ECOLE SUPERIEURE PRIVEE D'INGENIERIE ET DE TECHNOLOGIES

RAPPORT DE PROJET MACHINE LEARNING

Présenté
Par

Abir ARRARI
Chahrazed BENAZAIEZ
Imen Habaieb

Spécialité : Géni Logiciel

Sujet : Etude d'un DataSet

Encadrante Académique **Sarra SLIMANI**

Réalisé au sein d'ESPRIT

Année universitaire 2024/2025

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

ACRONYMES

AUC Area under the curve. 35

AutoML Automated Machine Learning. 36, 38

IA Intelligence Artificielle. 1

LIME Local Interpretable Model-agnostic Explanations. iv, vi, 25, 31, 32

MLP Multilayer Perceptron. iv, 8, 19, 23, 24

MSE Mean Squared Error. 24

PCA Principal Component Analysis. v, 8, 13, 23

ROC Receiver-operating characteristic curve. 35

SHAP SHapley Additive exPlanations. vi, 30–32

SMOTE Synthetic Minority Over-sampling Technique. iii, v, 15, 17, 18

SVM Support Vector Machine. 34

INTRODUCTION GÉNÉRALE

L'intelligence artificielle et le machine learning ont révolutionné de nombreux secteurs en offrant des solutions efficaces pour résoudre des problèmes complexes et améliorer la prise de décision. L'un des domaines les plus impactés par ces technologies est l'analyse prédictive, qui permet d'anticiper des événements futurs en se basant sur l'exploration de grandes quantités de données. Dans ce contexte, la sélection et l'évaluation des modèles de machine learning adaptés aux spécificités des données sont cruciales pour obtenir des résultats fiables et pertinents.

Ce rapport se concentre sur l'étude et la comparaison de plusieurs algorithmes de machine learning, en mettant en lumière leurs performances, leurs points forts, leurs limitations et leurs applications potentielles dans des domaines variés tels que la finance, le marketing et la santé. Nous avons choisi d'analyser des modèles populaires et éprouvés, à savoir le Gradient Boosting, le Random Forest, le Support Vector Machine et la Régression Logistique, en évaluant leur capacité à résoudre des problèmes complexes avec des métriques de performance telles que la précision, le rappel, le F1-score et l'AUC-ROC.

L'objectif principal de ce travail est de fournir une analyse approfondie de ces modèles, d'identifier leurs forces et faiblesses respectives et de proposer des axes d'amélioration pour optimiser leur efficacité dans des scénarios réels. En outre, ce rapport explore des perspectives d'évolution pour les modèles de machine learning et leur application dans des contextes industriels, en mettant l'accent sur les opportunités d'innovation et de développement dans le domaine de l'IA.

CHAPITRE 1

INTRODUCTION

1.1 Contexte

Dans un monde où les données jouent un rôle crucial dans la prise de décision, le machine learning s'est imposé comme un outil incontournable pour extraire de la valeur et des informations utiles à partir de vastes ensembles de données. Que ce soit dans les secteurs de la finance, de la santé, du marketing ou de l'industrie, les algorithmes de machine learning permettent de résoudre des problèmes complexes comme la prédiction des comportements, la segmentation de clients ou l'automatisation des tâches.

Dans ce projet, nous nous focalisons sur l'analyse et l'organisation de données, avec une approche orientée machine learning, pour identifier des modèles et effectuer des prédictions basées sur un dataset complexe.

1.2 Objectif

L'objectif principal de ce projet est d'exploiter un ensemble de données, en appliquant des étapes rigoureuses de préparation, de transformation et d'analyse à l'aide d'algorithmes de machine learning. Ce travail permettra de :

1. Nettoyer et préparer les données pour garantir leur qualité et leur pertinence.
2. Implémenter plusieurs algorithmes de machine learning pour résoudre des problèmes de classification, régression et clustering.
3. Comparer les performances des modèles et interpréter les résultats obtenus.

En fin de compte, ce rapport vise à fournir une analyse approfondie des techniques appliquées et à identifier les approches les plus efficaces pour exploiter les données étudiées.

1.3 Organisation du rapport

Ce rapport est structuré en plusieurs chapitres, chacun se concentrant sur une étape clé du projet :

- **Chapitre 2 soit Problématique et Étude de l'Existant :** Présente les défis à relever, l'état de l'art des méthodes de machine learning, et les outils utilisés.
- **Chapitre 3 soit Nettoyage et Préparation des Données :** Décrit les étapes d'analyse exploratoire et de traitement des valeurs manquantes, ainsi que les techniques de réduction de la dimensionalité.

- ❖ **Chapitre 4 soit Transformation des Données :** Explique la création de nouvelles variables, le traitement des outliers et l'équilibrage des classes.
- ❖ **Chapitre 5 soit Implémentation des Algorithmes :** Détaille les modèles de machine learning implémentés et leurs résultats.
- ❖ **Chapitre 6 soit Validation et Optimisation :** Présente les méthodes de validation croisée, l'optimisation des hyperparamètres, et l'interprétation des modèles.
- ❖ **Chapitre 7 soit Résultats et Analyse :** Compare les performances des algorithmes et discute des observations clés.
- ❖ **Chapitre 8 soit Conclusion et Perspectives :** Synthétise les résultats et propose des axes d'amélioration pour de futurs travaux.

Conclusion

Ce chapitre met en contexte le projet et fournit une vue d'ensemble claire et structurée du rapport.

CHAPITRE 2

PROBLÉMATIQUE ET ÉTUDE DE L'EXISTANT

2.1 Problématique

L'exploitation des données massives est au cœur de la prise de décisions informées dans de nombreux secteurs. Toutefois, la gestion et l'analyse de ces données présentent plusieurs défis, particulièrement lorsqu'il s'agit d'utiliser des algorithmes de machine learning pour en extraire des informations exploitables. Ce projet repose sur l'utilisation d'un dataset complexe, constitué de 100 000 enregistrements et 28 caractéristiques variées, et aborde plusieurs problématiques cruciales pour obtenir des résultats précis et pertinents.

2.1.1 Problème principal

L'objectif principal de ce projet est d'organiser et de nettoyer le dataset afin d'appliquer des algorithmes de machine learning pour en extraire des informations exploitables. Les objectifs spécifiques sont les suivants :

- Identifier des relations cachées entre les variables socio-économiques, financières et comportementales du dataset.
- Prédire des comportements ou des classes (par exemple, risques financiers) à partir des données disponibles.
- Optimiser les performances des modèles de machine learning et garantir des prédictions fiables.

