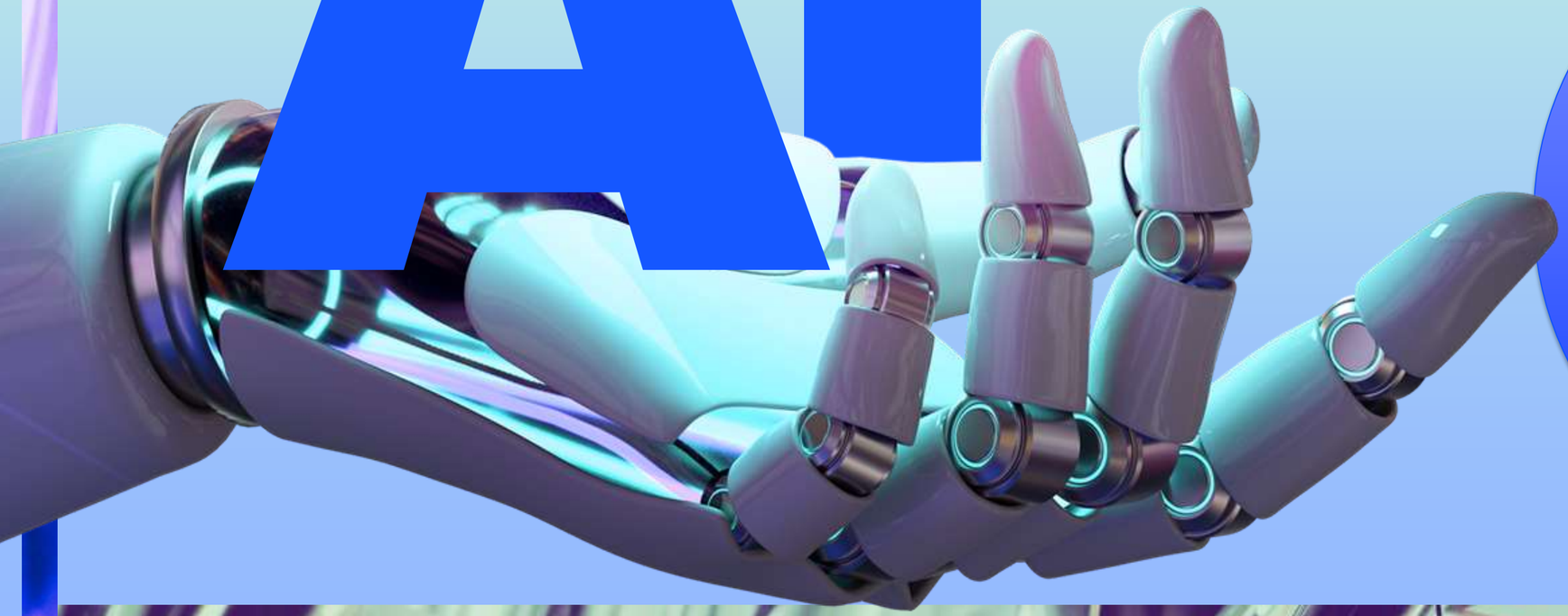
AI



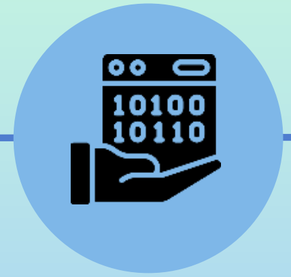
DETECTION D'ANOMALIES DANS LES PERFORMANCES DE MACHINES INDUSTRIELLES

