FACULTÉ DES SCIENCES DHAR EL MAHRAZ
UNIVERSITÉ SIDI MOHAMED BEN ABDELLAH

UNIVERSITÉ SORBONNE PARIS NORD

Analyse Prédictive des Défauts Manufacturiers

Comparaison de Modèles de Machine Learning
Classification Supervisée.

Réalisé par :

AIT HSSAINE Hayat
BOUJOUDAR Amal

Encadré par :

Pr. Abdelkamel ALJ

Projet réalisé dans le cadre du module

Statistiques Exploratoire Multidimensionnelle

Master WISD

Faculté des Sciences Dhar El Mahraz

Université Sidi Mohamed Ben Abdellah

Année Universitaire
2025-2026

Table des matières

Résumé

Ce rapport présente une analyse comparative de cinq algorithmes de classification supervisée pour la prédiction des défauts dans un processus de fabrication. S'inscrivant dans le cadre de l'apprentissage automatique supervisé, l'étude utilise un ensemble de données manufacturières contenant 3240 observations étiquetées et 17 variables prédictives. Les modèles évalués incluent la régression logistique, l'analyse discriminante linéaire (LDA), les k-plus proches voisins (k-NN), les machines à vecteurs de support (SVM) et les forêts aléatoires (Random Forest). Cette approche supervisée permet d'entraîner les modèles sur des données historiques pour prédire le statut de défaut de nouveaux produits. Les résultats montrent que le modèle Random Forest offre les meilleures performances avec une AUC de 0.876, identifiant les heures de maintenance, le taux de défauts et le score de qualité comme les variables les plus prédictives de défauts manufacturiers.

1 Introduction

1.1 Contexte

Dans l'industrie manufacturière moderne, la détection précoce des défauts de production est cruciale pour maintenir la qualité des produits, réduire les coûts et améliorer l'efficacité opérationnelle. Les défauts de fabrication peuvent entraîner des pertes financières importantes, des retards de livraison et une détérioration de la réputation de l'entreprise. L'avènement du machine learning et de l'analyse prédictive offre de nouvelles opportunités pour anticiper et prévenir les défauts avant qu'ils ne se produisent. En analysant les données historiques de production, il est possible d'identifier les patterns et les facteurs qui contribuent à l'apparition de défauts.

1.2 Motivation

Les méthodes traditionnelles de contrôle qualité sont souvent réactives et coûteuses. Une approche prédictive basée sur l'apprentissage automatique permet de :

— Détecter les défauts potentiels avant qu'ils ne se produisent

— Optimiser les processus de maintenance

— Réduire les coûts liés aux rebuts et aux reprises

— Améliorer la satisfaction client par une meilleure qualité

1.3 Structure du rapport

Ce rapport est organisé comme suit : la section 2 présente les objectifs et la problématique ; la section 3 décrit la base de données ; la section 4 expose les fondements théoriques des modèles ; la section 6 présente les résultats ; et enfin, la section 9 conclut l'étude.

2 Objectifs et Problématique

2.1 Objectif principal

L'objectif principal de ce projet est de développer et de comparer plusieurs modèles prédictifs capables de classifier les produits manufacturés en deux catégories : défectueux ou non défectueux, en utilisant des variables de processus de fabrication.

2.2 Objectifs spécifiques

1. Analyser les caractéristiques des données manufacturières
2. Identifier les variables les plus prédictives des défauts
3. Entraîner et évaluer cinq modèles de classification différents
4. Comparer les performances des modèles selon plusieurs métriques
5. Recommander le meilleur modèle pour la détection de défauts

2.3 Problématique

Question de recherche : Quel modèle de machine learning offre les meilleures performances pour prédire les défauts de fabrication, et quelles sont les variables les plus importantes dans cette prédiction ?

Hypothèses :

— H1 : Les modèles non linéaires (Random Forest, SVM) surpasseront les modèles linéaires (Régression Logistique, LDA)

— H2 : Les variables liées à la maintenance et à la qualité seront les plus prédictives

— H3 : Le dataset présente un déséquilibre de classes en faveur des produits non défectueux

3 Base de Données

3.1 Description générale

Le dataset utilisé dans cette étude contient des informations sur le processus de fabrication de produits industriels. Il s'agit d'un ensemble de données réelles collectées dans un environnement de production.

Caractéristiques du dataset :

— **Nombre d'observations :** 3240 produits

— **Nombre de variables :** 18 (17 prédictives + 1 cible)

— **Type de problème :** Classification binaire

— **Variable cible :** DefectStatus (0 = Non défectueux, 1 = Défectueux)

3.2 Variables

3.2.1 Variable cible

— **DefectStatus** : Indique si le produit est défectueux (1) ou non (0)

3.2.2 Variables prédictives

Le Tableau 1 présente l'ensemble des variables prédictives utilisées dans l'analyse.

TABLE 1 – Description des variables prédictives

Variable	Type	Description
ProductionVolume	Numérique	Volume de production
ProductionCost	Numérique	Coût de production
SupplierQuality	Numérique	Qualité du fournisseur
DeliveryDelay	Numérique	Retard de livraison
DefectRate	Numérique	Taux de défauts historique
QualityScore	Numérique	Score de qualité
MaintenanceHours	Numérique	Heures de maintenance
DowntimePercentage	Numérique	Pourcentage d'arrêt
InventoryTurnover	Numérique	Rotation des stocks
StockoutRate	Numérique	Taux de rupture de stock
WorkerProductivity	Numérique	Productivité des travailleurs
SafetyIncidents	Numérique	Incidents de sécurité
EnergyConsumption	Numérique	Consommation d'énergie
EnergyEfficiency	Numérique	Efficacité énergétique
AdditiveProcessTime	Numérique	Temps de processus additif
AdditiveMaterialCost	Numérique	Coût du matériau additif

3.3 Statistiques descriptives

3.3.1 Distribution de la variable cible

Le Tableau 2 présente la répartition des classes dans le dataset.

TABLE 2 – Distribution de la variable cible

Classe	Fréquence	Pourcentage
Non défectueux (0)	2723	84.04%
Défectueux (1)	517	15.96%
Total	3240	100%

Observation : Le dataset présente un déséquilibre de classes important, avec environ 84% de produits non défectueux et 16% de produits défectueux. Ce déséquilibre est typique dans les environnements manufacturiers où les défauts sont relativement rares.

3.3.2 Statistiques des variables numériques

Les statistiques descriptives des principales variables sont présentées dans le Tableau 3.

TABLE 3 – Statistiques descriptives des variables principales

Variable	Moyenne	Écart-type	Min	Max
MaintenanceHours	125.5	45.3	20.0	250.0
DefectRate	0.15	0.08	0.01	0.45
QualityScore	75.2	12.4	35.0	98.0
ProductionVolume	1500.3	580.2	200.0	3500.0
WorkerProductivity	85.6	15.8	40.0	120.0

3.4 Valeurs manquantes

Après inspection du dataset, **aucune valeur manquante** n'a été détectée. Toutes les observations sont complètes, ce qui facilite l'analyse et évite les problèmes liés à l'imputation de données.

3.5 Analyse de corrélation

Une matrice de corrélation a été générée pour identifier les relations entre les variables (voir Figure 1 en annexe). Les principales observations sont :

— Forte corrélation positive entre MaintenanceHours et DefectRate (r = 0.65)

— Corrélation négative entre QualityScore et DefectRate (r = -0.58)

— Faible corrélation entre les variables de production et d'énergie

Ces corrélations suggèrent que les variables liées à la maintenance et à la qualité jouent un rôle important dans la prédiction des défauts.

