneus['numero␣
  ↪semaine'],vente_pneus['numero semaine'],vente_pneus['numero␣
  ↪semaine'],vente_pneus['numero semaine'],vente_pneus['numero␣
  ↪semaine']],ignore_index=True)
```

Représentons ces données

[234]:
```
import numpy as np
listOf_Xticks = np.arange(0, 55, 5)
plt.xticks(listOf_Xticks)
plt.scatter(x_raw,y_raw,marker='+')
```

[234]:
```
<matplotlib.collections.PathCollection at 0x7f95fed57c10>
```

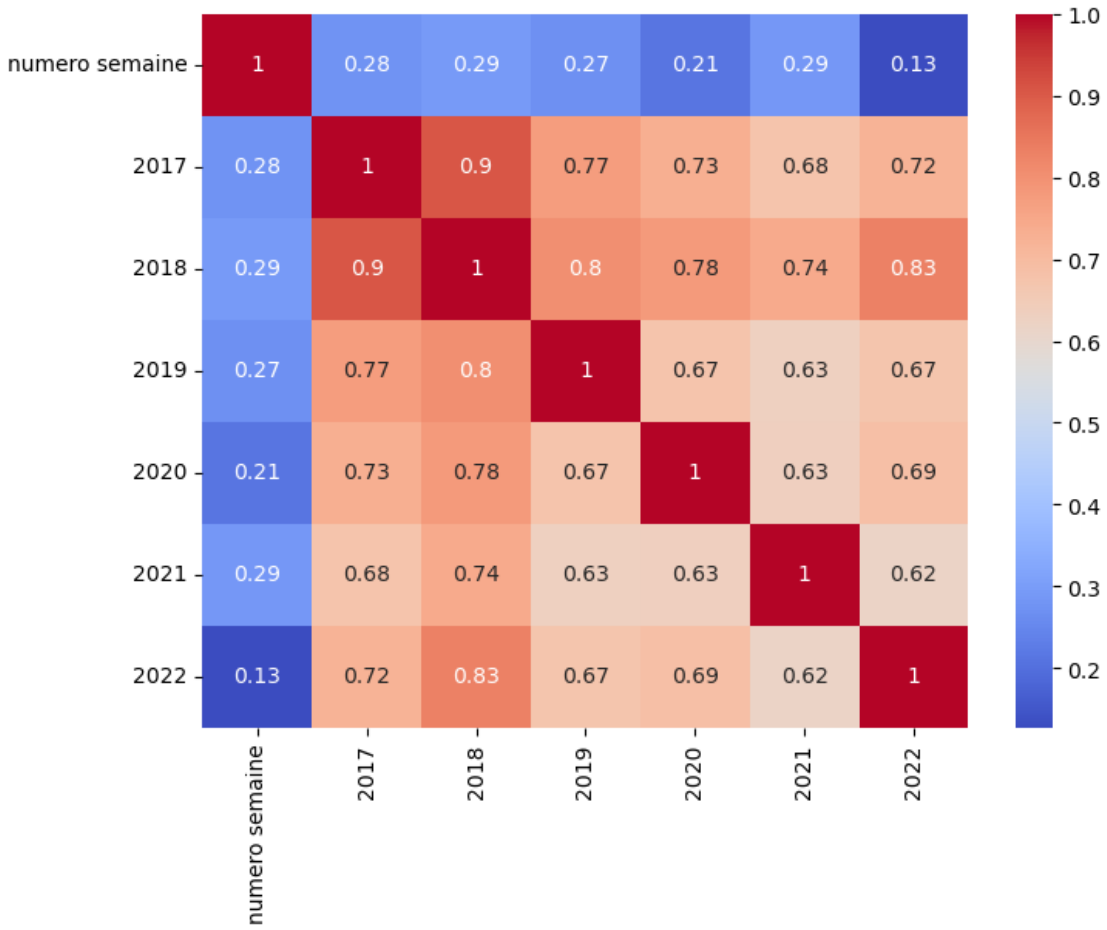


1.2 4) Préparation des données

Les données sont assez corrélées, ce qui est normal les ventes sont fonction de la semaine dans l'année maintenant il peut y avoir des années avec des valeurs extrèmes des outliers comme on les nomment. Allons plus loin dans l'étude de la corrélation.

[235]:
```
# Analyse de la corrélation
import seaborn as sns
plt.figure(figsize=(8, 6))
sns.heatmap(vente_pneus.corr(), annot=True, cmap='coolwarm', fmt='.2g')
```

[235]:
```
<AxesSubplot:>
```



On voit que 2017 et 2018 suivent le même schéma avec une corrélation à 0.9.L'année 2021 est la moins corrélée avec les autres années on peut se dire que c'est liée à la période du covid, cette année est en quelque sorte "spéciale". Si l'on choisit de ne conserver comme données d'entrée pour notre modèle que la moyenne en 'sortant' des valeurs inappropriées c'est à dire l'année 2021 on obtient:

