**Régression Linéaire : Définition et Fonctionnement**

La régression linéaire modélise la relation entre une variable dépendante yyy et une ou plusieurs variables indépendantes xxx en ajustant une ligne droite (également appelée "ligne de régression") à travers les points de données. L'équation de cette ligne est généralement de la forme :

Une image contenant texte, Police, ligne, capture d’écran

Description générée automatiquement

**Exemple en Python avec scikit-learn (Voir jupiter Notebook)**

Scikit-learn est une bibliothèque populaire pour le machine learning en Python. Elle offre des outils simples et efficaces pour l'analyse de données et la modélisation statistique, y compris la régression linéaire.

Imaginons un cas simple où nous avons des données sur la durée de vie restante de moteurs d'hélicoptères en fonction de l'heure totale de fonctionnement. Nous pouvons utiliser ces données pour entraîner un modèle de régression linéaire. Voici comment on pourrait faire cela en Python :

Dans le contexte de la régression linéaire, les matrices colonne (ou vecteurs colonne) sont souvent utilisées pour représenter les données d'entrée (features) et les étiquettes (labels) en raison de la manière dont les algorithmes de régression et d'apprentissage machine traitent les données. Voici pourquoi :

1. **Compatibilité avec les algorithmes** :
   * Les algorithmes de régression linéaire, tels que celui fourni par la bibliothèque scikit-learn, s'attendent à ce que les données d'entrée soient sous forme de matrice 2D où chaque colonne représente une caractéristique (feature) et chaque ligne représente une observation.
   * En utilisant une matrice colonne pour XXX, nous explicitons que chaque observation (heure de fonctionnement du moteur) est une caractéristique unique, ce qui est nécessaire pour l'algorithme de régression linéaire.
2. **Forme des données** :
   * Les algorithmes de machine learning nécessitent souvent une distinction claire entre les observations (lignes) et les caractéristiques (colonnes). Même si dans ce cas spécifique, chaque observation n'a qu'une seule caractéristique, il est toujours nécessaire de le formater en une matrice 2D.



1. **Compatibilité avec l'apprentissage supervisé** :
   * La régression linéaire implique la multiplication matricielle entre les coefficients (poids) et les caractéristiques des données d'entrée. Pour que cette multiplication soit valide, XXX doit être une matrice 2D.
2. **Consistance des prédictions** :
   * Lorsque vous effectuez des prédictions, la méthode predict s'attend également à ce que les nouvelles données soient sous forme de matrice 2D. C'est pourquoi np.array([[700]]) est utilisé pour les nouvelles données de prédiction, garantissant que les dimensions sont compatibles avec les attentes du modèle.



Le graphique montre une relation linéaire négative entre les heures de fonctionnement du moteur (X) et la durée de vie restante (y). Chaque point bleu représente une observation des données, et la ligne rouge est la ligne de régression linéaire ajustée. La pente de la ligne indique que la durée de vie restante du moteur diminue à mesure que les heures de fonctionnement augmentent.

Ce tracé peut être utilisé pour prédire la durée de vie restante du moteur pour un nombre donné d'heures de fonctionnement qui ne figurent pas dans le jeu de données initial.

--

**Régression Logistique : Définition et Fonctionnement**

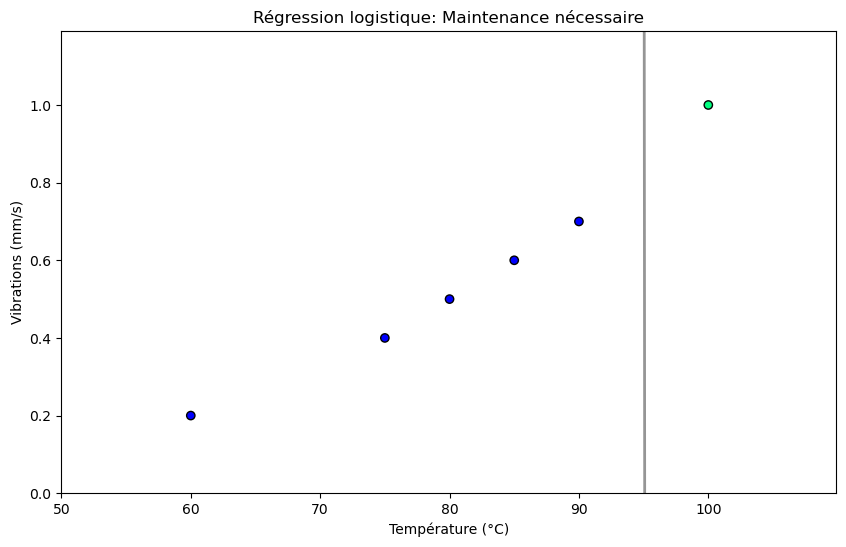
C’est un autre algorithme très courant en machine learning, particulièrement adapté à la classification binaire, où la variable cible prend deux valeurs distinctes (comme "oui" ou "non", "défectueux" ou "non défectueux").

La régression logistique est utilisée pour estimer la probabilité qu'une observation appartienne à une certaine catégorie. Au lieu de simplement ajuster une ligne droite comme en régression linéaire, la régression logistique ajuste une courbe en forme de S (fonction logistique ou sigmoïde) aux données. Cette courbe permet de prédire la probabilité de l'une des deux catégories possibles.

Une image contenant texte, Police, ligne, algèbre

Description générée automatiquement

Pour illustrer, imaginons que nous voulons classifier des moteurs d'hélicoptères comme nécessitant une maintenance ou non, basé sur les lectures des capteurs (par exemple, la température et les vibrations du moteur).



**Interprétation du Graphique**

1. **Points de Données**:
   * Les points bleus indiquent les observations où la maintenance n'est pas nécessaire (étiquetés avec 0).
   * Le point vert indique une observation où la maintenance est nécessaire (étiqueté avec 1).

 **Distribution des Points**:

* La plupart des points bleus (pas de maintenance nécessaire) sont situés à des températures plus basses et des niveaux de vibration inférieurs.
* Le point vert (maintenance nécessaire) est situé à une température élevée de 100°C et à un niveau de vibration de 1.0 mm/s, ce qui suggère que des valeurs élevées de ces deux mesures peuvent indiquer une probabilité accrue qu'une maintenance soit nécessaire.

 **Frontière de Décision**:

* La ligne verticale noire semble représenter une frontière de décision proposée par le modèle. Cependant, cette frontière apparaît comme une ligne verticale, ce qui est inhabituel pour un modèle de régression logistique traitant de deux caractéristiques. Normalement, on s'attendrait à voir une courbe ou une ligne oblique qui divise l'espace des caractéristiques en deux zones basées sur les probabilités calculées par le modèle.

-

Les coefficients du modèle indiqueront combien chaque caractéristique (température et vibrations) contribue à la décision :

* Un coefficient plus élevé pour la température signifie que la température a une influence plus significative sur la probabilité de maintenance.
* Un coefficient plus faible pour les vibrations signifie que les vibrations ont une influence moindre.

 **Distribution des Données**:

* Si la caractéristique des vibrations n'apporte pas beaucoup de variance ou ne contribue pas significativement à la décision, le modèle peut se concentrer principalement sur la température. Dans votre ensemble de données, il semble que la température ait une plus grande influence sur la décision de maintenance.

 **Fonctionnement du Modèle**:

* La régression logistique crée une frontière de décision linéaire entre les classes. Si une seule des caractéristiques est dominante (ici, la température), la frontière de décision peut sembler alignée sur cette caractéristique.

 **Balance des Données**:

* Si le modèle détecte que les changements dans la température ont un impact beaucoup plus significatif sur la nécessité de maintenance comparé aux vibrations, il peut prioriser cette caractéristique.

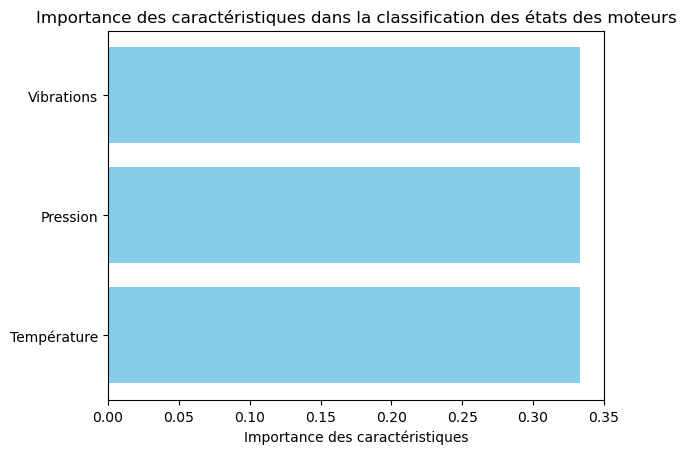
**Forêts Aléatoires : Définition et Fonctionnement**

les Forêts Aléatoires (Random Forests) sont une technique puissante et très populaire en machine learning pour aborder à la fois les problèmes de classification et de régression.

La méthode des forêts aléatoires est un algorithme d'apprentissage supervisé qui utilise un ensemble d'arbres de décision pour améliorer la stabilité et la précision des prédictions. Elle combine les résultats de plusieurs arbres de décision, construits à partir de sous-ensembles aléatoires de l'ensemble de données, pour produire une prédiction finale. Cette approche est particulièrement efficace pour réduire le risque de surapprentissage (overfitting) et pour améliorer la précision générale des prédictions.