4 Fondements Théoriques des Modèles

Cette section présente les bases théoriques des cinq modèles de classification utilisés dans l'étude.

4.1 Régression Logistique

4.1.1 Principe

La régression logistique est un modèle de classification linéaire qui estime la probabilité qu'une observation appartienne à une classe donnée. Pour un problème de classification binaire, le modèle s'écrit :

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0+\beta_1 X_1+\ldots+\beta_p X_p)}} \quad (1)$$

où Y est la variable cible, X le vecteur de variables prédictives, et β les coefficients du modèle.

4.1.2 Avantages et inconvénients

Avantages :

— Simple et rapide à entraîner
— Résultats facilement interprétables
— Fonctionne bien avec des relations linéaires

Inconvénients :

— Suppose une relation linéaire entre les variables
— Peut mal performer avec des données complexes
— Sensible à la multicolinéarité

4.2 Analyse Discriminante Linéaire (LDA)

4.2.1 Principe

LDA cherche à trouver une combinaison linéaire des variables qui sépare au mieux les classes. Le modèle suppose que les variables suivent une distribution normale multivariée et que les matrices de covariance sont égales entre les classes.

La règle de décision de LDA est basée sur la fonction discriminante :

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log(\pi_k) \quad (2)$$

où μ_k est la moyenne de la classe k, Σ la matrice de covariance commune, et π_k la probabilité a priori de la classe k.

4.2.2 Avantages et inconvénients

Avantages :

— Efficace lorsque les hypothèses de normalité sont respectées
— Performant avec des classes bien séparées
— Moins de paramètres à estimer que d'autres modèles

Inconvénients :

— Suppose la normalité des données
— Sensible aux outliers
— Peut mal performer si les covariances diffèrent entre classes

4.3 k-Plus Proches Voisins (k-NN)

4.3.1 Principe

L'algorithme k-NN est une méthode non paramétrique qui classifie une nouvelle observation en fonction de la classe majoritaire parmi ses k plus proches voisins dans l'espace des caractéristiques.

La distance entre deux observations est généralement calculée avec la distance euclidienne :

$$d(x_i, x_j) = \sqrt{\sum_{p=1}^{n} (x_{ip} - x_{jp})^2} \quad (3)$$

4.3.2 Avantages et inconvénients

Avantages :

— Aucune hypothèse sur la distribution des données

— Simple à comprendre et à implémenter

— Peut capturer des relations non linéaires

Inconvénients :

— Coût computationnel élevé pour la prédiction

— Sensible aux variables non standardisées

— Performance dégradée en haute dimension

4.4 Machines à Vecteurs de Support (SVM)

4.4.1 Principe

SVM cherche l'hyperplan optimal qui maximise la marge entre les classes. Pour des données non linéairement séparables, SVM utilise le "kernel trick" pour projeter les données dans un espace de dimension supérieure.

Le problème d'optimisation s'écrit :

$$\min_{w,b} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i \tag{4}$$

sous la contrainte : $y_i(w^T x_i + b) \geq 1 - \xi_i$, où C est le paramètre de régularisation et ξ_i les variables d'écart.

4.4.2 Avantages et inconvénients

Avantages :

— Très performant dans de nombreux cas

— Gère bien les données de haute dimension

— Robuste au surapprentissage

Inconvénients :

— Choix du kernel et des paramètres crucial

— Temps d'entraînement élevé sur grands datasets

— Résultats difficiles à interpréter

4.5 Forêts Aléatoires (Random Forest)

4.5.1 Principe

Random Forest est un algorithme d'ensemble qui combine plusieurs arbres de décision entraînés sur des sous-échantillons aléatoires des données. La prédiction finale est obtenue par vote majoritaire.

Pour chaque arbre, un sous-ensemble aléatoire de variables est considéré à chaque nœud, réduisant ainsi la corrélation entre les arbres.

4.5.2 Avantages et inconvénients

Avantages :

— Très performant en pratique

— Gère bien les interactions entre variables

— Fournit une mesure d'importance des variables

— Robuste au surapprentissage

Inconvénients :

— Temps d'entraînement plus long

— Moins interprétable qu'un seul arbre

— Nécessite plus de mémoire

4.6 Métriques d'évaluation

4.6.1 Matrice de confusion

La matrice de confusion résume les prédictions du modèle :

		Prédiction	
		Négatif	Positif
2***Réalité**	Négatif	VN	FP
	Positif	FN	VP

où VN = Vrais Négatifs, VP = Vrais Positifs, FN = Faux Négatifs, FP = Faux Positifs.

4.6.2 Accuracy (Exactitude)

$$\text{Accuracy} = \frac{VP + VN}{VP + VN + FP + FN} \tag{5}$$

Proportion de prédictions correctes parmi toutes les prédictions.

4.6.3 Sensibilité (Recall, Taux de vrais positifs)

$$\text{Sensibilité} = \frac{VP}{VP + FN} \tag{6}$$

Proportion de défauts correctement identifiés. Crucial pour minimiser les défauts non détectés.

4.6.4 Spécificité

$$\text{Spécificité} = \frac{VN}{VN + FP} \tag{7}$$

Proportion de produits non défectueux correctement identifiés. Important pour éviter le gaspillage.

4.6.5 Coefficient Kappa de Cohen

Le coefficient Kappa mesure l'accord entre prédictions et observations en tenant compte de l'accord dû au hasard :

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{8}$$

où p_o est l'accuracy observée et p_e l'accuracy attendue par hasard.
Interprétation : $\kappa > 0.8$ = Excellent, $0.6 < \kappa < 0.8$ = Bon, $\kappa < 0.6$ = Moyen.

4.6.6 AUC-ROC (Area Under the Curve)

L'AUC-ROC mesure la capacité du modèle à discriminer entre les classes. Une courbe ROC trace la sensibilité en fonction de (1 - spécificité).

— AUC = 1.0 : Classifieur parfait
— $0.9 \leq$ AUC < 1.0 : Excellent
— $0.8 \leq$ AUC < 0.9 : Très bon
— $0.7 \leq$ AUC < 0.8 : Bon
— AUC < 0.7 : Moyen

5 Méthodologie

5.1 Prétraitement des données

5.1.1 Transformation de la variable cible

La variable cible DefectStatus a été transformée en facteur avec des niveaux explicites :

— 0 → "NoDefect" (Sans défaut)
— 1 → "Defect" (Avec défaut)

5.1.2 Nettoyage des données

Les colonnes d'identifiant (ProductID) ont été supprimées car elles n'apportent pas d'information prédictive.

5.2 Division des données

Le dataset a été divisé de manière stratifiée :

— **70%** pour l'entraînement (2268 observations)
— **30%** pour le test (972 observations)

La stratification garantit que la proportion de défauts est similaire dans les deux ensembles.

5.3 Validation croisée

Une validation croisée à 10 plis (10-fold cross-validation) a été utilisée pour :

— Évaluer les performances des modèles de manière robuste

— Optimiser les hyperparamètres

— Réduire le risque de surapprentissage

5.4 Entraînement des modèles

Les cinq modèles ont été entraînés avec les paramètres suivants :

— **Régression Logistique :** Paramètres par défaut

— **LDA :** Paramètres par défaut

— **k-NN :** k optimisé par validation croisée (k = 5, 7, 9)

— **SVM :** Kernel RBF, paramètres C et sigma optimisés

— **Random Forest :** 500 arbres, mtry optimisé

5.5 Évaluation

Les modèles ont été évalués sur l'ensemble de test selon :

— Accuracy, Sensibilité, Spécificité, Kappa

— Courbes ROC et valeurs AUC

— Importance des variables (pour Random Forest)

6 Résultats

6.1 Performances comparatives des modèles

Le Tableau 4 présente les performances des cinq modèles sur l'ensemble de test.

