

Azure Machine Learning Project

« *Air Quality Prediction* »



Équipe : SENECHAL Morgan

Table des matières

Introduction	3
I. Environnement Azure.....	4
Avantages d'un Environnement Personnalisé	4
Création et Gestion de l'Environnement.....	4
II. Description du jeu de données et étapes de prétraitement.	5
Présentation des données	5
Stockage des données	6
Exploration des données	6
Analyse des données	7
Encodage des données	13
Train-Test Split	13
III. Model selection.	14
IV. Optimisation des hyperparamètres.....	17
V. Processus d'entraînement et d'évaluation	19
VI. Étapes de déploiement et implémentation de l'interface Web.	24
Sauvegarde et Enregistrement du Modèle	24
Création d'un Endpoint et déploiement en Ligne	25
Implémentation de l'Interface Web	25
Monitoring de l'endpoint.....	27
VII. Résultats, difficultés rencontrées et axes d'amélioration.	29
Résultats.....	29
Difficultés rencontrées	30
Améliorations potentielles.....	30
Conclusion	32

Introduction

Dans le contexte actuel, où les projets de data science et d'apprentissage automatique nécessitent une scalabilité et une efficacité accrue, l'utilisation du cloud, et en particulier d'Azure Machine Learning, se démarque comme une solution de choix. Contrairement aux environnements locaux, Azure offre une infrastructure flexible et évolutive, permettant de gérer facilement de grandes quantités de données, d'effectuer des entraînements intensifs de modèles et de déployer des applications en production. Azure Machine Learning se distingue par ses fonctionnalités intégrées telles que le suivi des expériences, l'optimisation des hyperparamètres et le déploiement simplifié de modèles sous forme d'API ou de web services. Ces outils permettent non seulement de gagner en productivité, mais également de garantir une collaboration fluide au sein des équipes grâce à l'accessibilité et à la centralisation des ressources.

Ce projet s'appuie sur Azure Machine Learning pour concevoir, optimiser et déployer un modèle d'apprentissage automatique end-to-end. En travaillant avec le jeu de données Air Quality and Pollution Assessment de Kaggle, l'objectif principal est de prédire la qualité de l'air en fonction de divers facteurs environnementaux et démographiques. Les étapes incluent la préparation des données, le développement et l'optimisation du modèle, ainsi que la création d'une interface utilisateur via Streamlit ou une Azure Web App pour des prédictions en temps réel. L'intégration d'outils comme MLflow pour le suivi des expériences et Optuna pour l'optimisation des hyperparamètres est également explorée. Ce projet illustre ainsi l'importance et la valeur des outils cloud dans la mise en œuvre de solutions d'apprentissage automatique modernes et performantes.

I. Environnement Azure

Dans le cadre de ce projet, un environnement personnalisé a été choisi au lieu d'un environnement préconfiguré (curated environment). Cette décision repose sur plusieurs justifications stratégiques et techniques.

Avantages d'un Environnement Personnalisé

Flexibilité dans les Dépendances :

- L'environnement personnalisé permet d'intégrer des bibliothèques spécifiques, telles qu'Optuna pour l'optimisation des hyperparamètres, MLflow pour la gestion du cycle de vie des modèles, scikit-learn pour l'application des techniques et modèles de machine learning, ainsi que des outils de visualisation comme Plotly et Seaborn, et des bibliothèques de manipulation de données comme pandas et numpy, entre autres.
- Les dépendances sont spécifiquement définies dans un fichier conda.yaml, garantissant un contrôle précis des versions et des canaux utilisés.

Compatibilité avec des Scénarios Complexes :

- Contrairement aux environnements préconfigurés, l'environnement personnalisé offre une intégration fluide avec des bibliothèques comme azureml-inference-server-http et azureml-mlflow, nécessaires pour le déploiement et le suivi sur Azure Machine Learning.

Reproductibilité et Traçabilité :

- En spécifiant un fichier conda.yaml et une image Docker explicite (mcr.microsoft.com/azureml/openmpi4.1.0-ubuntu20.04:latest), l'environnement est reproductible sur plusieurs exécutions et versions.

Optimisation pour des Besoins Spécifiques :

- Cet environnement inclut uniquement les bibliothèques essentielles au projet, ce qui réduit les risques de surcharge inutile et garantit des performances optimales.

Création et Gestion de l'Environnement

L'environnement a été défini dynamiquement grâce au fichier conda.yaml généré par le script suivant :

```
conda_content = """
name: custom-mlflow-env
channels:
  - defaults
  - conda-forge
dependencies:
  - python=3.8
  - pip
  - numpy
  - pandas
  - scikit-learn
  - matplotlib
  - seaborn
  - plotly
  - mlflow
  - optuna
  - joblib
  - pip:
    - azureml-mlflow
    - azureml-inference-server-http
    - azureml-defaults
    - mltable
    - azure-ai-ml
    - mlflow.pyfunc
"""

# Set and save the environment
custom_env = Environment(
    name="custom-mlflow-env",
    description="Environnement personnalisé avec MLflow et scikit-learn",
    conda_file=conda_file_path, # Generated conda.yaml file
    image="mcr.microsoft.com/azureml/openmpi4.1.0-ubuntu20.04:latest",
    version="2.0" # Explicitly set a version
)

# Create or update the environment
custom_env = ml_client.environments.create_or_update(custom_env)
print(f"Personalized environment '{custom_env.name}' with version '{custom_env.version}' successfully created.")
```

Ce fichier, combiné à une image Docker robuste, permet une configuration adaptée aux besoins du projet tout en offrant une grande compatibilité avec les outils Azure :

Comparaison avec un Environnement Préconfiguré

Les environnements préconfigurés, bien que pratiques pour des projets simples, ne permettent pas :

- Une personnalisation poussée des dépendances.
- L'utilisation de versions spécifiques de certaines bibliothèques nécessaires à ce projet.
- Un contrôle total sur les configurations, essentiel pour garantir une cohérence entre les environnements de développement et de production.

Le choix d'un environnement personnalisé garantit un contrôle total, une compatibilité optimale, et une adaptabilité aux exigences du projet, le rendant plus robuste et évolutif.

II. Description du jeu de données et étapes de prétraitement.

Présentation des données

Le jeu de données utilisé dans ce projet provient de la plateforme Kaggle, sous le titre Air Quality and Pollution Assessment (source). Ce jeu de données se concentre sur l'évaluation de la qualité de l'air dans différentes régions, avec un total de 5000 échantillons. Il contient des informations clés sur des facteurs environnementaux et démographiques qui influencent les niveaux de pollution.

Caractéristiques principales des données

Facteurs environnementaux :

- **Température (°C)** : Température moyenne enregistrée dans la région.
- **Humidité (%)** : Taux d'humidité relative dans l'air.

Concentrations de polluants :

- **PM2.5 ($\mu\text{g}/\text{m}^3$)** : Niveau de particules fines, potentiellement nocives pour la santé.
- **PM10 ($\mu\text{g}/\text{m}^3$)** : Niveau de particules plus grossières.
- **NO2 (ppb)** : Concentration de dioxyde d'azote, un gaz polluant.
- **SO2 (ppb)** : Concentration de dioxyde de soufre.
- **CO (ppm)** : Concentration de monoxyde de carbone.

Facteurs démographiques et industriels :

- **Proximité des zones industrielles (km)** : Distance au site industriel le plus proche.
- **Densité de population (personnes/km²)** : Nombre moyen d'habitants par kilomètre carré dans la région.

Variable cible

La qualité de l'air est évaluée selon quatre catégories :

- **Good** : Air propre avec de faibles niveaux de pollution.
- **Moderate** : Qualité de l'air acceptable malgré la présence de certains polluants.

- **Poor** : Pollution notable, pouvant affecter les groupes sensibles.
- **Hazardous** : Pollution élevée avec des risques sérieux pour la santé.

Ce jeu de données constitue une base solide pour analyser les facteurs influençant la qualité de l'air et pour identifier les zones à risque ou nécessitant des mesures correctives.

Stockage des données

Choix du stockage dans un Data Asset sur Azure Machine Learning

Nous avons décidé de stocker le fichier `pollution_dataset.csv` en tant que Data Asset dans Azure Machine Learning pour les raisons suivantes :

- **Centralisation** : Facilite l'accès et le partage des données depuis une plateforme unique.
- **Versionnement** : Permet de suivre et restaurer les modifications du dataset.
- **Intégration** : S'intègre directement aux pipelines d'analyse et d'entraînement.
- **Sécurité** : Garantit la protection des données avec des contrôles d'accès et un chiffrement intégré.
- **Flexibilité** : S'adapte à nos besoins, même si le dataset évolue.

Ce choix simplifie la gestion des données tout en assurant efficacité et fiabilité pour les étapes suivantes du projet.

Exploration des données

L'exploration des données a permis de mieux comprendre la structure et les caractéristiques du jeu de données.

Structure et types de données :

L'appel à `df.info()` montre que le dataset contient 5000 enregistrements et 10 colonnes. Les variables comprennent principalement des données numériques (`float64` et `int64`) ainsi qu'une variable catégorielle, `Air Quality`, qui représente la qualité de l'air. Aucune valeur manquante n'est présente dans le dataset, comme confirmé par `df.isnull().sum()`.

Résumé statistique :

La fonction `df.describe()` fournit des statistiques descriptives pour les variables numériques. Par exemple, la température varie entre 13,4 °C et 58,6 °C, avec une moyenne de 30 °C. Les concentrations de particules fines (PM2.5) et grossières (PM10) présentent des écarts importants, atteignant des valeurs maximales de 295 µg/m³ et 315,8 µg/m³ respectivement. Certaines valeurs négatives sont également observées pour PM10 et SO₂, suggérant des possibles anomalies ou erreurs dans les données. La densité de population varie entre 188 et 957 personnes par km², avec une moyenne de 497 personnes/km².

Répartition de la qualité de l'air :

L'analyse de la variable `Air Quality` révèle que :

- 2000 échantillons sont classés comme `Good`.

