Rapport du projet de Machine Learning : Prédiction de sentiment dans les tweets liés à la finance

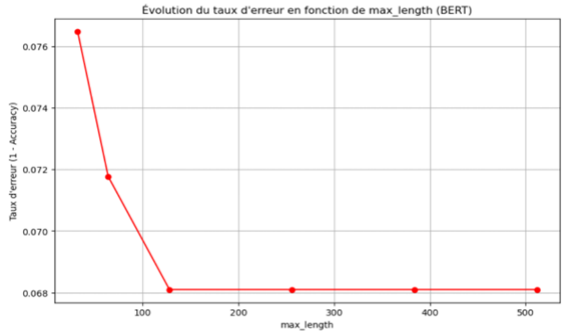
Nadège S., Danélius A., Nercy Chancelle N., Hippolyte S., Vivien G.

1. Implementation des algorithms (Negatif vs Non-négatif)

Modèle	Précision	Recall	F1-score pondéré	Temps d'exécution	Principaux hyperparamètres
Régression logistique	0,87	0,85	0,86	0,23s	`max_iter=1000`; `class_weight='balanced'`; `random_state=42`
K-Nearest Neighbors	0,84	0,85	0,81	0,46s	`n_neighbors=3`
Random Forest	0,87	0,88	0,86	74,47s (1 min 14s)	`n_estimators=500`; `random_state=42`
Réseaux de neurones	0,86	0,87	0,86	69,23s (1 min 09s)	`hidden_layer_sizes=(100, 100)`; `max_iter=500`; `activation='relu'`; `solver='adam'`; `random_state=42`
Transformers (BiDirectional Encoder Representations from Transformers); modèle pré-entraîné : 'bert-base-uncased'	0,93	0,93	0,93	2h10 min 44 s	`num_train_epochs=3`; `per_device_train_batch_size=16`; `per_device_eval_batch_size=64`; `warmup_steps=500`
BiLSTM avec Embeddings (BiDirectional Long Short-Term Memory)	0,88	0,89	0,88	181,46s (3 min 02s)	`Embedding: input_dim=5000, output_dim=128`; `LSTM: couches bidirectionnelles (64 et 32 unités)`; `Dense: 64 unités`; `activation='relu'`; `optimizer='adam'`; `epochs=5`; `batch_size=64`

2. Optimisation des modèles

2. 1 Sélection des variables exploratoires

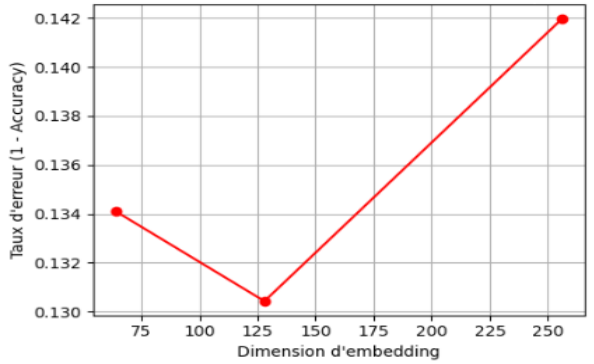


La longueur optimale des séquences de texte après tokenisation "max_length" pour le modèle BERT est 128

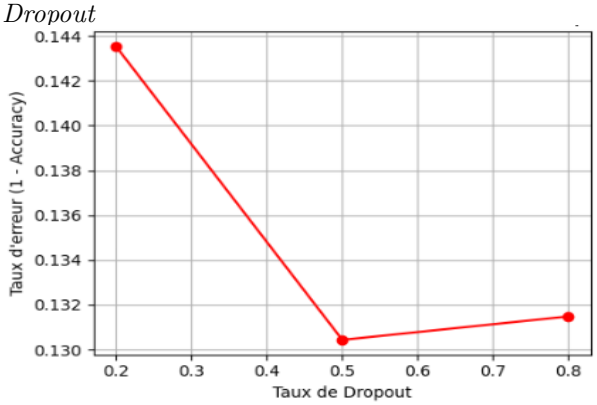
2.2 Sélection des Hyperparamètres

Modèle Bidirectional Long Short-Term Memory (BiLSTM) avec Embeddings

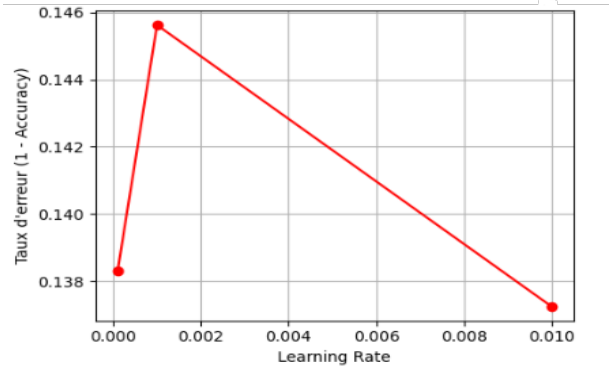
Évolution du taux d'erreur en fonction d'Embedding Dimension