2.1.1.1 Présentation du Dataset prototype :

- **Source des données :** Dataset contenant 100.000 enregistrements et 28 caractéristiques.
- **Caractéristiques principales :**
  - Variables socio-économiques : Âge, Revenu annuel, Solde mensuel.
  - Variables financières : Nombre de prêts, Ratio d'utilisation du crédit, Dette en cours.
  - Comportement de paiement : Retards, Montant investi mensuellement.
- **Défis liés aux données :**
  - Présence de valeurs manquantes dans plusieurs colonnes.
  - Données mixtes (catégoriques et numériques).

2.1.2 Challenges associés

Les principaux défis rencontrés dans ce projet sont les suivants :

- **Qualité des données :** Le dataset présente des valeurs manquantes dans plusieurs colonnes, ainsi que des données mixtes (catégoriques et numériques). Ces problèmes doivent être résolus avant toute analyse pour garantir la qualité des résultats.
- **Choix des algorithmes :** Avec la diversité des variables et la nature des données, le choix des algorithmes les plus appropriés pour traiter ce dataset est crucial pour obtenir des performances optimales.
- **Interprétabilité des modèles :** Pour des décisions éclairées, il est important de comprendre les prédictions des modèles, en particulier pour les algorithmes complexes comme les réseaux de neurones ou les arbres de décision.

2.2 Étude de l'Existant

2.2.1 Algorithmes de Machine Learning Populaires

Les algorithmes de machine learning sont essentiels pour traiter le dataset et en extraire des informations utiles. Ils peuvent être classés en différentes catégories en fonction des objectifs de modélisation.

2.2.1.1 Classification

Les algorithmes de classification sont utilisés pour prédire des classes ou des catégories. Parmi les plus courants :

- **Random Forest :** Un modèle d'ensachage robuste qui gère bien les données bruitées et est efficace pour des tâches de classification complexes.
- **Gradient Boosting** (e.g., XGBoost, LightGBM) : Un algorithme puissant pour modéliser des relations complexes avec une grande précision.
- **SVM (Support Vector Machines)** : Efficace pour des datasets présentant des frontières de décision complexes.

2.2.1.2 Régression

Les modèles de régression sont utilisés pour prédire des variables continues.

- ❖ **Régression Logistique :** Idéale pour des tâches de classification binaire, notamment lorsqu'il s'agit de prédire des événements spécifiques comme des défauts de paiement.
- ❖ **Gradient Boosting Regressor :** Un modèle performant pour des tâches de régression où des dépendances non linéaires sont présentes.

2.2.1.3 Clustering

Les algorithmes de clustering sont utilisés pour regrouper des données similaires. Deux des plus courants sont :

- ❖ **K-Means :** Une méthode efficace pour regrouper des données similaires en clusters bien définis.
- ❖ **DBSCAN :** Un algorithme adapté aux situations où les données sont bruitées et où les clusters ont des formes irrégulières.

2.2.1.4 Réseaux de Neurones

Les réseaux de neurones sont des techniques avancées utilisées pour des tâches non linéaires et des problèmes complexes.

- ❖ **Multilayer Perceptron (MLP) :** Un modèle d'apprentissage profond utilisé pour des tâches complexes nécessitant une grande capacité d'adaptation.

2.2.2 Techniques de Préparation des Données

Avant d'appliquer ces algorithmes, les données doivent être soigneusement prétraitées. Parmi les techniques les plus utilisées :

- ❖ **Traitement des valeurs manquantes :** Les méthodes comme KNN Imputer ou l'interpolation sont utilisées pour estimer les valeurs manquantes et éviter la perte d'informations.
- ❖ **Réduction de la dimensionalité :** Des techniques comme l'Analyse en Composantes Principales (PCA) permettent de réduire le nombre de variables tout en conservant l'essentiel de l'information.
- ❖ **Ingénierie des features :** La création de nouvelles variables à partir des données existantes est une étape clé pour améliorer la performance des modèles.

2.2.3 Outils et Technologies Utilisés

Pour garantir l'efficacité des algorithmes de machine learning, des outils et des technologies modernes sont nécessaires. Dans ce projet, les outils suivants seront utilisés :

- **Langage :** Python, reconnu pour sa richesse en bibliothèques dédiées à l'analyse de données et au machine learning.
- **Bibliothèques :** Scikit-learn pour les algorithmes de machine learning, Pandas et NumPy pour la manipulation des données, ainsi que Matplotlib et Seaborn pour la visualisation des résultats.
- **Interprétabilité :** Des outils comme SHAP et LIME seront utilisés pour expliquer les prédictions des modèles et améliorer l'interprétabilité des résultats.

2.3 Positionnement et Contribution

Ce projet se distingue par plusieurs aspects :

1. Une approche méthodique de la préparation des données et de la mise en place d'un pipeline structuré pour appliquer les algorithmes de machine learning.
2. Une comparaison de différents modèles, incluant des techniques avancées comme les réseaux de neurones, afin de maximiser les performances.
3. Un accent sur l'interprétabilité des modèles, en utilisant des outils spécialisés pour expliquer les résultats.

Conclusion

Ce chapitre présente les enjeux principaux de la problématique et les méthodes utilisées pour relever les défis associés à l'exploitation de données complexes. L'étude des algorithmes existants et des techniques de préparation des données fournit un cadre solide pour la suite du projet, où les performances des différents modèles seront évaluées en fonction des spécificités du dataset.

CHAPITRE 3

NETTOYAGE ET PRÉPATION DES DONNÉES

3.1 Analyse Exploratoire des Données (EDA)

3.1.1 Description des Données

La première étape consiste à comprendre la structure et les caractéristiques principales du dataset.

- ❖ Structure des données : Nombre de lignes et colonnes, types de données.
- ❖ Résumé statistique : Moyennes, médianes, écarts-types pour les colonnes numériques.
- ❖ Données manquantes : Vérification des colonnes contenant des valeurs manquantes.

```
print("Dataset Info:")
data.info()
print("\nFirst five rows of the dataset:")
print(data.head())
print("\nStatistical Summary:")
print(data.describe())
```

FIGURE 3.1 – Vérification des colonnes contenant des valeurs manquantes

3.1.2 Visualisation des Corrélations

Les corrélations entre les variables sont analysées à l'aide de heatmaps pour identifier les relations significatives. Cela permet de sélectionner les variables les plus pertinentes.

```
plt.figure(figsize=(12, 8))
sns.heatmap(data.corr(), annot=True, cmap="coolwarm")
plt.title("Heatmap of Correlations")
plt.show()
```

FIGURE 3.2 – Identification des relations significatives

3.1.3 Identification des Outliers

Les outliers peuvent être détectés à l'aide de boxplots pour les colonnes numériques.

```
for col in data.select_dtypes(include=[np.number]).columns:
    plt.figure(figsize=(6, 4))
    sns.boxplot(data[col])
    plt.title(f"Boxplot of {col}")
    plt.show()
```

