

Atelier 2 : Régression avec réseau de neurones

Objectifs :

Dans cet atelier, vous allons découvrir comment développer un modèle de réseau de neurones **Multi-layer perceptron (MLP)** pour la régression **de l'assurance automobile suédoise**.

Cet atelier a pour objectif de développer et évaluer des techniques de régression basées sur MLP pour prédire le paiement total pour les sinistres en milliers de couronnes suédoises utilisant la base de données « **Auto Insurance** » et le package **scikit-learn**

1- Charger et visualiser la base de données « Auto Insurance »

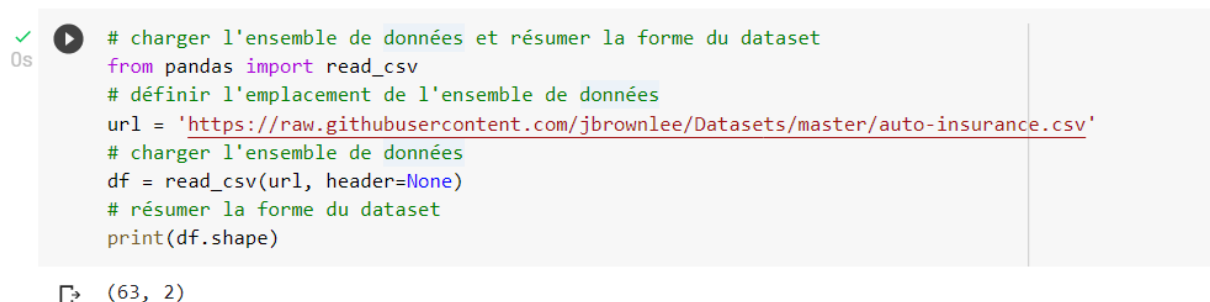
La première étape consiste à définir et à explorer la base de données « **Auto Insurance** » qui décrit l'assurance automobile suédoise. Cette base de données contient une seule variable d'entrée (le nombre de sinistres), et la variable cible (paiement total pour les sinistres en milliers de couronnes suédoises). L'objectif est de prédire le paiement total en fonction du nombre de sinistres.

Vous pouvez en savoir plus sur le dataset ici :

- [Auto Insurance Dataset \(auto-insurance.csv\)](#)
- [Auto Insurance Dataset Details \(auto-insurance.names\)](#)

a- Charger la base de données

Nous pouvons charger la base de données comme un DataFrame pandas directement à partir de l'URL (Figure 1).



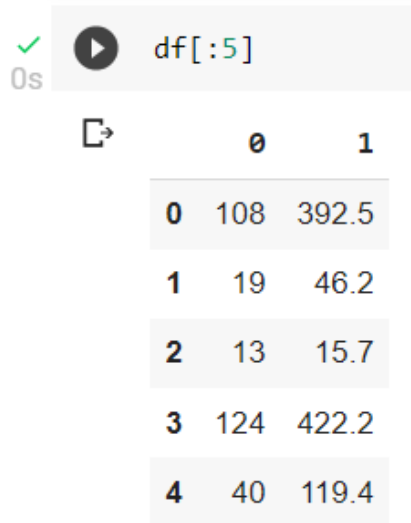
```
✓ 0s # charger l'ensemble de données et résumer la forme du dataset
from pandas import read_csv
# définir l'emplacement de l'ensemble de données
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/auto-insurance.csv'
# charger l'ensemble de données
df = read_csv(url, header=None)
# résumer la forme du dataset
print(df.shape)
```

↳ (63, 2)

Figure 1

Le résultat montre que la base de données comporte deux variables (une entrée et une sortie) et 63 lignes de données.