Simulation

Les étapes de simulation

Appellations des bibliothèques

Collection et analyse des données



entraînement du modèle

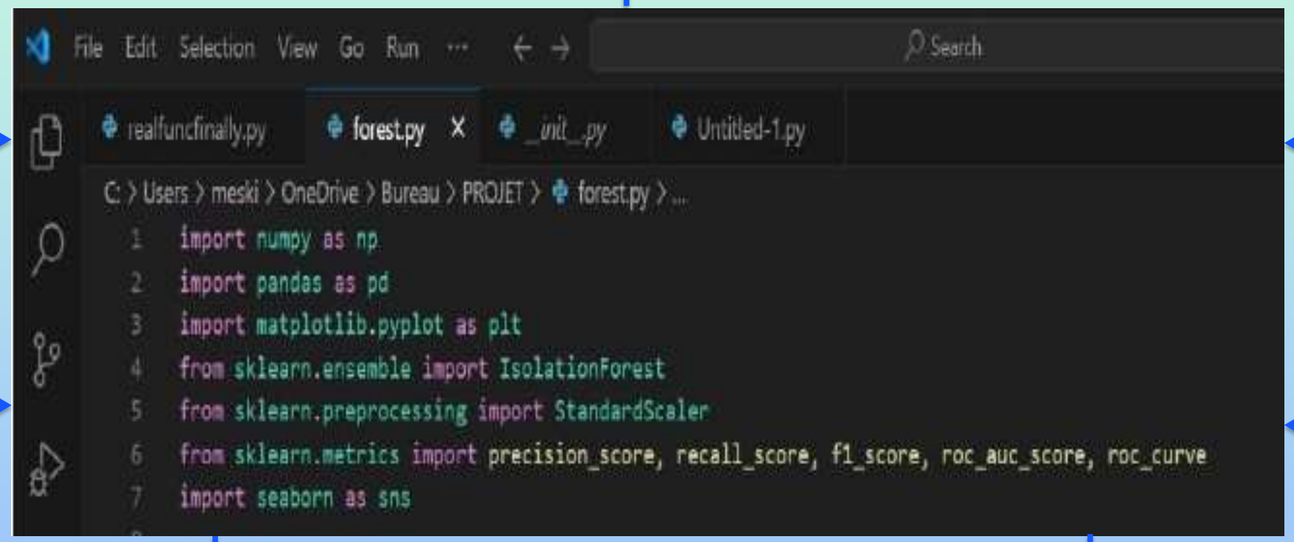
Mesure de performance

Appellations des bibliothèques

numpy

seaborn

matplotlib
. pyplot



pandas

sklearn.ensemble
.IsolationForest

sklearn.preprocessing
.StandardScaler

klearn.
metrics

Bibliothèque	Rôle principal	Utilisation courante dans ce contexte
numpy	Manipulation de tableaux numériques multidimensionnels (arrays)	Calculs numériques, opérations matricielles, traitement de données numériques
pandas	Structure de données haute performance (DataFrames) pour l'analyse de données	Chargement, nettoyage, manipulation et analyse de données tabulaires
matplotlib. pyplot	Création de visualisations statiques, de graphiques et de diagrammes	Visualisation des données, exploration de données
sklearn.ensemble.IsolationForest	Algorithme d'apprentissage automatique pour la détection d'anomalies	Identification des points de données qui s'écartent de la normale
sklearn.preprocessing.StandardScaler	Transformation des données pour avoir une moyenne nulle et une variance unitaire	Prétraitement des données pour les algorithmes d'apprentissage automatique
sklearn.metrics	Calcul de métriques d'évaluation pour les modèles d'apprentissage automatique	Évaluation des performances du modèle (précision, rappel, F1-score, AUC, etc.)
seaborn	Création de visualisations statistiques attrayantes	Visualisation de données de manière plus élaborée et esthétiquement plaisante

Collection et analyse des données

```python
# Chargement des données
file_path = r'C:\Users\meski\OneDrive\Bureau\synthetic_motor_temperatures_no_labels.xlsx'
df = pd.read_excel(file_path)

# Ajout de la vérité terrain
df['Ground Truth'] = ((df['Temperature (°C)'] < 20) | (df['Temperature (°C)'] > 100)).astype(int)

# Extraction des températures
temperatures = df['Temperature (°C)'].values.reshape(-1, 1)

# Normalisation des données
scaler = StandardScaler()
temperatures_scaled = scaler.fit_transform(temperatures)
```

Analyse et Identification des Anomalies dans les Températures avec Python

1

Chargement des données : Lecture du fichier Excel avec pd.read_excel(file_path), structurant les données dans un DataFrame.

2

Ajout de la vérité terrain : Création de la colonne 'Ground Truth' pour identifier les anomalies (1) et les valeurs normales (0)

3

Extraction des températures : Conversion de la colonne ' Temperature (°C)' en une matrice adaptée aux algorithmes de machine learning.

4

Normalisation des données : Application de StandardScaler pour obtenir une moyenne de 0 et un écart-type de 1, améliorant la performance des modèles.