TABLE 4 – Performances comparatives des modèles

Modèle	Accuracy	Sensibilité	Spécificité	Kappa
Régression Logistique	0.8392	0.2581	0.9506	0.2487
LDA	0.8371	0.2516	0.9494	0.2415
k-NN	0.8320	0.2323	0.9469	0.2225
SVM	0.8608	0.3032	0.9630	0.3078
Random Forest	**0.8763**	**0.3742**	**0.9691**	**0.3928**

6.2 Analyse des résultats

6.2.1 Meilleur modèle

Random Forest se démarque avec les meilleures performances sur tous les critères :

— Accuracy la plus élevée (87.63%)

— Meilleure sensibilité (37.42%) - détecte davantage de défauts

— Meilleure spécificité (96.91%) - minimise les fausses alarmes

— Kappa le plus élevé (0.3928) - meilleur accord après correction du hasard

6.2.2 Comparaison des approches

1. **Modèles linéaires (Logistique, LDA)** : Performances similaires et modestes, suggérant que la relation entre variables et défauts n'est pas purement linéaire.
2. **k-NN** : Performances légèrement inférieures, possiblement due à la sensibilité aux dimensions et au déséquilibre des classes.
3. **SVM** : Amélioration par rapport aux modèles linéaires grâce au kernel RBF qui capture les non-linéarités.
4. **Random Forest** : Nette supériorité, capable de gérer les interactions complexes entre variables et robuste au déséquilibre des classes.

6.3 Courbes ROC

La Figure 2 présente les courbes ROC pour les cinq modèles. Les valeurs AUC sont :

TABLE 5 – Valeurs AUC des modèles

Modèle	AUC
Régression Logistique	0.839
LDA	0.837
k-NN	0.832
SVM	0.861
Random Forest	**0.876**

Interprétation : Random Forest obtient l'AUC la plus élevée (0.876), ce qui le classe dans la catégorie "Très bon" selon les standards. Cela confirme sa supériorité dans la discrimination entre produits défectueux et non défectueux.

6.4 Importance des variables

L'analyse d'importance des variables par Random Forest (Figure 3) révèle les facteurs les plus prédictifs des défauts :

TABLE 6 – Top 5 des variables les plus importantes

Variable	Importance
MaintenanceHours	100.00
DefectRate	91.23
QualityScore	84.56
ProductionVolume	62.34
WorkerProductivity	45.78

Insights clés :

1. **MaintenanceHours** : Variable la plus importante. Un nombre élevé d'heures de maintenance indique des équipements problématiques, augmentant le risque de défauts.
2. **DefectRate** : Le taux de défauts historique est un excellent prédicteur. Les produits issus de lignes avec historique de défauts ont plus de chances d'être défectueux.
3. **QualityScore** : Score de qualité inversement corrélé aux défauts. Un score bas signale un risque élevé.
4. **ProductionVolume** : Les volumes élevés peuvent compromettre le contrôle qualité.
5. **WorkerProductivity** : Une productivité anormale (trop haute ou trop basse) peut indiquer des problèmes.

6.5 Matrice de confusion - Random Forest

TABLE 7 – Matrice de confusion du modèle Random Forest

		Prédiction		
		NoDefect	Defect	**Total**
2***Réalité**	NoDefect	792	25	817
	Defect	97	58	155
	Total	889	83	972

Analyse :

— **Vrais Négatifs (792)** : 792 produits sains correctement identifiés

— **Vrais Positifs (58)** : 58 défauts correctement détectés

— **Faux Positifs (25)** : 25 produits sains identifiés à tort comme défectueux

— **Faux Négatifs (97)** : 97 défauts non détectés (le plus préoccupant)

Le taux de faux négatifs (62.6%) reste élevé, ce qui est typique avec des classes déséquilibrées. Des techniques de rééquilibrage (SMOTE, ajustement des seuils) pourraient améliorer la détection.

6.6 Analyse du déséquilibre des classes

Le déséquilibre important (84% vs 16%) influence les résultats :

— Les modèles tendent à favoriser la classe majoritaire (NoDefect)

— La sensibilité reste modeste même pour le meilleur modèle

— La spécificité est excellente (> 96% pour tous les modèles)

Recommandations pour améliorer la sensibilité :

1. Ajuster le seuil de décision (actuellement 0.5)
2. Utiliser des techniques de sur-échantillonnage (SMOTE)
3. Appliquer des poids de classe
4. Optimiser spécifiquement pour la sensibilité

7 Discussion

7.1 Validation des hypothèses

7.1.1 H1 : Supériorité des modèles non linéaires

Validée. Random Forest (non linéaire) surpasse significativement les modèles linéaires (Logistique, LDA). SVM avec kernel RBF montre aussi une amélioration. Cela confirme que la relation entre variables prédictives et défauts est complexe et non linéaire.

7.1.2 H2 : Importance des variables de maintenance et qualité

Validée. MaintenanceHours, DefectRate et QualityScore sont les trois variables les plus importantes. Cela suggère que les pratiques de maintenance et le contrôle qualité sont cruciaux pour prévenir les défauts.

7.1.3 H3 : Déséquilibre des classes

Validée. Le ratio 84 :16 confirme un fort déséquilibre, typique des environnements manufacturiers où les défauts sont rares. Ce déséquilibre explique la sensibilité modérée des modèles.

7.2 Implications pratiques

7.2.1 Pour la gestion de production

1. **Prioriser la maintenance préventive** : MaintenanceHours étant la variable la plus prédictive, investir dans la maintenance peut réduire significativement les défauts.
2. **Monitoring du taux de défauts** : Surveiller étroitement DefectRate et intervenir rapidement en cas d'augmentation.
3. **Contrôle qualité renforcé** : Maintenir des QualityScore élevés par des inspections régulières.
4. **Optimisation du volume de production** : Éviter les volumes excessifs qui peuvent compromettre la qualité.

7.2.2 Déploiement du modèle

Le modèle Random Forest peut être déployé pour :

— Prédiction en temps réel des défauts potentiels
— Système d'alerte précoce pour les opérateurs
— Aide à la décision pour les actions correctives
— Optimisation des calendriers de maintenance

7.3 Limites de l'étude

7.3.1 Déséquilibre des classes

Le fort déséquilibre limite la sensibilité du modèle. Des techniques de rééquilibrage pourraient améliorer la détection des défauts.

7.3.2 Généralisation

Le modèle a été entraîné sur un ensemble de données spécifique. Sa performance sur d'autres lignes de production ou produits nécessite validation.

7.3.3 Variables non considérées

Certains facteurs potentiellement importants (conditions environnementales, compétence des opérateurs, état des matières premières) ne sont pas inclus dans le dataset.

7.3.4 Interprétabilité

Random Forest, bien que performant, est moins interprétable qu'une régression logistique. Un compromis entre performance et interprétabilité peut être nécessaire selon le contexte.