b- Afficher les 5 premières lignes de la base de données



0s

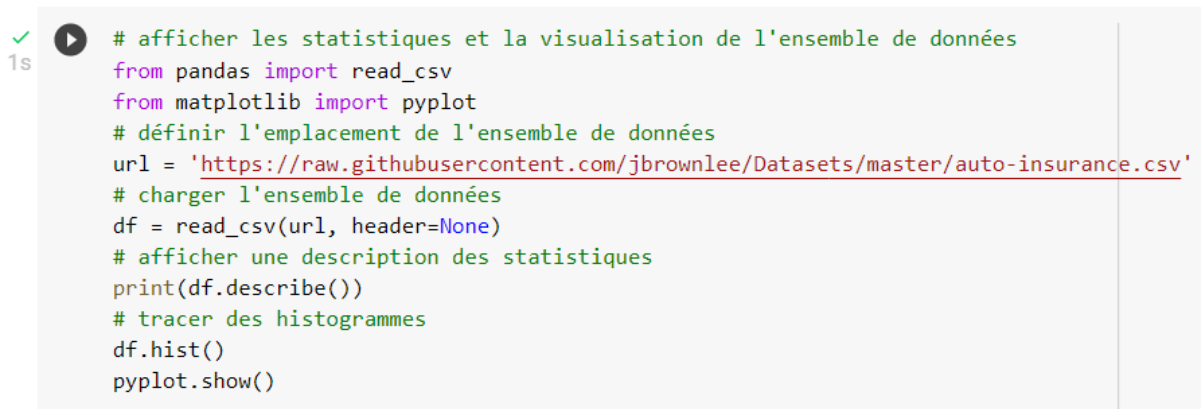
df[:5]

	0	1
0	108	392.5
1	19	46.2
2	13	15.7
3	124	422.2
4	40	119.4

Figure 2

c- Visualisation des attributs

Ensuite, nous pouvons en apprendre davantage sur la base de données (Figure 3).



1s

```
# afficher les statistiques et la visualisation de l'ensemble de données
from pandas import read_csv
from matplotlib import pyplot
# définir l'emplacement de l'ensemble de données
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/auto-insurance.csv'
# charger l'ensemble de données
df = read_csv(url, header=None)
# afficher une description des statistiques
print(df.describe())
# tracer des histogrammes
df.hist()
pyplot.show()
```

Figure 3

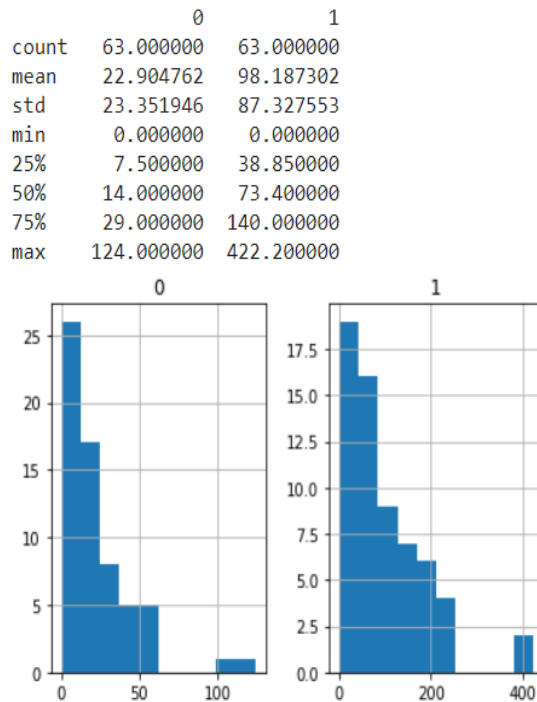


Figure 4

Que peut-on dire des deux variables de la base de données depuis la figure 4 ?

2- Multilayers perceptrons (MLP)

Dans cette partie, nous allons développer un modèle de multilayers perceptron (MLP) pour notre base de données en utilisant TensorFlow.

Nous allons utiliser une simple **répartition train/test** des données et examiner les graphiques des courbes d'apprentissage. Cela nous aidera à voir si nous sommes en **sur-apprentissage** ou en **sous-apprentissage (over-learning or under-learning)** ; nous pourrions alors adapter la configuration en conséquence.