Simulation

Isolation Forest

IF

Local Outlier Factor

LOF

Isolation Forest

entraînement du modèle

```python
# Step 4: Apply the Isolation Forest algorithm
isolation_forest = IsolationForest(n_estimators=100, contamination=0.05, random_state=42)
isolation_forest.fit(temperatures_scaled)
y_pred = isolation_forest.predict(temperatures_scaled)
anomaly_scores = isolation_forest.decision_function(temperatures_scaled)  # Anomaly scores

# Convert predictions: -1 for anomalies, 1 for normal
df['Anomaly'] = (y_pred == -1).astype(int)
```

- **Initialisation du modèle :** Création d'un Isolation Forest avec 100 arbres et une contamination de 5 %.Entraînement : Ajustement du modèle sur les températures normalisées.
- **Prédiction :** Détection des anomalies (-1) et des valeurs normales (1).Scores d'anomalie : Évaluation du degré d'anormalité des données.
- **Enregistrement des résultats :** Conversion et ajout des anomalies (1) et normales (0) au DataFrame.

```python
# Step 5: Identify anomalies and normal values
anomalies = df[df['Anomaly'] == 1]['Temperature (°C)']
normal = df[df['Anomaly'] == 0]['Temperature (°C)']

# Save anomalies to a CSV file for further analysis
df[df['Anomaly'] == 1].to_csv("isolation_forest_anomalies.csv", index=False)
```

- **Identification et sauvegarde des anomalies.**
- **Extraction des anomalies et valeurs normales, puis sauvegarde des anomalies dans un fichier CSV pour analyse ultérieure.**

Entraînement du modèle

```
plt.figure(figsize=(10, 6))
sns.histplot(temperatures.flatten(), bins=30, kde=True, color='lightblue', edgecolor='black')
plt.xlabel('Température (°C)')
plt.ylabel('Fréquence')
plt.title('Distribution des températures avec densité')
plt.savefig("temperature_distribution_isolation_forest.png")  # Save the plot
plt.show()
```

Génération d'un histogramme avec densité

- Création d'une figure pour l'affichage.
- Tracé d'un histogramme avec 30 intervalles (bins=30).
- Ajout d'une courbe de densité (kde=True).
- Étiquetage des axes et ajout d'un titre.
- Enregistrement du graphique (temperature_distribution_isolation_forest.png).
- Affichage du graphique avec plt.show().