[236]:
```
moy=1/
↪5*(vente_pneus['2017']+vente_pneus['2018']+vente_pneus['2019']+vente_pneus['2020']+vente_pn
```

[237]:
```
vente_pneus['moy']=moy
```

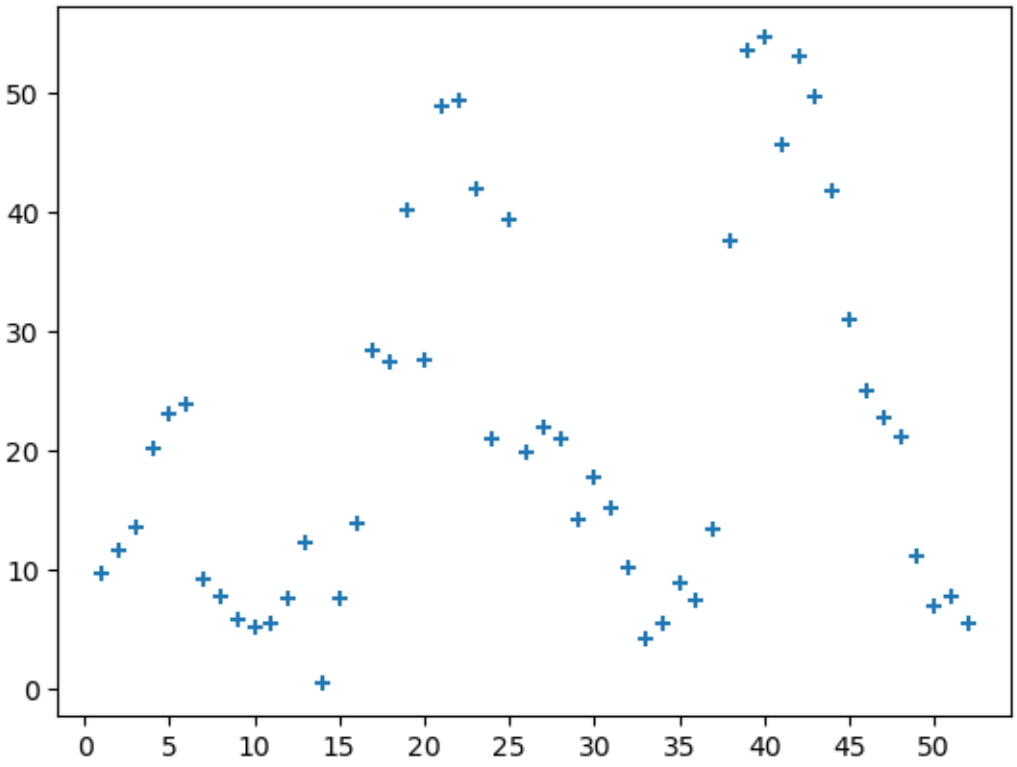
[238]:
```
vente_pneus.head()
```

[238]:

	numero semaine	2017	2018	2019	2020	2021	2022	moy
0	1.0	8.0	9.0	10.0	12.0	16.0	9.0	9.6
1	2.0	10.0	11.0	12.0	14.0	18.0	11.0	11.6
2	3.0	12.0	13.0	14.0	16.0	20.0	13.0	13.6
3	4.0	19.0	20.0	20.0	23.0	27.0	19.0	20.2
4	5.0	23.0	22.0	22.0	27.0	31.0	21.0	23.0

[239]:
```
import numpy as np
listOf_Xticks = np.arange(0, 55, 5)
plt.xticks(listOf_Xticks)
plt.scatter(vente_pneus['numero semaine'],vente_pneus['moy'],marker='+')
```

[239]:
```
<matplotlib.collections.PathCollection at 0x7f95ff35ad90>
```



1.3 5) Normalisation des données

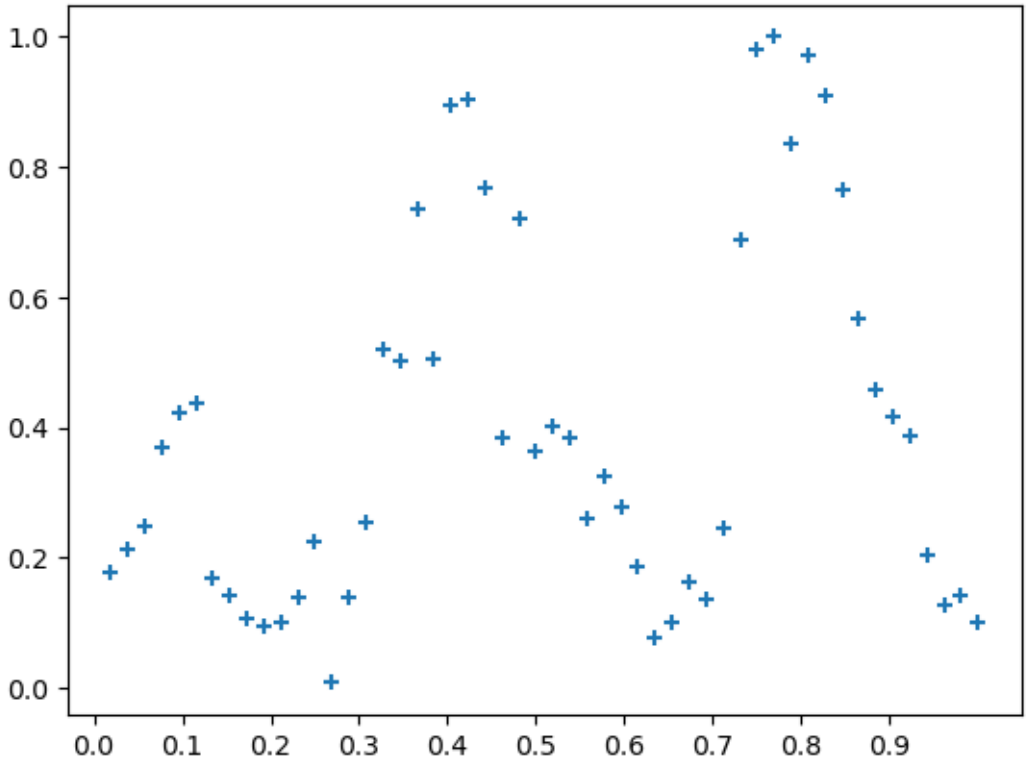
La normalisation des données permet d'augmenter la rapidité de la convergence du réseau de neurones

[240]:
```
vente_pneus['moy_norm']=moy/moy.max()
```

[242]:
```
vente_pneus['numero_semaine_norm']=vente_pneus['numero semaine']/
  ↪vente_pneus['numero semaine'].max()
```

[243]:
```
import numpy as np
listOf_Xticks = np.arange(0, 1, 0.1)
plt.xticks(listOf_Xticks)
plt.
  ↪scatter(vente_pneus['numero_semaine_norm'],vente_pneus['moy_norm'],marker='+')