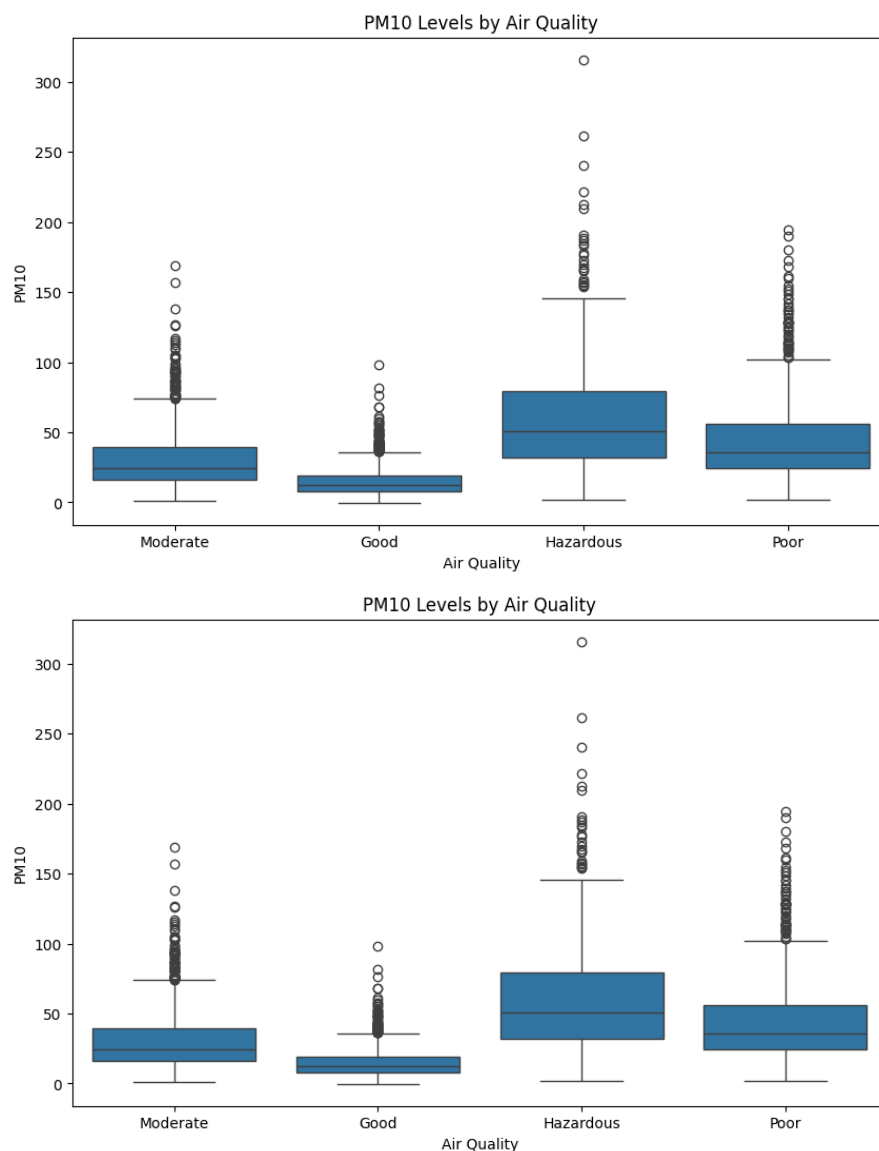
- 1500 comme Moderate.
- 1000 comme Poor.
- 500 comme Hazardous.

Cette distribution montre une majorité d'observations avec une qualité de l'air acceptable ou bonne, mais une proportion notable de pollution élevée ou dangereuse.

Ces analyses préliminaires confirment que le dataset est complet et prêt pour des étapes d'analyse plus approfondies, tout en nécessitant une attention particulière aux éventuelles anomalies.

Analyse des données

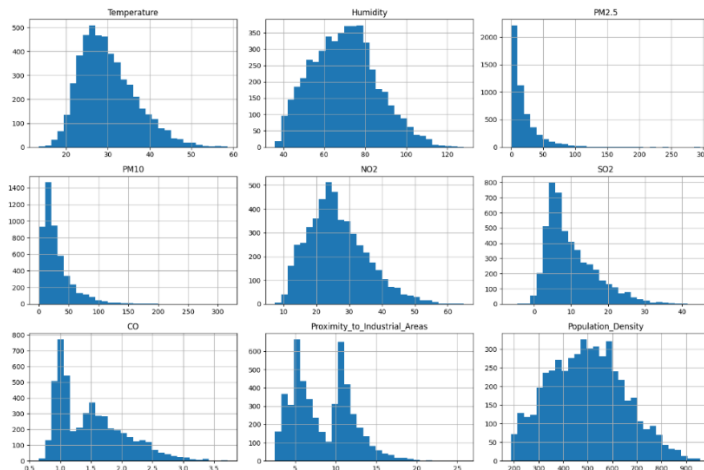
Niveaux de PM10 et PM2.5 selon la qualité de l'air :



Les graphiques des niveaux de PM10 (particules grossières) et PM2.5 (particules fines) selon la qualité de l'air révèlent une tendance similaire : plus la qualité de l'air se dégrade, plus les concentrations de ces particules augmentent. Pour une qualité de l'air "Good", les niveaux de

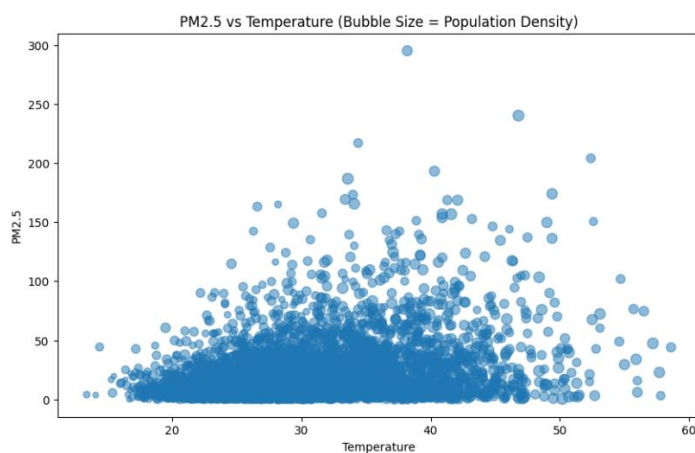
PM10 et PM2.5 restent faibles, avec des médianes bien en dessous des seuils critiques. À mesure que la qualité passe à "Moderate" et "Poor", les concentrations augmentent progressivement, traduisant une pollution croissante. Dans la catégorie "Hazardous", les niveaux atteignent leurs pics les plus élevés, avec des médianes significatives et de nombreuses valeurs extrêmes dépassant $300 \mu\text{g}/\text{m}^3$ pour les deux types de particules. Ces observations montrent une corrélation forte entre la détérioration de la qualité de l'air et l'augmentation des niveaux de particules fines et grossières, ces dernières jouant un rôle crucial dans la pollution atmosphérique et ses impacts sur la santé.

Distribution des données :



Le graphique montre que les variables Temperature et Humidity suivent des distributions normales autour de 30°C et 70% , respectivement. Les polluants PM2.5, PM10, NO2, SO2, et CO présentent des distributions asymétriques avec des valeurs majoritairement faibles, mais des pics indiquent des zones fortement polluées. La Proximity to Industrial Areas est bimodale, reflétant des régions proches et éloignées des sites industriels, tandis que la Population Density est normalement répartie autour de $500 \text{ habitants}/\text{km}^2$. Ces distributions révèlent des disparités marquées entre les régions étudiées.

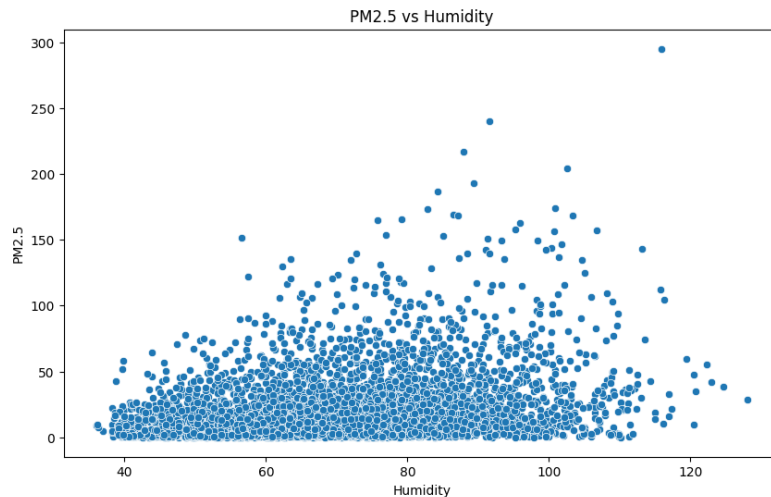
Niveaux de PM2.5 selon la température :



Le graphique illustre la relation entre la température et les niveaux de PM2.5, avec la taille des bulles représentant la densité de population. On observe une concentration élevée de

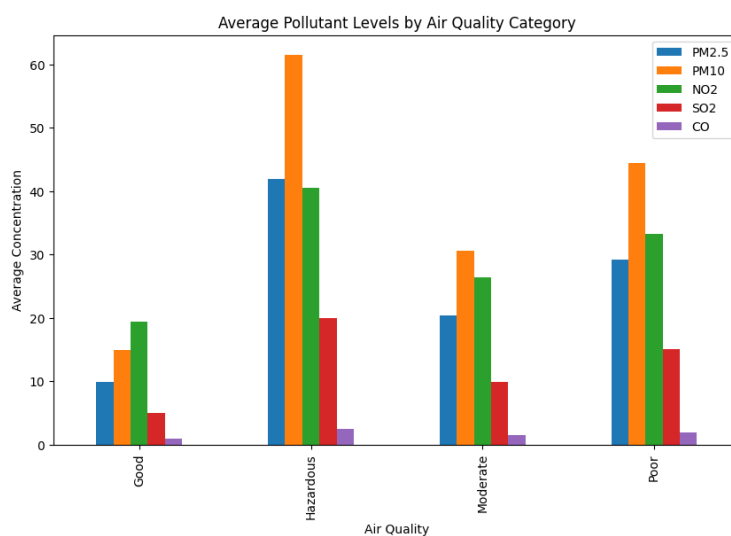
particules fines (PM_{2.5}) à des températures comprises entre 20 °C et 40 °C, avec une légère augmentation des valeurs extrêmes à mesure que la température augmente. Les bulles plus grandes, associées à une densité de population élevée, montrent que les zones densément peuplées tendent à enregistrer des niveaux de PM_{2.5} plus élevés. Cela suggère une corrélation entre la densité de population, les activités humaines, et la pollution par les particules fines.

Niveaux de PM_{2.5} selon l'humidité :



Le graphique montre la relation entre les niveaux de PM_{2.5} et l'humidité. La majorité des observations se situent à des niveaux faibles de PM_{2.5} (inférieurs à 50 µg/m³), quelle que soit l'humidité, qui varie principalement entre 40 % et 100 %. On observe toutefois une légère augmentation des valeurs extrêmes de PM_{2.5} à mesure que l'humidité dépasse 80 %, avec des pics atteignant plus de 300 µg/m³. Cette tendance pourrait refléter un lien entre des conditions d'humidité élevée et des concentrations accrues de particules fines, bien que l'effet semble marginal dans la majorité des cas.

