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

from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler, OneHotEncoder
from sklearn.metrics import mean_squared_error, mean_absolute_error

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.utils import to_categorical

X_train_clean = pd.read_csv("./X_train_clean.csv", index_col=0)
X_val_clean = pd.read_csv("./X_val_clean.csv", index_col=0)
X_test_clean = pd.read_csv("./X_test_clean.csv", index_col=0)
y_train = pd.read_csv("./y_train.csv", index_col=0)
y_val = pd.read_csv("./y_val.csv", index_col=0)
y_test = pd.read_csv("./y_test.csv", index_col=0)

print(X_train_clean.shape)
print(X_test_clean.shape)
print(X_val_clean.shape)
print(y_train.shape)
print(y_test.shape)
print(y_val.shape)
```

```
(3897, 12)
(1300, 12)
(1300, 12)
(3897, 1)
(1300, 1)
(1300, 1)
```

```python
y_train_clean = to_categorical(y_train, num_classes=10)
y_val_clean = to_categorical(y_val, num_classes=10)
y_test_clean = to_categorical(y_test, num_classes=10)

print(y_train.iloc[0])
print(y_train_clean[0])
print(y_train_clean.shape[1])
```

```
quality    6
Name: 1413, dtype: int64
[0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
10
```

Création du modèle de classification

Dans un premier temps, on crée un modèle qui réalise un classification sur 10 notes (quality).

```
input_dim = X_train_clean.shape[1]
output_dim = y_train_clean.shape[1]

def create_model():
  model = Sequential()

  model.add(Input(shape=(input_dim,)))

  model.add(Dense(20, activation="relu"))
  model.add(Dense(30, activation="relu"))
  model.add(Dense(40, activation="relu"))
  model.add(Dense(20, activation="relu"))

  model.add(Dense(output_dim, activation="softmax"))

  model.compile(optimizer="adam", loss="categorical_crossentropy",
metrics=["accuracy"])

  model.summary()

  return model

model = create_model()
```

```
Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                        ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
━━━━━━━━━━━━━━━━━━┩
│ dense (Dense)                       │ (None, 20)                  │
260 │
├───────────────────────────────────────┼─────────────────────────────┤
──────────────────┤
│ dense_1 (Dense)                     │ (None, 30)                  │
630 │
├───────────────────────────────────────┼─────────────────────────────┤
──────────────────┤
│ dense_2 (Dense)                     │ (None, 40)                  │
1,240 │
├───────────────────────────────────────┼─────────────────────────────┤
──────────────────┤
│ dense_3 (Dense)                     │ (None, 20)                  │
820 │
├───────────────────────────────────────┼─────────────────────────────┤
```

```
────────────────────┤
│ dense_4 (Dense)                      │ (None, 10)                     │
210 │
└──────────────────────────────────────┴──────────────────────────────┴
────────────────────┘

 Total params: 3,160 (12.34 KB)

 Trainable params: 3,160 (12.34 KB)

 Non-trainable params: 0 (0.00 B)

nb_epochs = 100

history = model.fit(X_train_clean,
          y_train_clean,
          epochs = nb_epochs,
          validation_data= (X_val_clean, y_val_clean),
          verbose=2
)

Epoch 1/100
122/122 - 3s - 24ms/step - accuracy: 0.4403 - loss: 1.5092 -
val_accuracy: 0.4292 - val_loss: 1.2368
Epoch 2/100
122/122 - 1s - 6ms/step - accuracy: 0.4865 - loss: 1.2124 -
val_accuracy: 0.5038 - val_loss: 1.1494
Epoch 3/100
122/122 - 0s - 2ms/step - accuracy: 0.5219 - loss: 1.1482 -
val_accuracy: 0.5146 - val_loss: 1.1150
Epoch 4/100
122/122 - 1s - 5ms/step - accuracy: 0.5250 - loss: 1.1201 -
val_accuracy: 0.5115 - val_loss: 1.1100
Epoch 5/100
122/122 - 0s - 2ms/step - accuracy: 0.5361 - loss: 1.1081 -
val_accuracy: 0.5246 - val_loss: 1.0990
Epoch 6/100
122/122 - 0s - 2ms/step - accuracy: 0.5250 - loss: 1.1037 -
val_accuracy: 0.5100 - val_loss: 1.1429
Epoch 7/100
122/122 - 0s - 2ms/step - accuracy: 0.5284 - loss: 1.1025 -
val_accuracy: 0.5162 - val_loss: 1.0955
Epoch 8/100
122/122 - 0s - 2ms/step - accuracy: 0.5432 - loss: 1.0919 -
val_accuracy: 0.5254 - val_loss: 1.0858
Epoch 9/100
122/122 - 0s - 3ms/step - accuracy: 0.5481 - loss: 1.0906 -
val_accuracy: 0.5246 - val_loss: 1.0947
Epoch 10/100
122/122 - 0s - 2ms/step - accuracy: 0.5448 - loss: 1.0880 -
val_accuracy: 0.5377 - val_loss: 1.0833
```