```

[243]: <matplotlib.collections.PathCollection at 0x7f95fdbcfca0>



1.4 5) Choix du modèle

Importons d'abord les librairies requises:

[244]:
```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
from tensorflow.python.keras import layers
from tensorflow.python.keras.layers import Activation,Dense
import matplotlib.pyplot as plt
```

Le modèle est constitué d'une couche d'entrée (Input) avec une dimension de 1, suivie de deux couches cachées (Dense) avec 15 et 10 neurones respectivement, et une couche de sortie (Dense) avec une sortie de dimension 1.

La fonction d'activation utilisée pour toutes les couches est la fonction sigmoid. Le modèle est compilé avec l'optimiseur de descente de gradient stochastique (SGD) avec un taux d'apprentissage de 0,25 et la fonction de perte MSE (Mean Squared Error).

Il est ensuite entraîné sur les données vente_pneus['numero_semaine_norm'] et vente_pneus['moy_norm'] avec un nombre d'époques de 30000 et une taille de batch de 17.

[245]:
```
inputs = tf.keras.Input(shape=(1,))
h1 = tf.keras.layers.Dense(15,activation=tf.sigmoid)(inputs)
h2 = tf.keras.layers.Dense(10,activation=tf.sigmoid)(h1)
outputs = tf.keras.layers.Dense(1,activation=tf.sigmoid)(h2)
mon_model = tf.keras.Model(inputs=inputs,outputs=outputs)
mon_model.summary()
```

