Pratique de la data science

March 28, 2025

1 TP 5 - Réseaux de Neurones

Librairies à installer :

```
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

Objectif: Implémenter trois types de réseaux de neurones (MLP, RNN, LSTM) pour prédire les valeurs des actions à J+1.
Même tâche que le TP4, on pourra donc comparer les performances avec les algorithmes de ML.

Tout doit être codé sous forme de fonctions. L'objectif est de pouvoir comparer l'utilisation de chaque modèle sur cette tâche.

1.1 Création du dataset pour la régression

On reprend les datasets créés pour chaque entreprise dans le TP4.

1.2 Modèles de Deep Learning

On utilise la librairie **TensorFlow**.
Pour chaque modèle, on crée une fonction qui prend en entrée l'ensemble des hyperparamètres (hidden_dims, dropout_rate, activation, optimizer, learning_rate...). L'objectif est de **comparer différentes architectures pour chaque modèle pour comprendre l'influence de ces hyperparamètres.**

1.2.1 Création des modèles

Layers MLP : `tf.keras.layers.Dense()`
Layers RNN : `tf.keras.layers.SimpleRNN()`
Layers LSTM : `tf.keras.layers.LSTM()`

Pour chaque modèle, créer la fonction :

```
def build_..._model(. . . .):
    model = tf.keras.Sequential()
    .
    .
    .
    model.add(tf.keras.layers.Dense(1)) # Neurone de sortie
    model.compile(optimizer=optimizer, loss='mean_squared_error')
    return model
```

1.2.2 Entrainement des modèles

```
def train_model(model_type, X_train, y_train, input_shape, . . .):

    if model_type == "MLP":
        . . .
    elif model_type == "RNN":
        . . .
    elif model_type == "LSTM":
        . . .
    model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, verbose=1)

    return model
```

1.2.3 Prédiction (Test du modèle)

Créer la fonction : `def predict(model, X_test, y_test, scaler, model_type):`

- Applique le modèle sur X_test.
- Inverse la standardisation(`scaler.inverse_transform(y_pred)`).
- Affiche les résultats (MAE, RMSE).
- Affiches les 10 premières valeurs prédites vs 10 premières valeurs réelles.

1.2.4 Choix des modèles et comparaison des résultats

- **Pour chaque type de modèle :** Comparer différentes architectures et conserver la plus efficace (expliquer).
- Appliquer les meilleurs modèles et **afficher les résultats** (MAE, MSE, valeurs prédites vs réelles)
- **Représenter graphiquement** les valeurs prédites vs vraies valeurs (cf TP4).
- Tableau de résultats à comparer avec les modèles de ML (Expliquer).

1.3 Améliorations possibles :

Itérer sur plusieurs jours : Pour prédiction à J+2, on prend la valeur prédite à J+1 et les 29 valeurs précédentes (on itére sur plusieurs jours).

Ajouter d'autres variables : Indicateurs techniques utilisés en TP3, variables temporelles, macroéconomiques...

Tester des modèles hybrides (Exemple : LSTM + MLP . . .)