**Fonctionnement des Forêts Aléatoires**

1. **Sélection aléatoire de sous-ensembles** : À partir de l'ensemble de données original, des sous-ensembles sont créés en sélectionnant des échantillons de manière aléatoire avec remplacement (bootstrap samples).
2. **Construction d'arbres de décision** : Un arbre de décision est construit pour chaque sous-ensemble. Pour chaque nœud de l'arbre, un nombre limité de caractéristiques est sélectionné aléatoirement pour déterminer la meilleure scission.
3. **Agrégation (Bagging)** : Les prédictions de chaque arbre de décision sont ensuite agrégées pour former la prédiction finale. Pour la classification, c'est généralement le mode des prédictions (la classe la plus fréquente) ; pour la régression, c'est la moyenne

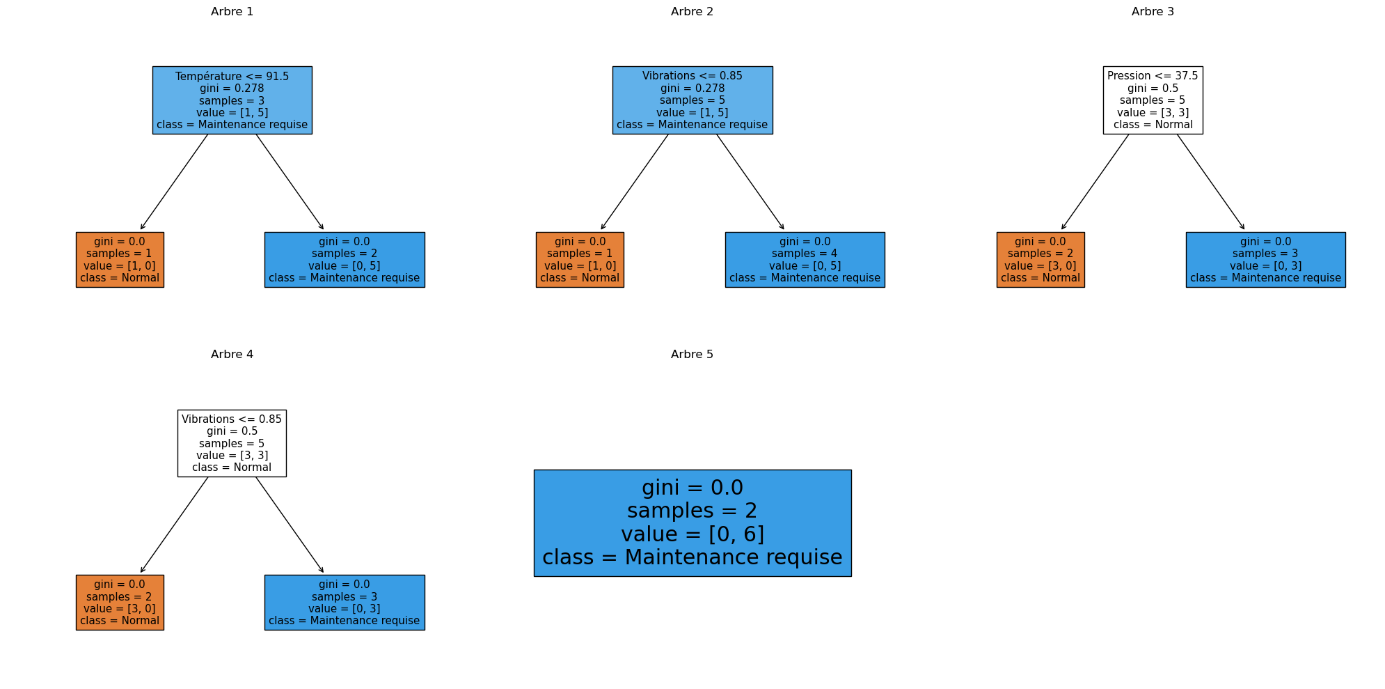


le graphique des importances des caractéristiques dans la classification des états des moteurs utilisant l'algorithme des forêts aléatoires. Chaque barre représente l'importance relative d'une caractéristique (température, pression, vibrations) dans la prédiction de l'état du moteur (normal ou nécessitant une maintenance). Les importances sont calculées en fonction de combien chaque caractéristique contribue à l'amélioration de la prédiction lors de la construction des arbres de décision.

**Consensus dans les Forêts Aléatoires**

Le consensus dans une forêt aléatoire (ou random forest) est obtenu par une méthode appelée **"majority voting"** pour la classification et **"averaging"** pour la régression.

1. **Classification** : Chaque arbre de la forêt fait une prédiction de classe pour une observation donnée. La classe qui obtient le plus de "votes" (prédite par le plus grand nombre d'arbres) est choisie comme la prédiction finale de la forêt aléatoire.
2. **Régression** : Chaque arbre fait une prédiction numérique pour une observation donnée. La prédiction finale est la moyenne de toutes ces prédictions numériques.



**Remarque :** Si vous avez besoin de visualiser la logique de décision collective de la forêt, vous devriez vous concentrer sur des métriques comme l'importance des caractéristiques, les courbes ROC, ou d'autres formes d'évaluation statistique, plutôt que d'essayer de visualiser l'ensemble de la structure arborescente, ce qui n'est pas possible avec les forêts aléatoires en tant que telles.

des exemples individuels de décisions prises par des arbres de décision au sein d'une forêt aléatoire. Chaque arbre est indépendant des autres et est construit en utilisant un sous-ensemble de données aléatoires, ce qui aide à diversifier les décisions et à réduire le surapprentissage.

Les arbres montrent des décisions basées sur les caractéristiques des moteurs d'hélicoptères (température, pression, vibrations), et classifient chaque moteur soit comme 'Normal' soit comme nécessitant 'Maintenance'.

**Observation de la foret aléatoire**

**Arbre 1**

1. **Nœud Racine** :
   * **Critère** : Température <= 91.5
   * **Indice Gini** : 0.278
     + L'indice Gini mesure l'impureté d'un nœud. Un Gini de 0 indique un nœud parfaitement pur (toutes les instances sont de la même classe). Un indice plus élevé indique une plus grande mixité des classes.
   * **Échantillons** : 6
     + Nombre total d'échantillons de données passant par ce nœud.
   * **Valeur** : [1, 5]
     + Le tableau montre le nombre d'échantillons dans chaque classe; ici, 1 pour 'Normal' et 5 pour 'Maintenance requise'.
   * **Classe** : Maintenance requise
     + La classe majoritaire pour ce nœud est 'Maintenance requise', donc toute nouvelle observation qui atteint ce nœud serait classée ainsi si aucune subdivision supplémentaire n'était faite.
2. **Branches du Nœud Racine** :
   * **Gauche (Vrai)** : Tous les moteurs avec une température <= 91.5, classés comme 'Normal'. Gini = 0.0, indiquant une pureté parfaite.
   * **Droite (Faux)** : Tous les moteurs avec une température > 91.5, également classés comme 'Maintenance requise'. Gini = 0.0, indiquant aussi une pureté parfaite.

**Arbre 2**

1. **Nœud Racine** :
   * **Critère** : Vibrations <= 0.85
   * **Indice Gini** : 0.278
   * **Échantillons** : 6
   * **Valeur** : [1, 5]
   * **Classe** : Maintenance requise
2. **Branches du Nœud Racine** :
   * **Gauche (Vrai)** : Tous les moteurs avec des vibrations <= 0.85, classés comme 'Normal'. Gini = 0.0.
   * **Droite (Faux)** : Tous les moteurs avec des vibrations > 0.85, classés comme 'Maintenance requise'. Gini = 0.0.

**Comment interpréter ces informations ?**

* **Critère de Division** : Chaque nœud utilise un critère pour diviser les données. Ce critère est basé sur une caractéristique qui est la plus efficace pour réduire l'incertitude (impureté) concernant la classe cible.
* **Indice Gini** : Donne une idée de l'efficacité de la séparation effectuée par le nœud. Un indice Gini faible à un nœud signifie que la séparation a bien clarifié la classification des échantillons entre les classes.
* **Valeur** : Le nombre d'échantillons de chaque classe qui passe par ce nœud aide à comprendre la distribution des classes à ce stade de la décision.

Rmq : Dans le contexte des forêts aléatoires et plus généralement de l'apprentissage supervisé, les "feuilles" ou "nœuds terminaux" de chaque arbre de décision servent à faire des prédictions basées sur les données d'entrée. Le but ultime est d'utiliser ces prédictions pour prendre des décisions éclairées ou faire des classifications précises

Dans le contexte des forêts aléatoires et plus généralement de l'apprentissage supervisé, les "feuilles" ou "nœuds terminaux" de chaque arbre de décision servent à faire des prédictions basées sur les données d'entrée. Le but ultime est d'utiliser ces prédictions pour prendre des

Dans le cadre de notre projet sur la classification des moteurs d'hélicoptères.

**Objectif des Feuilles dans les Forêts Aléatoires**

1. **Finalisation des Prédictions** :
   * Chaque feuille dans un arbre de décision représente une décision finale basée sur les conditions (les tests dans les nœuds précédents) menant à cette feuille. Les feuilles stockent généralement la classe majoritaire parmi les échantillons de données qui atteignent cette feuille.
2. **Agrégation des Prédictions** :
   * Dans une forêt aléatoire, pour une classification, la prédiction finale pour une observation donnée est déterminée par un vote majoritaire des prédictions de toutes les feuilles correspondantes de différents arbres (chaque arbre donne un "vote" basé sur sa prédiction).