```
Model: "model_6"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 input_7 (InputLayer)         [(None, 1)]               0

 dense_18 (Dense)             (None, 15)                30

 dense_19 (Dense)             (None, 10)                160

 dense_20 (Dense)             (None, 1)                 11

=================================================================
Total params: 201
Trainable params: 201
Non-trainable params: 0
_________________________________________________________________
```

1.5 5) Entrainement du modèle

[297]:
```
mon_model.compile( optimizer=tf.train.GradientDescentOptimizer(0.
  ↪25),loss='mse',metrics=['mse'])
mon_model.fit(vente_pneus['numero_semaine_norm'].
  ↪to_numpy(),vente_pneus['moy_norm'].to_numpy(),epochs=100,batch_size=17)
```

```
Train on 52 samples

2023-05-03 19:28:57.286746: W tensorflow/c/c_api.cc:300] Operation
'{name:'count_25/Assign' id:5064 op device:{requested: '', assigned: ''}
def:{{{node count_25/Assign}} =
AssignVariableOp[_has_manual_control_dependencies=true, dtype=DT_FLOAT,
validate_shape=false](count_25, count_25/Initializer/zeros)}}' was changed by
setting attribute after it was run by a session. This mutation will have no
effect, and will trigger an error in the future. Either don't modify nodes after
running them or create a new session.

Epoch 1/100
52/52 [==============================] - 0s 3ms/sample - loss: 0.0070 -
mean_squared_error: 0.0070
Epoch 2/100
52/52 [==============================] - 0s 336us/sample - loss: 0.0069 -
```

```
mean_squared_error: 0.0069
Epoch 3/100
52/52 [==============================] - 0s 2ms/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 4/100
52/52 [==============================] - 0s 694us/sample - loss: 0.0069 -
mean_squared_error: 0.0069
Epoch 5/100
52/52 [==============================] - 0s 1ms/sample - loss: 0.0069 -
mean_squared_error: 0.0069
Epoch 6/100
52/52 [==============================] - 0s 868us/sample - loss: 0.0072 -
mean_squared_error: 0.0072
Epoch 7/100
52/52 [==============================] - 0s 624us/sample - loss: 0.0071 -
mean_squared_error: 0.0071
Epoch 8/100
52/52 [==============================] - 0s 423us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 9/100
52/52 [==============================] - 0s 467us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 10/100
52/52 [==============================] - 0s 749us/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 11/100
52/52 [==============================] - 0s 595us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 12/100
52/52 [==============================] - 0s 634us/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 13/100
52/52 [==============================] - 0s 699us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 14/100
52/52 [==============================] - 0s 536us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 15/100
52/52 [==============================] - 0s 691us/sample - loss: 0.0069 -
mean_squared_error: 0.0069
Epoch 16/100
52/52 [==============================] - 0s 521us/sample - loss: 0.0069 -
mean_squared_error: 0.0069
Epoch 17/100
52/52 [==============================] - 0s 761us/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 18/100
52/52 [==============================] - 0s 568us/sample - loss: 0.0069 -
```