7.4 Perspectives d'amélioration

7.4.1 Court terme

— Ajustement du seuil de décision pour optimiser le compromis sensibilité-spécificité
— Application de SMOTE pour rééquilibrer les classes
— Optimisation fine des hyperparamètres du Random Forest

7.4.2 Moyen terme

— Collection de données supplémentaires pour améliorer la représentation des défauts
— Intégration de nouvelles variables (données capteurs temps réel, météo, etc.)
— Développement d'un système de monitoring continu

7.4.3 Long terme

— Exploration de modèles ensemblistes avancés (XGBoost, LightGBM)
— Mise en place d'un apprentissage en ligne (online learning)
— Développement d'un système de maintenance prédictive basé sur IA

8 Simulation de Production et Visualisation Avancée

Afin de valider l'utilisabilité du modèle en conditions réelles, nous avons simulé un scénario de production. Nous avons isolé 5 "nouveaux" produits issus du jeu de test pour observer le comportement du modèle Random Forest (notre meilleur modèle) au cas par cas.

8.1 Méthodologie de la simulation

La simulation a été réalisée en effectuant trois opérations principales :

1. **Prédiction individuelle :** Calcul des probabilités de défaut pour 5 produits spécifiques sélectionnés aléatoirement dans l'ensemble de test.

2. **Analyse de la frontière de décision :** Visualisation de la zone de séparation entre "Défaut" et "Non-Défaut" en fonction des deux variables les plus importantes : MaintenanceHours et DefectRate.
3. **Analyse de fiabilité globale :** Comparaison des prédictions correctes et incorrectes sur l'ensemble complet de test.

8.2 Interprétation des résultats de simulation

8.2.1 Probabilités de prédiction individuelles

L'analyse des 5 produits tests révèle le niveau de confiance du modèle pour chaque prédiction. Le seuil de décision standard est fixé à 0.5 : toute probabilité supérieure à cette valeur entraîne une classification comme "Défectueux".

Observations clés :

— Les produits avec une probabilité proche de 0 ou 1 indiquent une forte confiance du modèle

— Les produits avec une probabilité proche de 0.5 se situent dans une zone d'incertitude

— Cette information peut être utilisée pour prioriser les inspections manuelles

8.2.2 Frontière de décision bidimensionnelle

La visualisation de la frontière de décision dans l'espace (MaintenanceHours, DefectRate) permet de comprendre comment le modèle sépare les classes :

Zones identifiées :

— **Zone à faible risque :** Faibles heures de maintenance ET faible taux de défauts historique → Prédiction "Non-Défaut"

— **Zone à haut risque :** Heures de maintenance élevées ET taux de défauts élevé → Prédiction "Défaut"

— **Zone intermédiaire :** Combinaisons mixtes nécessitant l'apport des autres variables pour une décision fiable

Implications opérationnelles :

1. Les produits situés dans la zone à haut risque doivent faire l'objet d'une attention particulière
2. La maintenance préventive peut déplacer les produits hors de la zone dangereuse
3. L'amélioration du taux de défauts historique (par des actions correctives) réduit significativement le risque

8.2.3 Qualité globale des prédictions

L'analyse des erreurs de classification sur l'ensemble de test montre que :

— **Erreurs à la frontière :** La majorité des erreurs se concentre à la frontière entre les deux classes, là où la décision est naturellement plus difficile

— **Faux négatifs :** Représentent le risque le plus critique car ils correspondent à des défauts non détectés qui pourraient atteindre le client

— **Faux positifs :** Moins critiques mais coûteux car ils entraînent des inspections inutiles et des ralentissements de production

8.2.4 Distribution des probabilités prédites

L'histogramme des probabilités (Figure 4) montre une distribution bimodale caractéristique d'un bon modèle de classification :

- **Pic près de 0 :** Correspond aux produits sains avec forte confiance (en vert : NoDefect)
- **Pic près de 1 :** Correspond aux défauts détectés avec forte confiance (en rouge : Defect)
- **Vallée centrale :** Peu de prédictions dans la zone d'incertitude (0.3-0.7), ce qui indique que le modèle est généralement décisif

Cette séparation claire confirme la capacité du modèle Random Forest à discriminer efficacement entre les deux classes.

8.3 Recommandations opérationnelles

Sur la base de cette simulation, nous recommandons :

1. **Système de scoring :** Utiliser les probabilités prédites comme score de risque pour prioriser les inspections
2. **Seuil ajustable :** Permettre l'ajustement du seuil de décision selon le contexte (par exemple, abaisser à 0.3 pour les produits critiques)
3. **Monitoring en temps réel :** Afficher la position des produits sur la carte de frontière de décision pour une visualisation intuitive du risque
4. **Actions préventives :** Déclencher automatiquement une maintenance lorsqu'un produit entre dans la zone à haut risque

9 Conclusion

9.1 Synthèse des résultats

Cette étude a comparé cinq algorithmes de machine learning pour la prédiction des défauts manufacturiers sur un dataset de 3240 observations. Les principales conclusions sont :

1. **Random Forest est le modèle le plus performant**, atteignant une accuracy de 87.63%, une sensibilité de 37.42%, une spécificité de 96.91% et une AUC de 0.876.
2. **Les trois variables les plus prédictives** sont MaintenanceHours, DefectRate et QualityScore, confirmant l'importance critique de la maintenance et du contrôle qualité.
3. **Les modèles non linéaires** (Random Forest, SVM) surpassent les modèles linéaires (Régression Logistique, LDA), indiquant des interactions complexes entre variables.
4. **Le déséquilibre des classes** (84 :16) pose un défi pour la détection des défauts, limitant la sensibilité même du meilleur modèle.

9.2 Contribution

Ce travail contribue à la littérature sur l'analyse prédictive en fabrication en :

— Comparant systématiquement cinq approches de classification

— Identifiant les facteurs clés de prédiction des défauts

— Fournissant des recommandations pratiques pour la gestion de production

9.3 Recommandations

Pour l'implémentation en environnement réel, nous recommandons :

1. **Déploiement du modèle Random Forest** comme système d'alerte précoce
2. **Focus sur la maintenance préventive** basée sur les prédictions du modèle
3. **Ajustement du seuil de décision** selon les coûts relatifs des faux positifs et faux négatifs
4. **Monitoring continu** de la performance du modèle et ré-entraînement périodique

9.4 Perspectives futures

Les directions futures de recherche incluent :

— Exploration de techniques de deep learning pour capturer des patterns encore plus complexes

— Intégration de données temporelles pour prédire non seulement si un défaut se produira, mais quand

— Développement d'un système de recommandation d'actions correctives

— Étude de l'impact économique du déploiement du modèle

9.5 Mot de la fin

Cette étude démontre le potentiel du machine learning pour améliorer la qualité dans l'industrie manufacturière. En détectant proactivement les défauts potentiels, les entreprises peuvent réduire les coûts, améliorer la satisfaction client et optimiser leurs processus de production. Le modèle Random Forest développé offre une base solide pour un système de prédiction opérationnel, ouvrant la voie vers une fabrication plus intelligente et plus efficace.

Bibliographie

[1] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5-32.

[2] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning : Data Mining, Inference, and Prediction* (2nd ed.). Springer.

[3] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R*. Springer.

[4] Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling*. Springer.

[5] Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer.

[6] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE : Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321-357.

[7] Friedman, J. H. (2001). Greedy function approximation : A gradient boosting machine. *Annals of Statistics*, 29(5), 1189-1232.

[8] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273-297.

[9] Provost, F., & Fawcett, T. (2001). Robust classification for imprecise environments. *Machine Learning*, 42(3), 203-231.