Dans votre projet, les "feuilles" de chaque arbre dans la forêt aléatoire servent à faire des prédictions précises sur l'état des moteurs d'hélicoptères, en utilisant les informations fournies par les capteurs. L'objectif est d'anticiper les besoins de maintenance avant que des problèmes graves ne surviennent, optimisant ainsi les opérations de maintenance et la durabilité des moteurs.

**Utilité des Forêts Aléatoires**

1. **Robustesse** : Les forêts aléatoires sont moins susceptibles de surapprendre par rapport aux arbres de décision individuels car elles agrègent les résultats de nombreux arbres, ce qui réduit la variance sans augmenter le biais.
2. **Précision** : En combinant plusieurs arbres, cet algorithme tend à offrir une précision de prédiction élevée, même pour des ensembles de données complexes avec de nombreuses variables.
3. **Gestion des Données Manquantes** : Les forêts aléatoires peuvent gérer les valeurs manquantes dans les données et maintenir une bonne précision même quand une partie des données est manquante.
4. **Importance des Caractéristiques** : Elles permettent de mesurer l'importance relative de chaque caractéristique dans la prédiction, ce qui peut fournir des insights précieux pour la compréhension du problème.

**Finalité de Prédiction**

* **But** : Prédire la sortie (classe pour classification, valeur pour régression) pour de nouvelles données basées sur les apprentissages des données historiques.
* **Base** : Les prédictions se fondent sur les modèles appris de l'ensemble des arbres, où chaque arbre a été entraîné sur un sous-ensemble aléatoire des données avec des caractéristiques sélectionnées aléatoirement. Cela assure diversité et généralisation.
* **Aide à la Décision** : En fournissant des prédictions précises et des mesures de l'importance des différentes variables, les forêts aléatoires aident les décideurs à comprendre les facteurs influençant les résultats et à prendre des décisions éclairées.

La décision basée sur une forêt aléatoire s'appuie généralement sur deux résultats principaux : les prédictions de classe (ou de valeur, dans le cas d'une régression) et les probabilités associées à ces prédictions. Voyons comment ces éléments sont utilisés pour prendre des décisions concrètes et pratiques, en utilisant l'exemple de la maintenance prédictive des moteurs d'hélicoptères.

**Résultats Concrets d'une Forêt Aléatoire**

1. **Prédictions de Classe** :
   * Dans un contexte de classification, la forêt aléatoire prédit la classe de chaque entrée (par exemple, "Maintenance requise" ou "Pas de maintenance requise").
   * La décision de maintenance peut être directement basée sur cette prédiction.
2. **Probabilités de Classe** :
   * Pour chaque prédiction, la forêt aléatoire peut également fournir la probabilité de chaque classe. Cela signifie combien le modèle est sûr de sa prédiction.
   * Par exemple, pour un moteur donné, la forêt aléatoire pourrait prédire "Maintenance requise" avec une probabilité de 80%.

**Utilisation des Probabilités pour la Prise de Décision**

* **Seuil de Décision** :
  + Vous pouvez définir un seuil de probabilité pour déclencher une action. Par exemple, si la probabilité que le moteur nécessite une maintenance est supérieure à 70%, alors une maintenance est programmée.
  + Ce seuil peut être ajusté en fonction de la tolérance au risque, des coûts de maintenance inutile, ou de la gravité potentielle d'une défaillance du moteur.

**Visualisation Simple et Concrète pour la Décision**

1. **Graphiques de Probabilité** :
   * Des graphiques peuvent être créés pour montrer la probabilité de maintenance nécessaire pour chaque moteur. Ces graphiques aident les gestionnaires à visualiser rapidement quels moteurs sont à risque.
2. **Tableaux de Bord** :
   * Un tableau de bord peut afficher à la fois les prédictions et les probabilités pour chaque moteur, permettant aux utilisateurs de voir facilement quels moteurs requièrent une attention immédiate.

**Forêts Aléatoires pour la Régression**

**Fonctionnement**

Les forêts aléatoires en régression fonctionnent sur le même principe que pour la classification, mais au lieu de voter pour la classe la plus fréquente, elles calculent la moyenne des prédictions de tous les arbres pour une donnée d'entrée pour prédire une valeur continue.

**Fonctionnement des Forêts Aléatoires en Régression**

1. **Construction de l'Arbre** :
   * **Échantillonnage** : Comme en classification, les arbres sont construits à partir de sous-ensembles aléatoires de données (échantillonnage bootstrap).
   * **Sélection de Caractéristiques** : À chaque nœud de l'arbre, un nombre limité de caractéristiques est choisi aléatoirement pour décider de la meilleure division, similaire à la classification.
2. **Division des Nœuds** :
   * En régression, la division à chaque nœud de l'arbre est déterminée de manière à minimiser la variance ou l'erreur quadratique moyenne entre les valeurs réelles et les valeurs prédites par le nœud.
3. **Prédiction Finale** :
   * **Moyenne des Prédictions** : La prédiction d'une forêt aléatoire en régression pour une entrée donnée est la moyenne des prédictions de tous les arbres de la forêt pour cette entrée. Cela est différent de la classification, où c'est le vote majoritaire qui détermine la classe prédite.

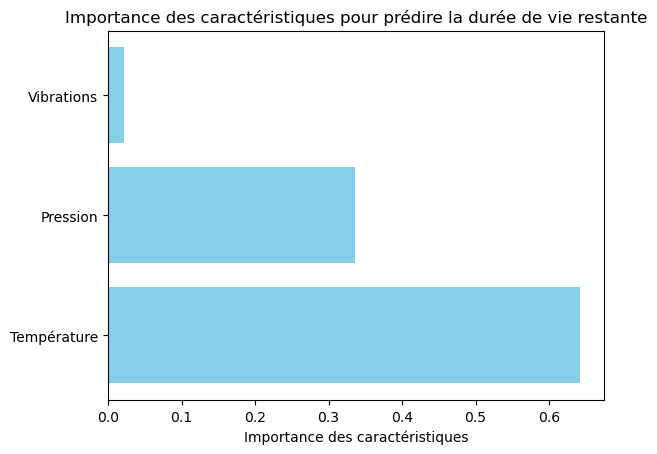
**Différences entre Régression et Classification avec Forêts Aléatoires**

* **Type de Sortie** :
  + **Classification** : Les sorties sont des étiquettes de classe.
  + **Régression** : Les sorties sont des valeurs continues.
* **Mesure de Pureté (ou Impureté)** :
  + **Classification** : Utilise des mesures comme l'indice de Gini ou l'entropie pour évaluer la qualité des divisions.
  + **Régression** : Utilise la variance ou l'erreur quadratique moyenne pour évaluer les divisions.
* **Prédiction Finale** :
  + **Classification** : Basée sur le mode (la classe la plus fréquente parmi les prédictions des arbres individuels).
  + **Régression** : Basée sur la moyenne des prédictions des arbres.

Rmq :

Pour adapter le concept des forêts aléatoires à la régression dans un contexte aéronautique, supposons que nous souhaitons prédire la durée de vie restante (RUL - Remaining Useful Life) d'un moteur d'hélicoptère en fonction des données opérationnelles telles que les heures de fonctionnement, la température, la pression et les vibrations. Ce genre d'analyse peut aider à planifier la maintenance préventive et à éviter des pannes inopinées.

Imaginons que nous voulons prédire le nombre d'heures avant la prochaine maintenance nécessaire (un problème de régression), basé sur les mêmes caractéristiques (température, pression, vibrations)



L'importance des caractéristiques dans les modèles de forêts aléatoires, qu'ils soient utilisés pour la classification ou la régression, est calculée de manière similaire. Cependant, la manière dont ces importances sont interprétées et les implications pour chaque type de modèle peuvent différer. Explorons comment ces importances sont calculées et pourquoi elles peuvent apparaître différentes selon que le modèle est de classification ou de régression.

**Calcul de l'Importance des Caractéristiques**

L'importance des caractéristiques dans une forêt aléatoire est généralement mesurée en examinant combien chaque caractéristique contribue à réduire l'impureté (pour la classification) ou la variance (pour la régression) dans les arbres de la forêt. Voici les étapes détaillées de ce calcul :

1. **Réduction de l'Impureté ou de la Variance** :
   * Pour chaque arbre de la forêt, le modèle calcule la réduction de l'impureté (indice de Gini ou entropie pour la classification) ou de la variance (pour la régression) apportée par chaque caractéristique à chaque nœud où cette caractéristique est utilisée pour diviser le jeu de données.
   * Cette réduction est pondérée par le nombre de points de données qui passent par le nœud pour tenir compte de l'influence de chaque nœud dans l'ensemble du modèle.
2. **Agrégation sur Tous les Arbres** :
   * Les réductions de l'impureté ou de la variance pour chaque caractéristique sont ensuite agrégées sur tous les arbres de la forêt pour donner une mesure moyenne de l'importance de cette caractéristique.
3. **Normalisation** :
   * Les valeurs d'importance calculées sont ensuite normalisées en divisant chaque importance par la somme de toutes les importances, ce qui garantit que la somme de toutes les importances dans le modèle est égale à 1.