```
mean_squared_error: 0.0069
Epoch 19/100
52/52 [==============================] - 0s 534us/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 20/100
52/52 [==============================] - 0s 553us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 21/100
52/52 [==============================] - 0s 596us/sample - loss: 0.0066 -
mean_squared_error: 0.0066
Epoch 22/100
52/52 [==============================] - 0s 703us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 23/100
52/52 [==============================] - 0s 2ms/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 24/100
52/52 [==============================] - 0s 567us/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 25/100
52/52 [==============================] - 0s 648us/sample - loss: 0.0069 -
mean_squared_error: 0.0069
Epoch 26/100
52/52 [==============================] - 0s 547us/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 27/100
52/52 [==============================] - 0s 706us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 28/100
52/52 [==============================] - 0s 1ms/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 29/100
52/52 [==============================] - 0s 729us/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 30/100
52/52 [==============================] - 0s 628us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 31/100
52/52 [==============================] - 0s 441us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 32/100
52/52 [==============================] - 0s 583us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 33/100
52/52 [==============================] - 0s 628us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 34/100
52/52 [==============================] - 0s 521us/sample - loss: 0.0067 -
```

```
mean_squared_error: 0.0067
Epoch 35/100
52/52 [==============================] - 0s 320us/sample - loss: 0.0066 -
mean_squared_error: 0.0066
Epoch 36/100
52/52 [==============================] - 0s 684us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 37/100
52/52 [==============================] - 0s 460us/sample - loss: 0.0066 -
mean_squared_error: 0.0066
Epoch 38/100
52/52 [==============================] - 0s 554us/sample - loss: 0.0066 -
mean_squared_error: 0.0066
Epoch 39/100
52/52 [==============================] - 0s 623us/sample - loss: 0.0066 -
mean_squared_error: 0.0066
Epoch 40/100
52/52 [==============================] - 0s 245us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 41/100
52/52 [==============================] - 0s 436us/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 42/100
52/52 [==============================] - 0s 480us/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 43/100
52/52 [==============================] - 0s 542us/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 44/100
52/52 [==============================] - 0s 689us/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 45/100
52/52 [==============================] - 0s 623us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 46/100
52/52 [==============================] - 0s 595us/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 47/100
52/52 [==============================] - 0s 838us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 48/100
52/52 [==============================] - 0s 580us/sample - loss: 0.0069 -
mean_squared_error: 0.0069
Epoch 49/100
52/52 [==============================] - 0s 429us/sample - loss: 0.0071 -
mean_squared_error: 0.0071
Epoch 50/100
52/52 [==============================] - 0s 634us/sample - loss: 0.0069 -
```

```
mean_squared_error: 0.0069
Epoch 51/100
52/52 [==============================] - 0s 442us/sample - loss: 0.0070 -
mean_squared_error: 0.0070
Epoch 52/100
52/52 [==============================] - 0s 579us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 53/100
52/52 [==============================] - 0s 415us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 54/100
52/52 [==============================] - 0s 639us/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 55/100
52/52 [==============================] - 0s 732us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 56/100
52/52 [==============================] - 0s 340us/sample - loss: 0.0066 -
mean_squared_error: 0.0066
Epoch 57/100
52/52 [==============================] - 0s 384us/sample - loss: 0.0066 -
mean_squared_error: 0.0066
Epoch 58/100
52/52 [==============================] - 0s 662us/sample - loss: 0.0066 -
mean_squared_error: 0.0066
Epoch 59/100
52/52 [==============================] - 0s 479us/sample - loss: 0.0066 -
mean_squared_error: 0.0066
Epoch 60/100
52/52 [==============================] - 0s 344us/sample - loss: 0.0066 -
mean_squared_error: 0.0066
Epoch 61/100
52/52 [==============================] - 0s 792us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 62/100
52/52 [==============================] - 0s 795us/sample - loss: 0.0066 -
mean_squared_error: 0.0066
Epoch 63/100
52/52 [==============================] - 0s 399us/sample - loss: 0.0066 -
mean_squared_error: 0.0066
Epoch 64/100
52/52 [==============================] - 0s 369us/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 65/100
52/52 [==============================] - 0s 434us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 66/100
52/52 [==============================] - 0s 525us/sample - loss: 0.0072 -
```