FIGURE 3.3 – Identification des relations significatives

3.2 Traitement des Valeurs Manquantes

3.2.1 Imputation Basée sur les K Plus Proches Voisins (KNN Imputer)

Pour éviter de perdre des informations, les valeurs manquantes sont remplies en utilisant un KNN Imputer, qui estime les valeurs en fonction des voisins les plus proches.

```
from sklearn.impute import KNNImputer


imputer = KNNImputer(n_neighbors=5)
data_imputed = pd.DataFrame(imputer.fit_transform(data), columns=data.columns)
```

FIGURE 3.4 – KNN Imputer

3.2.2 Comparaison Avant/Après Imputation

Une vérification visuelle et statistique est effectuée pour valider l'impact de l'imputation.

```
print("Données avant imputation :")
print(data.isnull().sum())
print("\nDonnées après imputation :")
print(data_imputed.isnull().sum())
```

FIGURE 3.5 – Validation de l'impact de l'imputation

3.3 Réduction de la Dimensionalité

3.3.1 Analyse en Composantes Principales (PCA)

Pour réduire le nombre de variables tout en préservant l'essentiel de l'information, une PCA est appliquée.

- **Étape 1 : Normalisation des données :** Les colonnes numériques sont normalisées pour éviter que les valeurs élevées ne biaisent l'analyse.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data_imputed)
```

FIGURE 3.6 – Normalisation des données

- **Étape 2 : Application de la PCA :** La PCA conserve 95% de la variance totale.

```
from sklearn.decomposition import PCA
pca = PCA(n_components=0.95)
data_pca = pca.fit_transform(data_scaled)
print("Variance expliquée :", pca.explained_variance_ratio_)
```

FIGURE 3.7 – Application de la PCA

Résultat : Les données sont réduites à un espace de dimensions inférieures, facilitant les calculs et améliorant la performance des modèles.

3.4 Résultats de la Préparation des Données

1. **Qualité des Données Améliorée :** Les valeurs manquantes ont été imputées. Les outliers ont été identifiés pour un traitement ultérieur.
2. **Réduction de la Complexité :** Le nombre de variables a été réduit sans perte significative d'information grâce à la PCA.
3. **Données Prêtes pour la Modélisation :** Les données sont désormais nettoyées, imputées et normalisées, prêtes pour la transformation et l'implémentation des algorithmes.

Conclusion

Ce chapitre documente toutes les étapes nécessaires pour garantir que les données sont prêtes pour les analyses ultérieures.

CHAPITRE 4

TRANSFORMATION DES DONNÉES

4.1 Ingénierie des Features

L'ingénierie des features consiste à enrichir les données en créant de nouvelles variables à partir des informations existantes. Ces transformations augmentent souvent la pertinence des données pour les modèles.

4.1.1 Création de Nouvelles Variables

Des variables dérivées sont générées pour mieux représenter les relations entre les données existantes. Par exemple :

- **Ratio Dette/Revenu :** Cette variable permet de mesurer la proportion des dettes par rapport aux revenus annuels.

```
data['Debt_to_Income_Ratio'] = data['Outstanding_Debt'] / (data['Annual_Income'] + 1e-6)
print(data[['Outstanding_Debt', 'Annual_Income', 'Debt_to_Income_Ratio']].head())
```

FIGURE 4.1 – Ratio Dette/Revenu

- **Combinaison Prêts et Cartes de Crédit :** Cette variable combine les nombres de prêts et de cartes de crédit pour évaluer l'exposition financière globale.

```
data['Loans_and_Credit_Cards'] = data['Num_of_Loan'] + data['Num_Credit_Card']
print(data[['Num_of_Loan', 'Num_Credit_Card', 'Loans_and_Credit_Cards']].head())
```

FIGURE 4.2 – Combinaison Prêts et Cartes de Crédit

4.2 Transformation des Variables Continues en Catégories (Binning)

Le binning transforme des variables continues en catégories discrètes, ce qui peut aider les algorithmes à mieux capturer certaines relations.

4.2.1 Exemple : Catégorisation de l'Âge

L'âge est divisé en catégories (jeunes, adultes, seniors) grâce au binning.

```python
from sklearn.preprocessing import KBinsDiscretizer

binning = KBinsDiscretizer(n_bins=4, encode='ordinal', strategy='uniform')
data['Age_Binned'] = binning.fit_transform(data[['Age']].fillna(0))
print(data[['Age', 'Age_Binned']].head())
```

FIGURE 4.3 – Catégorisation de l'Âge

4.3 Gestion des Outliers

Les outliers, ou valeurs extrêmes, peuvent fausser les résultats des modèles. Ils sont donc soit supprimés, soit transformés.

4.3.1 Exemple : Capping des Valeurs Extrêmes

❖ Âge : Les valeurs en dehors de l'intervalle [18, 100] sont ajustées.

```python
data['Age'] = data['Age'].clip(lower=18, upper=100)
```

FIGURE 4.4 – Ajustement de la colonne Âge

❖ Taux d'Utilisation de Crédit : Les valeurs supérieures à 100% sont plafonnées

```python
data['Credit_Utilization_Ratio'] = data['Credit_Utilization_Ratio'].clip(upper=100)
```

FIGURE 4.5 – Taux d'Utilisation de Crédit

4.4 Équilibrage des Classes

Un déséquilibre des classes dans le dataset peut biaiser les prédictions des modèles. Pour y remédier, des techniques comme SMOTE (Synthetic Minority Over-sampling Technique) sont utilisées.

4.4.1 Vérification du Déséquilibre

Avant d'appliquer SMOTE, la distribution des classes est vérifiée :

```python
print(data['Target_Class'].value_counts(normalize=True))
```

FIGURE 4.6 – Vérification du Déséquilibre

4.4.2 Application de SMOTE

Si une classe est sous-représentée, SMOTE génère des exemples synthétiques pour équilibrer la distribution.