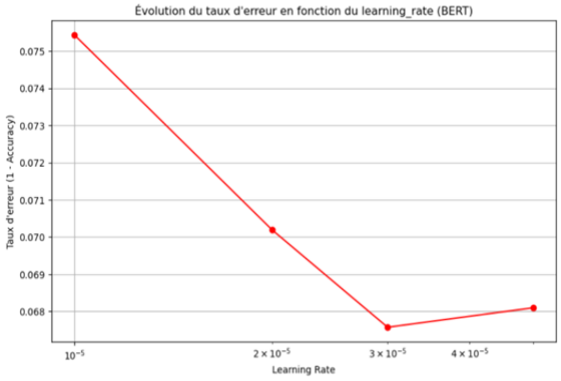
Évolution du taux d'erreur en fonction taux de Dropout



Évolution du taux d'erreur en fonction de Learning Rate



Modèle Bidirectional Encoder Representations from Transformers (BERT)

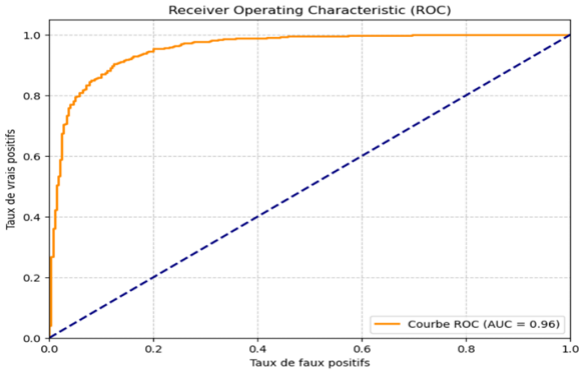


3. Sélection et Evaluation du Modèle le plus Performant

Après avoir testé plusieurs modèles, nous avons sélectionné BERT pour sa performance supérieure (**Précision** : 0.93 ; **Recall** : 0.93 ; **F1-score pondéré** : 0.93). Pré-entraîné sur de vastes quantités de données, BERT excelle à comprendre le contexte textuel. Nous avons configuré les hyperparamètres suivants pour optimiser son entraînement : trois époques pour éviter le sur-apprentissage (`num_train_epochs=3`), une taille de batch d'entraînement de 16 (`per_device_train_batch_size=16`) pour équilibrer stabilité et contraintes matérielles, et une taille de batch d'évaluation de 64 (`per_device_eval_batch_size=64`) pour réduire le temps d'inférence. Le taux d'apprentissage est fixé à 3e-5 (`learning_rate=3e-5`) pour stabiliser l'entraînement, avec 500 étapes de warmup (`warmup_steps=500`) pour augmenter progressivement ce taux. Une légère pénalisation des poids (`weight_decay=0.01`) est appliquée pour réduire le sur-apprentissage.

Les tweets, convertis en tokens via le BertTokenizer, servent de variables explicatives sans nécessiter de feature engineering manuel.

Courbe ROC



Faiblesse des Transformers

Le modèle BERT présente plusieurs limites à prendre en compte. Il est restreint à une longueur maximale de 512 tokens (128 dans ce cas), ce qui peut entraîner une perte d'information pour les tweets plus longs. De plus, un déséquilibre entre les classes peut biaiser les prédictions en faveur de la classe majoritaire. Le modèle peut également souffrir de surapprentissage si les performances diffèrent fortement entre l'entraînement et le test. Par ailleurs, BERT pré-entraîné sur des données générales peut mal interpréter le jargon financier, et reste peu performant face aux subtilités linguistiques comme l'ironie ou le sarcasme. Son coût computationnel élevé peut limiter son usage dans des environnements contraints. Enfin, les biais présents dans les données d'entraînement d'origine peuvent affecter la justesse des résultats, notamment dans des domaines spécifiques comme la finance.

4. Méthodes de classification multi-classes (Négatif, Positif, Neutre)

L'objectif ici de prédire les sentiments (négatif, neutre, positif). Nous avons utilisé le modèle logistique grâce aux données au format TF-IDF. Pour la validation croisée, nous avons utilisé le principe de Train/Validation/Test split qui consiste en une validation croisée sur le jeu d'entrainement avec jeu de test indépendant pour l'évaluation finale uniquement.
Le modèle de classification a été évalué à l'aide d'une validation croisée stratifiée sur 5 folds avec un score moyen de validation croisée de 76,88%.

Performances du modèle multi-classes avec validation croisée

Précision	**Recall**	**Accuracy**	**F1-score pondéré**	**Temps d'exécution (en sec)**
0,76	0,75	0,75	0,75	23