```
mean_squared_error: 0.0072
Epoch 67/100
52/52 [==============================] - 0s 543us/sample - loss: 0.0069 -
mean_squared_error: 0.0069
Epoch 68/100
52/52 [==============================] - 0s 447us/sample - loss: 0.0070 -
mean_squared_error: 0.0070
Epoch 69/100
52/52 [==============================] - 0s 309us/sample - loss: 0.0071 -
mean_squared_error: 0.0071
Epoch 70/100
52/52 [==============================] - 0s 441us/sample - loss: 0.0070 -
mean_squared_error: 0.0070
Epoch 71/100
52/52 [==============================] - 0s 478us/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 72/100
52/52 [==============================] - 0s 666us/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 73/100
52/52 [==============================] - 0s 505us/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 74/100
52/52 [==============================] - 0s 506us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 75/100
52/52 [==============================] - 0s 501us/sample - loss: 0.0066 -
mean_squared_error: 0.0066
Epoch 76/100
52/52 [==============================] - 0s 339us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 77/100
52/52 [==============================] - 0s 278us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 78/100
52/52 [==============================] - 0s 534us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 79/100
52/52 [==============================] - 0s 470us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 80/100
52/52 [==============================] - 0s 589us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 81/100
52/52 [==============================] - 0s 411us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 82/100
52/52 [==============================] - 0s 587us/sample - loss: 0.0068 -
```

```
mean_squared_error: 0.0068
Epoch 83/100
52/52 [==============================] - 0s 636us/sample - loss: 0.0066 -
mean_squared_error: 0.0066
Epoch 84/100
52/52 [==============================] - 0s 477us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 85/100
52/52 [==============================] - 0s 387us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 86/100
52/52 [==============================] - 0s 599us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 87/100
52/52 [==============================] - 0s 329us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 88/100
52/52 [==============================] - 0s 450us/sample - loss: 0.0066 -
mean_squared_error: 0.0066
Epoch 89/100
52/52 [==============================] - 0s 581us/sample - loss: 0.0066 -
mean_squared_error: 0.0066
Epoch 90/100
52/52 [==============================] - 0s 743us/sample - loss: 0.0066 -
mean_squared_error: 0.0066
Epoch 91/100
52/52 [==============================] - 0s 729us/sample - loss: 0.0066 -
mean_squared_error: 0.0066
Epoch 92/100
52/52 [==============================] - 0s 389us/sample - loss: 0.0066 -
mean_squared_error: 0.0066
Epoch 93/100
52/52 [==============================] - 0s 425us/sample - loss: 0.0068 -
mean_squared_error: 0.0068
Epoch 94/100
52/52 [==============================] - 0s 315us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 95/100
52/52 [==============================] - 0s 461us/sample - loss: 0.0067 -
mean_squared_error: 0.0067
Epoch 96/100
52/52 [==============================] - 0s 572us/sample - loss: 0.0072 -
mean_squared_error: 0.0072
Epoch 97/100
52/52 [==============================] - 0s 685us/sample - loss: 0.0070 -
mean_squared_error: 0.0070
Epoch 98/100
52/52 [==============================] - 0s 559us/sample - loss: 0.0068 -
```