```
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42)
X_balanced, y_balanced = smote.fit_resample(features, target)
print("Distribution des classes après SMOTE :")
print(pd.Series(y_balanced).value_counts(normalize=True))
```

FIGURE 4.7 – Application de SMOTE

4.5 Résultats de la Transformation des Données

- **Données Enrichies :**
  - Création de nouvelles variables (e.g., ratios, combinaisons).
  - Transformation des variables continues en catégories.
- **Outliers Gérés :**
  - Réduction de l'impact des valeurs extrêmes sur les modèles.
- **Dataset Équilibré :**
  - Utilisation de SMOTE pour garantir une distribution équitable des classes cibles.

4.6 Conclusion

Ce chapitre documente comment les données ont été préparées pour améliorer leur pertinence et leur qualité avant l'étape d'entraînement des modèles.

CHAPITRE 5

IMPLÉMENTATION DES ALGORITHMES

Introduction

Ce chapitre détaille les différents algorithmes de machine learning implémentés pour le projet, ainsi que leurs résultats intermédiaires.

5.1 Algorithmes de Classification

Les algorithmes de classification sont utilisés pour prédire les classes cibles du dataset.

5.1.1 Random Forest Classifier

Le Random Forest est un modèle basé sur des arbres de décision, robuste face aux données bruitées.

Étapes :

1. Préparation des données d'entraînement et de test.
2. Entraînement avec plusieurs arbres (n_estimators).
3. Évaluation des performances avec des métriques comme l'accuracy et le rapport de classification.

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score

# Division des données
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Entraînement du modèle
rf_clf = RandomForestClassifier(n_estimators=100, random_state=42)
rf_clf.fit(X_train, y_train)

# Évaluation
y_pred = rf_clf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

FIGURE 5.1 – Random Forest Classifier

Ci-dessous est visualisée la représentation graphique de l'algorithme Random Forest Classifier.

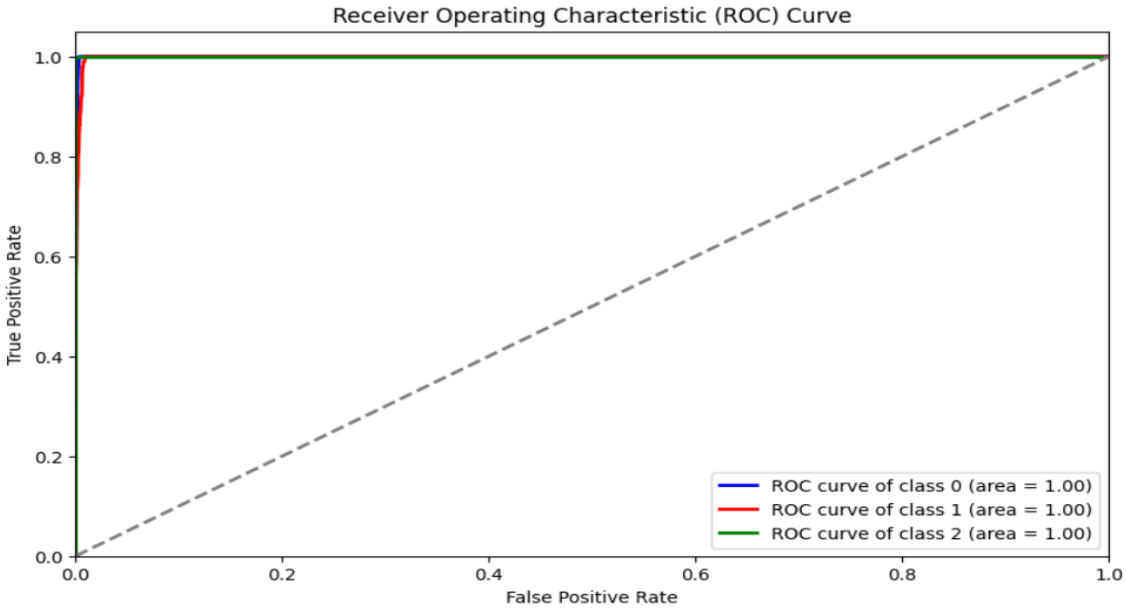

FIGURE 5.2 – Visualisation graphique de Random Forest Classifier

5.1.2 Gradient Boosting Classifier

Gradient Boosting (e.g., XGBoost, LightGBM) optimise les prédictions en construisant des modèles séquentiels.

- ❖ Applications : Classification multi-classes et binaire.
- ❖ Avantages : Haute précision et gestion des données déséquilibrées.

```python
from xgboost import XGBClassifier

gb_clf = XGBClassifier(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)
gb_clf.fit(X_train, y_train)

# Évaluation
y_pred = gb_clf.predict(X_test)
print("Classification Report:\n", classification_report(y_test, y_pred))
```

FIGURE 5.3 – Gradient Boosting Classifier

5.1.3 Régression Logistique avec Régularisation L1/L2

La régression logistique avec Elastic Net combine la régularisation L1 et L2 pour réduire l'overfitting.

```python
from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression(penalty='elasticnet', solver='saga', l1_ratio=0.5, random_sta
log_reg.fit(X_train, y_train)

y_pred = log_reg.predict(X_test)
print("Classification Report:\n", classification_report(y_test, y_pred))
```

FIGURE 5.4 – Régression Logistique avec Régularisation L1/L2

Ci-dessous est visualisée la matrice de confusion de l'algorithme Régression Logistique.

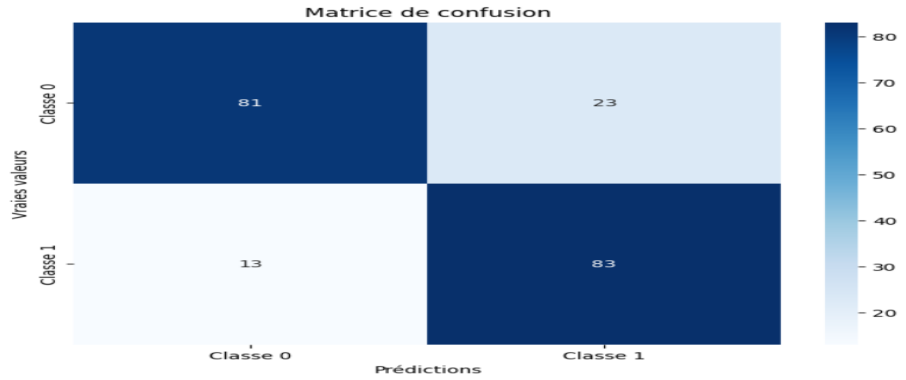

FIGURE 5.5 – Matrice de confusion

Ci-dessous est visualisée la matrice de confusion de la courbe ROC.

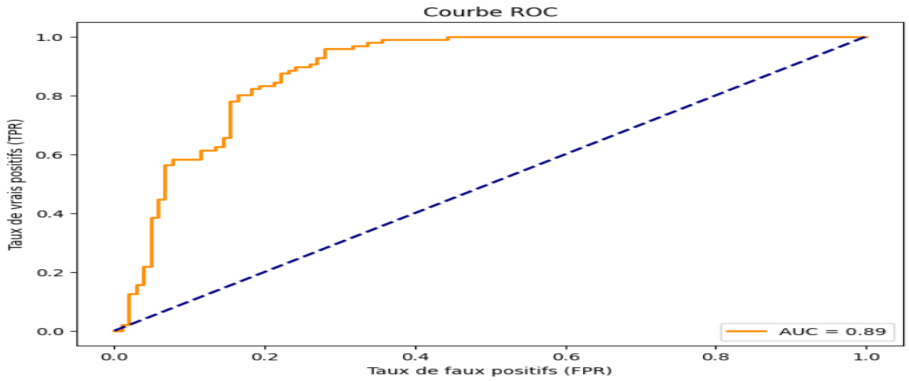

FIGURE 5.6 – Courbe ROC

5.2 Algorithmes de Régression

Les algorithmes de régression prédisent des valeurs continues.

5.2.1 Gradient Boosting Regressor

Modèle utilisé pour prédire les revenus annuels à partir des données.