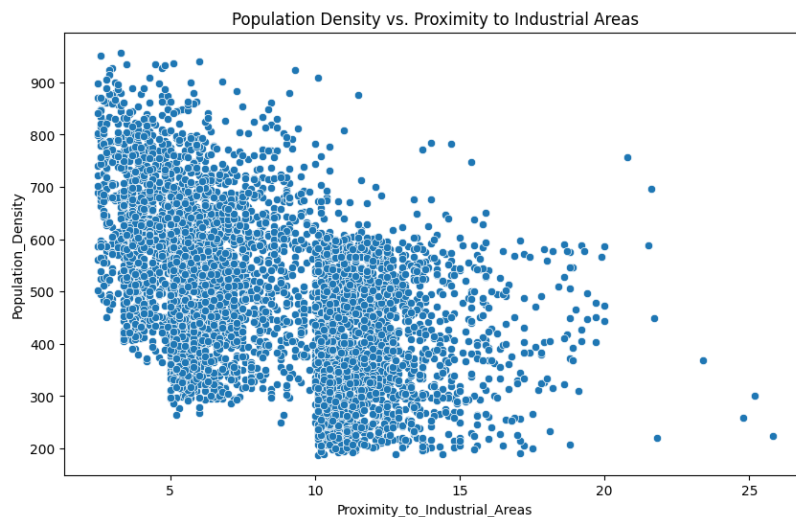
Niveaux moyens de polluant selon la qualité de l'air :



Le graphique montre les concentrations moyennes de différents polluants (PM_{2.5}, PM₁₀, NO₂, SO₂, CO) en fonction des catégories de qualité de l'air. On observe que les niveaux de tous les

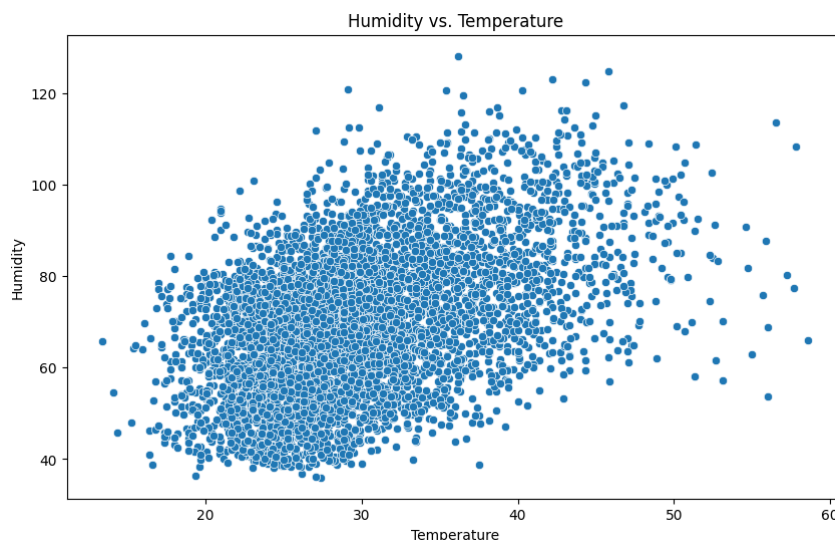
polluants augmentent de manière significative à mesure que la qualité de l'air se détériore, atteignant des pics dans la catégorie "Hazardous". Les particules fines (PM2.5) et grossières (PM10) présentent les concentrations les plus élevées, suivies de NO2. Les gaz SO2 et CO affichent des niveaux plus faibles mais augmentent également dans les zones où la qualité de l'air est mauvaise. Ces résultats soulignent le rôle dominant des particules en suspension dans la dégradation de la qualité de l'air et les risques sanitaires associés.

Densité de population et proximité des zones industrielles :



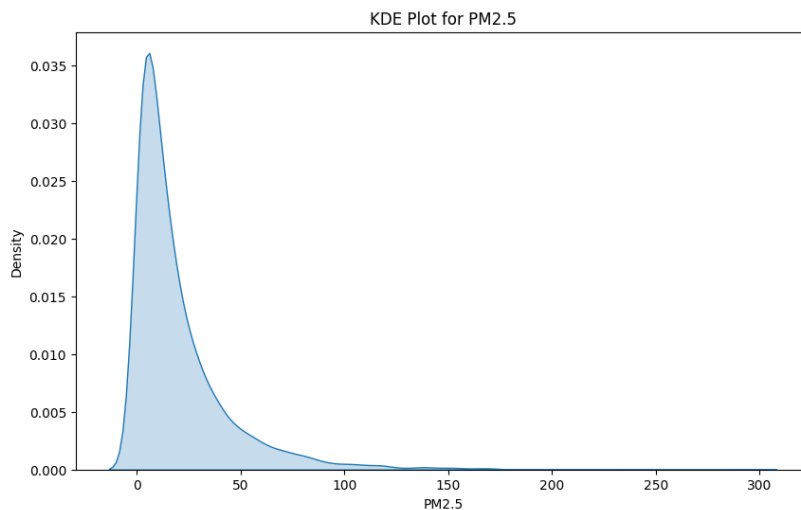
Le graphique illustre la relation entre la densité de population et la proximité aux zones industrielles. On observe une concentration de points dans deux groupes distincts : l'un autour de 5-10 km et l'autre autour de 10-15 km des zones industrielles. La densité de population est généralement plus élevée (500 à 900 habitants/km²) dans les zones plus proches des sites industriels (5-10 km), ce qui peut refléter l'implantation de zones urbaines ou résidentielles à proximité. À mesure que la distance augmente, la densité de population tend à diminuer, suggérant des zones moins peuplées dans les régions éloignées des activités industrielles. Ce schéma souligne l'impact potentiel de l'industrialisation sur la distribution démographique.

Humidité et température :



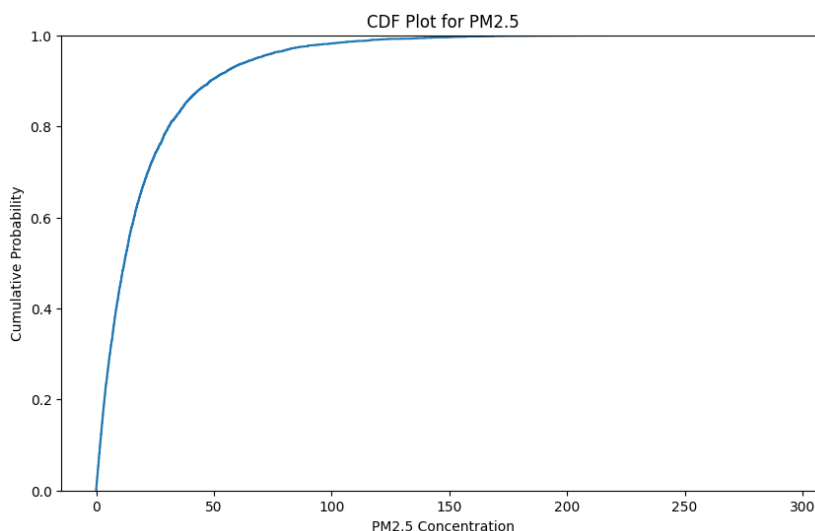
Le graphique montre la relation entre l'humidité et la température. Une tendance générale est observée : à mesure que la température augmente, l'humidité tend à diminuer, bien que les points soient assez dispersés. Les valeurs d'humidité se concentrent principalement entre 60 % et 80 %, avec une plage plus large à des températures modérées (20 °C à 40 °C). Cependant, des niveaux d'humidité plus élevés (supérieurs à 100 %) apparaissent rarement, souvent à des températures plus basses. Ce schéma reflète l'influence de la température sur la capacité de l'air à retenir l'humidité, une caractéristique courante des environnements climatiques.

Graphique KDE pour PM2.5 :



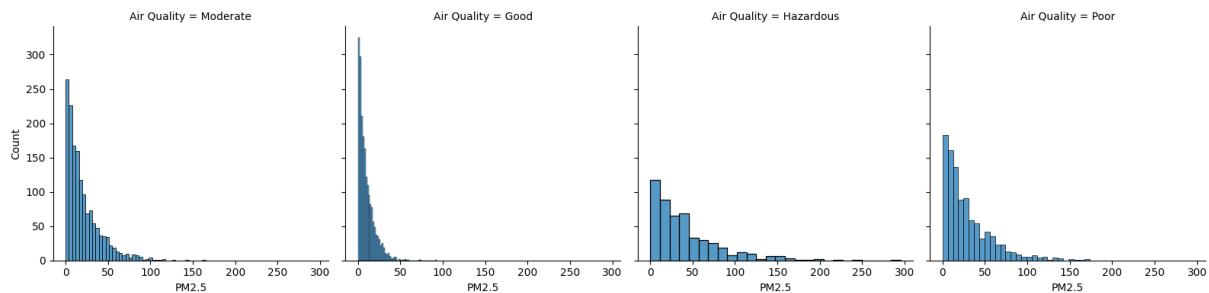
Le graphique KDE pour PM2.5 illustre une distribution fortement asymétrique. La densité est concentrée autour des faibles niveaux de PM2.5, avec un pic marqué entre 0 et 20 $\mu\text{g}/\text{m}^3$, indiquant que la majorité des observations présentent des concentrations faibles de particules fines. Cependant, une longue queue à droite révèle la présence de valeurs plus élevées, atteignant jusqu'à 300 $\mu\text{g}/\text{m}^3$, bien que celles-ci soient rares. Cette distribution reflète une situation où la pollution par les particules fines est généralement faible, mais des niveaux extrêmes sont observés dans certaines zones spécifiques.

Graphique CDF pour PM2,5 :



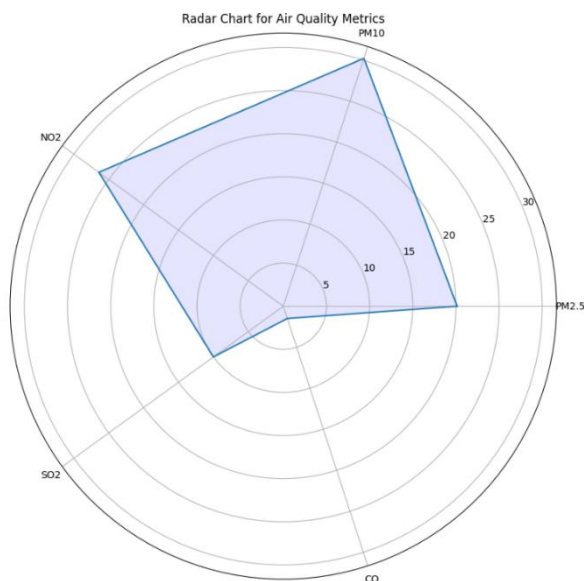
Le graphique CDF pour PM2.5 montre la probabilité cumulative des concentrations de particules fines dans l'air. On observe que plus de 80 % des valeurs de PM2.5 sont inférieures à $50 \mu\text{g}/\text{m}^3$, ce qui reflète une faible pollution pour la majorité des observations. Les concentrations plus élevées, au-delà de $100 \mu\text{g}/\text{m}^3$, deviennent rares et concernent moins de 10 % des données. Ce graphique met en évidence que, bien que les niveaux extrêmes existent, la majorité des zones étudiées présentent des concentrations modérées ou faibles de PM2.5.