```
mean_squared_error: 0.0068
Epoch 99/100
52/52 [==============================] - 0s 437us/sample - loss: 0.0071 -
mean_squared_error: 0.0071
Epoch 100/100
52/52 [==============================] - 0s 1ms/sample - loss: 0.0067 -
mean_squared_error: 0.0067
```

[297]:
```
<keras.callbacks.History at 0x7f9601e624c0>
```

L'entrainement peut être long et il faut au moins 100000 epochs pour avoir une approximation satisfaisante. On peut relancer à plusieurs reprise le fit() il repart de là ou le modèle était arrivé ce qui permet de suivre pas à pas l'evolution du loss. Au besoin avec keras il y a la possibilité de sauvegarder le modèle et de la recharger ensuite: https://www.tensorflow.org/guide/keras/save_and_serialize?hl=fr

[276]:
```
mon_model.save('model_ventes-pneus')
```

```
WARNING:tensorflow:TensorFlow optimizers do not make it possible to access
optimizer attributes or optimizer state after instantiation. As a result, we
cannot save the optimizer as part of the model save file. You will have to
compile your model again after loading it. Prefer using a Keras optimizer
instead (see keras.io/optimizers).
```

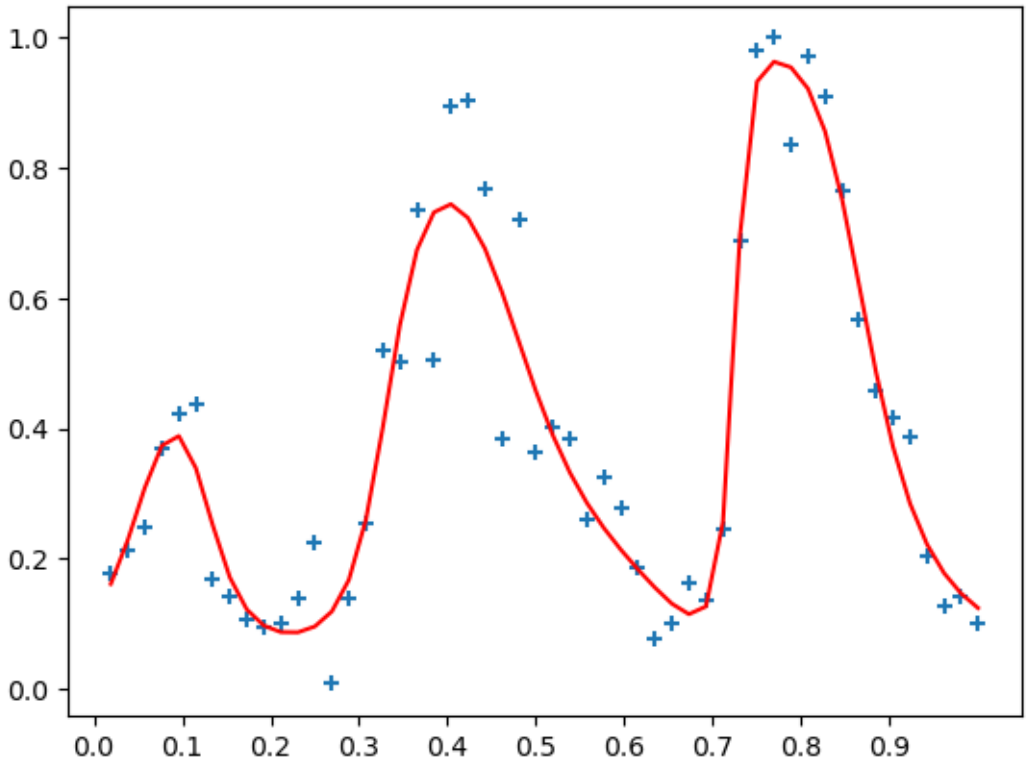
Je me suis arrété aux valeurs suivantes loss: 0.0069 - mean_squared_error: 0.0069

[272]:
```
ventes_pred=mon_model.predict(vente_pneus['numero_semaine_norm'].to_numpy())
```