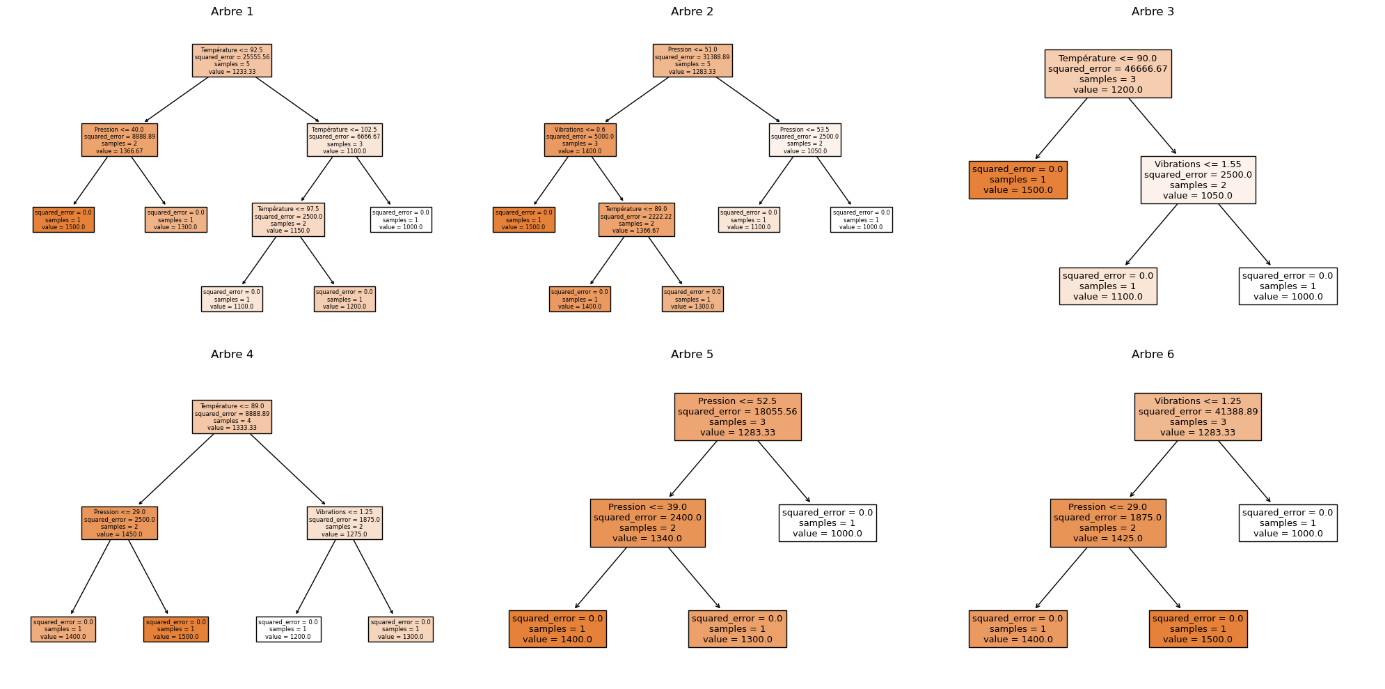
**Pourquoi les Importances Peuvent Différer entre Classification et Régression**

* **Nature des Sorties** :
  + En classification, les caractéristiques qui sont les meilleures pour séparer les classes distinctement auront une importance plus élevée, car elles réduisent l'impureté efficacement.
  + En régression, les caractéristiques qui contribuent le plus à réduire la variance des prédictions autour de la moyenne seront jugées plus importantes.
* **Sensibilité aux Échelles** :
  + En régression, l'importance peut être plus sensible aux échelles des caractéristiques. Par exemple, des caractéristiques avec une plus grande amplitude de variation peuvent souvent apparaître comme plus importantes, car de petites variations dans ces caractéristiques peuvent avoir de grands effets sur la sortie.
* **Différences dans les Distributions des Caractéristiques** :
  + La manière dont les caractéristiques sont distribuées et leur corrélation avec la variable cible peuvent influencer leur importance. En classification, des discontinuités claires dans les caractéristiques qui alignent bien avec les frontières de classe ressortiront comme importantes. En régression, des caractéristiques avec des relations linéaires ou non linéaires fortes avec la variable cible seront marquées comme importantes.

**Conclusion**

L'importance des caractéristiques fournit une perspective précieuse sur quelles entrées sont les plus influentes pour les prédictions du modèle. En régression, cela peut aider à identifier les facteurs qui ont le plus grand impact sur la quantité prévue, tandis qu'en classification, cela aide à comprendre quelles caractéristiques sont les plus utiles pour distinguer entre les classes. Ces informations sont cruciales pour affiner les modèles, pour la sélection des caractéristiques, et pour guider des décisions techniques et opérationnelles basées sur le modèle.

Chaque arbre effectue des divisions basées sur les valeurs des caractéristiques (température, pression, vibrations) pour prédire une valeur continue, en l'occurrence la durée de vie restante d'un moteur d'hélicoptère. Examinons de plus près les éléments d'un arbre pour comprendre comment ils contribuent à la prédiction finale.



Une image contenant texte, diagramme, ligne, Plan

Description générée automatiquement

Examinons en détail l'Arbre 1 que vous avez partagé :

**Structure de l'Arbre**

1. **Nœud Racine** :
   * **Critère** : Température <= 92.5
   * **Erreur au Carré** : 25555.56
   * **Échantillons** : 5
   * **Valeur Prédite** : 1233.33 heures
   * Ce nœud racine divise l'ensemble des données en deux groupes basés sur la température. Les moteurs avec une température inférieure ou égale à 92.5°C vont à gauche, et ceux avec une température supérieure vont à droite.
2. **Nœud de Gauche** :
   * **Critère** : Pression <= 40.0
   * **Erreur au Carré** : 8888.89
   * **Échantillons** : 2
   * **Valeur Prédite** : 1366.67 heures
   * Ce nœud subdivise le groupe de gauche en deux basé sur la pression.
3. **Feuilles de Gauche** :
   * **Erreur au Carré** : 0.0 pour les deux feuilles
   * **Échantillons** : 1 pour chaque feuille
   * **Valeurs Prédites** : 1500 et 1300 heures
   * Chaque feuille représente un cas où tous les échantillons restants après les divisions antérieures sont parfaitement prédits par la valeur moyenne de la feuille (aucune erreur).
4. **Nœud de Droite** :
   * **Critère** : Température <= 102.5
   * **Erreur au Carré** : 6666.67
   * **Échantillons** : 3
   * **Valeur Prédite** : 1100.0 heures
   * Ce nœud gère les moteurs de température supérieure à 92.5°C, et les divise à nouveau en fonction de la température.
5. **Feuilles de Droite** :
   * **Feuille Gauche** : Température <= 97.5, erreur 2500.0, 2 échantillons, valeur 1150.0 heures
   * **Feuille Droite** : Température > 97.5, erreur 0.0, 1 échantillon, valeur 1000.0 heures
   * Ces feuilles représentent des cas spécifiques où les moteurs sont classés selon des températures plus élevées, avec des valeurs prédites ajustées en fonction de divisions supplémentaires.

**Comment l'Arbre Fait des Prédictions**

Chaque chemin de la racine à une feuille représente une série de décisions basées sur les valeurs des caractéristiques (température, pression, vibrations). Lorsque vous voulez prédire la durée de vie restante d'un nouveau moteur, vous commencez par la racine et suivez le chemin dicté par les valeurs de ses caractéristiques jusqu'à ce que vous atteigniez une feuille. La valeur prédite à cette feuille est l'estimation de la durée de vie restante du moteur.

**Visualisation et Interprétation**

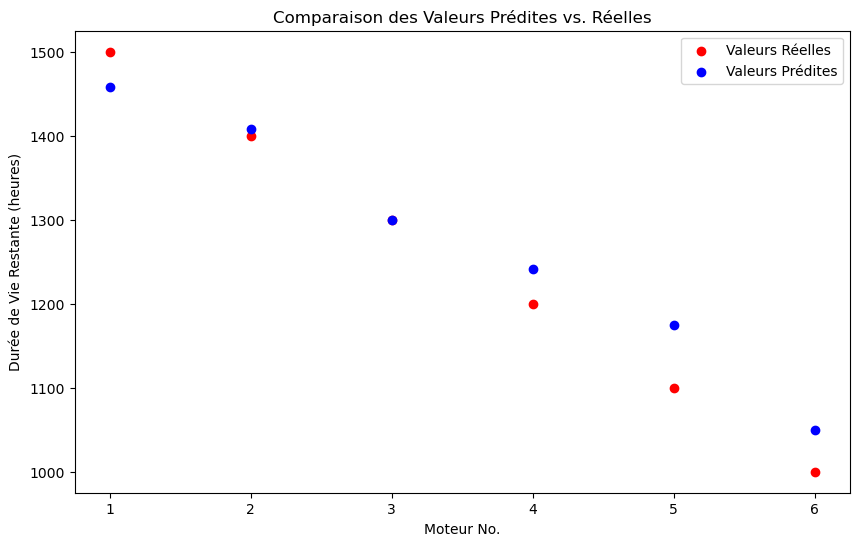
* **Critères de Division** : Chaque division est choisie pour réduire au maximum l'erreur quadratique moyenne entre les valeurs prédites et réelles des échantillons à ce nœud.
* **Erreur au Carré** : Une mesure de la performance du modèle à ce nœud. Une erreur de 0 signifie que tous les échantillons à ce nœud ont exactement la valeur prédite.
* **Échantillons** : Le nombre d'échantillons affectés à chaque nœud donne une idée de la quantité de données soutenant la prédiction à ce stade.