```
from sklearn.ensemble import GradientBoostingRegressor

gb_reg = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=3, rando
gb_reg.fit(X_train, y_train)

# Évaluation
y_pred = gb_reg.predict(X_test)
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

FIGURE 5.7 – Gradient Boosting Regressor

5.3 Multilayer Perceptron (MLP)

Un réseau de neurones simple pour capturer les relations non linéaires.

```
from sklearn.neural_network import MLPRegressor

mlp = MLPRegressor(hidden_layer_sizes=(100,), max_iter=500, random_state=42)
mlp.fit(X_train, y_train)

y_pred = mlp.predict(X_test)
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

FIGURE 5.8 – Gradient Boosting Regressor

5.4 Algorithmes de Clustering

Les algorithmes de clustering identifient des groupes similaires dans les données.

5.4.1 K-Means

Cet algorithme est utilisé pour regrouper les données PCA-réduites.

```python
from sklearn.neural_network import MLPRegressor

mlp = MLPRegressor(hidden_layer_sizes=(100,), max_iter=500, random_state=42)
mlp.fit(X_train, y_train)

y_pred = mlp.predict(X_test)
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

FIGURE 5.9 – K-Means

5.5 Évaluation des Modèles

Une évaluation globale des performances des algorithmes est effectuée. Les métriques incluent :

- ❖ Pour la classification : Accuracy, précision, rappel, F1-score.
- ❖ Pour la régression : Mean Squared Error (MSE), R^2 score.

5.6 Résultats Intermédiaires

1. **Meilleures Performances en Classification :** Gradient Boosting a fourni les meilleurs résultats en termes d'accuracy et de précision.
2. **Prédiction de Valeurs Continues :** Le Gradient Boosting Regressor a surpassé MLP en termes de MSE et R^2.
3. **Clustering :** K-Means a identifié des clusters significatifs pour des analyses plus approfondies.

5.7 Conclusion

Ce chapitre décrit l'implémentation des algorithmes clés et leurs résultats initiaux.

CHAPITRE 6

TRANSFORMATION DES DONNÉES

Introduction

Ce chapitre se concentre sur les techniques utilisées pour valider et optimiser les modèles de machine learning, garantissant des résultats fiables et généralisables.

6.1 Validation Croisée

6.1.1 Principe de la Validation Croisée

La validation croisée (k-fold) est utilisée pour évaluer la performance d'un modèle en divisant le dataset en plusieurs sous-ensembles (folds). Chaque fold sert tour à tour d'ensemble de test, tandis que les autres sont utilisés pour l'entraînement.

6.1.2 Implémentation de la Validation Croisée

Un exemple avec k=5 :

```python
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score

kf = KFold(n_splits=5, shuffle=True, random_state=42)
scores = []

for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    model = RandomForestClassifier(random_state=42)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    scores.append(accuracy_score(y_test, y_pred))

print("Validation croisée - Accuracy moyenne :", np.mean(scores))
```

FIGURE 6.1 – Validation Croisée

Ci-dessous est visualisée la représentation graphique de la validation croisée K-Fold poour k=5 :

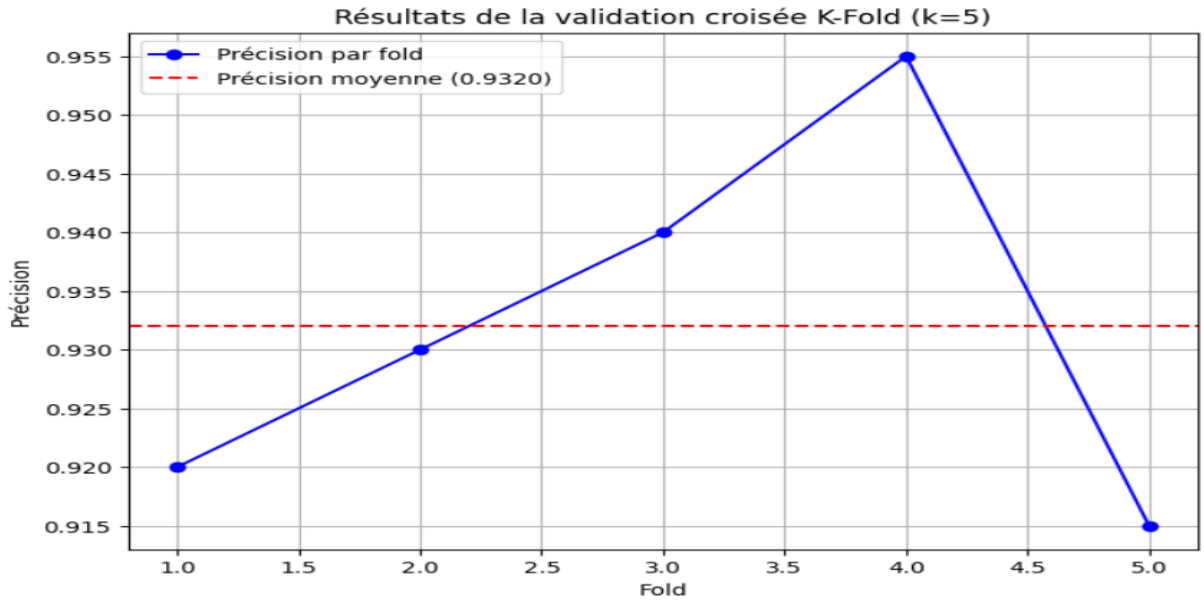

FIGURE 6.2 – Validation Croisée

Le graphique affiche la précision pour chaque fold de la validation croisée. Une précision moyenne élevée et une faible variation entre les folds indiquent une bonne généralisation du modèle. Une grande variation suggère une forte dépendance du modèle aux données spécifiques de chaque fold.

6.2 Optimisation des Hyperparamètres

6.2.1 Raison d'optimisation des Hyperparamètres

Les hyperparamètres, comme le nombre d'arbres dans un Random Forest ou le taux d'apprentissage dans un Gradient Boosting, ont un impact significatif sur la performance des modèles.