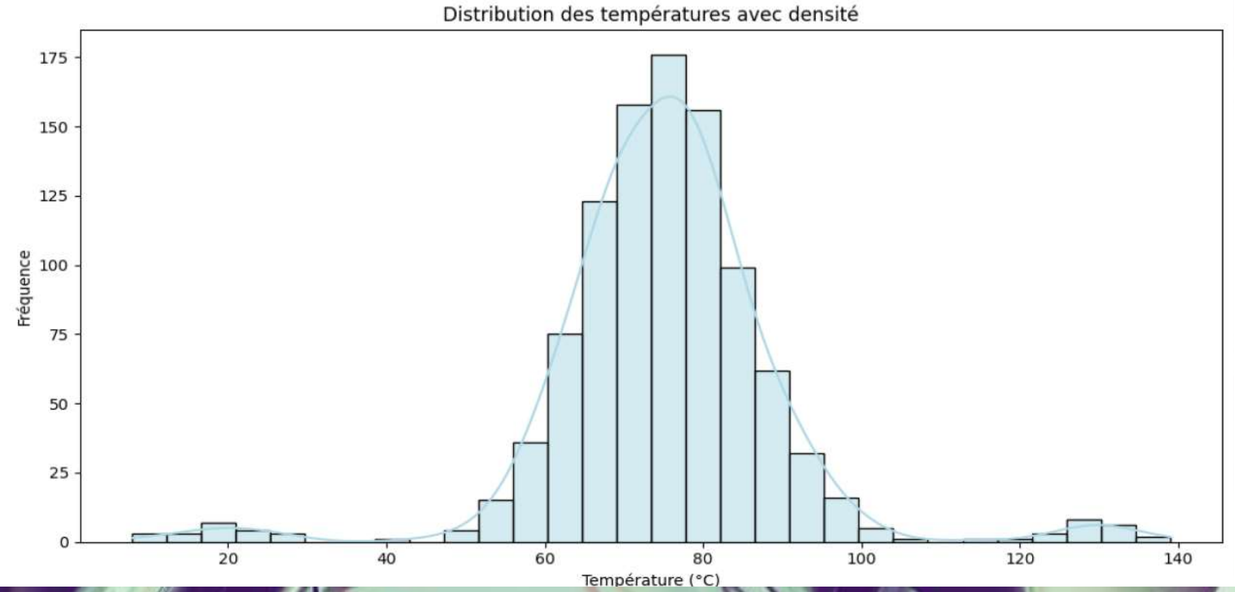
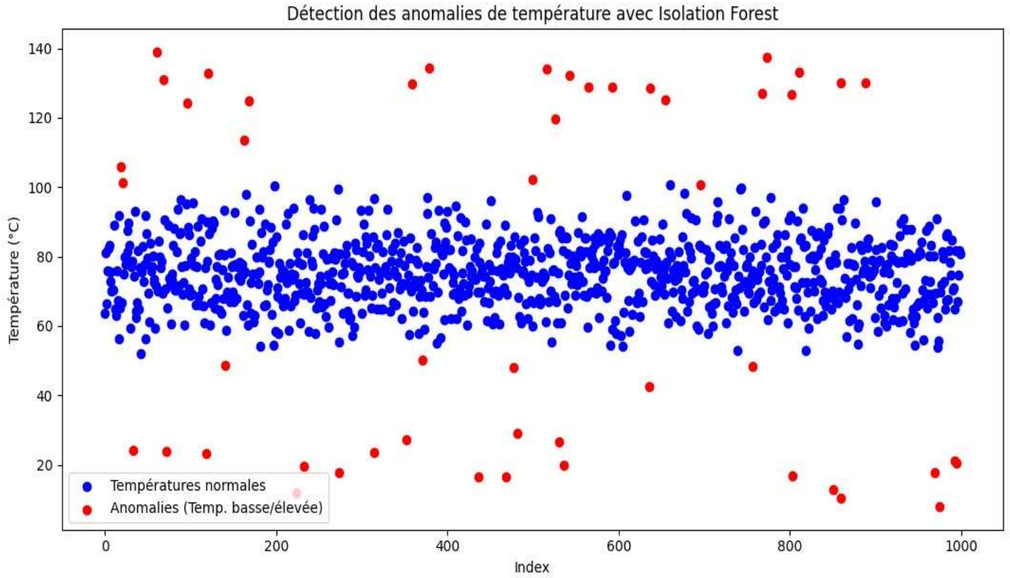
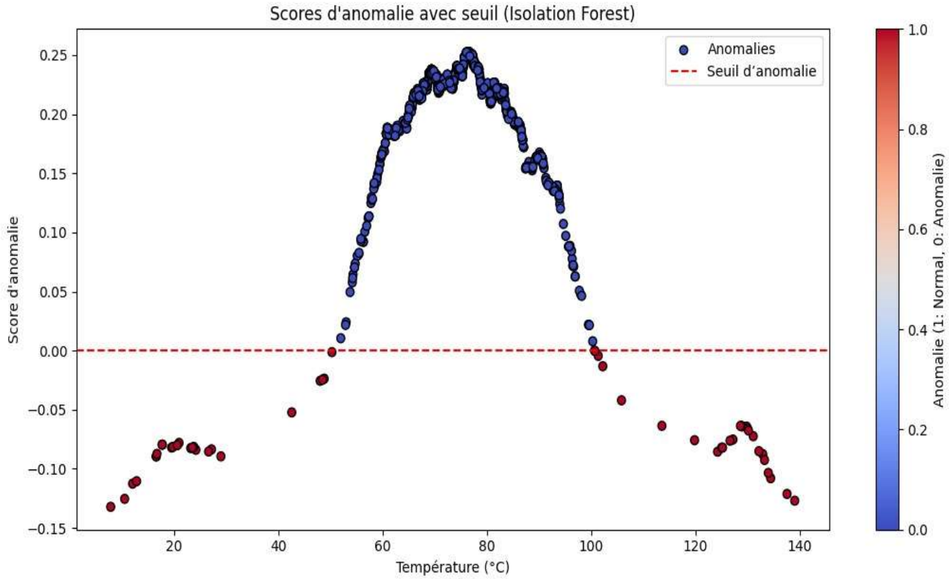


Figure : Distribution des Températures avec Densité

Mesure de performance



- Ce graphique montre la détection des anomalies de température avec Isolation Forest.
- Les points bleus représentent les températures normales, tandis que les points rouges indiquent les anomalies (températures trop basses ou trop élevées).



- Ce graphique représente les scores d'anomalie en fonction des températures détectées par Isolation Forest.
- Les points bleus indiquent les températures normales, tandis que les points rouges représentent les anomalies.
- La ligne rouge en pointillés marque le seuil d'anomalie, en dessous duquel une température est considérée comme anormale.