Cette approche permet de comprendre les facteurs influençant la durée de vie des moteurs et aide à ajuster les interventions de maintenance pour améliorer la performance et la fiabilité des équipements.

Dans le cas d'un modèle de régression utilisant les forêts aléatoires pour prédire des valeurs continues, telles que la durée de vie restante d'un moteur d'hélicoptère, les visualisations typiquement utilisées sont différentes de celles utilisées en classification (comme les probabilités de classe). Pour soutenir la prise de décision basée sur les prédictions d'un modèle de régression, les types de visualisations suivants sont souvent utilisés :

**1. Graphique de Prédictions vs. Valeurs Réelles**

Ce graphique permet de comparer directement les valeurs prédites par le modèle aux valeurs réelles observées. C'est un moyen efficace de visualiser la précision des prédictions du modèle.



**Analyse du Graphique**

1. **Points Bleus** : Représentent les valeurs prédites par le modèle pour la durée de vie restante de chaque moteur.
2. **Points Rouges** : Représentent les valeurs réelles de la durée de vie restante pour chaque moteur.

**Observations Clés**

* **Correspondance Générale** : On observe que, pour la majorité des moteurs (moteurs No. 1, 2, 3, 4, et 6), les prédictions semblent assez proches des valeurs réelles. Cela indique que le modèle a une bonne performance générale en termes de prédiction des durées de vie restantes.
* **Discrepances** :
  + **Moteur No. 3** : Il y a une légère divergence entre la valeur prédite et la valeur réelle, où le modèle sous-estime la durée de vie restante.
  + **Moteur No. 5** : On observe une plus grande divergence entre la prédiction et la valeur réelle. Cette divergence peut indiquer un problème spécifique avec les caractéristiques de ce moteur que le modèle n'a pas bien capturé ou un exemple de données où le modèle pourrait être amélioré.

**Interprétation**

* **Précision du Modèle** : En général, le modèle semble performant pour prédire la durée de vie restante des moteurs. La majorité des points prédits sont relativement proches de leurs homologues réels, indiquant que le modèle capture bien les tendances dans les données.
* **Potentiels d'Amélioration** :
  + Pour les moteurs où les prédictions ne correspondent pas bien, comme le moteur No. 5, il pourrait être utile de revoir les données d'entraînement pour ces cas spécifiques pour s'assurer qu'ils sont bien représentés ou de considérer des modèles plus complexes qui pourraient capter des nuances supplémentaires dans les données.
  + L'examen des caractéristiques utilisées pour ces moteurs peut également révéler si des informations supplémentaires sont nécessaires pour améliorer les prédictions.

**Conclusion**

Ce graphique est un outil efficace pour visualiser et évaluer la performance du modèle de régression. Il permet aux ingénieurs et aux décideurs de vérifier rapidement la précision des prédictions du modèle et d'identifier les cas où des ajustements ou des investigations supplémentaires peuvent être nécessaires. En fin de compte, ce type de visualisation aide à renforcer la confiance dans le modèle ou à identifier des opportunités d'amélioration continue.

**2. Graphique des Résidus**

Un graphique des résidus montre la différence entre les valeurs prédites et les valeurs réelles pour chaque prédiction. Cela peut aider à identifier des modèles d'erreur ou des tendances qui pourraient indiquer des problèmes avec le modèle.

Une image contenant texte, capture d’écran, ligne, diagramme

Description générée automatiquement

**Analyse du Graphique des Résidus**

1. **Axes** :
   * **Axe des X** : Valeurs prédites par le modèle.
   * **Axe des Y** : Résidus (différences entre valeurs prédites et réelles).
2. **Points** :
   * Chaque point représente un moteur, où la position horizontale du point indique la valeur prédite de la durée de vie restante du moteur, et la position verticale (le résidu) indique combien cette prédiction était éloignée de la valeur réelle.
3. **Ligne Rouge Pointillée** :
   * La ligne rouge pointillée indique le niveau zéro des résidus. Les points exactement sur cette ligne sont des prédictions parfaites sans erreur.

**Observations Clés**

* **Distribution des Résidus** :
  + La plupart des points sont au-dessus de la ligne zéro, ce qui indique que le modèle a tendance à sous-estimer la durée de vie restante des moteurs. Cela pourrait être dû à une tendance du modèle à prédire des valeurs plus conservatrices ou à un biais dans les données d'entraînement ou la méthode de modélisation.
* **Variabilité des Résidus** :
  + Les résidus montrent une certaine variabilité. Par exemple, certains points sont très proches de la ligne zéro, indiquant des prédictions très précises, tandis que d'autres sont assez éloignés, indiquant des erreurs plus importantes.
  + Le point le plus élevé (plus grande valeur positive de résidu) suggère une sous-estimation significative par le modèle.

**Implications et Actions Possibles**

* **Amélioration du Modèle** :
  + Examiner les cas où les résidus sont les plus élevés pour comprendre si certaines caractéristiques ou interactions spécifiques n'ont pas été adéquatement capturées par le modèle.
  + Considérer l'ajout de données supplémentaires, l'ajustement des paramètres du modèle, ou l'utilisation de techniques de modélisation plus complexes pour mieux capturer la complexité des données.
* **Évaluation de la Robustesse du Modèle** :
  + Réaliser des analyses de robustesse pour voir comment le modèle performe dans différents sous-ensembles de données ou dans des conditions de fonctionnement variées.
  + Utiliser des techniques de validation croisée pour évaluer la stabilité des prédictions du modèle sur différentes partitions des données.

**Conclusion**

Le graphique des résidus est un outil diagnostique essentiel en régression. Il aide à identifier les problèmes de précision des prédictions, à comprendre le comportement du modèle, et à guider les améliorations futures. En identifiant où et comment les prédictions dévient de la réalité, les ingénieurs et les analystes peuvent travailler à améliorer la fiabilité et l'exactitude des modèles prédictifs.

**Machines à Vecteurs de Support (SVM) : Fonctionnement et Application**

Les Machines à Vecteurs de Support (SVM) sont un type d'algorithme d'apprentissage supervisé très efficace pour la classification (et la régression avec le SVR - Support Vector Regression). La capacité des SVM à maximiser la marge entre les classes les rend particulièrement efficaces dans de nombreux contextes où la séparation claire des classes est importante.

**Fonctionnement de SVM**

1. **Hyperplan de Séparation** :
   * SVM fonctionne en trouvant l'hyperplan qui sépare le mieux les classes de données dans l'espace de caractéristiques. L'hyperplan est essentiellement une décision limite qui cherche à maximiser la distance (marge) entre les classes de données les plus proches de cet hyperplan (ces données sont appelées vecteurs de support).
2. **Maximisation de la Marge** :
   * La marge est l'espace entre les deux lignes parallèles aux côtés opposés de l'hyperplan. Les SVM optimisent cette marge pour améliorer la généralisation du modèle.
3. **Noyaux** :
   * Dans les cas où les données ne sont pas linéairement séparables, SVM utilise des fonctions noyau pour transformer les données dans un espace de dimensions supérieures où elles peuvent être séparées linéairement. Les noyaux communs incluent linéaire, polynomial, RBF (Radial Basis Function), et sigmoïde.

Une image contenant diagramme, ligne, texte, Tracé

Description générée automatiquement

**Application : Classification des États Opérationnels des Moteurs**

* **Classification Binaire** :
  + Par exemple, déterminer si un moteur est en état de fonctionner normalement ou s'il nécessite une maintenance. Les SVM peuvent classifier ces états en apprenant à partir des données des capteurs (température, pression, vibrations, etc.).
* **Multiclasse** :
  + Si plusieurs états opérationnels doivent être identifiés (normal, attention, critique), les SVM peuvent être étendus pour gérer plusieurs classes en utilisant des approches telles que one-vs-one ou one-vs-all.

Supposons que nous classifions les moteurs en fonction de leur nécessité de maintenance en utilisant les caractéristiques de température, pression, et vibrations :

**Division des données**

la division des données en ensembles d'entraînement et de test est une pratique courante et essentielle dans l'apprentissage automatique pour tous les algorithmes de machine learning, y compris les forêts aléatoires, la régression linéaire et la régression logistique. La raison pour laquelle cette division est si importante est qu'elle permet de tester le modèle sur des données qu'il n'a pas vues pendant l'entraînement, ce qui fournit une évaluation plus impartiale de ses performances.

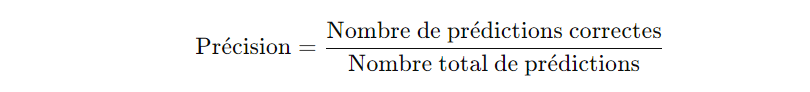
**C’est quoi X\_test ?**

X\_test est un sous-ensemble de l'ensemble des données d'origine X, utilisé spécifiquement pour évaluer la performance du modèle. Les données de test sont sélectionnées de manière aléatoire (ou en utilisant un seed pour reproductibilité) et ne sont pas utilisées pendant l'entraînement du modèle. Cela permet de tester le modèle sur des données qu'il n'a jamais vues auparavant, donnant ainsi une évaluation plus juste de ses performances.