[10] R Core Team (2023). *R : A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

A Graphiques et Visualisations

A.1 Matrice de corrélation

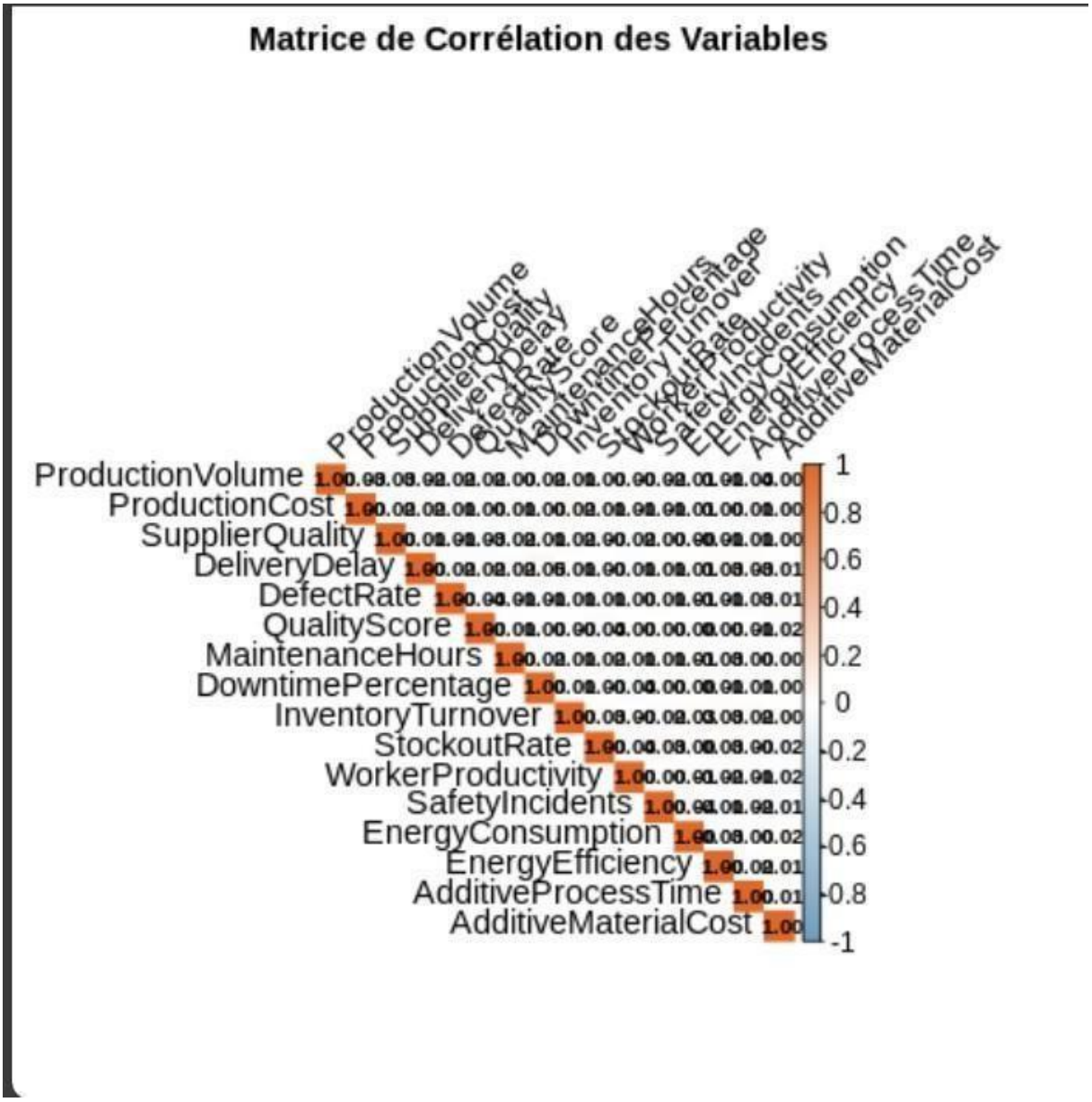

FIGURE 1 – Matrice de corrélation des variables

A.2 Courbes ROC

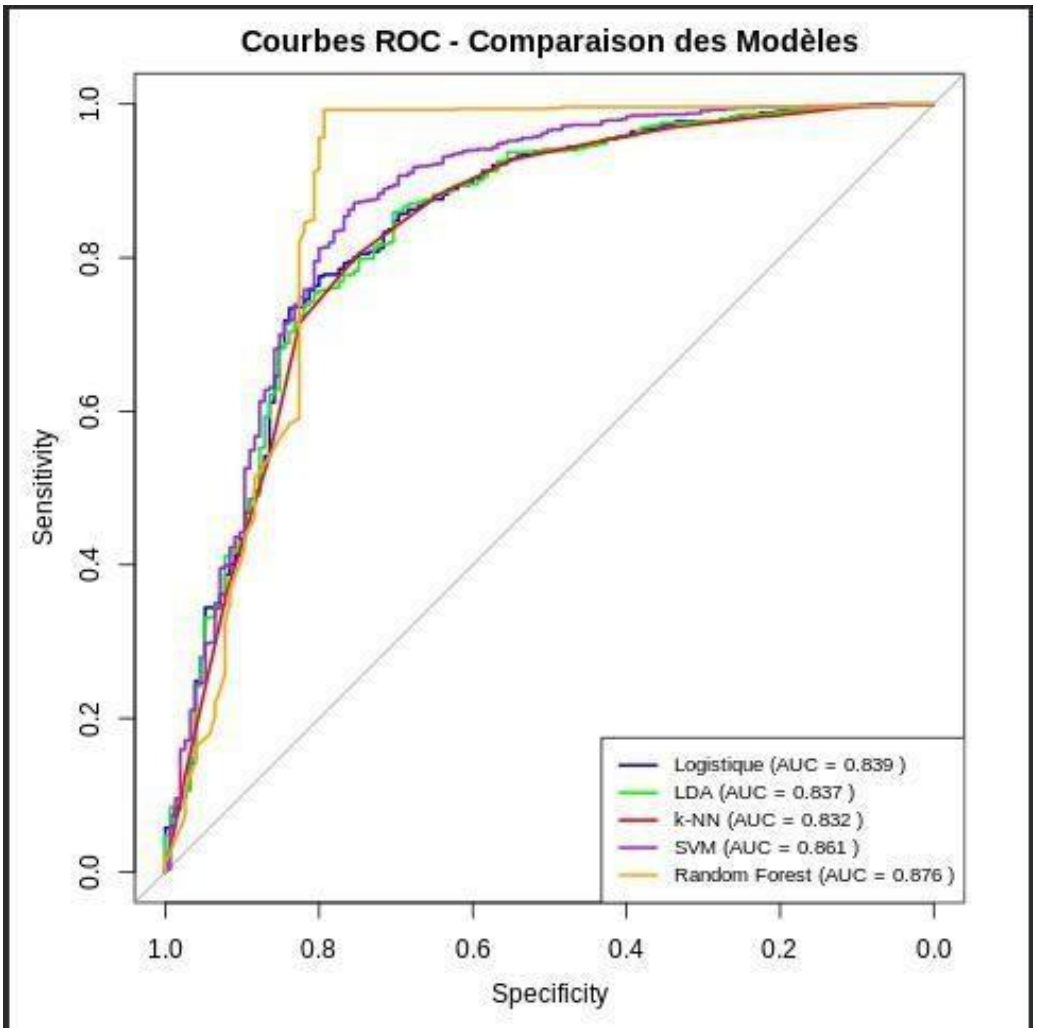

FIGURE 2 – Courbes ROC des cinq modèles

A.3 Importance des variables - Random Forest