Mesure de performance

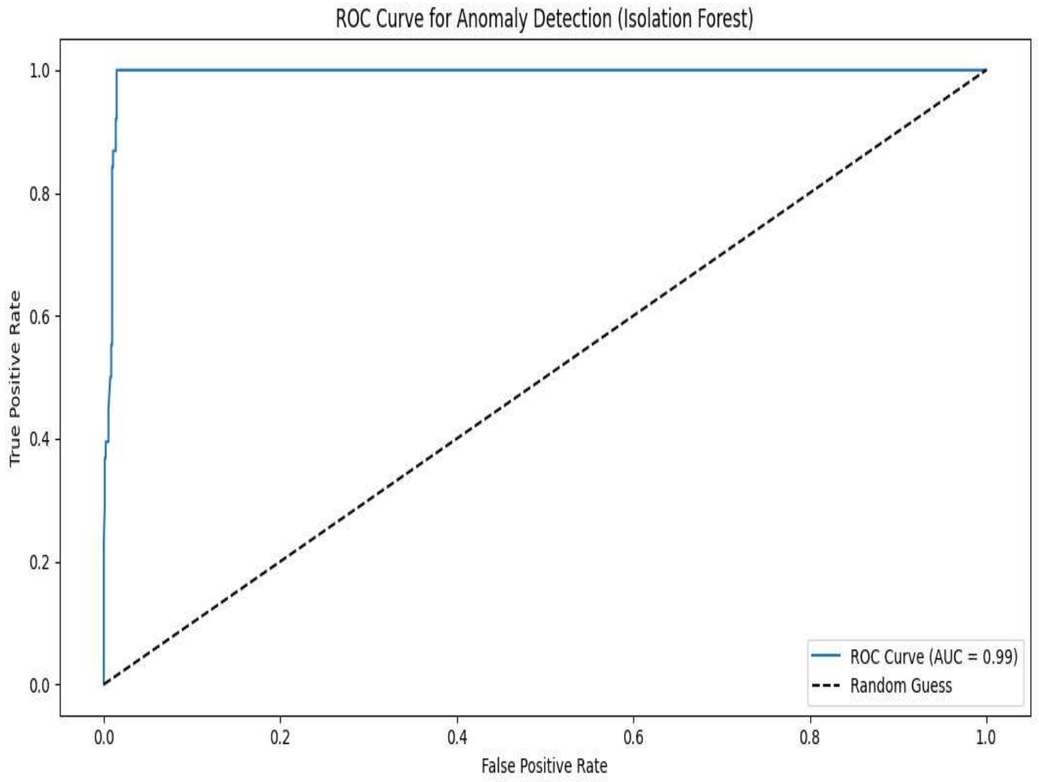


Figure :Ce graphique représente la courbe ROC (Receiver Operating Characteristic) pour l'évaluation des performances du modèle Isolation Forest dans la détection d'anomalies.

- L'axe des x correspond au taux de faux positifs, tandis que l'axe des y représente le taux de vrais positifs.
- La courbe bleue montre la performance du modèle avec une aire sous la courbe (AUC) de 0.99, indiquant une excellente capacité de classification.
- La ligne en pointillés noirs représente une prédiction aléatoire (AUC = 0.5).
- Plus la courbe est éloignée de cette ligne, meilleure est la performance du modèle.

Local Outlier Factor

entraînement du modèle:

Application d'algorithme:

- LOF détecte les anomalies en comparant la densité locale des points.
- N_neighbors =10 : Nombre de voisins pour l'analyse.
- - contamination=0.05 : 5 % des données sont considérées comme anomalies.

```
# Step 5: Apply the LOF algorithm
lof = LocalOutlierFactor(n_neighbors=10, contamination=0.05)
y_pred = lof.fit_predict(temperatures_scaled)
lof_scores = -lof.negative_outlier_factor_  # Higher scores indicate anomalies
```

```
# Step 5: Identify anomalies and normal values
anomalies = temperatures[y_pred == -1]
normal = temperatures[y_pred == 1]
```

Sauvegarde :

Les anomalies sont enregistrées dans anomalies.csv pour analyse.

```
# Save anomalies to a CSV file for further analysis
anomalies_df = df[y_pred == -1]
anomalies_df.to_csv("anomalies.csv", index=False)
```

Résultats :

y_pred == -1 → Anomalies

y_pred == 1 → Normales

lof_scores → Plus le score est élevé, plus l'anomalie est probable.

entraînement du modèle:

```
# Step 8: Evaluate performance (if ground truth labels are available)
if 'Ground Truth' in df.columns:
    ground_truth = df['Ground Truth'].values
    # Calculate precision, recall, F1-score
    precision = precision_score(ground_truth, y_pred == -1)
    recall = recall_score(ground_truth, y_pred == -1)
    f1 = f1_score(ground_truth, y_pred == -1)
    auc = roc_auc_score(ground_truth, lof_scores)

    # Print the metrics
    print(f"Precision: {precision:.2f}")
    print(f"Recall: {recall:.2f}")
    print(f"F1-Score: {f1:.2f}")
    print(f"ROC AUC Score: {auc:.2f}")
```