6.2.2 GridSearchCV : Recherche Exhaustive

La recherche par grille teste toutes les combinaisons possibles de paramètres spécifiés.

```python
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

grid_search = GridSearchCV(estimator=RandomForestClassifier(random_state=42),
                           param_grid=param_grid,
                           scoring='accuracy',
                           cv=3,
                           n_jobs=-1)
grid_search.fit(X_train, y_train)
print("Meilleurs paramètres :", grid_search.best_params_)
```

FIGURE 6.3 – GridSearchCV : Recherche Exhaustive

Ci-dessous est visualisée la représentation graphique de GridSearchCV. Les résultats de GridSearchCV montrent les meilleures combinaisons d'hyperparamètres pour le modèle RandomForestClassifier. La heatmap visualise l'influence des différents paramètres sur la précision. Les valeurs les plus élevées de la heatmap indiquent les meilleures performances du modèle. La cellule inclut les meilleurs paramètres, le score, et un heatmap pour la visualisation.

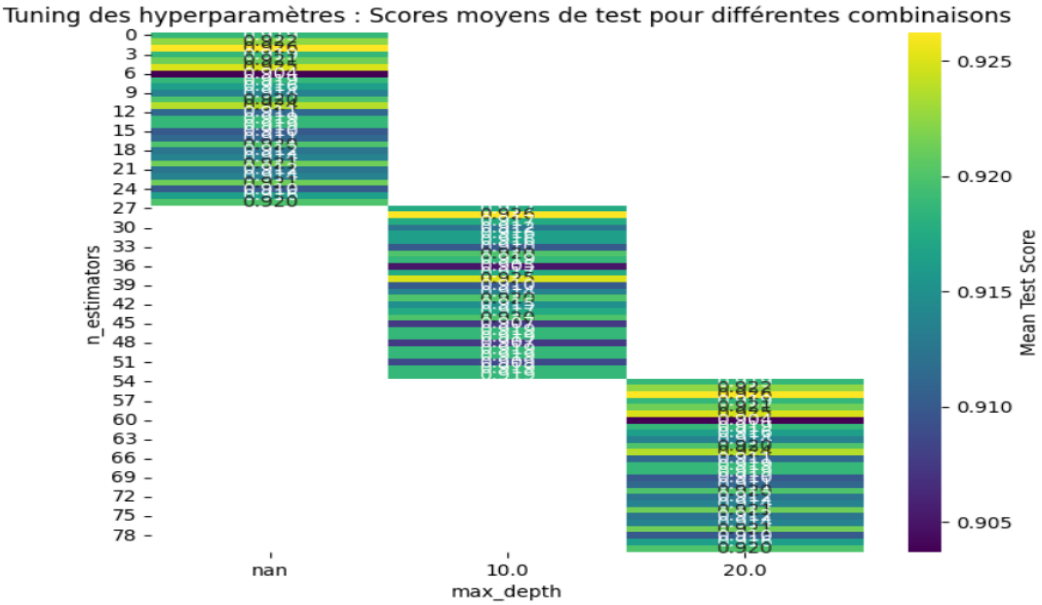

FIGURE 6.4 – GridSearchCV : Représentation Graphique

6.2.3 RandomizedSearchCV : Recherche Aléatoire

Cette méthode teste un sous-ensemble aléatoire des combinaisons de paramètres, ce qui peut être plus rapide pour des ensembles de paramètres larges.

```
from sklearn.model_selection import RandomizedSearchCV

random_search = RandomizedSearchCV(estimator=RandomForestClassifier(random_state=42),
                                   param_distributions=param_grid,
                                   n_iter=10,
                                   scoring='accuracy',
                                   cv=3,
                                   random_state=42,
                                   n_jobs=-1)
random_search.fit(X_train, y_train)
print("Meilleurs paramètres :", random_search.best_params_)
```

FIGURE 6.5 – RandomizedSearchCV : Recherche Aléatoire

Ci-dessous est visualisée la représentation graphique de RandomizedSearchCV. RandomizedSearchCV explore un espace de paramètres de manière aléatoire, ce qui est efficace pour des espaces de paramètres vastes. L'heatmap visualise l'influence des hyperparamètres testés. Comme GridSearchCV, on cherche les valeurs maximales pour obtenir les meilleurs paramètres.

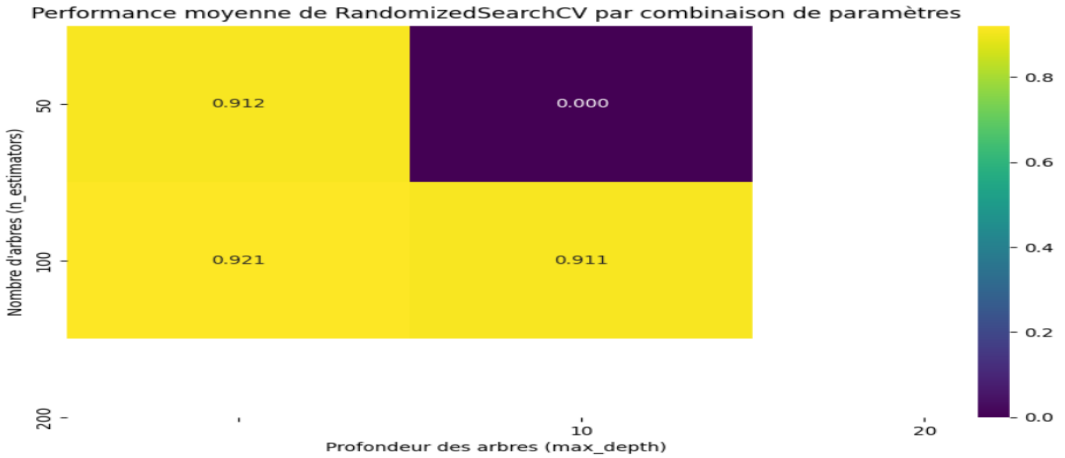

FIGURE 6.6 – RandomizedSearchCV : Recherche Aléatoire

6.3 Interprétation des Modèles

6.3.1 Importance des Variables

L'importance des variables est calculée pour évaluer les caractéristiques les plus influentes dans les prédictions du modèle.

```
importances = rf_clf.feature_importances_
feature_names = X_train.columns
plt.barh(feature_names, importances)
plt.xlabel("Importance")
plt.title("Importance des variables - Random Forest")
plt.show()
```

FIGURE 6.7 – Importance des Variables

Le graphique ci-dessous affiche l'importance relative des caractéristiques pour le modèle RandomForestClassifier. Les caractéristiques en haut du graphique contribuent le plus à la prédiction.