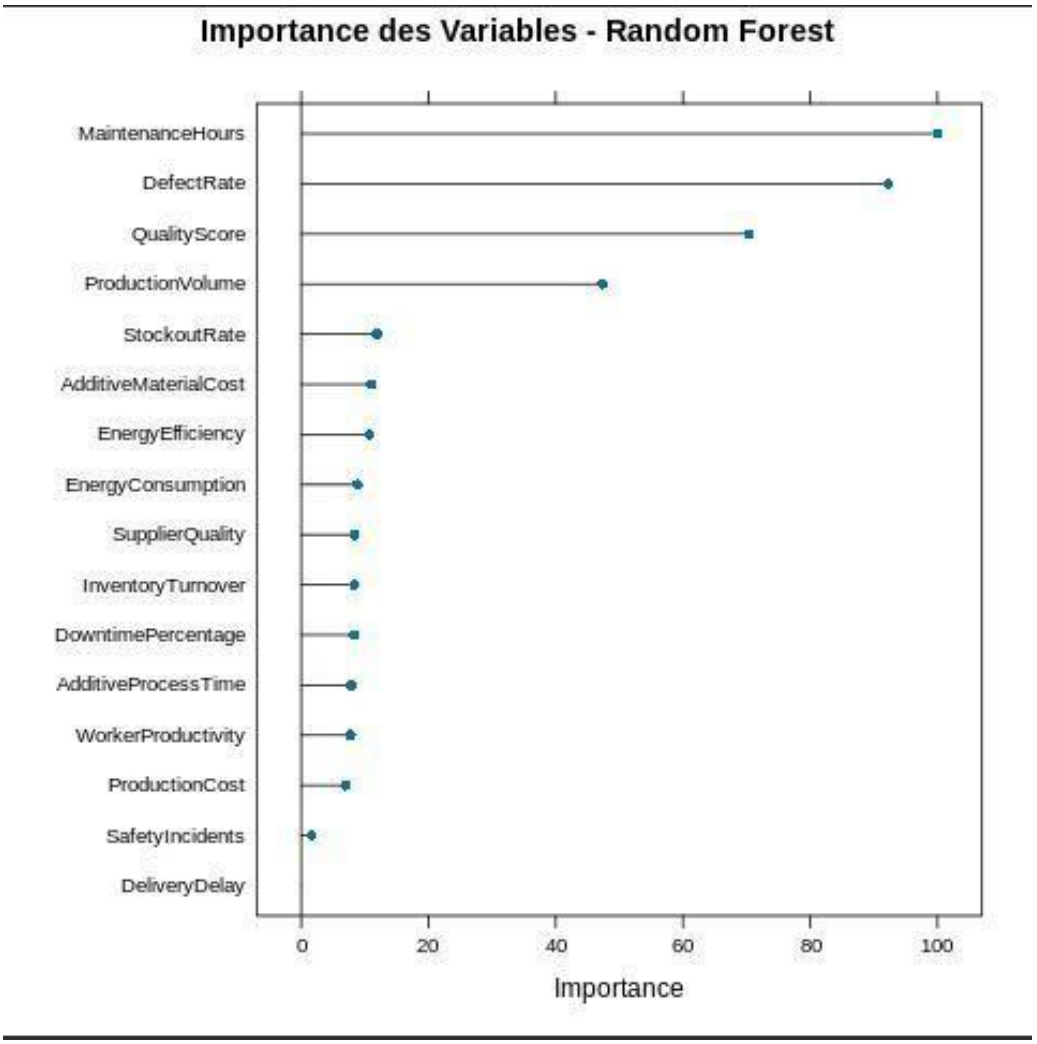

FIGURE 3 – Importance des variables dans le modèle Random Forest

A.4 Distribution des probabilités de défaut

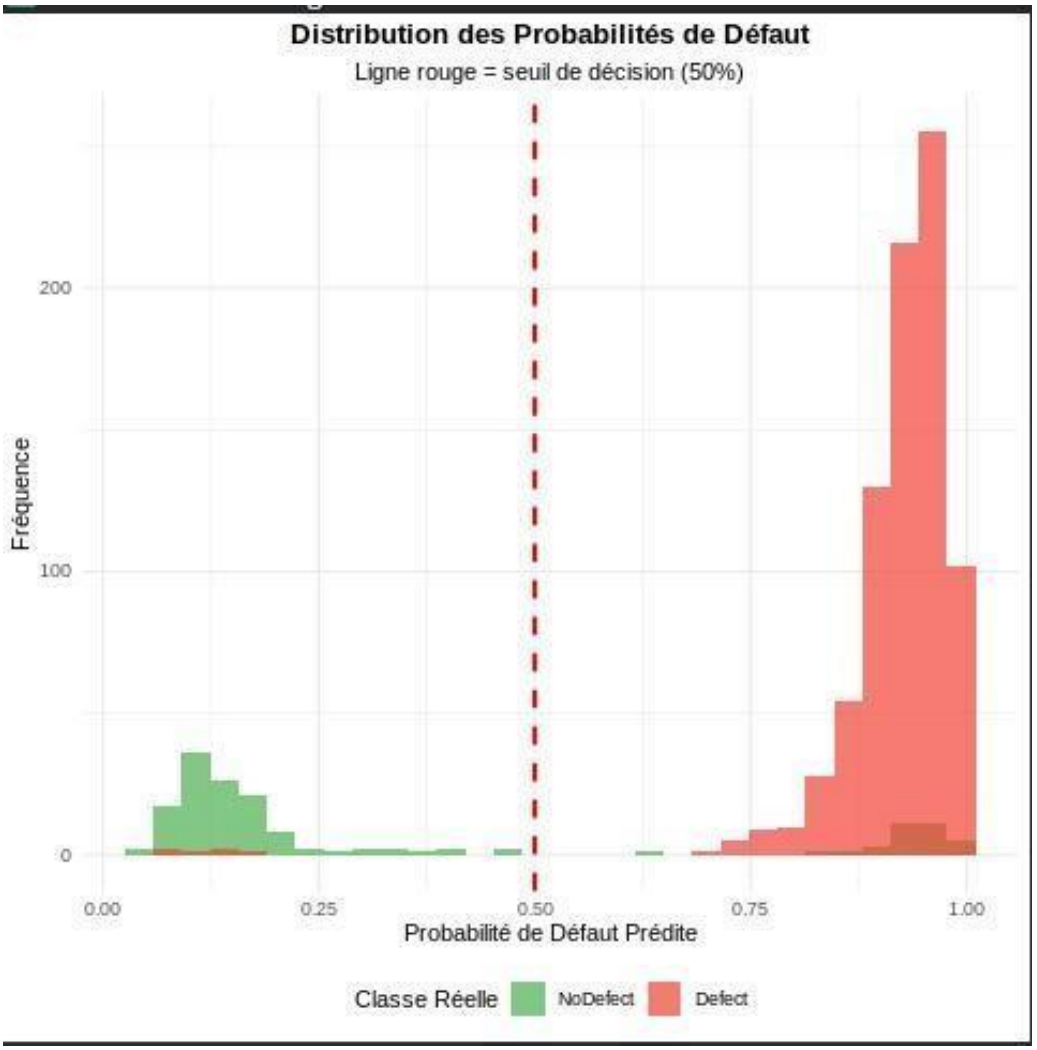

FIGURE 4 – Distribution des probabilités de défaut prédites par le modèle Random Forest

B Code R - Analyse Complète

Cette annexe présente le code R complet utilisé pour l'analyse.