Extraire la’’Ground Truth’’:

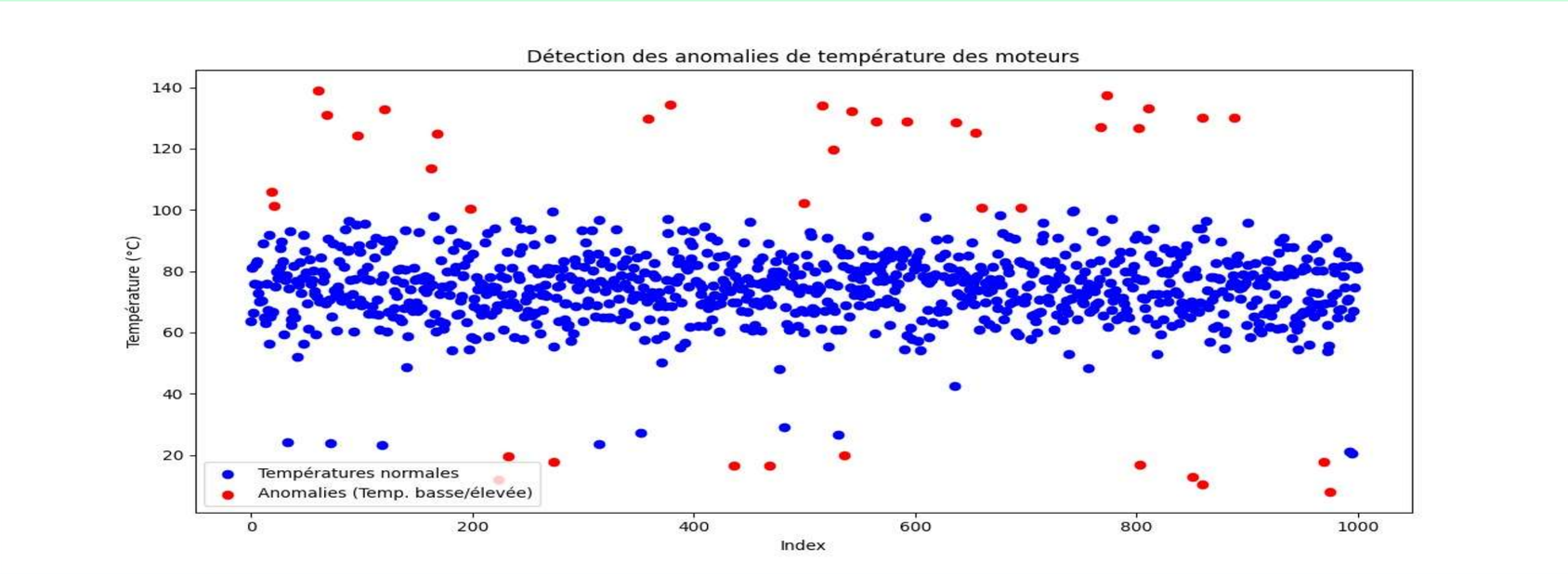
- Le modèle compare ses prédictions avec les valeurs réelles ("Ground Truth") pour vérifier si ses résultats sont corrects.

Calcule les métriques :

- Précision : Taux de détection correcte des anomalies.
- Recall : Proportion des anomalies correctement détectées.
- F1-score : Moyenne harmonique de précision et rappel.
- AUC (ROC) : Performance globale du modèle.

- Affiche les métriques avec deux décimales.

Mesure de performance



- Ce graphique montre la détection des anomalies de température avec LOF.
- Les points bleus représentent les températures normales, tandis que les points rouges indiquent les anomalies (températures trop basses ou trop élevées).