Distribution de la qualité de l'air :



Le graphique montre la distribution des niveaux de PM2.5 selon les différentes catégories de qualité de l'air (Good, Moderate, Poor, et Hazardous). Pour la catégorie Good, les niveaux de PM2.5 sont principalement concentrés en dessous de $50 \mu\text{g}/\text{m}^3$, reflétant une pollution faible. Dans la catégorie Moderate, une légère augmentation des niveaux est observée, bien que la majorité des valeurs reste inférieure à $100 \mu\text{g}/\text{m}^3$. Pour les catégories Poor et Hazardous, les distributions montrent des concentrations plus élevées, avec des pics allant au-delà de $150 \mu\text{g}/\text{m}^3$ dans la catégorie Hazardous, indiquant une pollution critique. Ces distributions confirment que des niveaux accrus de PM2.5 sont étroitement associés à une détérioration de la qualité de l'air.

Quantité de polluant moyen :



Le radar chart représente les différents polluants (**PM2.5**, **PM10**, **NO2**, **SO2**, **CO**) en termes de leur concentration moyenne. On observe que **PM10** et **PM2.5** dominent nettement, avec des valeurs bien plus élevées par rapport aux autres polluants, reflétant leur contribution

significative à la pollution atmosphérique. **NO2** occupe une position intermédiaire, tandis que **SO2** et **CO** présentent les concentrations les plus faibles. Ce graphique met en évidence l'importance des particules fines et grossières comme principaux indicateurs de la qualité de l'air, nécessitant une attention particulière dans les analyses et les politiques de gestion environnementale.

Encodage des données

L'encodage des données est une étape essentielle pour préparer les variables catégorielles à être utilisées dans les algorithmes d'apprentissage automatique, qui fonctionnent avec des valeurs numériques. Dans notre dataset, la variable **Air Quality** est catégoriel, avec quatre classes : **Good**, **Hazardous**, **Moderate**, et **Poor**. Pour permettre une intégration fluide dans notre pipeline, cette variable a été encodée en valeurs numériques.

Nous avons utilisé la technique de **Label Encoding** à l'aide de la bibliothèque **scikit-learn**. Cette méthode attribue un entier unique à chaque classe. Voici le mappage des classes résultant :

- **0 : Good**
- **1 : Hazardous**
- **2 : Moderate**
- **3 : Poor**

Ce choix de **Label Encoding** est approprié car les classes de la variable cible sont ordonnées, reflétant une progression dans la qualité de l'air (de bonne à dangereuse). Ce type d'encodage est rapide, simple à mettre en œuvre, et permet de préserver l'ordre naturel des catégories, ce qui peut être utile pour certains modèles d'apprentissage. Cette étape garantit que la variable **Air Quality** est prêt pour les algorithmes de machine learning tout en minimisant les erreurs potentielles liées au traitement des données catégorielles.

Train-Test Split

La séparation des données en ensembles d'entraînement (**train**) et de test (**test**) est une étape clé pour évaluer la performance de nos modèles d'apprentissage automatique de manière impartiale.

Création des variables X et y

Dans notre cas, nous avons créé :

- **X** : les variables explicatives ou indépendantes, obtenues en excluant la colonne **Air Quality**, car elles contiennent les informations utilisées pour prédire la qualité de l'air.
- **y** : la variable cible ou dépendante, qui correspond à **Air Quality**, car c'est celle que nous souhaitons prédire.

Cette séparation est essentielle pour différencier les données utilisées comme entrée dans le modèle (X) et la sortie attendue (y).

Paramètres de la fonction `train_test_split`

- **`test_size=0.2`** : Nous utilisons 20 % des données pour le test et 80 % pour l'entraînement. Ce ratio est un standard courant qui assure un bon équilibre : suffisamment de données pour entraîner le modèle tout en conservant une part représentative pour l'évaluation.
- **`random_state=42`** : Ce paramètre garantit la reproductibilité des résultats. En fixant une graine aléatoire, la division des données sera toujours identique lors des exécutions, ce qui facilite la comparaison des performances et la validation du modèle.

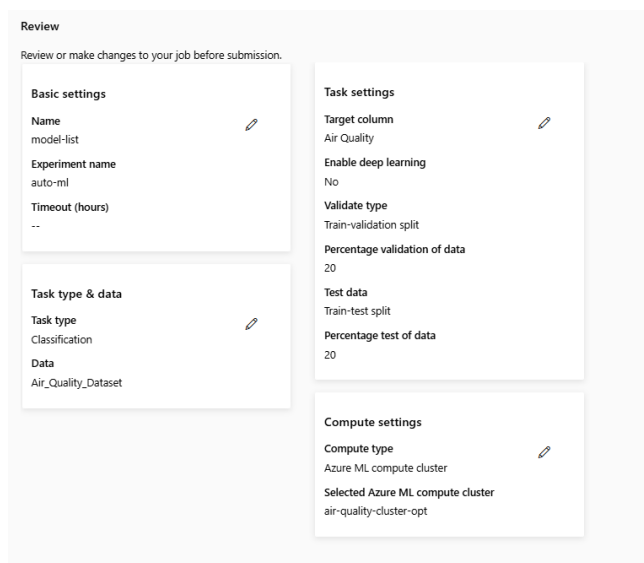
Cette étape garantit que notre modèle est entraîné sur une partie des données (**train**) et évalué sur une autre partie (**test**) qu'il n'a jamais vue, évitant ainsi le surapprentissage et assurant une mesure fiable de ses performances.

III. Model selection.

Dans le cadre de ce projet, l'étape de sélection et d'entraînement du modèle joue un rôle crucial pour garantir des prédictions précises et fiables. L'exploration de différents types de modèles est essentielle dans la résolution de problèmes de classification, car elle permet d'identifier les algorithmes les plus performants pour notre jeu de données spécifique. Cette approche comparative nous aide à évaluer les forces et les faiblesses des modèles, ainsi qu'à sélectionner celui offrant les meilleures performances.

Pour mener cette exploration de manière rapide et efficace, nous avons opté pour l'utilisation d'Azure AutoML. Cet outil puissant d'Azure Machine Learning automatise le processus de sélection de modèles, en testant une large gamme d'algorithmes adaptés à notre problème de classification. Azure AutoML évalue chaque modèle selon des métriques prédéfinies (comme l'accuracy), ce qui nous permet de repérer facilement les modèles les plus performants.

Voici la configuration choisie pour cette étape d'auto-ML :



The screenshot shows the 'Review' stage of an Azure AutoML job. It contains four main configuration panels:

- Basic settings:**
 - Name: model-list
 - Experiment name: auto-ml
 - Timeout (hours): --
- Task type & data:**
 - Task type: Classification
 - Data: Air_Quality_Dataset
- Task settings:**
 - Target column: Air Quality
 - Enable deep learning: No
 - Validate type: Train-validation split
 - Percentage validation of data: 20
 - Test data: Train-test split
 - Percentage test of data: 20
- Compute settings:**
 - Compute type: Azure ML compute cluster
 - Selected Azure ML compute cluster: air-quality-cluster-opt

Dans notre configuration d'Azure AutoML, nous avons fait des choix stratégiques pour garantir une exploration pertinente :

Choix des proportions de validation et de test : Nous avons opté pour une séparation des données avec une proportion de 20 % pour l'ensemble de validation (*train-validation-split*) et 20 % pour l'ensemble de test (*train-test-split*). Cette répartition permet de garantir une quantité suffisante de données pour entraîner le modèle (60 %), tout en réservant une portion équilibrée pour évaluer ses performances sur des données inconnues (validation) et pour mesurer sa capacité à généraliser sur un jeu de test final. Ce choix est particulièrement adapté à un jeu de données de taille moyenne (5000 échantillons), assurant une représentativité des classes dans les ensembles de validation et de test, notamment les catégories moins fréquentes comme "Hazardous".

Métrique principale : Nous avons sélectionné l'Accuracy comme métrique principale car elle mesure la proportion de prédictions correctes sur l'ensemble des données, ce qui est pertinent dans notre cas où, malgré une légère différence de proportions entre les classes, le déséquilibre reste modéré. Cette métrique reflète adéquatement les performances du modèle dans un problème où toutes les classes (Good, Moderate, Poor, Hazardous) doivent être prises en compte de manière équitable.

Modèles autorisés : Nous avons restreint AutoML à tester une sélection spécifique de modèles, comprenant :

Logistic Regression : La régression logistique est un choix pertinent pour démarrer avec un modèle simple et interprétable. Ce modèle permet de modéliser la probabilité d'appartenance à l'une des quatre catégories de qualité de l'air (Good, Moderate, Poor, Hazardous) en fonction des variables explicatives. Bien qu'il soit limité aux relations linéaires, il constitue un excellent point de départ pour analyser les contributions individuelles des variables telles que la température, la proximité des zones industrielles, ou la densité de population. Ce modèle est rapide à entraîner et offre des bases solides pour des analyses plus avancées.

Random Forest, Decision Tree, Extreme Random Trees: Les modèles basés sur des arbres de décision, comme Random Forest, Decision Tree, et Extreme Random Trees, sont adaptés à votre problème en raison de leur capacité à gérer des relations complexes et non linéaires entre les variables. Random Forest combine plusieurs arbres pour réduire le risque de surapprentissage et améliorer la robustesse, tandis qu'Extreme Random Trees accélère la génération de modèles grâce à des seuils aléatoires. Decision Tree, plus simple, est utile pour comprendre les décisions prises par le modèle, offrant une visualisation claire des règles sous-jacentes. Ces modèles s'adaptent bien aux variables continues comme la concentration de polluants ou la distance aux zones industrielles.

K-Nearest Neighbors (KNN) : Le KNN repose sur la similarité entre les points de données et est particulièrement utile si les différentes catégories de qualité de l'air forment des clusters distincts dans l'espace des variables. Avec vos données bien distribuées sur 5000 échantillons, ce modèle peut offrir une bonne performance, surtout après une normalisation appropriée des variables. Cependant, son efficacité diminue avec des données très volumineuses ou lorsque les classes se chevauchent, ce qui pourrait nécessiter des ajustements pour obtenir des résultats optimaux.

Stochastic Gradient Descent (SGD) : L'algorithme SGD est rapide et efficace pour traiter des données de grande dimension. Il est bien adapté pour des modèles linéaires sur des jeux de

données tabulaires et peut être utilisé pour des problèmes de classification multi-classes comme le vôtre. Toutefois, sa performance dépend du bon réglage des hyperparamètres, tels que le taux d'apprentissage, et peut nécessiter plusieurs itérations pour converger vers une solution optimale. Cet algorithme est particulièrement avantageux lorsque les ressources informatiques ou le temps d'entraînement sont limités.

