Pratique de la data science

March 20, 2025

1 TP 4 - Régression

Librairies à installer :

```
import pandas as pd
import glob
import ta
from sklearn.preprocessing import MinMaxScaler
from xgboost import XGBRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_errorxz
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
```

Objectif: Utiliser les algorithmes de régressoin pour prédire les valeurs des actions à J+1.

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

1.1 Création du dataset pour la régression

Pour cette application, on étudiera chaque entreprise séparément.

Pour chaque entreprise:

1. Charger les rendements scrappés en TP1 et ne conserver que la colonne "Close".
2. Standardiser et séparer les données en train et test.
3. Créer les datasets de features (X) et de labels (Y). Pour prédire Y, le "close" à J+1, on prend comme features les 30 derniers "close":

```
def create_target_features(df, n=n_days):
    x = []
    y = []
    for i in range(n,df.shape[0]):
        x.append(df[i-n:i,0])
        y.append(df[i,0])
    x = np.array(x)
    y = np.array(y)
    return x,y
```

4. Retourner les datasets obtenus pour l'utilisation des algorithmes de régression.

5. **Important:** Retourner également le `scaler` qui a été fit sur le dataset d'entrainement. Il faudra inverser cette transformation pour obtenir les valeurs des rendements prédits à la bonne échelle.

1.2 Algorithmes de régression

Objectif: Créer une fonction par algorithme de régression : Xgboost, Random Forest, KNN, Régression linéaire.

1. Effectuer un gridsearch (`GridSearchCV()`) pour déterminer les meilleurs hyperparamètres.
2. Appliquer le meilleur modèle.
3. Afficher les MSE et RMSE.
4. Représenter graphiquement les valeurs prédites vs vraies valeurs :

```
fig,ax = plt.subplots(figsize=(10,6))
ax.plot(range(len(company_returns_close)), company_returns_close,
color='red', label='Valeurs reelles')
ax.plot(range(len(y_train)+30, len(y_train)+30+len(xgb_pred)), xgb_pred,
color='blue',label='XGboost Predictions')
```

(On peut représenter toutes les courbes sur le même graphique pour la comparaison.)

Extraire toutes les performances dans un tableau pour comparer les modèles

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...