Mesure de performance

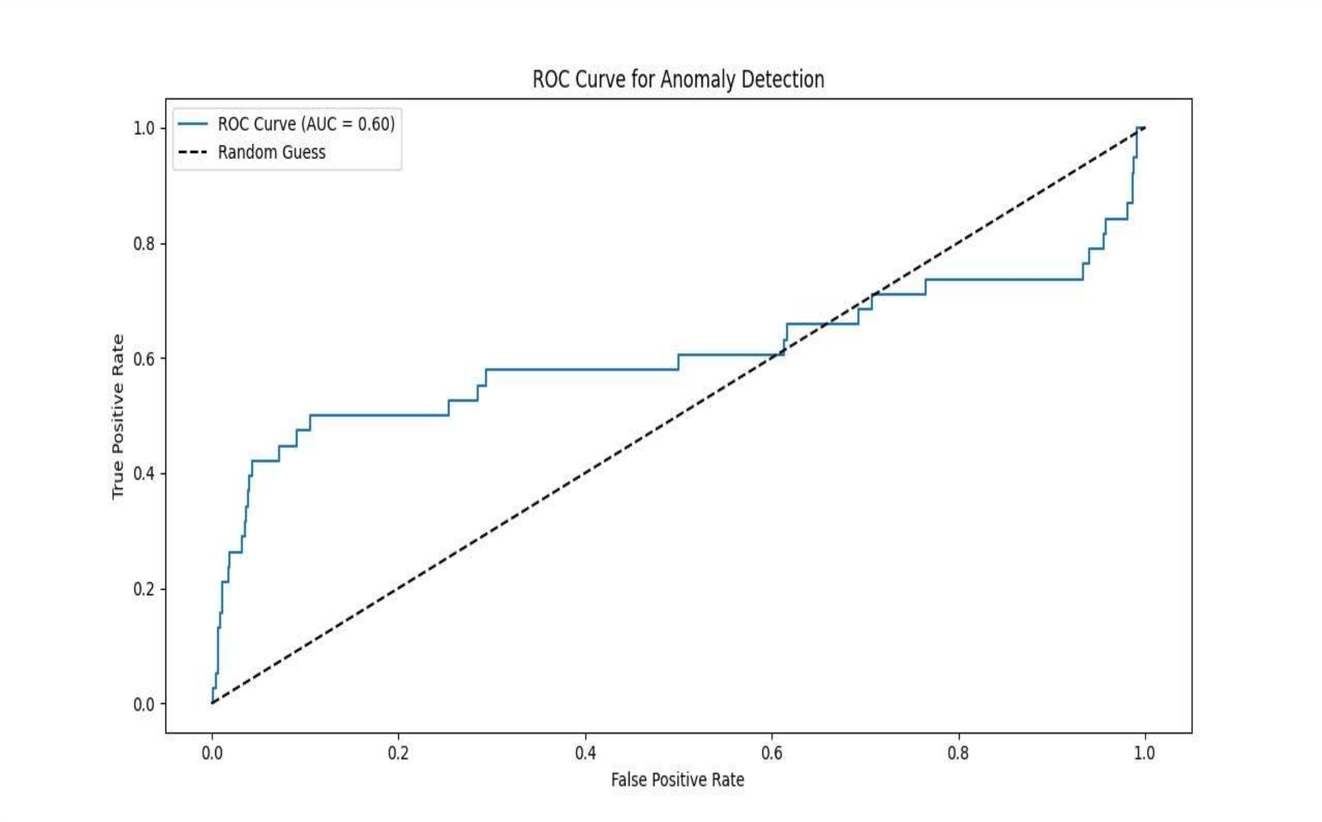
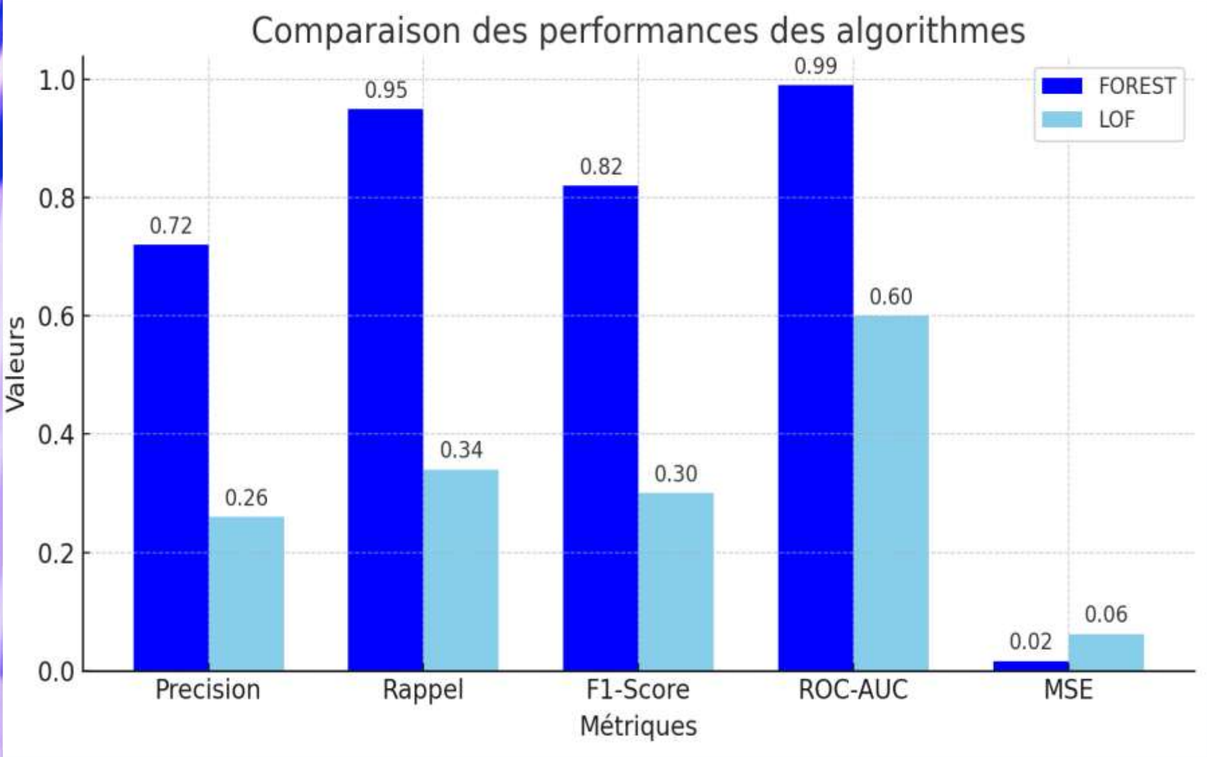


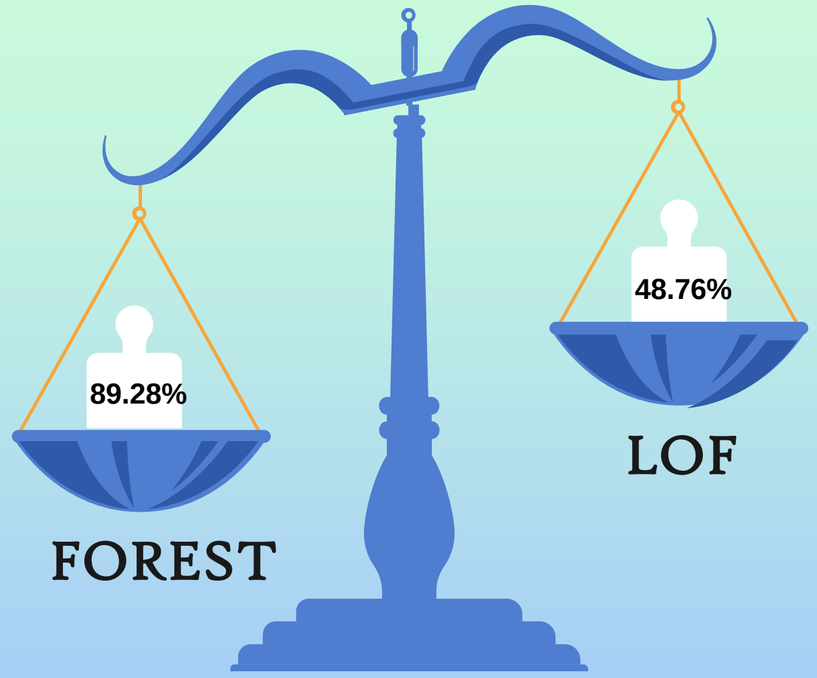
Figure : Courbe ROC pour la détection d'anomalies

- Axe X : Taux de faux positifs (FPR), représentant les erreurs du modèle.
- Axe Y : Taux de vrais positifs (TPR), représentant la capacité du modèle à détecter correctement les anomalies.
- La diagonale (ligne pointillée) : Représente un modèle aléatoire (AUC = 0.5), ce qui signifie une classification au hasard .
- L'AUC (Area Under Curve) : Mesure la qualité du modèle. Une AUC proche de 1 indique une très bonne performance, tandis qu'une AUC proche de 0.5 indique une performance proche du hasard . Dans ton cas, l'AUC est 0.60, ce qui signifie que le modèle a une capacité de détection des anomalies légèrement meilleure que le hasard, mais qui pourrait être améliorée.

Comparaison des performances



Le graphique compare les performances des algorithmes FOREST et LOF. FOREST surpasse LOF sur toutes les métriques, offrant une meilleure précision, un rappel plus élevé et une erreur plus faible.



Pour calculer **un pourcentage global de performance pour chaque algorithme**, nous faisons la moyenne des valeurs des cinq.

- **FOREST : (0.72+0.95+0.82+0.99+(1−0.016))/5 = 89.28%**
- **LOF : (0.26+0.34+0.30+0.60+(1−0.062))/5 = 48.76%**