**Standardisation des Données**

Lors de la standardisation avec StandardScaler, chaque caractéristique de X est transformée en soustrayant la moyenne de cette caractéristique et en la divisant par son écart type. Cela signifie que les valeurs originales sont transformées pour avoir une distribution centrée autour de 0 avec un écart type de 1.

La précision (accuracy) d'un modèle de machine learning est une mesure de la proportion de prédictions correctes effectuées par le modèle par rapport à l'ensemble total des prédictions. Elle est calculée comme suit :

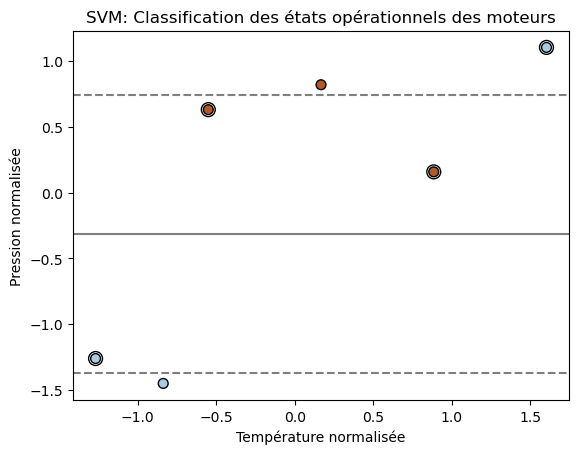


**Causes Potentielles d'une Précision de 0.0**

1. **Modèle Mal Entraîné** :
   * Le modèle n'a pas suffisamment appris des données d'entraînement, peut-être en raison d'un surapprentissage (overfitting) ou sous-apprentissage (underfitting).
2. **Problème avec les Données d'Entraînement** :
   * Les données d'entraînement pourraient être déséquilibrées ou ne pas représenter adéquatement les cas présents dans les données de test.
3. **Prétraitement des Données** :
   * Des erreurs dans le prétraitement des données, telles que des caractéristiques mal standardisées ou des données mal étiquetées, peuvent affecter les performances du modèle.
4. **Hyperparamètres** :
   * Les hyperparamètres du modèle, comme le type de noyau pour un SVM ou la régularisation, pourraient ne pas être optimaux.

**Comment Améliorer la Précision**

1. **Réévaluer les Données** :
   * Vérifiez que les données sont correctement étiquetées et qu'elles représentent bien le problème que vous essayez de résoudre.
2. **Changer les Hyperparamètres** :
   * Essayez d'ajuster les hyperparamètres de votre modèle pour améliorer les performances.
3. **Améliorer l'Ensemble de Données** :
   * Ajoutez plus de données d'entraînement ou utilisez des techniques de rééchantillonnage pour équilibrer les classes.
4. **Choisir un Autre Modèle** :
   * Essayez d'autres algorithmes de machine learning pour voir si un autre modèle donne de meilleurs résultats pour votre problème spécifique.



Le graphique ci-dessus illustre la classification des états opérationnels des moteurs utilisant un modèle SVM avec un noyau linéaire. Les éléments clés de la visualisation incluent :

1. **Données Classifiées** :
   * Les points colorés (bleu et rouge) représentent les moteurs, colorés selon leur état (normal ou nécessitant une maintenance). Les points sont positionnés selon les valeurs normalisées de la température et de la pression.
2. **Hyperplan de Décision** :
   * La ligne droite (en noir) au milieu représente l'hyperplan qui sépare les deux classes. C'est la frontière de décision que le SVM a apprise pour distinguer entre les états opérationnels.
3. **Marges** :
   * Les lignes pointillées de chaque côté de l'hyperplan représentent les marges. Elles montrent la distance minimale entre l'hyperplan et les points de données les plus proches de chaque classe, connus sous le nom de vecteurs de support.
4. **Vecteurs de Support** :
   * Les points entourés d'un cercle noir sont les vecteurs de support. Ce sont les points les plus proches de l'hyperplan, et ils jouent un rôle crucial dans la définition de l'hyperplan et des marges.

Ce graphique aide à visualiser comment le modèle SVM linéaire classifie les données et quelles données influencent le plus la frontière de décision. Cela permet de comprendre visuellement la séparation entre les classes et l'efficacité du modèle SVM pour ce scénario spécifique.

**Interprétation des Résultats**

* **Séparation Efficace** :
  + Le modèle a efficacement trouvé un hyperplan qui sépare les classes avec une marge maximale. Cela suggère que le modèle SVM linéaire est approprié pour ces données, bien que les données soient assez simples et linéairement séparables.
* **Classification Potentielle** :
  + Les moteurs classés dans la région au-dessus de l'hyperplan sont identifiés comme nécessitant une maintenance (points rouges), tandis que ceux en dessous sont considérés comme normaux (points bleus). Cette classification correspond bien à l'étiquetage des données.
* **Stabilité et Robustesse** :
  + La présence de marges équitables et le positionnement correct des vecteurs de support indiquent que le modèle est non seulement bien ajusté, mais aussi potentiellement robuste contre de légères variations dans les données d'entrée.

**Implications et Recommandations**

* **Évaluation de Nouvelles Données** :
  + Tester le modèle avec de nouvelles données de moteurs serait une étape essentielle pour évaluer sa performance en conditions réelles et sa capacité à généraliser au-delà de l'ensemble de données actuel.
* **Exploration de Modèles Plus Complexes** :
  + Si des données futures montrent des relations non linéaires plus complexes entre les caractéristiques et les classes, envisager l'utilisation de noyaux non linéaires (comme RBF ou polynomial) pour améliorer la capacité du modèle à capturer ces complexités.

Ce graphique est un excellent outil pour visualiser et comprendre la manière dont le modèle SVM fonctionne avec les données spécifiques du projet, et il fournit une base solide pour prendre des décisions informées sur la maintenance des moteurs basée sur les prédictions du modèle.

**K-Plus Proches Voisins (K-NN) : Fonctionnement et Application**

Le K-Plus Proches Voisins (K-NN) est un algorithme d'apprentissage supervisé simple et intuitif utilisé pour la classification et parfois pour la régression. Il est particulièrement efficace pour des scénarios où la décision de classification peut être prise en fonction de la similitude des caractéristiques de l'instance à classer avec celles des instances déjà classées.

**Fonctionnement de K-NN**

1. **Choix de K** :
   * Le paramètre 'k' représente le nombre de voisins les plus proches à considérer. Il est crucial de choisir une valeur optimale pour 'k', car un nombre trop faible peut rendre l'algorithme trop sensible au bruit, tandis qu'un nombre trop élevé peut le rendre insensible aux réelles structures des données.
2. **Mesure de Distance** :
   * K-NN fonctionne en calculant la distance entre l'instance à classer et toutes les autres dans l'ensemble d'apprentissage. Les distances couramment utilisées incluent la distance euclidienne, la distance de Manhattan, et la distance de Minkowski.
3. **Vote Majoritaire** :
   * Pour chaque nouvelle observation, K-NN identifie les 'k' échantillons les plus proches dans les données d'apprentissage, puis assigne la classe la plus fréquente parmi ces k voisins comme la prédiction pour l'observation.

**Application : Diagnostics Rapides des Moteurs**

* **Comparaison des Conditions Actuelles aux Historiques** :
  + K-NN peut être utilisé pour comparer rapidement les conditions actuelles d'un moteur (par exemple, température, pression, vibrations) avec des enregistrements historiques pour diagnostiquer si un moteur fonctionne normalement ou si des anomalies indiquent un besoin de maintenance.
* **Avantages pour le Diagnostic** :
  + **Simplicité** : Pas de phase d'apprentissage explicite, ce qui est idéal pour les systèmes où les données arrivent en continu et où les modèles doivent être fréquemment mis à jour.
  + **Efficacité** : Permet des diagnostics rapides en comparant de nouvelles mesures avec des historiques spécifiques.

**Paramètres Importants**

* **n\_neighbors=k** : Ce paramètre spécifie le nombre de voisins à considérer pour le vote majoritaire. Dans cet exemple, k indique que l'algorithme doit regarder les trois échantillons les plus proches de la nouvelle observation pour déterminer sa classe.

**Fonctionnement de K-NN avec n\_neighbors=k**

1. **Calcul de Distance** :
   * Pour chaque nouvelle observation que vous souhaitez classifier, K-NN calcule la distance entre cette observation et toutes les autres observations dans l'ensemble d'apprentissage. Ces distances peuvent être calculées de différentes manières (euclidienne, Manhattan, etc.), mais la distance euclidienne est la plus couramment utilisée par défaut.
2. **Identification des Plus Proches Voisins** :
   * Après avoir calculé ces distances, l'algorithme identifie les trois observations les plus proches (dans cet exemple, car n\_neighbors=k).
3. **Vote Majoritaire** :
   * Une fois les trois voisins les plus proches identifiés, leurs étiquettes de classe sont consultées pour procéder à un vote majoritaire. La classe qui apparaît le plus souvent parmi les trois voisins est assignée à la nouvelle observation.

**Implications de n\_neighbors=k**

* **Sensibilité au Bruit** : Utiliser un petit nombre de voisins (comme k) peut rendre l'algorithme plus sensible au bruit et aux valeurs aberrantes, car la classification d'une observation est influencée par un très petit nombre de voisins proches.