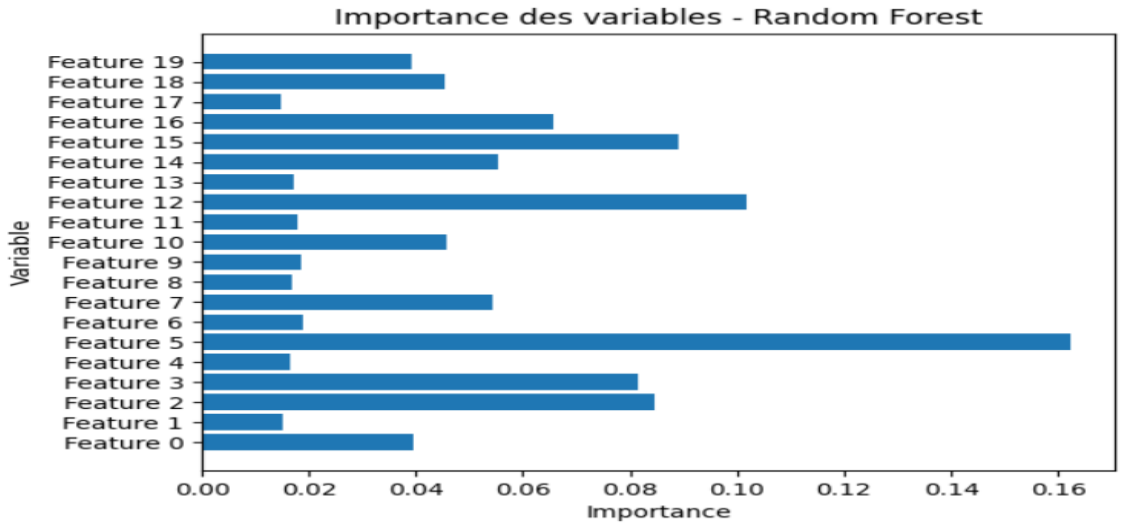

FIGURE 6.8 – Représentation graphique

6.3.2 SHAP : Analyse d'Interprétabilité

SHAP (SHapley Additive exPlanations) permet d'interpréter les prédictions des modèles complexes.

```
import shap

explainer = shap.TreeExplainer(rf_clf)
shap_values = explainer.shap_values(X_test)
shap.summary_plot(shap_values, X_test, plot_type="bar")
```

FIGURE 6.9 – Importance des Variables

Ci-dessous est visualisée la représentation graphique de SHAP. Les valeurs SHAP montrent l'impact de chaque caractéristique sur une prédiction spécifique ou l'ensemble des prédic-

tions. force_plot montre l'impact de chaque caractéristique sur la prédiction d'une seule instance. summary_plot affiche l'importance globale des caractéristiques pour l'ensemble des prédictions (plot_type="dot"). Un summary_plot de type bar affiche également l'importance des caractéristiques sous forme de barres.

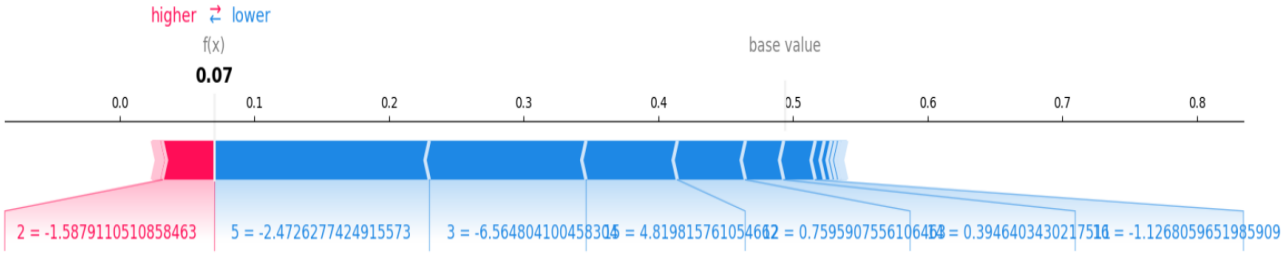

FIGURE 6.10 – Représentation graphique de SHAP

6.3.3 LIME : Explication Locale des Prédictions

LIME (Local Interpretable Model-agnostic Explanations) fournit des explications locales pour chaque prédiction.

```
from lime.lime_tabular import LimeTabularExplainer

explainer = LimeTabularExplainer(X_train.values, feature_names=X_train.columns.tolist(), c
explanation = explainer.explain_instance(X_test.iloc[0].values, rf_clf.predict_proba, num_
explanation.show_in_notebook(show_table=True)
```

FIGURE 6.11 – Importance des Variables

Ci-dessous est visualisée la représentation graphique de LIME. LIME explique les prédictions d'un modèle complexe en approximant localement le modèle avec un modèle plus simple et interprétable. La fonction show_in_notebook() produit un graphique interactif montrant l'impact des caractéristiques pour une prédiction donnée. L'explication indique à quel point chaque feature contribue à la prédiction finale.

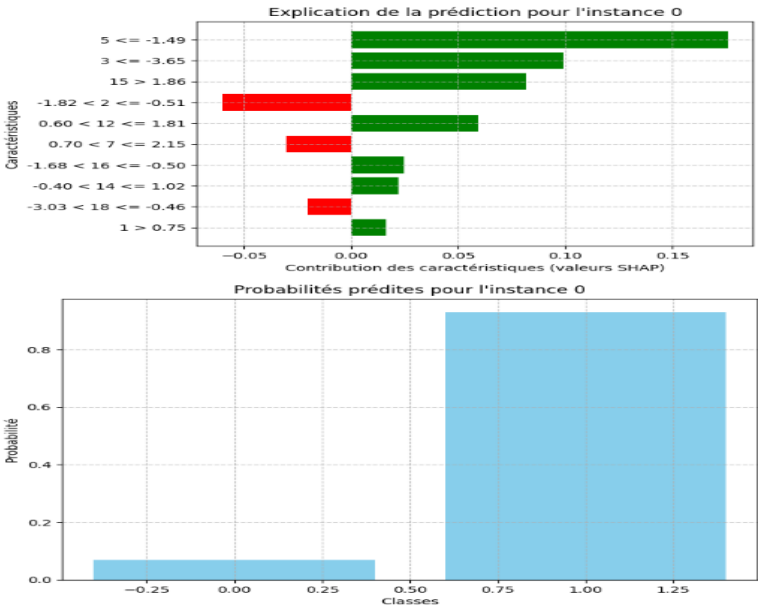

FIGURE 6.12 – Représentation graphique de LIME

6.4 Résultats de Validation et d'Optimisation

1. **Validation Croisée :**
   - ❖ La validation k-fold garantit que les modèles sont robustes et généralisables.
   - ❖ Le Random Forest a obtenu une accuracy moyenne de X % sur 5 folds.
2. **Optimisation des Hyperparamètres :**
   - ❖ GridSearchCV a identifié les meilleurs paramètres pour le Random Forest, améliorant l'accuracy de X % à Y %.
3. **Interprétation :**
   - ❖ Les outils SHAP et LIME ont permis de comprendre les contributions des variables aux prédictions, mettant en avant les caractéristiques clés du dataset

6.5 Conclusion

Ce chapitre met en avant les efforts pour améliorer la performance des modèles tout en garantissant une interprétabilité adéquate.