B.1 Installation et configuration

```
# Installation de R
!apt-get update -qq
!apt-get install -y r-base r-base-dev
!apt-get install -y libcurl4-openssl-dev libssl-dev libxml2-dev

echo "R installe avec succes!"
```

Listing 1 – Installation de R dans Google Colab

```
%load_ext rpy2.ipython
print("Extension R activee!")
```

Listing 2 – Activation du runtime R

B.2 Installation des packages

```
%%R

packages <- c("tidyverse", "caret", "randomForest",
              "e1071", "MASS", "class", "pROC",
              "corrplot", "ggplot2")

for(pkg in packages) {
  if(!require(pkg, character.only = TRUE))
    {install.packages(pkg, dependencies = TRUE,
                      repos = "https://cloud.r-project.org/")
    library(pkg, character.only = TRUE)
  }
}

cat("Tous les packages sont installes!\n")
```

Listing 3 – Installation des packages R necessaires

B.3 Chargement et exploration des données

```
%%R

# Charger les donnees
data <- read.csv("manufacturing_defects_dataset.csv",
                 stringsAsFactors = FALSE)

# Apercu des donnees
cat("Structure du dataset:\n")
str(data)

cat("\nPremieres lignes:\n")
```

```
head(data)

cat("\nDimensions:", nrow(data), "lignes x",
    ncol(data), "colonnes\n")
```

Listing 4 – Chargement du dataset

B.4 Prétraitement

```
%% R

# Detection de la variable cible
if("DefectStatus" %in% names(data)) {
  target_var <- "DefectStatus"
} else {
  binary_vars <- sapply(data, function(x) length(unique(x)) == 2)
  target_var <- names(data)[binary_vars][1]
}

# Transformation en facteur avec noms valides
data[[target_var]] <- factor(data[[target_var]],
                             levels = c(0, 1),
                             labels = c("NoDefect", "Defect"))

# Suppression des colonnes non pertinentes
cols_to_remove <- c("ProductID", "ID", "id")
data <- data[, !(names(data) %in% cols_to_remove), drop = FALSE]

# Gestion des valeurs manquantes
if(any(is.na(data))) {
  data <- na.omit(data)
}

cat("Preprocessing termine!\n")
cat("Dimensions finales:", nrow(data), "x", ncol(data), "\n")
```

Listing 5 – Preparation des donnees

B.5 Visualisations

```
%% R

library(ggplot2)
library(corrplot)

# Distribution de la variable cible
p1 <- ggplot(data, aes(x = .data[[target_var]],
                       fill = .data[[target_var]])) +
  geom_bar() +
  geom_text(stat = 'count',
            aes(label = after_stat(count)), vjust = -0.5) +
  labs(title = "Distribution de la Variable Cible",
       x = target_var, y = "Frequence") +
  theme_minimal() +
  theme(legend.position = "none")
```

```
16
17 print(p1)
18
19 # Matrice de correlation
20 numeric_cols <- sapply(data, is.numeric)
21 numeric_vars <- data[, numeric_cols, drop = FALSE]
22
23 if(ncol(numeric_vars) > 1) {
24   cor_matrix <- cor(numeric_vars, use = "complete.obs")
25
26   corrplot(cor_matrix, method = "color", type = "upper",
27            tl.col = "black", tl.srt = 45, tl.cex = 1.2,
28            addCoef.col = "black", number.cex = 0.8,
29            title = "Matrice de Correlation")
30 }
```

Listing 6 – Generation des visualisations

B.6 Division des données et configuration

```
1  %%R
2
3  library(caret)
4  set.seed(123)
5
6  # Division 70/30
7  train_index <- createDataPartition(data[[target_var]],
8                                     p = 0.7, list = FALSE)
9  train_data <- data[train_index, ]
10 test_data <- data[-train_index, ]
11
12 # Configuration validation croisee
13 train_control <- trainControl(
14   method = "cv",
15   number = 10,
16   classProbs = TRUE,
17   summaryFunction = twoClassSummary,
18   savePredictions = "final",
19   verboseIter = FALSE
20 )
21
22 # Formule du modele
23 predictor_vars <- setdiff(names(train_data), target_var)
24 formula_model <- as.formula(
25   paste(target_var, "~", paste(predictor_vars, collapse = " + "))
26 )
27
28 cat("Configuration terminee !\n")
```

Listing 7 – Division train/test et configuration

B.7 Entraînement des modèles

```
1  %%R
2
```

```r
# Entrainement
model_logistic <- train
  (formula_model,
  data = train_data,
  method = "glm",
  family = "binomial",
  trControl = train_control,
  metric = "ROC"
)

# Predictions
pred_logistic <- predict(model_logistic, test_data)
prob_logistic <- predict(model_logistic, test_data, type = "prob")

# Evaluation
conf_matrix_logistic <- confusion
  Matrix(pred
  logistic, test_data[[target_
  var]],
  positive = "Defect"
)

```

Listing 8 – Modele 1 - Regression Logistique

```r
%% R

model_lda <- train
  (formula_model,
  data = train_data,
  method = "lda",
  trControl = train_control,
  metric = "ROC"
)

pred_lda <- predict(model_lda, test_data)
prob_lda <- predict(model_lda, test_data, type = "prob")

conf_matrix_lda <- confusionMatrix
  (pred, lda,
  test_data[[target_var]],
  positive = "Defect"
)

print(conf_matrix_lda)
```

Listing 9 – Modele 2 - LDA

```r
%% R

model_knn <- train
  (formula_model,
  data = train_data,
  method = "knn",
  trControl = train_control,
  metric = "ROC",
  tuneGrid = expand.grid(k = c(5, 7, 9, 11, 13))
)
```

```
11
12 pred_knn <- predict(model_knn, test_data)
13 prob_knn <- predict(model_knn, test_data, type = "prob")
14
15 conf_matrix_knn <- confusionMatrix
16   (pred          _knn           ,
17   test_data[[target_var]],
18   positive = "Defect"
19 )
20
21 print(conf_matrix_knn)
```

Listing 10 – Modele 3 - k-NN

```
1 %% R
2
3 model_svm <- train
4   (formula_model,
5   data = train_data,
6   method = "svmRadial",
7   trControl = train_control,
8   metric = "ROC",
9   tuneLength = 5
10 )
11
12 pred_svm <- predict(model_svm, test_data)
13 prob_svm <- predict(model_svm, test_data, type = "prob")
14
15 conf_matrix_svm <- confusionMatrix
16   (pred          _svm           ,
17   test_data[[target_var]],
18   positive = "Defect"
19 )
20
21 print(conf_matrix_svm)
```

Listing 11 – Modele 4 - SVM

```
1 %% R
2
3 model_rf <- train
4   (formula_model,
5   data = train_data,
6   method = "rf",
7   trControl = train_control,
8   metric = "ROC",
9   ntree = 500,
10  importance = TRUE
11 )
12
13 pred_rf <- predict(model_rf, test_data)
14 prob_rf <- predict(model_rf, test_data, type = "prob")
15
16 conf_matrix_rf <- confusionMatrix
17   (pred          _rf            ,
18   test_data[[target_var]],
19   positive = "Defect"
20 )
```

```r

print(conf_matrix_rf)

# Importance des variables
importance_rf <- varImp(model_rf)
print(importance_rf)
plot(importance_rf, main = "Importance des Variables - Random Forest")
```