**Choix de k**

Le choix de k est crucial et peut dépendre de plusieurs facteurs, y compris :

* La distribution des classes dans les données.
* La présence de bruit et de valeurs aberrantes.
* La taille de l'ensemble d'apprentissage (un k plus grand peut être nécessaire avec un grand ensemble de données pour éviter un surajustement)

On veut visualiser l'efficacité de l'algorithme K-Plus Proches Voisins (K-NN) et comprendre comment il classe les données, Voici un graphique montrant les données, les frontières de décision et les voisins les plus proches pour chaque point de test. Cette visualisation aide à comprendre les régions d'influence de chaque classe et à voir comment les décisions sont prises pour de nouveaux points.

Une image contenant texte, capture d’écran, diagramme, pente

Description générée automatiquement

**Cohérence avec les 'k' Voisins** :

* Les régions de couleur doivent correspondre logiquement aux 'k' voisins les plus proches de tout point hypothétique dans cette région. Par exemple, dans une région bleue, les trois voisins les plus proches (si k=3) de tout point devraient majoritairement être bleus.

**Analyse du Graphique**

Sur la base de votre description, voici quelques points d'analyse spécifiques :

* **Distribution et Classification** :
  + Il semble que les points soient correctement classifiés selon leurs régions. Cela indique une bonne performance du modèle sur cet ensemble de données spécifique.
* **Frontières de Décision** :
  + La frontière semble complexe et peut-être irrégulière, ce qui est typique dans les classifications K-NN surtout avec un petit 'k'. Cela montre que la frontière est fortement influencée par la disposition locale des points.
* **Possibles Anomalies ou Erreurs de Classification** :
  + S'il y a des points rouges dans la zone bleue ou vice versa, cela pourrait indiquer des erreurs de classification ou des anomalies dans les données. Cela peut aussi simplement refléter la nature de K-NN où les frontières ne sont pas toujours parfaites ou lisses.

**Conclusion**

Si le graphique montre une classification logique et que les frontières semblent respecter la règle des 'k' voisins les plus proches, alors il est probablement correct. Assurez-vous que la visualisation aide à comprendre la classification et à identifier les zones où des ajustements pourraient être nécessaires, soit dans le choix de 'k', soit dans le traitement des données pour améliorer la performance du modèle.

**Qu'est-ce qu'un Réseau de Neurones ?**

Un réseau de neurones est une structure composée de plusieurs couches de neurones, où chaque couche effectue des transformations sur les données d'entrée pour les transformer en une sortie.

**Les Couches dans un Réseau de Neurones**

1. **Couches (Layers) :**
   * Une **couche** est une structure fondamentale dans un réseau de neurones. Elle prend des entrées, effectue des calculs dessus et produit des sorties.
   * Les couches sont empilées les unes sur les autres dans un modèle séquentiel.
2. **Couche Dense (Dense Layer) :**
   * Une **couche Dense** est une couche de neurones complètement connectés, ce qui signifie que chaque neurone de cette couche est connecté à chaque neurone de la couche précédente.
   * Par exemple, Dense(1) signifie une couche Dense avec un seul neurone.
3. **Couche LSTM :**
   * Une **couche LSTM (Long Short-Term Memory)** est une couche spéciale utilisée pour traiter les séquences temporelles et retenir des informations sur de longues périodes.
   * Par exemple, LSTM(50) signifie une couche LSTM avec 50 unités de mémoire, ce qui permet de retenir les informations sur 50 neurones.

**Création et Compilation du Modèle**

Un modèle de réseau de neurones est une séquence de couches. Voici comment nous construisons et compilons un modèle.

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

model = Sequential()  # Initialise un modèle séquentiel

model.add(LSTM(50, activation='relu', input\_shape=(1, 1)))  # Ajoute une couche LSTM

model.add(Dense(1))  # Ajoute une couche Dense avec 1 neurone de sortie

 **Sequential()** : Crée un modèle séquentiel où les couches sont ajoutées les unes après les autres.

 **LSTM(50, activation='relu', input\_shape=(1, 1))** :

* 50 : Le nombre d'unités LSTM dans la couche. Chaque unité peut retenir une information sur les séquences temporelles.
* activation='relu' : Fonction d'activation ReLU (Rectified Linear Unit) pour introduire la non-linéarité.
* input\_shape=(1, 1) : La forme des données d'entrée, ici 1 pas de temps et 1 caractéristique.

 **Dense(1)** : Ajoute une couche Dense avec un seul neurone, ce qui produit une seule valeur numérique en sortie.

2. **Compilation du Modèle**

 **optimizer='adam'** : Adam est un optimiseur utilisé pour ajuster les poids du réseau. Il combine les avantages de deux autres méthodes d'optimisation, AdaGrad et RMSProp, pour obtenir une convergence plus rapide.

 **loss='mse'** : La fonction de perte MSE (Mean Squared Error) mesure l'erreur quadratique moyenne entre les valeurs prédites et les valeurs réelles. Elle est utilisée comme critère pour ajuster les poids afin de minimiser cette erreur.

**Reshape des Données pour LSTM**

Les LSTM nécessitent que les données soient formatées de manière spécifique : [samples, time steps, features].

**Exemple de Reshape**

data = np.array([[85], [88], [90], [93], [95], [97]])

data = data.reshape((data.shape[0], 1, data.shape[1]))

* data.shape[0] : Nombre de samples (échantillons) dans les données.
* 1 : Nombre de time steps (pas de temps) par échantillon.
* data.shape[1] : Nombre de features (caractéristiques) par pas de temps.

**Fonctionnement du Réseau de Neurones**

1. **Entrée** : La température à un certain moment.
2. **Couche LSTM** : Traite la séquence d'entrée et retient des informations grâce à ses unités de mémoire.
3. **Couche Dense** : Prend la sortie de la couche LSTM et produit une prédiction (une valeur numérique).
4. **Optimisation** : L'optimiseur Adam ajuste les poids du réseau pour minimiser l'erreur mesurée par la fonction de perte MSE.

**Pourquoi ces Choix ?**

* **LSTM** : Utilisé pour sa capacité à retenir des informations temporelles sur de longues séquences.
* **Dense Layer** : Utilisé pour produire la valeur prédite finale.
* **Optimiseur Adam** : Choisi pour sa capacité à converger rapidement et efficacement.
* **Fonction de Perte MSE** : Choisie car elle mesure bien l'erreur pour les problèmes de régression où nous prédisons des valeurs continues.

**Conclusion**

En reformulant les données pour correspondre aux attentes des LSTM et en utilisant un modèle séquentiel avec des couches LSTM et Dense, nous pouvons entraîner le modèle à prédire des valeurs futures à partir de données séquentielles. L'optimiseur Adam et la fonction de perte MSE aident à ajuster les poids du réseau pour minimiser l'erreur et améliorer les prédictions. Cette approche permet de capturer les dépendances temporelles et les motifs récurrents dans les données, ce qui est essentiel pour les séries temporelles.

**Réseaux de Neurones et Deep Learning**

Les réseaux de neurones et le deep learning offrent des capacités avancées pour l'analyse et la prédiction basées sur des données complexes et volumineuses. Pour des applications telles que la surveillance en temps réel des conditions des moteurs et la prédiction de leur maintenance, ces techniques peuvent fournir des insights précieux qui ne sont pas facilement capturables avec des modèles plus simples ou linéaires. Leur capacité à apprendre des relations non linéaires et à modéliser des dépendances temporelles longues en fait des outils puissants pour la maintenance prédictive dans l'aéronautique.

**Fonctionnement des Réseaux de Neurones**

1. **Structure de Base** :
   * Un réseau de neurones est composé de couches de neurones artificiels. Chaque neurone reçoit des entrées, effectue une somme pondérée (y compris un biais), passe le résultat à travers une fonction d'activation (qui peut être une sigmoïde, ReLU, tanh, etc.), et envoie la sortie à la couche suivante.
2. **Propagation Avant (Forward Propagation)** :
   * Les données d'entrée sont passées à travers les couches de neurones depuis l'entrée (input layer) jusqu'à la sortie (output layer). Chaque neurone dans une couche reçoit des entrées de tous les neurones de la couche précédente, formant un réseau dense.
3. **Rétropropagation (Backpropagation)** :
   * Après la propagation avant, le réseau calcule l'erreur entre la sortie obtenue et la sortie attendue (vraie valeur). L'erreur est ensuite propagée en arrière à travers le réseau pour ajuster les poids de manière à minimiser l'erreur, en utilisant des techniques d'optimisation telles que la descente de gradient.

**Application : Analyse de Séries Temporelles des Données des Capteurs**

* **Prédiction Basée sur le Temps** :
  + Les réseaux de neurones sont particulièrement efficaces pour analyser et prédire les comportements à long terme à partir de séries temporelles, comme celles générées par des capteurs sur les moteurs. Ces modèles peuvent identifier des motifs complexes et récurrents dans les données temporelles.
* **Réseaux de Neurones Récurrents (RNN) et LSTM** :
  + Pour les séries temporelles, les Réseaux de Neurones Récurrents (RNN) et en particulier les Long Short-Term Memory networks (LSTM) sont souvent utilisés. Ils ont la capacité de retenir des informations passées (mémoire) grâce à leurs structures de boucles internes, ce qui est crucial pour comprendre le contexte temporel dans les données de capteurs.

