Pratique de la data science

March 20, 2025

1 TP 3 - Classification

Librairies à installer :

```
import pandas as pd
import glob
import ta
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
import shap
```

Objectif: Utiliser les algorithmes de classification pour prédire une consigne ("buy", "hold", "sell") pour chaque action.

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

1.1 Préparation du dataset

Self-supervised learning: On créé la labellisation.

- On considère qu'il faut acheter l'action ("**buy**"), si le rendement en 1 mois est de plus de 5%. Il faut la vendre ("**sell**") si le rendement est de moins de -5%. Il ne faut rien faire ("**hold**") sinon.

1.1.1 Création des labels

Pour chaque entreprise:

1. Charger les rendements scrappés en TP1 et ne conserver que la colonne "Close".
2. Créer une colonne "Close_horizon" contenant la valeur de close 20 jours plus tard : `company_returns['Close_Horizon'] = company_returns['Close'].shift(-20)`
3. Créer la colonne "horizon_return" qui calcule le rendement sur ces 20 jours.
4. Créer la colonne "label" ("buy" : 2, "hold" : 1, "sell" : 0)

1.1.2 Conversion des données pour problème de classification classique

Transformer les informations séquentielles en features pour les algorithmes de classification (moyenne mobile, volatilité...) grâce à la librairie `ta` (voir documentation).

Pour chaque entreprise:

1. A partir de la colonne "Close" et de la libraire `ta`, créer les features : `['SMA_20', 'EMA_20', 'RSI_14', 'MACD', 'MACD_Signal', 'Bollinger_High', 'Bollinger_Low', 'Rolling_Volatility_20', 'ROC_10']`

On obtient un dataframe par entreprise

1.1.3 Processing des datasets obtenus pour le pipeline de classification

1. Concaténer l'ensemble des dataframes obtenus pour obtenir une large base de données.
2. Séparer les features explicative (X) et la colonne de labels (Y). $\Longrightarrow$ supprimer de X les variables `['Label', 'Close_Horizon', 'Weekly_return', 'Next Day Close']`
3. Standardiser les données de X.
4. Utiliser `train_test_split` pour créer les datasets d'entrainement et de test.

1.2 Algorithmes de classification

Objectif: Créer une fonction par algorithme de classification : Xgboost, Random Forest, KNN, Régression logistique, SVM.

Pour chaque algorithme:

1. Effectuer un gridsearch (`GridSearchCV()`) pour déterminer les meilleurs hyperparamètres.
2. Appliquer le meilleur modèle.
3. Afficher les résultats grâce à la fonction `classification_report`
4. Afficher les `shap.summary_plot` pour expliquer la prédiction :

```
explainer = shap.TreeExplainer(best_model)
shap_values = explainer.shap_values(X_test)
# Prediction "buy" :
shap.summary_plot(shap_values[:, :, 2], X_test, feature_names=X.columns)
# Prediction "Sell" :
shap.summary_plot(shap_values[:, :, 0], X_test, feature_names=X.columns)
```

Analyser les classsifcation report (Effet du déséquilibre des classes, recall, précision...)

Extraire toutes les performances dans un tableau pour comparer les modèles

1.3 Propositions d'améliorations

Ajout de nouvelles variables : Indicateurs macroéconomiques, temporels (effet de la période)...

Gestion du déséquilibre des classes : SMOTE ou weighted Loss.