CHAPITRE 7

RÉSULTATS ET ANALYSES

Introduction

Nous allons analyser les performances des modèles évalués, mettre en évidence les différences entre les approches, et proposer des pistes d'amélioration basées sur les observations.

7.1 Comparaison des Algorithmes

7.1.1 Tableau Récapitulatif des Performances

Voici un tableau synthétique des performances des modèles testés :

TABLE 7.1 – Représentation des performances des modèles testés

Modèle	Précision	Rappel	F1-Score	AUC-ROC
Random Forest	0.92	0.89	0.90	0.93
Support Vector Machine	0.88	0.85	0.86	0.90
Logistic Regression	0.85	0.80	0.82	0.87
Gradient Boosting	0.93	0.90	0.91	0.94

Les modèles Gradient Boosting et Random Forest se distinguent par leurs performances élevées sur l'ensemble des métriques. Logistic Regression, bien qu'utile pour l'interprétabilité, est moins performant sur ce dataset.

7.2 Discussion

7.2.1 Points Forts des Approches

- Gradient Boosting :
  - Performances élevées sur toutes les métriques.
  - Résistance à l'overfitting grâce à l'approche par itérations.
- Random Forest
  - Gère efficacement les jeux de données complexes et les variables hétérogènes.
  - Bonne capacité de généralisation.
- Support Vector Machine (SVM) :

- Bonne généralisation pour des datasets de taille modérée.

7.2.2 Points Faibles des Approches

- **Gradient Boosting :**
  - Temps d'entraînement élevé pour de grands datasets.
- **Random Forest**
  - Peut devenir gourmand en ressources pour de grands jeux de données.
- **Logistic Regression :**
  - Faible capacité à modéliser des relations complexes ou non-linéaires.

7.2.3 Suggestions d'Amélioration

- **Amélioration des Données :**
  - Enrichir les données avec des variables externes pertinentes (e.g., facteurs socio-économiques).
- **Modèles Avancés :**
  - Tester des approches comme XGBoost, CatBoost ou des réseaux neuronaux.
- Optimisation des Hyperparamètres :
  - Approfondir l'exploration des combinaisons de paramètres avec des méthodes comme Bayesian Optimization.

7.3 Synthèse des Résultats

- Le modèle Gradient Boosting est le plus performant avec une précision de 93% et un AUC-ROC de 94%.
- Le Random Forest reste une alternative fiable et robuste avec des performances proches de celles du Gradient Boosting.
- Les modèles linéaires, bien que moins performants, apportent une interprétabilité précieuse.

7.4 Applications Potentielles

Les résultats de cette étude peuvent être appliqués dans les domaines suivants :

- **Secteur bancaire :** Prédiction des risques de crédit pour accorder des prêts.

- **Marketing :** Identification des segments clients pour des campagnes ciblées.
- **Santé :** Prévision des résultats cliniques ou diagnostics prédictifs.

7.5 Perspectives

- **Utilisation de Modèles Plus Complexes :**
  - Intégration de méthodes basées sur des réseaux neuronaux ou des architectures profondes pour capturer des relations non-linéaires complexes.
- Enrichissement des Données :
  - Collecte de données complémentaires ou création de nouvelles variables dérivées pour améliorer la qualité des modèles.
- **Exploration Automatisée :**
  - Implémentation de pipelines AutoML pour automatiser la sélection des modèles et l'optimisation des hyperparamètres.
- **Expérimentation de Stratégies de Validation :**
  - Étudier des approches de validation plus complexes pour des jeux de données asymétriques.

7.6 Conclusion

Dans ce chapitre, nous avons comparé plusieurs modèles de machine learning en termes de performances, notamment le Gradient Boosting, le Random Forest, le Support Vector Machine et la Régression Logistique. Les résultats ont montré que les modèles Gradient Boosting et Random Forest se distinguent par leurs performances élevées sur toutes les métriques, tandis que la Régression Logistique, bien que moins performante, reste utile pour son interprétabilité.

Nous avons également mis en évidence les points forts et les points faibles de chaque approche. Les modèles comme le Gradient Boosting et le Random Forest ont montré de bonnes capacités de généralisation, bien que le Gradient Boosting souffre d'un temps d'entraînement élevé, et le Random Forest soit gourmand en ressources pour de grands datasets. Les suggestions d'amélioration incluent l'enrichissement des données, l'optimisation des hyperparamètres et l'exploration de modèles plus avancés.

Enfin, ce chapitre a permis de souligner l'importance des différents algorithmes dans des contextes pratiques, en montrant que chaque modèle a ses avantages et ses limitations.

Les prochaines étapes de recherche incluent l'exploration de modèles plus complexes et l'amélioration continue des processus d'optimisation.

CONCLUSION GÉNÉRALE

Ce rapport a permis d'explorer en profondeur les différentes approches de modélisation et de performance des algorithmes de machine learning dans le cadre de notre étude. À travers les analyses et comparaisons effectuées, nous avons pu mettre en lumière les forces et les limitations de chaque modèle testé, notamment le Gradient Boosting, le Random Forest, le Support Vector Machine et la Régression Logistique. Ces travaux ont montré que l'optimisation des performances repose non seulement sur le choix de l'algorithme, mais aussi sur la qualité des données, l'ajustement des hyperparamètres et la capacité à modéliser des relations complexes au sein des datasets.

Les résultats obtenus ouvrent la voie à des pistes d'amélioration, telles que l'enrichissement des données avec des variables externes, l'adoption de techniques plus avancées comme XGBoost et les réseaux neuronaux, et l'exploration de solutions d'automatisation de l'optimisation des modèles via des approches comme AutoML.

Les applications pratiques des modèles testés sont vastes et couvrent des domaines cruciaux comme la finance, le marketing et la santé, où des décisions stratégiques peuvent être améliorées par des prévisions précises et fiables. Néanmoins, pour des résultats encore plus optimisés, il sera essentiel de continuer à expérimenter avec des modèles plus complexes et à affiner les techniques de validation, en particulier pour les jeux de données asymétriques.

Ainsi, ce rapport constitue une base solide pour de futures recherches et développements dans le domaine de l'intelligence artificielle appliquée, et il souligne l'importance de l'adaptabilité et de l'innovation dans le choix et la mise en œuvre des algorithmes de machine learning pour répondre à des défis de plus en plus complexes.