Multinomial Naive Bayes, Bernoulli Naive Bayes: Les modèles Naive Bayes sont rapides à entraîner et conviennent bien à des données catégoriques ou textuelles. Bien que vos données soient principalement continues, ces modèles peuvent être explorés pour vérifier des hypothèses simplifiées ou pour des tâches spécifiques liées à des sous-ensembles de données catégoriques. Leur efficacité dépend toutefois de la distribution sous-jacente des données, et leur performance pourrait être inférieure à celle d'autres modèles plus sophistiqués dans ce contexte.

SVM et Linear SVM : Les Machines à Vecteurs de Support (SVM) sont idéales pour des données complexes où les frontières entre les classes ne sont pas immédiatement évidentes. Linear SVM est un bon choix initial pour évaluer les séparations linéaires, tandis qu'une version avec noyau (non mentionnée ici) pourrait améliorer la performance sur des relations non linéaires. Ce modèle est particulièrement robuste dans des jeux de données à haute dimension, bien qu'il puisse être coûteux en temps d'entraînement avec un ensemble de données de 5000 échantillons.

Gradient Boosting, LightGBM, XGBoostClassifier : Les algorithmes de boosting, comme Gradient Boosting, LightGBM, et XGBoostClassifier, sont parmi les choix les plus performants pour des problèmes de classification complexes. Ces modèles excellent dans la gestion des relations non linéaires, des interactions complexes entre les variables, et des déséquilibres de classes. LightGBM est particulièrement rapide et efficace pour des données volumineuses, tandis que XGBoost est connu pour sa robustesse et sa capacité à généraliser. Ces modèles nécessitent un réglage précis des hyperparamètres, mais leur capacité à gérer des données variées en fait un choix privilégié.

Gestion de l'équilibre des classes :

Avec une distribution déséquilibrée des classes (Good : 2000, Hazardous : 500), certains modèles comme Random Forest et XGBoost peuvent directement ajuster les poids pour traiter cet déséquilibre. Pour les autres modèles comme Logistic Regression ou SVM, des approches prétraitées, telles que le sur-échantillonnage ou le sous-échantillonnage, pourraient être nécessaires pour éviter que les classes majoritaires dominent les prédictions. L'utilisation de ces techniques garantit une classification équitable entre toutes les catégories.

Voici les résultats de l'auto-ML obtenus :

Grâce à Azure auto-ML, nous avons pu gagner du temps et optimiser les efforts nécessaires à cette phase cruciale. L'outil non seulement propose les algorithmes les mieux adaptés, mais il offre également des informations détaillées sur les hyperparamètres utilisés, permettant une optimisation supplémentaire si nécessaire. Cette approche garantit un équilibre entre rapidité d'exécution et qualité des résultats.

IV. Optimisation des hyperparamètres.

Dans ce projet, nous avons utilisé Optuna, une bibliothèque avancée et flexible dédiée à la recherche d'hyperparamètres, pour optimiser les performances des modèles de machine learning. Optuna se distingue par sa capacité à explorer efficacement un vaste espace de recherche grâce à des stratégies avancées telles que l'optimisation bayésienne. Cette approche nous a permis d'identifier les combinaisons d'hyperparamètres les plus performantes pour maximiser les résultats de nos modèles.

L'utilisation d'Optuna s'est imposée en raison des nombreux avantages qu'il offre :

- **Efficacité** : Optuna utilise une recherche guidée pour identifier rapidement les zones prometteuses dans l'espace des hyperparamètres, réduisant ainsi le temps nécessaire à l'optimisation.
- **Flexibilité** : Il prend en charge des types variés de distributions d'hyperparamètres (discrètes, continues, catégoriques) et s'adapte aisément à différents modèles et frameworks.
- **Simplicité d'intégration** : Avec une interface intuitive, Optuna s'intègre facilement aux pipelines existants, notamment ceux basés sur scikit-learn ou d'autres outils de machine learning.

Justification des Modèles Sélectionnés

Les modèles optimisés avec Optuna ont été choisis en s'appuyant sur les résultats préliminaires d'Azure AutoML. Ce dernier a permis d'identifier un sous-ensemble de modèles prometteurs pour notre problématique de classification. Nous avons donc concentré nos efforts sur les modèles suivants :

- **Logistic Regression** : Apprécié pour sa simplicité et son interprétabilité, il offre souvent de bonnes performances pour les problèmes de classification binaire.
- **K-Nearest Neighbors (KNN)** : Idéal pour les données où les relations locales jouent un rôle important.
- **Support Vector Classifier (SVC)** : Efficace pour traiter des données avec des marges bien définies, particulièrement adapté aux problèmes complexes.
- **Random Forest** : Reconnu pour sa robustesse et sa capacité à bien généraliser sur des données complexes.
- **Gradient Boosting** : Excellente option pour capturer des relations non linéaires entre les variables.

Définition et Choix des Hyperparamètres

Pour chaque modèle, les hyperparamètres optimisés ont été choisis en fonction de leur impact sur les performances. Voici les détails des configurations :

Logistic Regression :

- **C** (continu, [0.01, 10.0]) : Régule la pénalisation pour éviter le surapprentissage.

K-Nearest Neighbors (KNN) :

- `n_neighbors` (discret, [3, 15]) : Définit le nombre de voisins à considérer pour la classification, équilibrant biais et variance.

SVC :

- `C` (continu, [0.01, 10.0]) : Régule la marge entre les classes.
- `kernel` (catégorique, ['linear', 'rbf', 'poly']) : Permet de choisir la transformation de données dans un espace à haute dimension.

Random Forest :

- `n_estimators` (discret, [50, 200]) : Nombre d'arbres dans la forêt, influençant la robustesse et la stabilité du modèle.
- `max_depth` (discret, [5, 50]) : Limite la profondeur des arbres pour éviter un surapprentissage.

Gradient Boosting :

- `learning_rate` (continu, [0.01, 0.3]) : Détermine la vitesse d'apprentissage pour éviter une convergence trop rapide ou trop lente.
- `n_estimators` (discret, [50, 200]) : Contrôle le nombre d'étapes de boosting.

Méthodologie d'Optimisation

Chaque modèle a été évalué à l'aide d'une validation croisée (CV=3) en utilisant la métrique Accuracy comme référence. Une fonction d'objectif a été définie dans Optuna pour explorer dynamiquement l'espace des hyperparamètres et maximiser cette métrique. Ce processus a permis une exploration structurée et exhaustive, garantissant des résultats reproductibles.

Résultats

L'optimisation menée avec Optuna a permis d'identifier le modèle et les hyperparamètres offrant les meilleures performances pour notre problématique. Le modèle retenu est le Random Forest avec les hyperparamètres suivants :

- `n_estimators` : 88
- `max_depth` : 10

Ce modèle a atteint une **accuracy de 95,50%** sur les données de validation. Ces résultats témoignent de la robustesse et de la capacité de généralisation du modèle optimisé, confirmant son adéquation à notre problématique.

Conclusion

Grâce à l'utilisation d'Optuna, nous avons optimisé les performances des modèles en ajustant précisément leurs hyperparamètres. Cette démarche rigoureuse a permis d'améliorer l'accuracy d'environ 0.1 par rapport aux résultats obtenus avec l'AutoML, tout en adoptant une approche structurée et reproductible pour la sélection des hyperparamètres optimaux. Ce processus d'optimisation a non seulement validé le choix final du modèle pour le processus d'entraînement, mais a également assuré l'utilisation d'hyperparamètres optimaux. Cependant,

il est important de noter que la différence relativement faible entre les résultats d'Optuna et ceux de l'AutoML suggère que, dans le cadre spécifique de nos données et de notre problème de classification, l'AutoML pourrait constituer une solution suffisamment performante pour répondre aux besoins de ce problème de classification.

V. Processus d'entraînement et d'évaluation

Après avoir identifié Random Forest comme le meilleur modèle à l'aide de la recherche d'hyperparamètres avec Optuna, nous avons procédé à l'entraînement et à l'évaluation de ce modèle en exploitant les hyperparamètres optimaux trouvés.

Intégration de MLflow pour le suivi des expériences :

Étant donné que le projet est développé sur Azure Machine Learning, le choix d'utiliser MLflow pour le suivi des expériences s'est imposé naturellement. MLflow permet :

- **Traçabilité complète** : Tous les hyperparamètres, métriques, et artefacts (comme les graphiques de matrice de confusion) sont enregistrés automatiquement, facilitant la reproductibilité.
- **Visualisation intégrée** : Les résultats des expériences peuvent être visualisés directement dans le portail Azure ML.
- **Flexibilité** : MLflow s'intègre parfaitement avec Azure ML, offrant une gestion centralisée des modèles.

Nous avons activé l'autologging de MLflow pour capturer automatiquement les détails de l'entraînement et des évaluations.

Ajout de métriques supplémentaires :

En complément de l'accuracy, nous avons calculé les métriques suivantes pour évaluer plus précisément les performances du modèle :

- **Recall** : Le recall, évalue la capacité du modèle à détecter toutes les occurrences positives d'une classe donnée. Dans le contexte de la qualité de l'air, il est essentiel de ne pas manquer des zones classées comme "Poor" ou "Hazardous", car cela pourrait empêcher la mise en œuvre d'interventions vitales. Une bonne valeur de recall garantit que le modèle est sensible aux risques et capable d'identifier efficacement les régions nécessitant une attention particulière, même si elles sont sous-représentées dans les données.
- **F1-Score** : Le F1-score est la moyenne harmonique entre la précision et le recall, offrant un équilibre précieux entre ces deux métriques. Il est particulièrement pertinent pour les classes déséquilibrées comme "Hazardous," où il est crucial de maintenir à la fois une haute précision pour éviter les fausses alertes et un bon rappel pour capturer les cas critiques. En combinant ces deux aspects, le F1-score fournit une vue d'ensemble des performances du modèle pour chaque classe, ce qui est indispensable dans des situations où les erreurs ont des implications réelles.
- **Support** : Le support représente le nombre d'échantillons réels de chaque classe dans le jeu de test, fournissant un contexte important pour interpréter les métriques. Par

exemple, une classe comme "Moderate," avec un faible support, pourrait influencer les moyennes globales des métriques comme l'accuracy ou le F1-score. Comprendre le support permet d'ajuster notre analyse et de mieux juger les performances du modèle, en tenant compte de la répartition déséquilibrée des données. Cela garantit que les classes minoritaires ne sont pas négligées lors de l'évaluation.