Les points clés à retenir sont :

* **LSTM** est bien adapté pour les séries temporelles en raison de sa capacité à retenir des informations sur de longues séquences.
* **Reshape** des données est crucial pour préparer les données d'entrée dans le format attendu par les LSTM.
* **Entraînement** du modèle avec les données préparées et prédiction des valeurs futures.

Rmq : Nous utilisons un modèle LSTM pour prédire les valeurs futures de température basées sur les valeurs passées. Le modèle apprend à partir des séquences de température et utilise cette connaissance pour faire des prédictions sur les températures futures. Cette approche est particulièrement puissante pour analyser des séries temporelles où les données successives sont corrélées et où la compréhension du contexte temporel est cruciale.

**Structure du Modèle et Entraînement**

1. **Modèle LSTM** :
   * Un modèle LSTM est utilisé car il peut capturer les dépendances temporelles et les motifs récurrents dans les données à long terme dans les données séquentielles ou temporelles.

Par exemple on peut prédire la valeur future d'une série temporelle basée sur les valeurs passées. Plus précisément :

* **Données d'entrée (data)** : Les valeurs de température enregistrées à différents moments.
* **Cible (target)** : Les valeurs de température enregistrées aux moments suivants. Autrement dit, pour chaque valeur de température d'entrée, la cible est la température au prochain moment.

**Entraînement du Modèle**

* Lors de l'entraînement, nous utilisons les données d’entrée et Cible pour entraîner le modèle à prédire la/les températures futures. Le modèle apprend les relations entre les températures successives.

**Prédiction**

* Après l'entraînement, le modèle est utilisé pour prédire les températures futures basées sur les températures passées.

**Visualisation du Processus**

**Étapes Détaillées**

1. **Préparation des données** :
   * data : Températures passées.
   * target : Températures futures correspondantes.
2. **Reshape des données pour LSTM** :
   * Les données sont reformattées pour correspondre à l'attente de la couche LSTM [samples, time steps, features].
3. **Création et compilation du modèle LSTM** :
   * Une couche LSTM de 50 unités avec activation ReLU.
   * Une couche de sortie Dense avec 1 neurone.
4. **Entraînement** :
   * Les données d'entraînement sont passées au modèle pour qu'il apprenne à prédire la prochaine valeur dans la séquence.
5. **Prédiction** :
   * Le modèle est utilisé pour prédire les futures valeurs de température.

**Conclusion**

En résumé, dans cet exemple de code, nous utilisons un modèle LSTM pour prédire les valeurs futures de température basées sur les valeurs passées. Le modèle apprend à partir des séquences de température et utilise cette connaissance pour faire des prédictions sur les températures futures. Cette approche est particulièrement puissante pour analyser des séries temporelles où les données successives sont corrélées et où la compréhension du contexte temporel est importante

**Exemple Pratique**

Imaginons que vous ayez les données de température d'un moteur collectées chaque mois. Vous utilisez ces données pour entraîner un modèle à prédire la température du mois suivant en se basant sur la température actuelle.

**Données d'Entrée et Cibles**

Pour entraîner le modèle, vous structurez les données comme suit :

* **Données d'entrée (data)** : Température du mois en cours.
  + [85] pour janvier
  + [88] pour février
  + [90] pour mars
  + [93] pour avril
  + [95] pour mai
  + [97] pour juin
* **Cible (target)** : Température du mois suivant.
  + [88] pour février
  + [90] pour mars
  + [93] pour avril
  + [95] pour mai
  + [97] pour juin
  + [99] pour juillet (prédiction future)

N.B : Cette approche est applicable à toute série temporelle où les données sont collectées de manière séquentielle et où les valeurs futures dépendent des valeurs passées. le modèle utilise la température actuelle pour prédire la température future, ce qui est utile pour des applications comme la maintenance prédictive, où la prévision des conditions futures peut aider à planifier les interventions nécessaires.

**2. Reshape des Données pour LSTM**

**Pourquoi Reshape ?**

Les réseaux de neurones LSTM attendent les données d'entrée dans un format spécifique : [samples, time steps, features].

* **samples** : Nombre d'échantillons ou de séquences dans votre jeu de données.
* **time steps** : Nombre de pas de temps dans chaque séquence.
* **features** : Nombre de caractéristiques (ou de variables) à chaque pas de temps.

Dans l‘exemple sur Jupiter notebook, chaque température est une seule caractéristique observée à un seul moment (un pas de temps), donc nous avons :

* **samples** : 6 (il y a 6 observations dans les données).
* **time steps** : 1 (chaque observation est indépendante et non divisée en plusieurs pas de temps).
* **features** : 1 (chaque observation a une seule caractéristique, la température).

**Code de Reshape**

# Reshaping data for LSTM [samples, time steps, features]

data = data.reshape((data.shape[0], 1, data.shape[1]))

**3. Création et Compilation du Modèle LSTM**

**Création du Modèle LSTM**

Le modèle LSTM est créé en utilisant le modèle séquentiel Keras, ce qui signifie que les couches sont ajoutées les unes après les autres.

model = Sequential()

model.add(LSTM(50, activation='relu', input\_shape=(1, 1)))

 Sequential() : Crée un modèle séquentiel.

 LSTM(50, activation='relu', input\_shape=(1, 1)) :

* 50 : Nombre d'unités LSTM dans la couche. Cela signifie que la couche LSTM aura 50 unités de mémoire.
* activation='relu' : Fonction d'activation ReLU (Rectified Linear Unit) utilisée dans les unités LSTM pour introduire la non-linéarité.
* input\_shape=(1, 1) : La forme des données d'entrée. Ici, 1 pour les pas de temps et 1 pour les caractéristiques.

**Reshape des Données pour LSTM**

* Les LSTM nécessitent que les données soient formatées de manière spécifique : [samples, time steps, features].

data = np.array([[85], [88], [90], [93], [95], [97]])

data = data.reshape((data.shape[0], 1, data.shape[1]))

 data.shape[0] : Nombre de samples (échantillons) dans les données.

 1 : Nombre de time steps (pas de temps) par échantillon.

 data.shape[1] : Nombre de features (caractéristiques) par pas de temps.

**Fonctionnement du Réseau de Neurones**

1. **Entrée** : La température à un certain moment.
2. **Couche LSTM** : Traite la séquence d'entrée et retient des informations grâce à ses unités de mémoire.
3. **Couche Dense** : Prend la sortie de la couche LSTM et produit une prédiction (une valeur numérique).
4. **Optimisation** : L'optimiseur Adam ajuste les poids du réseau pour minimiser l'erreur mesurée par la fonction de perte MSE.

**Pourquoi ces Choix ?**

* **LSTM** : Utilisé pour sa capacité à retenir des informations temporelles sur de longues séquences.
* **Dense Layer** : Utilisé pour produire la valeur prédite finale.
* **Optimiseur Adam** : Choisi pour sa capacité à converger rapidement et efficacement.
* **Fonction de Perte MSE** : Choisie car elle mesure bien l'erreur pour les problèmes de régression où nous prédisons des valeurs continues.

**Entraînement du Modèle**

Lors de l'entraînement d'un modèle de réseau de neurones, deux paramètres importants sont les **epochs** et le **verbose**.

*  **Epochs** : Définissent le nombre de fois que le modèle verra l'ensemble complet des données d'entraînement. Un nombre plus élevé d'epochs permet au modèle d'apprendre plus longtemps, mais peut aussi entraîner un surapprentissage (overfitting).
*  **Verbose** : Contrôle la quantité de sortie (logs) que vous voyez pendant l'entraînement. verbose=0 pour aucun affichage, verbose=1 pour une barre de progression détaillée, et verbose=2 pour une ligne par epoch.

Rmq : Le surapprentissage, ou overfitting, est un phénomène en apprentissage automatique où un modèle apprend non seulement les tendances générales présentes dans les données d'entraînement, mais aussi les détails spécifiques et le bruit de ces données. Cela conduit à un modèle qui fonctionne très bien sur les données d'entraînement mais qui se généralise mal aux nouvelles données, c'est-à-dire qu'il performe mal sur les données de test ou toute autre donnée non vue.

**Modèle Sur-entraîné (Overfitting)** :

* Un modèle trop complexe (par exemple, une courbe très sinueuse) pourrait s'ajuster parfaitement aux données d'entraînement, y compris au bruit et aux anomalies.
* Résultat : excellente performance sur les données d'entraînement, mais mauvaise performance sur les données de test car le modèle n'a pas généralisé les tendances générales des données.

 **Modèle Sous-entraîné (Underfitting)** :

* Un modèle trop simple (par exemple, une ligne droite) pourrait ne pas capturer la tendance générale des données.
* Résultat : mauvaise performance à la fois sur les données d'entraînement et de test.

 **Modèle Bien Entraîné** :

* Un modèle de complexité appropriée capture la tendance générale sans se soucier du bruit.
* Résultat : bonne performance à la fois sur les données d'entraînement et de test.

**Conclusion**

En reformulant les données pour correspondre aux attentes des LSTM et en utilisant un modèle séquentiel avec des couches LSTM et Dense, nous pouvons entraîner le modèle à prédire des valeurs futures à partir de données séquentielles. L'optimiseur Adam et la fonction de perte MSE aident à ajuster les poids du réseau pour minimiser l'erreur et améliorer les prédictions. Cette approche permet de capturer les dépendances temporelles et les motifs récurrents dans les données, ce qui est essentiel pour les séries temporelles.