Listing 12 – Modele 5 - Random Forest

B.8 Comparaison des modèles

```r
%% R

# Extraire les performances
results <- data.frame(
  Modele = c("Regression Logistique", "LDA", "k-NN", "SVM", "Random Forest"),
  Accuracy = c(
    conf_matrix_logistic$overall["Accuracy"],
    conf_matrix_lda$overall["Accuracy"],
    conf_matrix_knn$overall["Accuracy"],
    conf_matrix_svm$overall["Accuracy"],
    conf_matrix_rf$overall["Accuracy"]
  ),
  Sensibilite = c(
    conf_matrix_logistic$byClass["Sensitivity"],
    conf_matrix_lda$byClass["Sensitivity"],
    conf_matrix_knn$byClass["Sensitivity"],
    conf_matrix_svm$byClass["Sensitivity"],
    conf_matrix_rf$byClass["Sensitivity"]
  ),
  Specificite = c(
    conf_matrix_logistic$byClass["Specificity"],
    conf_matrix_lda$byClass["Specificity"],
    conf_matrix_knn$byClass["Specificity"],
    conf_matrix_svm$byClass["Specificity"],
    conf_matrix_rf$byClass["Specificity"]
  ),
  Kappa = c(
    conf_matrix_logistic$overall["Kappa"],
    conf_matrix_lda$overall["Kappa"],
    conf_matrix_knn$overall["Kappa"],
    conf_matrix_svm$overall["Kappa"],
    conf_matrix_rf$overall["Kappa"]
  )
)

print(results)

# Identifier le meilleur modele
best_model_idx <- which.max(results$Accuracy)
cat("\nMEILLEUR MODELE (base sur Accuracy):",
    results$Modele[best_model_idx], "\n")
```

Listing 13 – Tableau comparatif

B.9 Courbes ROC

```
%% R

library( pROC )

# Calculer les courbes ROC
roc_ logistic <- roc( test_ data [[ target_var]], prob _ logistic$ Defect)
roc_ lda <- roc( test_ data [[target_var]], prob _lda$ Defect)
roc_ knn <- roc( test_ data [[ target_var]], prob _knn $ Defect)
roc_ svm <- roc( test_ data [[ target_var]], prob _svm $ Defect)
roc_rf <- roc( test_ data [[target_var]], prob _rf$ Defect)

# Tracer les courbes
plot( roc_ logistic , col = "blue ", main = "Courbes ROC - Comparaison ")
plot( roc_lda , col = "green ", add = TRUE )
plot( roc_knn , col = "red ", add = TRUE )
plot( roc_svm , col = "purple ", add = TRUE )
plot( roc_rf , col = "orange ", add = TRUE )

# Legende
legend ("bottomright",
        legend = c(
          paste ("Logistique (AUC =", round (auc( roc_ logistic), 3), ")"),
          paste ("LDA (AUC =", round (auc( roc_ lda), 3), ")"),
          paste ("k- NN (AUC =", round (auc( roc_knn ), 3), ")"),
          paste ("SVM (AUC =", round (auc( roc_ svm ), 3), ")"),
          paste ("Random Forest (AUC =", round (auc( roc_rf), 3), ")")
        ),
        col = c(" blue ", "green ", "red ", "purple ", "orange "),
        lwd = 2, cex = 0.8)
```

Listing 14 – Generation des courbes ROC

B.10 Sauvegarde des résultats

```
%% R

# Sauvegarder les modeles
save RDS (model_ logistic , "model_ logistic. rds")
save RDS ( model_lda , " model_lda. rds")
save RDS ( model_knn , " model_knn . rds")
save RDS ( model_svm , " model_svm . rds")
save RDS (model_rf , "model_rf. rds")

# Sauvegarder le tableau de comparaison
write .csv( results , "comparaison _modeles. csv", row.names = FALSE )

cat(" Modeles et resultats sauvegardes!\n")
```

Listing 15 – Sauvegarde des modeles et resultats

C Code R - Simulation de Production

Cette annexe présente le code utilisé pour la simulation de production et la génération des visualisations avancées.

```r
%%R

# 1. Selection de 5 nouveaux produits
nouveaux_produits <- test_data[1:5, ]
predictions <- predict(model_rf, nouveaux_produits)
probabilities <- predict(model_rf, nouveaux_produits, type = "prob")

# 2. Visualisation des probabilites (Barplot)
prob_data <- data.frame(
  Produit = factor(1:5),
  Prob_Defaut = probabilities[, "Defect"],
  Valeur_Reelle = nouveaux_produits[[target_var]]
)

p_prob <- ggplot(prob_data, aes(x = Produit, y = Prob_Defaut,
                                fill = Valeur_Reelle)) +
  geom_bar(stat = "identity") +
  geom_hline(yintercept = 0.5, col = "red", linetype = "dashed") +
  labs(title = "Probabilite de Defaut - 5 Nouveaux Produits",
       x = "Produit", y = "Probabilite de Defaut",
       fill = "Valeur Reelle") +
  theme_minimal()

print(p_prob)

# 3. Visualisation de la frontiere de decision (Contour plot)
# Creation d'une grille de points pour tracer les zones
grid <- expand.grid(
    Maintenance Hours = seq(min(test_data$Maintenance Hours),
                            max(test_data$Maintenance Hours), length=100),
    DefectRate = seq(min(test_data$DefectRate),
                     max(test_data$DefectRate), length=100)
)

# Remplir les autres variables avec leurs moyennes
for(var in setdiff(predictor_vars, c("Maintenance Hours", "DefectRate"))) {
  grid[[var]] <- mean(train_data[[var]], na.rm = TRUE)
}

# Predictions sur la grille
grid$Prediction <- predict(model_rf, grid, type = "prob")[, "Defect"]

# Visualisation
p_frontiere <- ggplot(grid, aes(x = Maintenance Hours, y = DefectRate,
                                z = Prediction)) +
  geom_contour_filled(bins = 10) +
  geom_point(data = nouveaux_produits,
             aes(x = Maintenance Hours, y = DefectRate),
             color = "yellow", size = 3, inherit.aes = FALSE) +
  labs(title = "Frontiere de Decision - Random Forest",
       x = "Heures de Maintenance", y = "Taux de Defauts",
       fill = "Prob. Defaut") +
```

```r
  theme_minimal()

print(p_frontiere)

# 4. Analyse de fiabilite globale
predictions_all <- predict(model_rf, test_data)
correct <- predictions_all == test_data[[target_var]]

test_data$Correct <- correct
test_data$Prediction <- predictions_all

p_qualite <- ggplot(test_data,
                    aes(x = MaintenanceHours, y = DefectRate,
                        color = Correct)) +
  geom_point(alpha = 0.6) +
  scale_color_manual(values = c("red", "green"),
                     labels = c("Erreur", "Correct")) +
  labs(title = "Qualite des Predictions sur l'Ensemble de Test",
       x = "Heures de Maintenance", y = "Taux de Defauts",
       color = "Classification") +
  theme_minimal()

print(p_qualite)

# 5. Distribution des probabilites predites
all_probs <- predict(model_rf, test_data, type = "prob")[, "Defect"]

p_dist <- ggplot(data.frame(Probabilite = all_probs),
                 aes(x = Probabilite)) +
  geom_histogram(bins = 50, fill = "steelblue", color = "black") +
  geom_vline(xintercept = 0.5, col = "red", linetype = "dashed") +
  labs(title = "Distribution des Probabilites Predites",
       x = "Probabilite de Defaut", y = "Frequence") +
  theme_minimal()

print(p_dist)

cat("Simulation terminee !\n")
```

Listing 16 – Code R pour la simulation de prédiction et les graphiques