- **Precision** : La précision mesure la proportion de prédictions positives correctes parmi toutes les prédictions positives faites par le modèle. Dans le cadre de notre projet, une précision élevée est essentielle pour éviter les fausses alertes, en particulier pour les classes critiques comme "Hazardous". Cela permet d'assurer que les alertes émises par le modèle sont fiables et exploitables, réduisant ainsi les interventions inutiles.
- **Log Loss (Logarithmic Loss)** : La log loss évalue la probabilité que le modèle attribue à chaque classe, pénalisant fortement les prédictions incorrectes avec une confiance élevée. Cette métrique est particulièrement utile pour mesurer la calibration du modèle, c'est-à-dire sa capacité à exprimer une confiance appropriée dans ses prédictions. Dans notre contexte, une faible log loss indique que le modèle est bien calibré et évite les surestimations ou les sous-estimations.
- **AUC (Area Under the Curve)** : L'AUC, basée sur la courbe ROC, mesure la capacité du modèle à distinguer correctement entre les classes. Avec un problème impliquant des classes critiques comme "Poor" ou "Hazardous", l'AUC fournit une évaluation robuste des performances globales, même en cas de déséquilibre des classes. Une AUC proche de 1 démontre une excellente séparation entre les classes et garantit que le modèle est capable d'identifier efficacement les zones à risque.

Ces métriques offrent une vision complète des performances, particulièrement utile dans des cas de classes déséquilibrées, comme dans notre problème.

Résultats et interprétations

Après l'entraînement et l'évaluation de notre modèle Random Forest optimisé, plusieurs métriques ont été calculées pour analyser les performances sur le jeu de test. Ces résultats permettent d'évaluer la précision globale du modèle ainsi que sa capacité à prédire correctement la qualité de l'air pour chaque catégorie.

Résultats globaux logger par MLflow :

train_accuracy 1	train_log_loss 0.03978298	train_f1_score 1	train_auc 1	train_recall 1	train_precision 1
test_accuracy 0.963	test_log_loss 0.1311294	test_f1_score 0.9630179	test_auc 0.9967642	test_recall 0.963	test_precision 0.9630952

- **Accuracy (96.3%)** : L'accuracy indique que 96.3% des échantillons du jeu de test ont été correctement classifiés. Ce résultat global témoigne de la robustesse du modèle dans la prise en compte des différents facteurs influençant la qualité de l'air.

- **Log Loss (0.1311)** : Un faible log loss montre que le modèle est bien calibré, attribuant des probabilités cohérentes aux prédictions pour chaque classe. Ce point est particulièrement crucial pour des cas sensibles comme les catégories "Poor" et "Hazardous", où une sous-estimation ou une surestimation pourrait entraîner des conséquences graves.
- **F1-Score (96.3%)** : Avec une moyenne pondérée de 96.3%, le F1-score reflète un équilibre solide entre précision et rappel pour toutes les classes. Cela est particulièrement important dans un contexte où les classes sont légèrement déséquilibrées.
- **AUC (99.7%)** : L'AUC quasi parfaite souligne l'excellente capacité du modèle à distinguer entre les différentes classes, même en cas de déséquilibre dans les données.
- **Recall (96.3%) et Precision (96.3%)** : Ces deux métriques, avec des scores élevés, démontrent que le modèle est à la fois sensible (capable de détecter les cas pertinents) et précis (réduisant les fausses alertes).

Analyse par classe :

Testing Metrics by Class:				
	precision	recall	f1-score	support
Class 0	1.00	1.00	1.00	409
Class 1	0.93	0.90	0.91	111
Class 2	0.97	0.97	0.97	294
Class 3	0.89	0.91	0.90	186
accuracy			0.96	1000
macro avg	0.95	0.94	0.95	1000
weighted avg	0.96	0.96	0.96	1000

Le rapport par classe montre les performances spécifiques du modèle pour chaque catégorie de qualité de l'air :

Class 0 (Good) :

- **Precision et Recall : 1.00 (parfaits)** : La classe "Good" est prédite avec une exactitude totale, reflétant une forte dominance de cette catégorie dans les données, ce qui a permis au modèle de l'apprendre efficacement.
- **F1-Score : 1.00** : Aucun compromis entre précision et rappel n'est nécessaire pour cette classe, étant donné ses performances parfaites.

Class 1 (Hazardous) :

- **Precision : 0.93, Recall : 0.90, F1-Score : 0.91** : Bien que légèrement moins performante que la classe précédente, la catégorie "Hazardous" présente des scores respectables. Le modèle montre une bonne capacité à capturer cette classe tout en limitant les erreurs.
- **Support : 111** : Un faible nombre d'échantillons rend cette classe plus difficile à prédire, ce qui explique des métriques légèrement inférieures.

Class 2 (Moderate) :

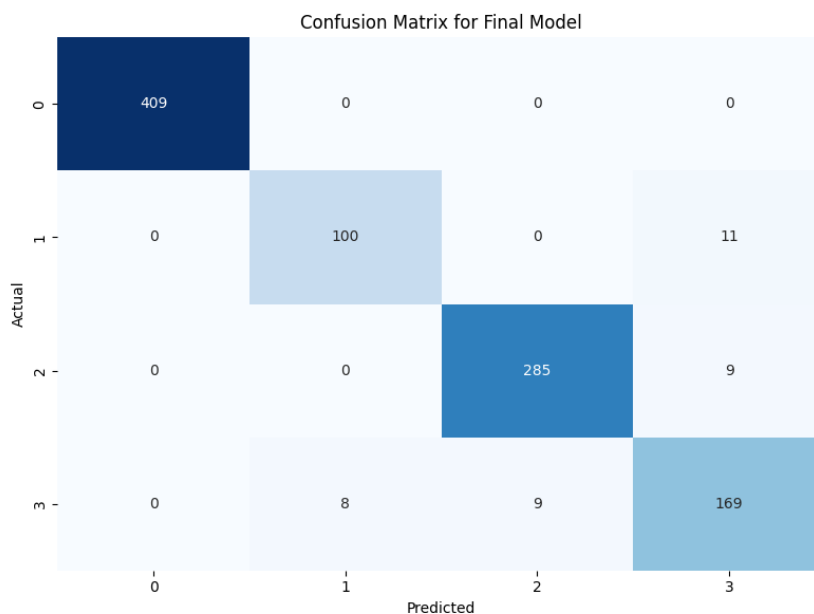
- **Precision : 0.97, Recall : 0.97, F1-Score : 0.97** : Les résultats élevés pour cette classe montrent que le modèle est fiable pour identifier les zones modérément polluées, ce qui est crucial pour des interventions préventives.

Class 3 (Poor) :

- **Precision : 0.89, Recall : 0.91, F1-Score : 0.90** : Bien que cette classe soit la plus difficile à prédire, le modèle maintient une bonne capacité à détecter les cas critiques, tout en minimisant les fausses alertes. Cela est crucial pour éviter des conséquences graves liées à des prédictions incorrectes.
- **Support : 186** : Le modèle gère efficacement cette classe malgré son importance relative moindre dans les données.

Interprétation des Graphiques logger par MLflow :

Matrice de Confusion :



La matrice de confusion visualise les performances du modèle pour chaque classe. Voici une analyse détaillée des résultats :

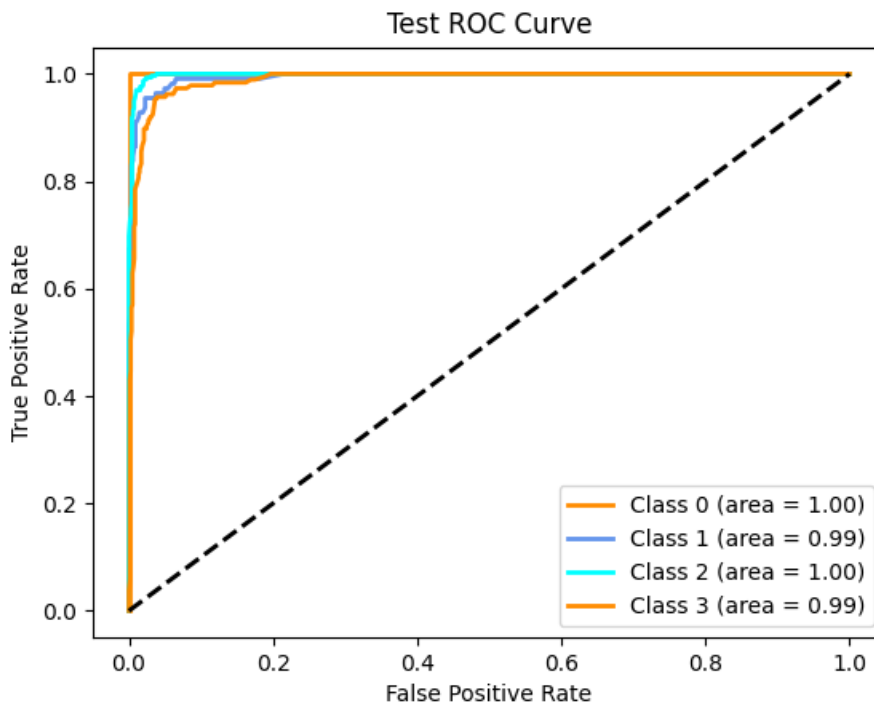
Classe 0 (Good) : Le modèle prédit parfaitement cette classe, avec 409 prédictions correctes et aucune erreur (aucune valeur dans les autres colonnes). Cela démontre que le modèle est bien formé pour identifier les cas où la qualité de l'air est "Good," probablement en raison de leur forte représentation dans le jeu de données.

Classe 1 (Hazardous) : Avec 100 prédictions correctes et 8 cas mal classifiés comme "Poor," le modèle montre une légère difficulté à différencier les classes "Hazardous" et "Poor." Cela pourrait être dû à des similarités dans les caractéristiques des échantillons de ces classes.

Classe 2 (Moderate) : Le modèle montre une excellente capacité à identifier les cas "Moderate," avec 285 prédictions correctes. Seuls 9 cas ont été incorrectement classifiés comme "Poor," indiquant une faible confusion entre ces deux classes.

Classe 3 (Poor) : Bien que cette classe soit la plus difficile à prédire, le modèle parvient à obtenir 169 prédictions correctes. Les erreurs sont réparties avec 11 échantillons classés à tort comme "Hazardous" et 9 comme "Moderate." Ces erreurs peuvent être liées à des variations dans les facteurs environnementaux ou démographiques proches des seuils de classification.

Courbe AUC-ROC :



La courbe ROC pour chaque classe indique la capacité du modèle à distinguer correctement cette classe des autres, basée sur des seuils de probabilité. Voici l'interprétation des résultats :

Classe 0 (Good) : Avec une AUC de 1.00, le modèle démontre une séparation parfaite entre les échantillons "Good" et les autres classes. Cela reflète une très bonne représentation des données de cette classe et une distinction claire dans l'espace des caractéristiques.

Classe 1 (Hazardous) : L'AUC de 0.99 pour "Hazardous" montre une performance excellente, bien que non parfaite. Cela est crucial dans un contexte où une mauvaise classification pourrait empêcher des interventions critiques.

Classe 2 (Moderate) : Une AUC de 1.00 reflète une séparation nette entre "Moderate" et les autres classes. Cette précision souligne la capacité du modèle à capturer les zones modérément polluées, souvent essentielles pour des interventions préventives.

Classe 3 (Poor) : Une AUC de 0.99 pour "Poor" indique que le modèle est également très performant pour identifier cette classe, mais il reste des améliorations possibles, en particulier pour réduire les confusions avec les classes adjacentes comme "Hazardous."

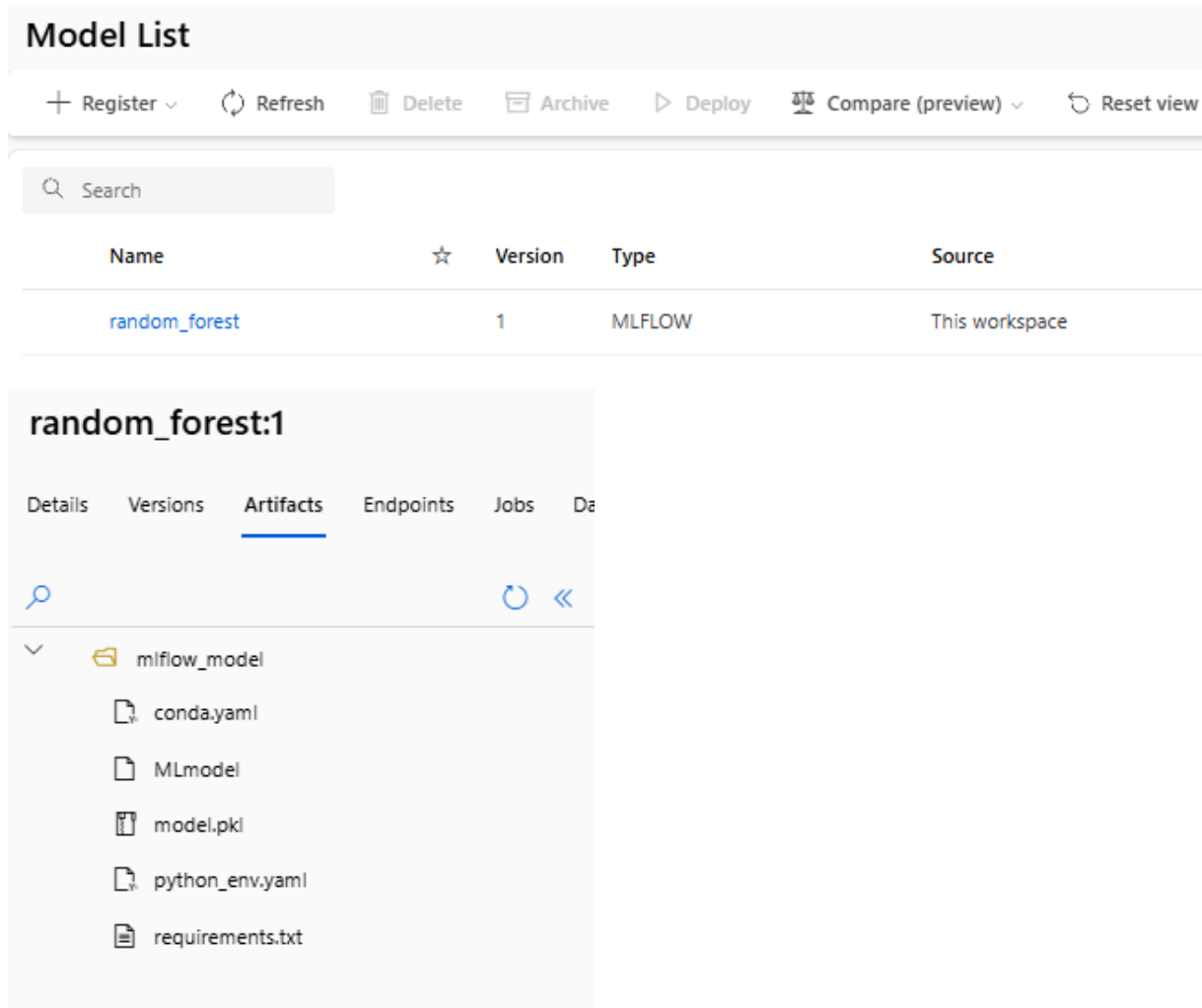
Les résultats obtenus mettent en évidence la robustesse du modèle Random Forest optimisé pour prédire la qualité de l'air. Avec une **accuracy globale de 96.3%**, un **F1-score pondéré élevé** et une **AUC quasi parfaite (99.7%)**, le modèle démontre sa capacité à classer

efficacement les différentes catégories de qualité de l'air, même dans un contexte de déséquilibre des classes. Les prédictions pour les classes majoritaires comme "Good" sont parfaites, tandis que les classes plus critiques comme "Hazardous" et "Poor" affichent des performances respectables malgré quelques confusions. Ces résultats soulignent la fiabilité du modèle pour identifier les zones à risque et soutenir des décisions éclairées en matière de gestion de la pollution atmosphérique.

VI. Étapes de déploiement et implémentation de l'interface Web.

La mise en production du modèle constitue une étape clé de ce projet, permettant son intégration dans un système utilisable pour des analyses en temps réel. Voici les étapes suivies pour le déploiement et l'implémentation de l'interface web :

Sauvegarde et Enregistrement du Modèle



Model List

+ Register ▾ Refresh Delete Archive Deploy Compare (preview) ▾ Reset view

Search

Name	☆	Version	Type	Source
random_forest		1	MLFLOW	This workspace

random_forest:1

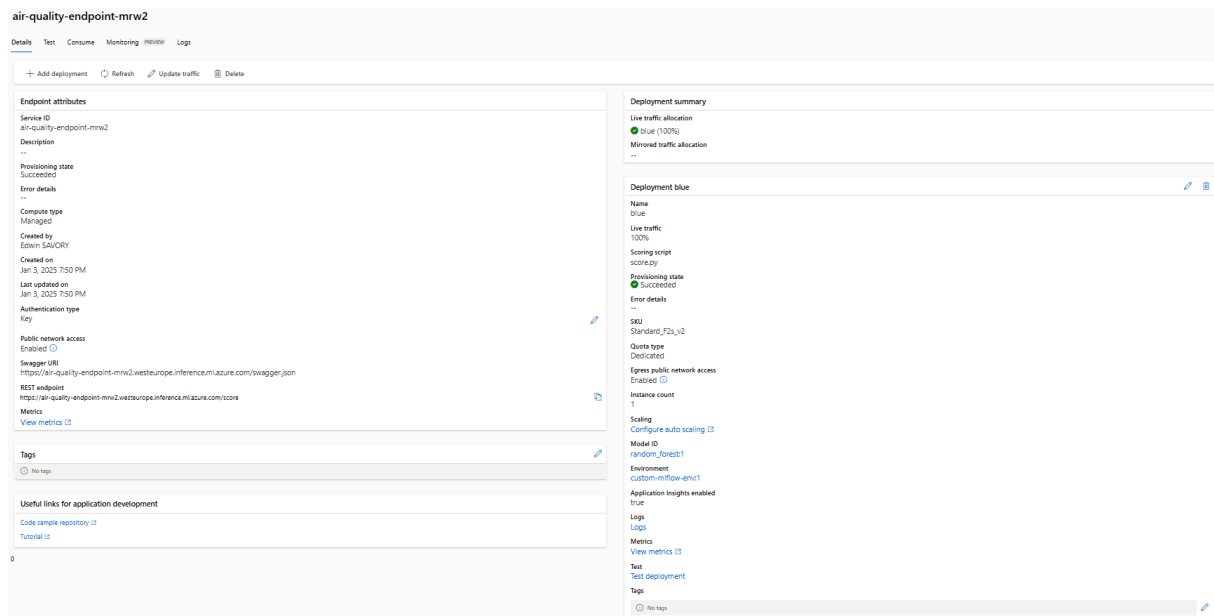
Details Versions Artifacts Endpoints Jobs Da

mlflow_model

- conda.yaml
- MLmodel
- model.pkl
- python_env.yaml
- requirements.txt

Le modèle final, optimisé avec Optuna, est sauvegardé au format MLflow, un choix standard et portable, parfaitement intégré avec Azure Machine Learning, garantissant une traçabilité complète incluant les hyperparamètres, les métriques d'entraînement et les artefacts. Il est ensuite enregistré sur Azure Machine Learning Model sous un nom unique basé sur ses hyperparamètres, simplifiant ainsi sa gestion et sa réutilisation dans divers environnements.

Création d'un Endpoint et déploiement en Ligne



The screenshot displays the Azure ML portal interface for an online endpoint. The left sidebar shows the 'Endpoint attributes' section with details such as Service ID, Provisioning state (Succeeded), and Authentication type (Key). The right sidebar shows the 'Deployment summary' and 'Deployment blue' configuration, including Name, Live traffic allocation (100%), Scoring script (score.py), Provisioning state (Succeeded), SKU (Standard_F2s_v2), and Environment (custom-mlflow-emp1).

Un endpoint en ligne unique, généré dynamiquement pour éviter les conflits, a été créé en utilisant un Managed Online Endpoint sur Azure ML, offrant une gestion simplifiée de l'infrastructure et permettant de déployer le modèle en production sans se soucier de l'administration des serveurs ou de la scalabilité. Pour ce déploiement, le modèle a été hébergé sur le endpoint à l'aide d'un environnement personnalisé MLflow et d'un script de scoring spécifique, avec une instance Standard_F2s_v2 sélectionnés pour offrir un bon compromis entre performance et coût dans cette première version de production. Enfin, tout le trafic a été routé vers le déploiement principal "blue", et Application Insights a été activé pour collecter en temps réel les métriques et les logs, facilitant ainsi le suivi des performances sur des données réelles et l'identification rapide des éventuels problèmes en production.

Implémentation de l'Interface Web

L'interface web constitue un pont entre l'utilisateur final et le modèle déployé, permettant de soumettre des données et de recevoir des prédictions sur la qualité de l'air de manière intuitive et rapide. Voici les étapes suivies pour sa conception et son implémentation :

Utilisation de Streamlit :

L'interface web a été développée en Streamlit, un framework Python léger et rapide à mettre en œuvre pour créer des applications web interactives. Le choix de Streamlit repose sur plusieurs facteurs :

- **Simplicité** : Une syntaxe intuitive et directe permet un développement rapide.
- **Visualisation interactive** : Offre des composants prêts à l'emploi (formulaires, graphiques) idéaux pour l'affichage et la manipulation des données.
- **Intégration native avec Python** : Facilite l'intégration avec le modèle et les bibliothèques Python existantes, évitant les surcharges techniques inutiles.