```
Epoch 11/100
122/122 - 0s - 2ms/step - accuracy: 0.5481 - loss: 1.0830 -
val_accuracy: 0.5123 - val_loss: 1.0990
Epoch 12/100
122/122 - 0s - 2ms/step - accuracy: 0.5432 - loss: 1.0851 -
val_accuracy: 0.5462 - val_loss: 1.0834
Epoch 13/100
122/122 - 0s - 2ms/step - accuracy: 0.5453 - loss: 1.0802 -
val_accuracy: 0.5369 - val_loss: 1.0832
Epoch 14/100
122/122 - 0s - 2ms/step - accuracy: 0.5476 - loss: 1.0801 -
val_accuracy: 0.5323 - val_loss: 1.0863
Epoch 15/100
122/122 - 0s - 3ms/step - accuracy: 0.5471 - loss: 1.0784 -
val_accuracy: 0.5385 - val_loss: 1.0757
Epoch 16/100
122/122 - 0s - 2ms/step - accuracy: 0.5486 - loss: 1.0758 -
val_accuracy: 0.5323 - val_loss: 1.0794
Epoch 17/100
122/122 - 0s - 2ms/step - accuracy: 0.5509 - loss: 1.0731 -
val_accuracy: 0.5338 - val_loss: 1.0712
Epoch 18/100
122/122 - 0s - 2ms/step - accuracy: 0.5494 - loss: 1.0703 -
val_accuracy: 0.5369 - val_loss: 1.0671
Epoch 19/100
122/122 - 0s - 2ms/step - accuracy: 0.5514 - loss: 1.0680 -
val_accuracy: 0.5469 - val_loss: 1.0737
Epoch 20/100
122/122 - 0s - 2ms/step - accuracy: 0.5502 - loss: 1.0677 -
val_accuracy: 0.5315 - val_loss: 1.0740
Epoch 21/100
122/122 - 0s - 2ms/step - accuracy: 0.5545 - loss: 1.0668 -
val_accuracy: 0.5369 - val_loss: 1.0732
Epoch 22/100
122/122 - 0s - 2ms/step - accuracy: 0.5502 - loss: 1.0648 -
val_accuracy: 0.5385 - val_loss: 1.0689
Epoch 23/100
122/122 - 0s - 2ms/step - accuracy: 0.5545 - loss: 1.0617 -
val_accuracy: 0.5285 - val_loss: 1.0710
Epoch 24/100
122/122 - 0s - 2ms/step - accuracy: 0.5514 - loss: 1.0616 -
val_accuracy: 0.5508 - val_loss: 1.0628
Epoch 25/100
122/122 - 0s - 2ms/step - accuracy: 0.5558 - loss: 1.0613 -
val_accuracy: 0.5254 - val_loss: 1.0647
Epoch 26/100
122/122 - 0s - 3ms/step - accuracy: 0.5540 - loss: 1.0559 -
val_accuracy: 0.5462 - val_loss: 1.0682
Epoch 27/100
```

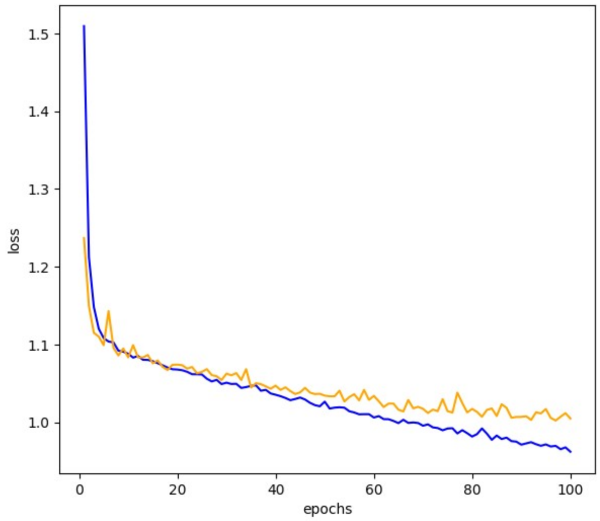
```
122/122 - 0s - 2ms/step - accuracy: 0.5535 - loss: 1.0524 -
val_accuracy: 0.5323 - val_loss: 1.0605
Epoch 28/100
122/122 - 0s - 3ms/step - accuracy: 0.5517 - loss: 1.0544 -
val_accuracy: 0.5369 - val_loss: 1.0591
Epoch 29/100
122/122 - 0s - 2ms/step - accuracy: 0.5586 - loss: 1.0490 -
val_accuracy: 0.5431 - val_loss: 1.0542
Epoch 30/100
122/122 - 0s - 2ms/step - accuracy: 0.5638 - loss: 1.0508 -
val_accuracy: 0.5385 - val_loss: 1.0624
Epoch 31/100
122/122 - 0s - 2ms/step - accuracy: 0.5579 - loss: 1.0491 -
val_accuracy: 0.5300 - val_loss: 1.0603
Epoch 32/100
122/122 - 0s - 2ms/step - accuracy: 0.5612 - loss: 1.0493 -
val_accuracy: 0.5423 - val_loss: 1.0631
Epoch 33/100
122/122 - 0s - 2ms/step - accuracy: 0.5563 - loss: 1.0440 -
val_accuracy: 0.5331 - val_loss: 1.0542
Epoch 34/100
122/122 - 0s - 2ms/step - accuracy: 0.5527 - loss: 1.0451 -
val_accuracy: 0.5377 - val_loss: 1.0682
Epoch 35/100
122/122 - 0s - 3ms/step - accuracy: 0.5530 - loss: 1.0466 -
val_accuracy: 0.5462 - val_loss: 1.0446
Epoch 36/100
122/122 - 1s - 6ms/step - accuracy: 0.5594 - loss: 1.0474 -
val_accuracy: 0.5338 - val_loss: 1.0501
Epoch 37/100
122/122 - 0s - 3ms/step - accuracy: 0.5617 - loss: 1.0404 -
val_accuracy: 0.5346 - val_loss: 1.0489
Epoch 38/100
122/122 - 0s - 3ms/step - accuracy: 0.5609 - loss: 1.0416 -
val_accuracy: 0.5431 - val_loss: 1.0461
Epoch 39/100
122/122 - 0s - 4ms/step - accuracy: 0.5599 - loss: 1.0369 -
val_accuracy: 0.5515 - val_loss: 1.0430
Epoch 40/100
122/122 - 1s - 5ms/step - accuracy: 0.5594 - loss: 1.0353 -
val_accuracy: 0.5400 - val_loss: 1.0470
Epoch 41/100
122/122 - 0s - 3ms/step - accuracy: 0.5668 - loss: 1.0334 -
val_accuracy: 0.5446 - val_loss: 1.0414
Epoch 42/100
122/122 - 1s - 4ms/step - accuracy: 0.5607 - loss: 1.0311 -
val_accuracy: 0.5431 - val_loss: 1.0448
Epoch 43/100
122/122 - 0s - 3ms/step - accuracy: 0.5635 - loss: 1.0283 -
```

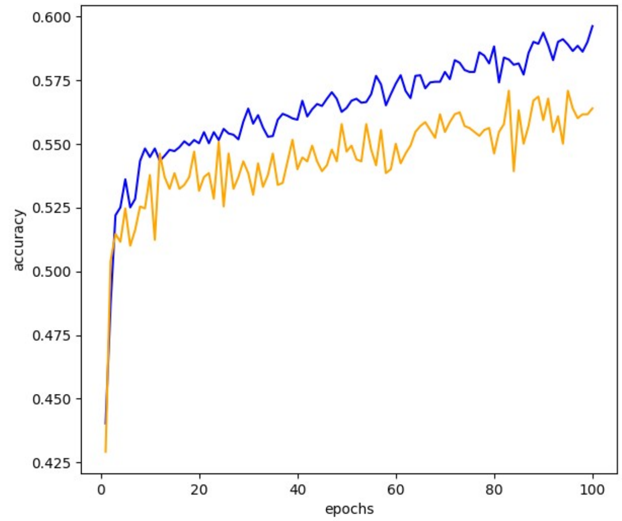
```
val_accuracy: 0.5492 - val_loss: 1.0400
Epoch 44/100
122/122 - 0s - 2ms/step - accuracy: 0.5656 - loss: 1.0298 -
val_accuracy: 0.5431 - val_loss: 1.0362
Epoch 45/100
122/122 - 0s - 2ms/step - accuracy: 0.5648 - loss: 1.0317 -
val_accuracy: 0.5392 - val_loss: 1.0384
Epoch 46/100
122/122 - 0s - 3ms/step - accuracy: 0.5676 - loss: 1.0291 -
val_accuracy: 0.5415 - val_loss: 1.0441
Epoch 47/100
122/122 - 0s - 2ms/step - accuracy: 0.5702 - loss: 1.0249 -
val_accuracy: 0.5477 - val_loss: 1.0380
Epoch 48/100
122/122 - 0s - 2ms/step - accuracy: 0.5676 - loss: 1.0219 -
val_accuracy: 0.5431 - val_loss: 1.0361
Epoch 49/100
122/122 - 0s - 2ms/step - accuracy: 0.5625 - loss: 1.0204 -
val_accuracy: 0.5577 - val_loss: 1.0365
Epoch 50/100
122/122 - 0s - 3ms/step - accuracy: 0.5640 - loss: 1.0264 -
val_accuracy: 0.5469 - val_loss: 1.0341
Epoch 51/100
122/122 - 0s - 2ms/step - accuracy: 0.5668 - loss: 1.0175 -
val_accuracy: 0.5492 - val_loss: 1.0333
Epoch 52/100
122/122 - 0s - 2ms/step - accuracy: 0.5676 - loss: 1.0187 -
val_accuracy: 0.5438 - val_loss: 1.0333
Epoch 53/100
122/122 - 0s - 2ms/step - accuracy: 0.5661 - loss: 1.0192 -
val_accuracy: 0.5431 - val_loss: 1.0404
Epoch 54/100
122/122 - 0s - 2ms/step - accuracy: 0.5663 - loss: 1.0186 -
val_accuracy: 0.5577 - val_loss: 1.0266
Epoch 55/100
122/122 - 0s - 2ms/step - accuracy: 0.5694 - loss: 1.0140 -
val_accuracy: 0.5477 - val_loss: 1.0323
Epoch 56/100
122/122 - 0s - 3ms/step - accuracy: 0.5766 - loss: 1.0125 -
val_accuracy: 0.5415 - val_loss: 1.0360
Epoch 57/100
122/122 - 0s - 3ms/step - accuracy: 0.5733 - loss: 1.0101 -
val_accuracy: 0.5554 - val_loss: 1.0280
Epoch 58/100
122/122 - 0s - 2ms/step - accuracy: 0.5651 - loss: 1.0103 -
val_accuracy: 0.5385 - val_loss: 1.0415
Epoch 59/100
122/122 - 0s - 2ms/step - accuracy: 0.5694 - loss: 1.0103 -
val_accuracy: 0.5400 - val_loss: 1.0287
```

```
Epoch 60/100
122/122 - 0s - 2ms/step - accuracy: 0.5735 - loss: 1.0060 -
val_accuracy: 0.5500 - val_loss: 1.0338
Epoch 61/100
122/122 - 0s - 2ms/step - accuracy: 0.5769 - loss: 1.0078 -
val_accuracy: 0.5423 - val_loss: 1.0269
Epoch 62/100
122/122 - 0s - 3ms/step - accuracy: 0.5707 - loss: 1.0039 -
val_accuracy: 0.5462 - val_loss: 1.0195
Epoch 63/100
122/122 - 0s - 3ms/step - accuracy: 0.5679 - loss: 1.0038 -
val_accuracy: 0.5492 - val_loss: 1.0238
Epoch 64/100
122/122 - 0s - 2ms/step - accuracy: 0.5766 - loss: 1.0016 -
val_accuracy: 0.5546 - val_loss: 1.0239
Epoch 65/100
122/122 - 0s - 2ms/step - accuracy: 0.5769 - loss: 0.9988 -
val_accuracy: 0.5569 - val_loss: 1.0160
Epoch 66/100
122/122 - 0s - 2ms/step - accuracy: 0.5717 - loss: 1.0031 -
val_accuracy: 0.5585 - val_loss: 1.0138
Epoch 67/100
122/122 - 0s - 2ms/step - accuracy: 0.5740 - loss: 0.9990 -
val_accuracy: 0.5554 - val_loss: 1.0285
Epoch 68/100
122/122 - 0s - 2ms/step - accuracy: 0.5743 - loss: 0.9996 -
val_accuracy: 0.5523 - val_loss: 1.0178
Epoch 69/100
122/122 - 0s - 2ms/step - accuracy: 0.5743 - loss: 0.9989 -
val_accuracy: 0.5615 - val_loss: 1.0198
Epoch 70/100
122/122 - 0s - 3ms/step - accuracy: 0.5781 - loss: 0.9954 -
val_accuracy: 0.5546 - val_loss: 1.0172
Epoch 71/100
122/122 - 0s - 2ms/step - accuracy: 0.5753 - loss: 0.9972 -
val_accuracy: 0.5585 - val_loss: 1.0118
Epoch 72/100
122/122 - 0s - 2ms/step - accuracy: 0.5828 - loss: 0.9933 -
val_accuracy: 0.5615 - val_loss: 1.0162
Epoch 73/100
122/122 - 0s - 2ms/step - accuracy: 0.5817 - loss: 0.9924 -
val_accuracy: 0.5623 - val_loss: 1.0142
Epoch 74/100
122/122 - 0s - 3ms/step - accuracy: 0.5789 - loss: 0.9895 -
val_accuracy: 0.5569 - val_loss: 1.0296
Epoch 75/100
122/122 - 0s - 3ms/step - accuracy: 0.5781 - loss: 0.9919 -
val_accuracy: 0.5562 - val_loss: 1.0141
Epoch 76/100
```

```
122/122 - 1s - 5ms/step - accuracy: 0.5781 - loss: 0.9922 -
val_accuracy: 0.5546 - val_loss: 1.0124
Epoch 77/100
122/122 - 1s - 6ms/step - accuracy: 0.5858 - loss: 0.9855 -
val_accuracy: 0.5531 - val_loss: 1.0380
Epoch 78/100
122/122 - 1s - 5ms/step - accuracy: 0.5846 - loss: 0.9897 -
val_accuracy: 0.5554 - val_loss: 1.0245
Epoch 79/100
122/122 - 0s - 4ms/step - accuracy: 0.5815 - loss: 0.9858 -
val_accuracy: 0.5562 - val_loss: 1.0125
Epoch 80/100
122/122 - 0s - 2ms/step - accuracy: 0.5881 - loss: 0.9815 -
val_accuracy: 0.5462 - val_loss: 1.0171
Epoch 81/100
122/122 - 0s - 2ms/step - accuracy: 0.5740 - loss: 0.9847 -
val_accuracy: 0.5546 - val_loss: 1.0132
Epoch 82/100
122/122 - 0s - 2ms/step - accuracy: 0.5838 - loss: 0.9919 -
val_accuracy: 0.5577 - val_loss: 1.0070
Epoch 83/100
122/122 - 0s - 3ms/step - accuracy: 0.5830 - loss: 0.9852 -
val_accuracy: 0.5708 - val_loss: 1.0157
Epoch 84/100
122/122 - 0s - 2ms/step - accuracy: 0.5810 - loss: 0.9773 -
val_accuracy: 0.5392 - val_loss: 1.0177
Epoch 85/100
122/122 - 0s - 3ms/step - accuracy: 0.5815 - loss: 0.9828 -
val_accuracy: 0.5631 - val_loss: 1.0079
Epoch 86/100
122/122 - 0s - 2ms/step - accuracy: 0.5771 - loss: 0.9783 -
val_accuracy: 0.5500 - val_loss: 1.0231
Epoch 87/100
122/122 - 0s - 2ms/step - accuracy: 0.5856 - loss: 0.9801 -
val_accuracy: 0.5569 - val_loss: 1.0184
Epoch 88/100
122/122 - 0s - 2ms/step - accuracy: 0.5899 - loss: 0.9756 -
val_accuracy: 0.5669 - val_loss: 1.0057
Epoch 89/100
122/122 - 0s - 3ms/step - accuracy: 0.5892 - loss: 0.9749 -
val_accuracy: 0.5685 - val_loss: 1.0067
Epoch 90/100
122/122 - 0s - 2ms/step - accuracy: 0.5935 - loss: 0.9710 -
val_accuracy: 0.5592 - val_loss: 1.0068
Epoch 91/100
122/122 - 0s - 2ms/step - accuracy: 0.5887 - loss: 0.9727 -
val_accuracy: 0.5677 - val_loss: 1.0076
Epoch 92/100
122/122 - 0s - 2ms/step - accuracy: 0.5828 - loss: 0.9743 -
```

```
val_accuracy: 0.5546 - val_loss: 1.0029
Epoch 93/100
122/122 - 0s - 2ms/step - accuracy: 0.5899 - loss: 0.9716 -
val_accuracy: 0.5608 - val_loss: 1.0126
Epoch 94/100
122/122 - 0s - 2ms/step - accuracy: 0.5910 - loss: 0.9696 -
val_accuracy: 0.5500 - val_loss: 1.0113
Epoch 95/100
122/122 - 0s - 2ms/step - accuracy: 0.5889 - loss: 0.9714 -
val_accuracy: 0.5708 - val_loss: 1.0169
Epoch 96/100
122/122 - 0s - 2ms/step - accuracy: 0.5863 - loss: 0.9687 -
val_accuracy: 0.5638 - val_loss: 1.0055
Epoch 97/100
122/122 - 0s - 2ms/step - accuracy: 0.5884 - loss: 0.9695 -
val_accuracy: 0.5600 - val_loss: 1.0021
Epoch 98/100
122/122 - 0s - 2ms/step - accuracy: 0.5861 - loss: 0.9652 -
val_accuracy: 0.5615 - val_loss: 1.0070
Epoch 99/100
122/122 - 0s - 3ms/step - accuracy: 0.5899 - loss: 0.9676 -
val_accuracy: 0.5615 - val_loss: 1.0116
Epoch 100/100
122/122 - 0s - 2ms/step - accuracy: 0.5961 - loss: 0.9620 -
val_accuracy: 0.5638 - val_loss: 1.0047

def plot_history(history):
    fig, axes = plt.subplots(1,2, figsize=(15,6))
    hist_data = history.history
    hist_data["epochs"] = list(range(1, len(history.history["loss"])
+1))

    hist_data = pd.DataFrame(hist_data)
    sns.lineplot(data=hist_data, x="epochs", y="loss", ax=axes[0],
color = "blue")
    sns.lineplot(data=hist_data, x="epochs", y="val_loss", ax=axes[0],
color = "orange")

    sns.lineplot(data=hist_data, x="epochs", y="accuracy", ax=axes[1],
color = "blue")
    sns.lineplot(data=hist_data, x="epochs", y="val_accuracy",
ax=axes[1], color = "orange")

plot_history(history)
```





Evaluation du modèle

```
loss, accuracy = model.evaluate(X_test_clean, y_test_clean)
print(f"Loss: {loss}")
print(f"Accuracy: {accuracy}")
```

```
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.5682 - loss:
1.0174
Loss: 0.9690523743629456
Accuracy: 0.5884615182876587
```

Pour calculer les notes des vins prédites, on utilise la fonction `argmax`. En effet, la note de qualité est donnée par l'indice (+1) de la classe dont la probabilité est la plus élevée dans le vecteur sorti par le réseau.

```
y_pred = model.predict(X_test_clean)
```

```
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
```

`y_pred` est une matrice de dimensions (nombre d'observations dans le jeu de test, nombre de classes) :

```
y_pred.shape
```

```
(1300, 10)
```

On crée un vecteur contenant les notes prédites avec la fonction `argmax`:

```
y_pred_final = np.argmax(y_pred,axis=1)+1
```

Ce vecteur peut être comparé à celui des vraies notes `y_test` avec une métrique MAE ou MSE:

```
mean_absolute_error(y_pred_final, y_test)
```

```
0.9961538461538462
```

Ce modèle de classification est donc moins performant pour prédire les notes de vin par rapport à celui réalisant une régression sur les notes (MAE ~= 0.16).

Création du 2e modèle de classification

On fusionne d'abord les classes en 3 classes : "<6", "6" et ">6".

```
def merge_classes(y):
  return np.column_stack((np.where(y.values < 6, 1, 0) ,
np.where(y.values == 6, 1, 0) , np.where(y.values > 6, 1, 0)))

y_train_clean = merge_classes(y_train)
y_val_clean = merge_classes(y_val)
y_test_clean = merge_classes(y_test)

print(y_train_clean.shape)
print(y_val_clean.shape)
print(y_test_clean.shape)
```

```
(3897, 3)
(1300, 3)
(1300, 3)
```

```
input_dim = X_train_clean.shape[1]
output_dim = y_train_clean.shape[1]

model_2 = create_model()
```

```
Model: "sequential_1"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                          ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
━━━━━━━━━━━━━━━━━┩
│ dense_5 (Dense)                       │ (None, 20)                  │
260 │
├───────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ dense_6 (Dense)                       │ (None, 30)                  │
630 │
├───────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ dense_7 (Dense)                       │ (None, 40)                  │
1,240 │
```

```
├──────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ dense_8 (Dense)                      │ (None, 20)                   │
820 │
├──────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ dense_9 (Dense)                      │ (None, 3)                    │
63 │
└──────────────────────────────────────┴──────────────────────────────┴
─────────────────┘
```

```
 Total params: 3,013 (11.77 KB)

 Trainable params: 3,013 (11.77 KB)

 Non-trainable params: 0 (0.00 B)
```

```
nb_epochs = 100

history = model_2.fit(X_train_clean,
          y_train_clean,
          epochs = nb_epochs,
          validation_data= (X_val_clean, y_val_clean),
          verbose=2
)
```

```
Epoch 1/100
122/122 - 2s - 18ms/step - accuracy: 0.4963 - loss: 1.0019 -
val_accuracy: 0.5315 - val_loss: 0.9286
Epoch 2/100
122/122 - 1s - 5ms/step - accuracy: 0.5566 - loss: 0.8889 -
val_accuracy: 0.5485 - val_loss: 0.9144
Epoch 3/100
122/122 - 0s - 4ms/step - accuracy: 0.5658 - loss: 0.8720 -
val_accuracy: 0.5554 - val_loss: 0.8949
Epoch 4/100
122/122 - 1s - 5ms/step - accuracy: 0.5666 - loss: 0.8638 -
val_accuracy: 0.5623 - val_loss: 0.8950
Epoch 5/100
122/122 - 1s - 5ms/step - accuracy: 0.5663 - loss: 0.8571 -
val_accuracy: 0.5623 - val_loss: 0.8968
Epoch 6/100
122/122 - 1s - 5ms/step - accuracy: 0.5763 - loss: 0.8604 -
val_accuracy: 0.5646 - val_loss: 0.8866
Epoch 7/100
122/122 - 0s - 4ms/step - accuracy: 0.5774 - loss: 0.8556 -
val_accuracy: 0.5562 - val_loss: 0.8829
Epoch 8/100
122/122 - 0s - 3ms/step - accuracy: 0.5735 - loss: 0.8526 -
val_accuracy: 0.5654 - val_loss: 0.8803
Epoch 9/100
```

```
122/122 - 0s - 2ms/step - accuracy: 0.5802 - loss: 0.8501 -
val_accuracy: 0.5446 - val_loss: 0.8903
Epoch 10/100
122/122 - 0s - 2ms/step - accuracy: 0.5740 - loss: 0.8525 -
val_accuracy: 0.5654 - val_loss: 0.8886
Epoch 11/100
122/122 - 0s - 2ms/step - accuracy: 0.5856 - loss: 0.8482 -
val_accuracy: 0.5700 - val_loss: 0.8788
Epoch 12/100
122/122 - 0s - 3ms/step - accuracy: 0.5843 - loss: 0.8450 -
val_accuracy: 0.5585 - val_loss: 0.9018
Epoch 13/100
122/122 - 0s - 2ms/step - accuracy: 0.5815 - loss: 0.8477 -
val_accuracy: 0.5700 - val_loss: 0.8731
Epoch 14/100
122/122 - 0s - 2ms/step - accuracy: 0.5804 - loss: 0.8436 -
val_accuracy: 0.5708 - val_loss: 0.8724
Epoch 15/100
122/122 - 0s - 2ms/step - accuracy: 0.5846 - loss: 0.8453 -
val_accuracy: 0.5600 - val_loss: 0.8709
Epoch 16/100
122/122 - 0s - 2ms/step - accuracy: 0.5871 - loss: 0.8465 -
val_accuracy: 0.5446 - val_loss: 0.8823
Epoch 17/100
122/122 - 0s - 2ms/step - accuracy: 0.5902 - loss: 0.8417 -
val_accuracy: 0.5731 - val_loss: 0.8690
Epoch 18/100
122/122 - 0s - 3ms/step - accuracy: 0.5971 - loss: 0.8366 -
val_accuracy: 0.5654 - val_loss: 0.8720
Epoch 19/100
122/122 - 0s - 2ms/step - accuracy: 0.5992 - loss: 0.8333 -
val_accuracy: 0.5754 - val_loss: 0.8655
Epoch 20/100
122/122 - 0s - 2ms/step - accuracy: 0.5912 - loss: 0.8328 -
val_accuracy: 0.5615 - val_loss: 0.8652
Epoch 21/100
122/122 - 0s - 2ms/step - accuracy: 0.5935 - loss: 0.8335 -
val_accuracy: 0.5608 - val_loss: 0.8632
Epoch 22/100
122/122 - 0s - 3ms/step - accuracy: 0.5994 - loss: 0.8292 -
val_accuracy: 0.5685 - val_loss: 0.8614
Epoch 23/100
122/122 - 1s - 5ms/step - accuracy: 0.5979 - loss: 0.8263 -
val_accuracy: 0.5662 - val_loss: 0.8652
Epoch 24/100
122/122 - 0s - 3ms/step - accuracy: 0.5958 - loss: 0.8300 -
val_accuracy: 0.5692 - val_loss: 0.8625
Epoch 25/100
122/122 - 1s - 4ms/step - accuracy: 0.5966 - loss: 0.8264 -
```

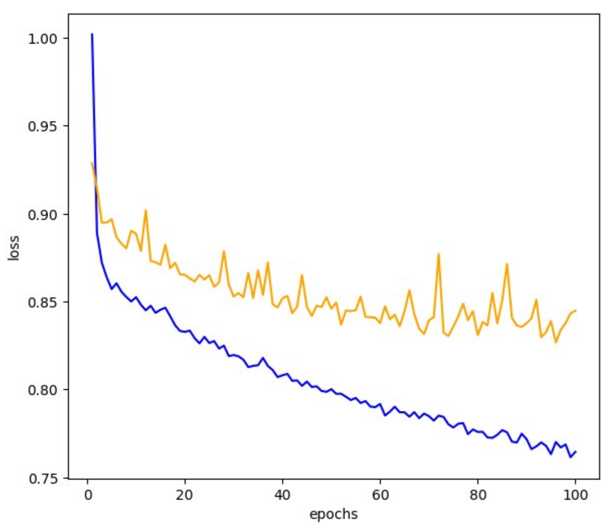
```
val_accuracy: 0.5600 - val_loss: 0.8650
Epoch 26/100
122/122 - 0s - 2ms/step - accuracy: 0.5935 - loss: 0.8276 -
val_accuracy: 0.5677 - val_loss: 0.8584
Epoch 27/100
122/122 - 0s - 3ms/step - accuracy: 0.5976 - loss: 0.8233 -
val_accuracy: 0.5662 - val_loss: 0.8609
Epoch 28/100
122/122 - 0s - 2ms/step - accuracy: 0.5974 - loss: 0.8250 -
val_accuracy: 0.5577 - val_loss: 0.8786
Epoch 29/100
122/122 - 0s - 2ms/step - accuracy: 0.5976 - loss: 0.8190 -
val_accuracy: 0.5692 - val_loss: 0.8596
Epoch 30/100
122/122 - 0s - 2ms/step - accuracy: 0.5956 - loss: 0.8196 -
val_accuracy: 0.5800 - val_loss: 0.8529
Epoch 31/100
122/122 - 0s - 2ms/step - accuracy: 0.6066 - loss: 0.8189 -
val_accuracy: 0.5785 - val_loss: 0.8548
Epoch 32/100
122/122 - 0s - 2ms/step - accuracy: 0.6007 - loss: 0.8169 -
val_accuracy: 0.5777 - val_loss: 0.8525
Epoch 33/100
122/122 - 0s - 2ms/step - accuracy: 0.6074 - loss: 0.8128 -
val_accuracy: 0.5615 - val_loss: 0.8663
Epoch 34/100
122/122 - 0s - 2ms/step - accuracy: 0.6112 - loss: 0.8135 -
val_accuracy: 0.5700 - val_loss: 0.8520
Epoch 35/100
122/122 - 0s - 2ms/step - accuracy: 0.6025 - loss: 0.8138 -
val_accuracy: 0.5669 - val_loss: 0.8678
Epoch 36/100
122/122 - 0s - 2ms/step - accuracy: 0.6033 - loss: 0.8180 -
val_accuracy: 0.5692 - val_loss: 0.8538
Epoch 37/100
122/122 - 0s - 4ms/step - accuracy: 0.6043 - loss: 0.8134 -
val_accuracy: 0.5700 - val_loss: 0.8723
Epoch 38/100
122/122 - 1s - 5ms/step - accuracy: 0.6074 - loss: 0.8111 -
val_accuracy: 0.5769 - val_loss: 0.8484
Epoch 39/100
122/122 - 1s - 5ms/step - accuracy: 0.6064 - loss: 0.8071 -
val_accuracy: 0.5754 - val_loss: 0.8467
Epoch 40/100
122/122 - 1s - 5ms/step - accuracy: 0.6015 - loss: 0.8081 -
val_accuracy: 0.5769 - val_loss: 0.8519
Epoch 41/100
122/122 - 1s - 5ms/step - accuracy: 0.6017 - loss: 0.8089 -
val_accuracy: 0.5708 - val_loss: 0.8533
```

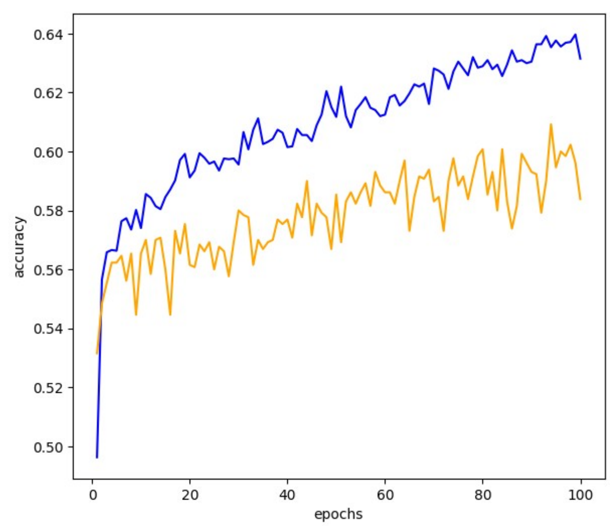
```
Epoch 42/100
122/122 - 0s - 2ms/step - accuracy: 0.6076 - loss: 0.8049 -
val_accuracy: 0.5823 - val_loss: 0.8434
Epoch 43/100
122/122 - 0s - 2ms/step - accuracy: 0.6056 - loss: 0.8052 -
val_accuracy: 0.5777 - val_loss: 0.8471
Epoch 44/100
122/122 - 1s - 5ms/step - accuracy: 0.6056 - loss: 0.8021 -
val_accuracy: 0.5900 - val_loss: 0.8650
Epoch 45/100
122/122 - 0s - 3ms/step - accuracy: 0.6035 - loss: 0.8045 -
val_accuracy: 0.5715 - val_loss: 0.8470
Epoch 46/100
122/122 - 0s - 2ms/step - accuracy: 0.6089 - loss: 0.8015 -
val_accuracy: 0.5823 - val_loss: 0.8418
Epoch 47/100
122/122 - 0s - 3ms/step - accuracy: 0.6125 - loss: 0.8018 -
val_accuracy: 0.5792 - val_loss: 0.8477
Epoch 48/100
122/122 - 0s - 2ms/step - accuracy: 0.6205 - loss: 0.7992 -
val_accuracy: 0.5777 - val_loss: 0.8469
Epoch 49/100
122/122 - 0s - 2ms/step - accuracy: 0.6151 - loss: 0.7987 -
val_accuracy: 0.5669 - val_loss: 0.8524
Epoch 50/100
122/122 - 0s - 2ms/step - accuracy: 0.6118 - loss: 0.8002 -
val_accuracy: 0.5854 - val_loss: 0.8460
Epoch 51/100
122/122 - 0s - 2ms/step - accuracy: 0.6220 - loss: 0.7976 -
val_accuracy: 0.5692 - val_loss: 0.8495
Epoch 52/100
122/122 - 0s - 2ms/step - accuracy: 0.6120 - loss: 0.7976 -
val_accuracy: 0.5831 - val_loss: 0.8369
Epoch 53/100
122/122 - 0s - 2ms/step - accuracy: 0.6082 - loss: 0.7959 -
val_accuracy: 0.5862 - val_loss: 0.8449
Epoch 54/100
122/122 - 0s - 2ms/step - accuracy: 0.6141 - loss: 0.7940 -
val_accuracy: 0.5823 - val_loss: 0.8446
Epoch 55/100
122/122 - 0s - 3ms/step - accuracy: 0.6161 - loss: 0.7952 -
val_accuracy: 0.5862 - val_loss: 0.8451
Epoch 56/100
122/122 - 1s - 5ms/step - accuracy: 0.6184 - loss: 0.7923 -
val_accuracy: 0.5892 - val_loss: 0.8528
Epoch 57/100
122/122 - 0s - 2ms/step - accuracy: 0.6148 - loss: 0.7935 -
val_accuracy: 0.5815 - val_loss: 0.8413
Epoch 58/100
```

```
122/122 - 0s - 2ms/step - accuracy: 0.6141 - loss: 0.7903 -
val_accuracy: 0.5931 - val_loss: 0.8411
Epoch 59/100
122/122 - 0s - 2ms/step - accuracy: 0.6120 - loss: 0.7900 -
val_accuracy: 0.5885 - val_loss: 0.8408
Epoch 60/100
122/122 - 0s - 2ms/step - accuracy: 0.6125 - loss: 0.7918 -
val_accuracy: 0.5862 - val_loss: 0.8377
Epoch 61/100
122/122 - 0s - 2ms/step - accuracy: 0.6184 - loss: 0.7852 -
val_accuracy: 0.5862 - val_loss: 0.8473
Epoch 62/100
122/122 - 0s - 2ms/step - accuracy: 0.6192 - loss: 0.7874 -
val_accuracy: 0.5823 - val_loss: 0.8399
Epoch 63/100
122/122 - 0s - 2ms/step - accuracy: 0.6156 - loss: 0.7902 -
val_accuracy: 0.5900 - val_loss: 0.8426
Epoch 64/100
122/122 - 0s - 2ms/step - accuracy: 0.6171 - loss: 0.7871 -
val_accuracy: 0.5969 - val_loss: 0.8360
Epoch 65/100
122/122 - 0s - 2ms/step - accuracy: 0.6197 - loss: 0.7871 -
val_accuracy: 0.5731 - val_loss: 0.8445
Epoch 66/100
122/122 - 0s - 2ms/step - accuracy: 0.6228 - loss: 0.7846 -
val_accuracy: 0.5846 - val_loss: 0.8565
Epoch 67/100
122/122 - 0s - 2ms/step - accuracy: 0.6220 - loss: 0.7872 -
val_accuracy: 0.5915 - val_loss: 0.8428
Epoch 68/100
122/122 - 0s - 3ms/step - accuracy: 0.6230 - loss: 0.7837 -
val_accuracy: 0.5908 - val_loss: 0.8346
Epoch 69/100
122/122 - 0s - 2ms/step - accuracy: 0.6161 - loss: 0.7863 -
val_accuracy: 0.5938 - val_loss: 0.8316
Epoch 70/100
122/122 - 0s - 2ms/step - accuracy: 0.6282 - loss: 0.7849 -
val_accuracy: 0.5831 - val_loss: 0.8393
Epoch 71/100
122/122 - 0s - 2ms/step - accuracy: 0.6274 - loss: 0.7823 -
val_accuracy: 0.5846 - val_loss: 0.8412
Epoch 72/100
122/122 - 0s - 2ms/step - accuracy: 0.6261 - loss: 0.7852 -
val_accuracy: 0.5731 - val_loss: 0.8771
Epoch 73/100
122/122 - 0s - 3ms/step - accuracy: 0.6212 - loss: 0.7844 -
val_accuracy: 0.5900 - val_loss: 0.8323
Epoch 74/100
122/122 - 1s - 6ms/step - accuracy: 0.6271 - loss: 0.7802 -
```

```
val_accuracy: 0.5977 - val_loss: 0.8305
Epoch 75/100
122/122 - 0s - 4ms/step - accuracy: 0.6305 - loss: 0.7784 -
val_accuracy: 0.5885 - val_loss: 0.8358
Epoch 76/100
122/122 - 0s - 4ms/step - accuracy: 0.6282 - loss: 0.7805 -
val_accuracy: 0.5915 - val_loss: 0.8416
Epoch 77/100
122/122 - 1s - 6ms/step - accuracy: 0.6259 - loss: 0.7809 -
val_accuracy: 0.5838 - val_loss: 0.8488
Epoch 78/100
122/122 - 0s - 4ms/step - accuracy: 0.6320 - loss: 0.7746 -
val_accuracy: 0.5915 - val_loss: 0.8394
Epoch 79/100
122/122 - 1s - 4ms/step - accuracy: 0.6284 - loss: 0.7773 -
val_accuracy: 0.5985 - val_loss: 0.8446
Epoch 80/100
122/122 - 0s - 3ms/step - accuracy: 0.6289 - loss: 0.7759 -
val_accuracy: 0.6008 - val_loss: 0.8309
Epoch 81/100
122/122 - 0s - 2ms/step - accuracy: 0.6310 - loss: 0.7760 -
val_accuracy: 0.5854 - val_loss: 0.8384
Epoch 82/100
122/122 - 0s - 2ms/step - accuracy: 0.6279 - loss: 0.7728 -
val_accuracy: 0.5931 - val_loss: 0.8365
Epoch 83/100
122/122 - 0s - 2ms/step - accuracy: 0.6295 - loss: 0.7725 -
val_accuracy: 0.5800 - val_loss: 0.8549
Epoch 84/100
122/122 - 0s - 3ms/step - accuracy: 0.6256 - loss: 0.7743 -
val_accuracy: 0.6008 - val_loss: 0.8376
Epoch 85/100
122/122 - 0s - 2ms/step - accuracy: 0.6295 - loss: 0.7769 -
val_accuracy: 0.5831 - val_loss: 0.8504
Epoch 86/100
122/122 - 0s - 2ms/step - accuracy: 0.6343 - loss: 0.7757 -
val_accuracy: 0.5738 - val_loss: 0.8713
Epoch 87/100
122/122 - 0s - 3ms/step - accuracy: 0.6305 - loss: 0.7704 -
val_accuracy: 0.5815 - val_loss: 0.8409
Epoch 88/100
122/122 - 0s - 2ms/step - accuracy: 0.6310 - loss: 0.7699 -
val_accuracy: 0.5992 - val_loss: 0.8364
Epoch 89/100
122/122 - 0s - 2ms/step - accuracy: 0.6300 - loss: 0.7749 -
val_accuracy: 0.5962 - val_loss: 0.8356
Epoch 90/100
122/122 - 0s - 2ms/step - accuracy: 0.6305 - loss: 0.7718 -
val_accuracy: 0.5931 - val_loss: 0.8377
```

```
Epoch 91/100
122/122 - 0s - 3ms/step - accuracy: 0.6364 - loss: 0.7661 -
val_accuracy: 0.5923 - val_loss: 0.8405
Epoch 92/100
122/122 - 0s - 2ms/step - accuracy: 0.6364 - loss: 0.7677 -
val_accuracy: 0.5792 - val_loss: 0.8510
Epoch 93/100
122/122 - 0s - 2ms/step - accuracy: 0.6392 - loss: 0.7699 -
val_accuracy: 0.5900 - val_loss: 0.8297
Epoch 94/100
122/122 - 0s - 2ms/step - accuracy: 0.6354 - loss: 0.7680 -
val_accuracy: 0.6092 - val_loss: 0.8327
Epoch 95/100
122/122 - 0s - 2ms/step - accuracy: 0.6377 - loss: 0.7633 -
val_accuracy: 0.5946 - val_loss: 0.8389
Epoch 96/100
122/122 - 0s - 2ms/step - accuracy: 0.6356 - loss: 0.7702 -
val_accuracy: 0.6000 - val_loss: 0.8268
Epoch 97/100
122/122 - 0s - 2ms/step - accuracy: 0.6369 - loss: 0.7671 -
val_accuracy: 0.5985 - val_loss: 0.8339
Epoch 98/100
122/122 - 0s - 2ms/step - accuracy: 0.6372 - loss: 0.7688 -
val_accuracy: 0.6023 - val_loss: 0.8378
Epoch 99/100
122/122 - 0s - 2ms/step - accuracy: 0.6397 - loss: 0.7615 -
val_accuracy: 0.5962 - val_loss: 0.8433
Epoch 100/100
122/122 - 0s - 2ms/step - accuracy: 0.6315 - loss: 0.7646 -
val_accuracy: 0.5838 - val_loss: 0.8448

plot_history(history)
```





```
loss, accuracy = model_2.evaluate(X_test_clean, y_test_clean)
print(f"Loss: {loss}")
print(f"Accuracy: {accuracy}")
```

```
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.5935 - loss: 
0.8262
Loss: 0.7969306707382202
Accuracy: 0.6176922917366028
```

```
y_pred = model_2.predict(X_test_clean)
```

```
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```

```
y_pred_final = np.argmax(y_pred,axis=1)+1
y_test_final = np.argmax(y_test,axis=1)+1
```

```
mean_absolute_error(y_pred_final, y_test_final)
```

```
0.6538461538461539
```