Tout d'abord, nous pouvons diviser l'ensemble de données en variables d'entrée et de sortie, puis en 67/33 ensembles de train-test (Figure 5)

```
# divisé en colonnes d'entrée et de sortie
X, y = df.values[:, :-1], df.values[:, -1]
# diviser en train-test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
```

Figure 5

Ensuite, nous pouvons définir notre modèle MLP (Figure 6). Nous allons utiliser :

- Une couche cachée (**hidden layer**) de 10 nœuds et une couche de sortie (**output layer**).
- La fonction d'activation ReLU dans la couche cachée et l'**initialisation des poids (weight initialization)** « he_normal ».
- La sortie du modèle est une activation linéaire et nous minimiserons la perte du (**mean squared error MSE**).
- Étant donné que l'ensemble de données est petit, une petite **taille de lot (batch size)** est probablement une bonne idée, par exemple 8 ou 16 lignes.
- Nous ajusterons le modèle pour 100 **épisodes d'apprentissage (training epochs)**.
- L'utilisation de la **version Adam de la descente de gradient stochastique (Adam version of stochastic gradient descent)** car elle adapte automatiquement le **taux d'apprentissage (learning rate)**.

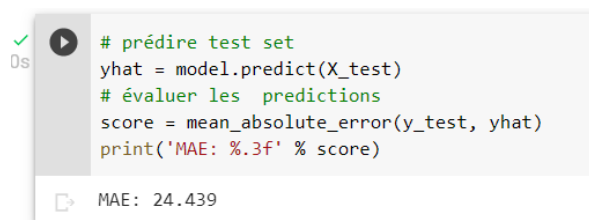


```
# déterminer le nombre de caractéristiques d'entrée
n_features = X.shape[1]
# définir le modèle
model = Sequential()
model.add(Dense(10, activation='relu', kernel_initializer='he_normal', input_shape=(n_features,)))
model.add(Dense(1))
# compiler le modèle
model.compile(optimizer='adam', loss='mse')

# fit model
history = model.fit(X_train, y_train, epochs=100, batch_size=8, verbose=0, validation_data=(X_test, y_test))
```

Figure 6

À la fin du training, nous évaluerons les performances du modèle sur l'ensemble de données de test et les rapporterons sous forme d'erreur absolue moyenne (MAE) comme indiqué dans la Figure 7.



```
# prédire test set
yhat = model.predict(X_test)
# évaluer les prédictions
score = mean_absolute_error(y_test, yhat)
print('MAE: %.3f' % score)
```

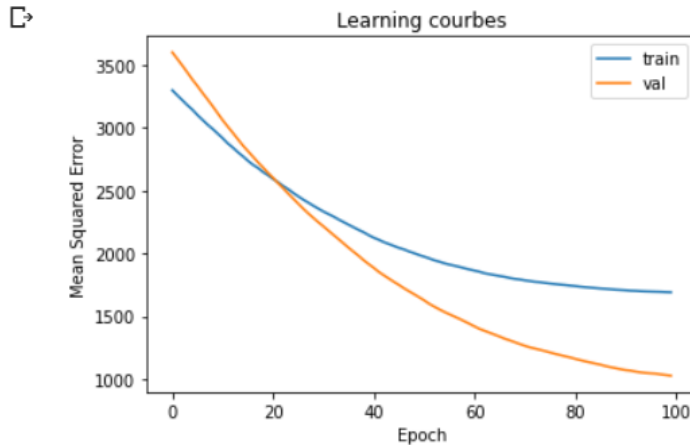
MAE: 24.439

Figure 7

Enfin, nous tracerons la courbe d'apprentissage de l'erreur MSE sur les ensembles du training et de test durant le training (Figure 8).

✓
0s

```
# tracer learning courbes
pyplot.title('Learning courbes')
pyplot.xlabel('Epoch')
pyplot.ylabel('Mean Squared Error')
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='val')
pyplot.legend()
pyplot.show()
```



Travail demandé

Dans cet exercice nous utiliserons l'ensemble de données "**abalone**". Déterminer l'âge d'un ormeau est une tâche qui prend du temps et il est souhaitable de déterminer l'âge à partir des seuls détails physiques.

Il s'agit d'un ensemble de données qui décrit les détails physiques de l'ormeau et qui nécessite de prédire le nombre d'anneaux de l'ormeau, qui est un indicateur de l'âge de la créature.

Vous pouvez en savoir plus sur ce jeu de données en cliquant ici :

1) Charger le dataset abalone à partir de l'url ci-dessus et afficher ses caractéristiques.

url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/abalone.csv'

2) Visualiser les 5 premières lignes.

3) Construite et évaluer un modèle MLP (architecture de votre choix) sur le critère d'évaluation MAE.