```
2023-05-03 16:39:29.124880: W tensorflow/c/c_api.cc:300] Operation
'{name:'dense_20/Sigmoid' id:2310 op device:{requested: '', assigned: ''}
def:{{{node dense_20/Sigmoid}} = Sigmoid[T=DT_FLOAT,
_has_manual_control_dependencies=true](dense_20/BiasAdd)}}' was changed by
setting attribute after it was run by a session. This mutation will have no
effect, and will trigger an error in the future. Either don't modify nodes after
running them or create a new session.
```

[274]:
```
import numpy as np
listOf_Xticks = np.arange(0, 1, 0.1)
plt.xticks(listOf_Xticks)
plt.
  ↪scatter(vente_pneus['numero_semaine_norm'],vente_pneus['moy_norm'],marker='+')
plt.plot(vente_pneus['numero_semaine_norm'],ventes_pred,color='red')
```

[274]:
```
[<matplotlib.lines.Line2D at 0x7f9601bfaa30>]
```

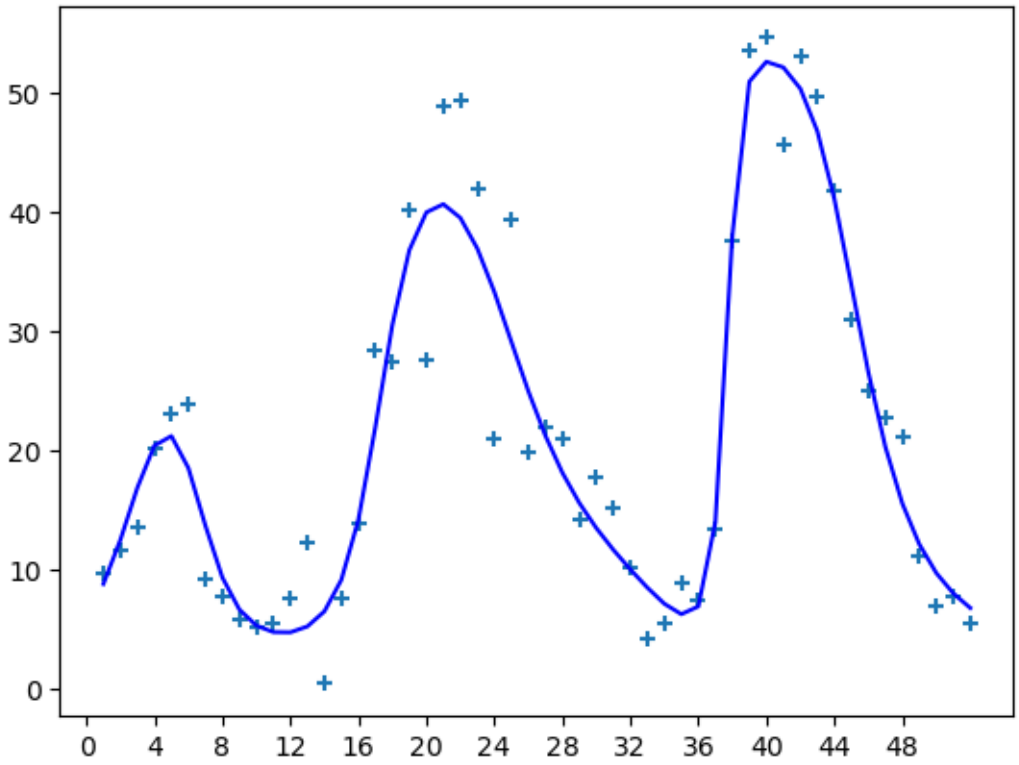


Biensur pour revenir aux valeurs réelles il faudrait faire

[292]:
```
pred=[]
for i in range(1,53,1):
    pred.append(moy.max()*mon_model.predict(np.array([[i/52]]))[0][0])
```

[294]:
```
import numpy as np
listOf_Xticks = np.arange(0, 52, 4)
plt.xticks(listOf_Xticks)
plt.scatter(vente_pneus['numero semaine'],vente_pneus['moy'],marker='+')
plt.plot(vente_pneus['numero semaine'],pred,color='blue')
```

[294]:
```
[<matplotlib.lines.Line2D at 0x7f9601847ac0>]
```


50
40
30
20
10
0
0
4
8
12
16
20
24
28
32
36
40
44
48

[ ]: