```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Dense, Conv2D, Input, Flatten, Dropout, MaxPooling2D
from tensorflow.keras.models import Model
import pandas as pd
from sklearn.metrics import classification_report,confusion_matrix
from tensorflow.keras.callbacks import EarlyStopping
import seaborn as sns

"""récupérer les données et partitionnement"""
dataset_fashion_mnsit = tf.keras.datasets.fashion_mnist
(X_train, y_train), (X_test, y_test) = dataset_fashion_mnsit.load_data()
"""vérifier la répartition des données sur les classes """
pd.DataFrame(y_train)[0].value_counts()
"""préparation des données"""
X_train = X_train / 255
X_test = X_test / 255
print(f"Données entrainement: {X_train.shape}, Test: {X_test.shape}")
"""Regardons une image Et son étiquette:"""
plt.imshow(X_train[0])
y_train[0]
"""ajouter 1 dimension (couleur : RVB). """
X_train = X_train.reshape(60000, 28, 28, 1)
X_test = X_test.reshape(10000, 28, 28, 1)
"""création du modèle"""
mon_cnn = tf.keras.Sequential()
# 3 couches de convolution, avec Nb filtres progressif 32, 64 puis 128
mon_cnn.add(Conv2D(filters=32, kernel_size=(3,3), input_shape=(28, 28, 1), activation='relu'))
mon_cnn.add(MaxPooling2D(pool_size=(2, 2)))
mon_cnn.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(28, 28, 1), activation='relu'))
mon_cnn.add(MaxPooling2D(pool_size=(2, 2)))
mon_cnn.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(28, 28, 1), activation='relu'))
mon_cnn.add(MaxPooling2D(pool_size=(2, 2)))
# remise à plat
mon_cnn.add(Flatten())
# Couche dense classique ANN
mon_cnn.add(Dense(512, activation='relu'))
# Couche de sortie (classes de 0 à 9)
mon_cnn.add(Dense(10, activation='softmax'))
"""déterminer meilleur nombres de epochs ==> early stop"""
early_stop = EarlyStopping(monitor='val_loss',patience=2)
"""compiler le model et afficher le résumé"""
mon_cnn.compile(optimizer='adam',
          loss='sparse_categorical_crossentropy',
          metrics=['accuracy'])
mon_cnn.summary()
"""train model"""
mon_cnn.fit(x=X_train,
         y=y_train,
         validation_data=(X_test, y_test),
         epochs=25,
         callbacks=[early_stop])
```

```
"""Evaluation du modèle  """
losses = pd.DataFrame(mon_cnn.history.history)
"""Affichage des courbes loss et accuracy  """
losses[['accuracy', 'val_accuracy']].plot()
losses[['loss', 'val_loss']].plot()

"""prédiction pour X_test """

pred=np.argmax(mon_cnn.predict( X_test ) , axis=-1)

"""Matrice de Confusion colorée """

plt.figure(figsize=(6,4))
sns.heatmap(confusion_matrix(y_test, pred),annot=True)

#Confution Matrix (textuelle) and Classification Report
print('Confusion Matrix')
cm=confusion_matrix(y_test, pred)
print(cm)
print('Classification Report')
print(classification_report(y_test, pred))
```