Fonctionnalités de l'Interface :

Air Quality Prediction Interface

Enter the necessary data to predict air quality using the deployed model.

User Inputs

Temperature (°C)

-30,0

Humidity (%)

30,0

PM2.5 Concentration ($\mu\text{g}/\text{m}^3$)

0.5

PM10 Concentration ($\mu\text{g}/\text{m}^3$)

0.3

NO₂ Concentration (ppb)

0.6

SO₂ Concentration (ppb)

0.8

CO Concentration (ppm)

0,60

Proximity to Industrial Areas (km)

30,0

Population Density (people/km²)

10

Predict

Air Quality Predictions:

prediction : class_index : 0, class_label : "Good", latency_seconds : 0.008066

L'application web présente un formulaire intuitif permettant à l'utilisateur d'entrer des paramètres environnementaux tels que la température, l'humidité et les concentrations de polluants atmosphériques. Une fois les données saisies, elles sont transmises au modèle via un endpoint Azure Machine Learning pour obtenir une prédiction.

- **Saisie des Données :** L'utilisateur peut saisir les paramètres nécessaires (température, humidité, PM2.5, PM10, etc.) grâce aux composants interactifs fournis par Streamlit.
- **Appel au Modèle :** Les données saisies sont formatées et transmises via une requête POST au modèle déployé sur Azure Machine Learning. Les appels sont réalisés à l'aide d'une clé API chargée directement depuis le script app.py.
- **Affichage des Résultats :** La prédiction est affichée de manière claire, avec des messages explicites en cas d'erreur (ex. problème d'authentification ou d'appel API).

Utilisation de ngrok :

Pour rendre l'interface accessible, ngrok a été utilisé afin de créer un tunnel sécurisé vers un serveur local. Ce choix offre plusieurs avantages :

- **Facilité de configuration** : Permet d'exposer rapidement l'application Streamlit sur une URL publique sans configuration complexe.
- **Sécurité** : Fournit des connexions HTTPS sécurisées, essentielles pour protéger les échanges de données sensibles.
- **Flexibilité** : Idéal pour des démonstrations ou des tests temporaires sans déploiement complet.

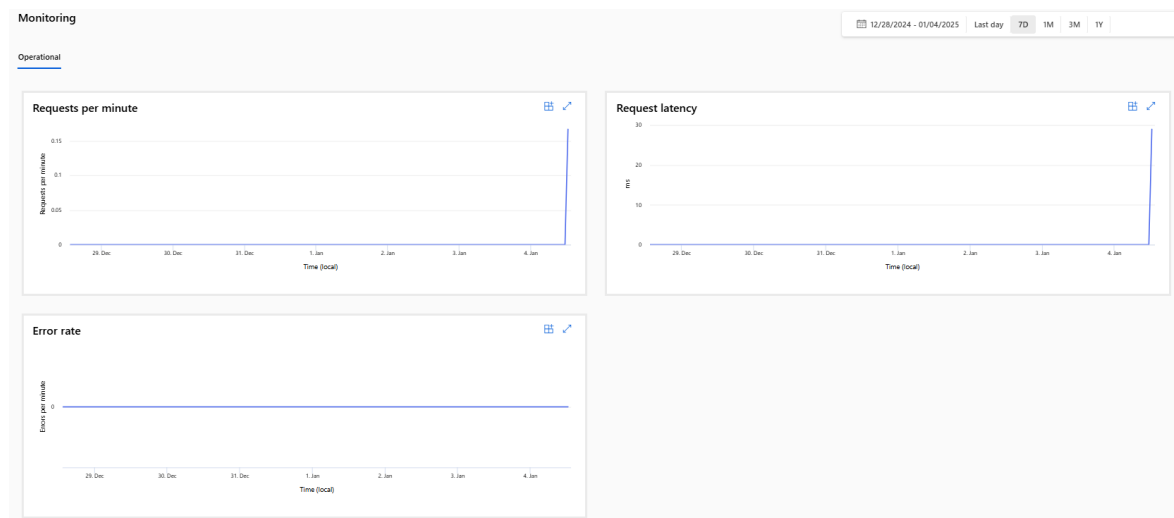
L'interface web simplifie l'accès au modèle et favorise son adoption par des utilisateurs non techniques, tout en assurant une intégration fluide avec Azure Machine Learning et une mise en production rapide grâce à Streamlit et ngrok. Elle constitue une solution robuste, interactive et sécurisée pour la prédiction de la qualité de l'air.

Monitoring de l'endpoint

Le suivi du déploiement est une étape cruciale pour garantir la fiabilité et les performances du modèle en production. Azure Machine Learning fournit une interface de monitoring intuitive accessible directement depuis la rubrique "Monitoring" de l'endpoint.

Dashboards de Visualisation

L'interface de monitoring offre plusieurs tableaux de bord en temps réel permettant de visualiser les métriques clés :

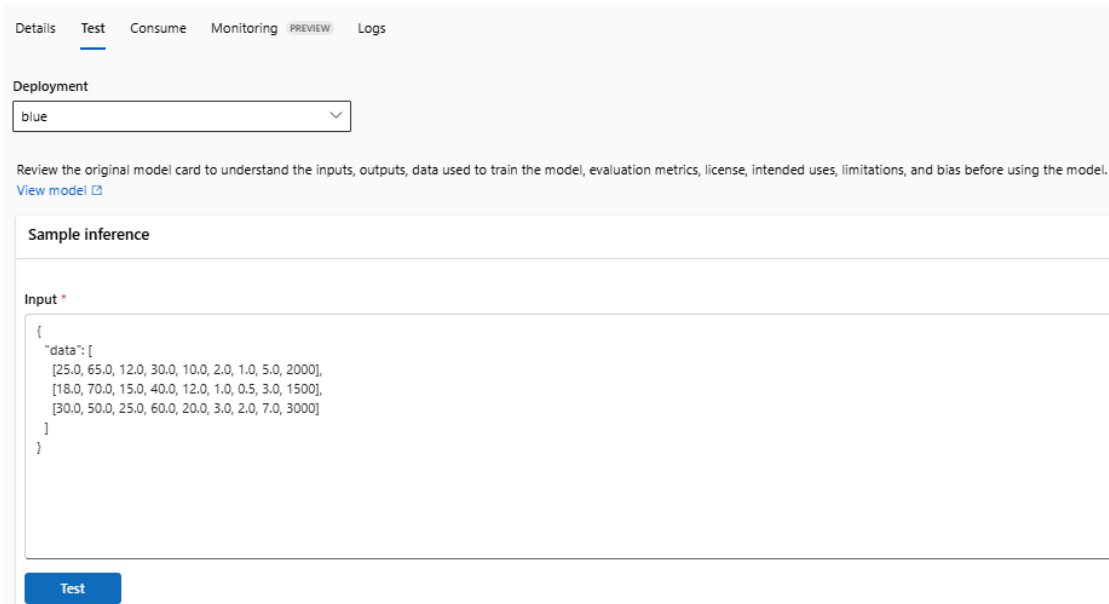


- **Request per minute** : Affiche le nombre de requêtes traitées par minute, utile pour évaluer la charge de travail du modèle.
- **Request latency** : Permet de surveiller le temps de réponse moyen des prédictions.
- **Error rate** : Suit le taux d'erreurs afin d'identifier rapidement d'éventuels problèmes.

Ces visualisations aident à détecter les anomalies et à optimiser les performances en fonction des besoins en production.

Test de l'Endpoint

L'onglet "Test" permet de simuler des requêtes avec des données d'entrée représentatives. Par exemple, en utilisant l'entrée suivante :



Details **Test** Consume Monitoring PREVIEW Logs

Deployment
blue

Review the original model card to understand the inputs, outputs, data used to train the model, evaluation metrics, license, intended uses, limitations, and bias before using the model.
[View model](#)

Sample inference

Input *

```
{
  "data": [
    [25.0, 65.0, 12.0, 30.0, 10.0, 2.0, 1.0, 5.0, 2000],
    [18.0, 70.0, 15.0, 40.0, 12.0, 1.0, 0.5, 3.0, 1500],
    [30.0, 50.0, 25.0, 60.0, 20.0, 3.0, 2.0, 7.0, 3000]
  ]
}
```

Test

Le modèle retourne les prédictions avec la latence de chaque requête ainsi que la latence totale pour toutes les prédictions, comme illustré ci-dessous :

jsonOutput

```
{
  "predictions": [
    "prediction : class_index : 2, class_label : \"Moderate\", latency_seconds : 0.00273",
    "prediction : class_index : 2, class_label : \"Moderate\", latency_seconds : 0.00273",
    "prediction : class_index : 2, class_label : \"Moderate\", latency_seconds : 0.00273"
  ],
  "total_latency_seconds": 0.008396
}
```

Ce retour permet d'analyser les performances précises du modèle, avec un focus sur la rapidité des prédictions et leur cohérence.

Analyse des Logs

L'onglet "Logs" fournit un suivi détaillé et en temps réel des interactions avec l'endpoint, incluant :

- Les requêtes reçues.
- Les prédictions réalisées.
- Les éventuelles erreurs rencontrées.

Voici un exemple de logs capturés lors de l'exécution :

```
INFO:entry_module:Received input data: {'data': [[25, 65, 12, 30, 10, 2, 1, 5, 2000], [18, 70, 15, 40, 12, 1, 0.5, 3, 1500], [30, 50, 25, 60, 20, 3, 2, 7, 3000]]}
INFO:entry_module:Predictions made: ['prediction : class_index : 2, class_label : "Moderate", latency_seconds : 0.00549', ...], total_latency=0.0166s
INFO:entry_module:custom_dimensions: {'prediction_latency': 0.01647, 'total_latency': 0.01663, 'input_data_size': 3}}
```

Ces informations permettent de :

- Suivre la performance globale du modèle.
- Identifier les éventuels goulets d'étranglement dans le traitement des requêtes.
- Analyser les erreurs pour une résolution rapide.

Le monitoring intégré d'Azure Machine Learning, couplé à ces outils de test et de suivi, assure une gestion robuste et une amélioration continue du déploiement en production.

VII. Résultats, difficultés rencontrées et axes d'amélioration.

Résultats

Les résultats de ce projet se distinguent par une solution complète et structurée, comprenant plusieurs composantes clés :

Application Streamlit interactive :

Une interface utilisateur intuitive développée en Streamlit, permettant d'interagir en temps réel avec un endpoint pour prédire la qualité de l'air à partir de divers facteurs environnementaux. Cette interface simplifie l'accès au modèle pour des utilisateurs non techniques.

Notebook explicatif : `data_exploration_to_deployment.ipynb`

Un notebook Python détaillé documentant l'intégralité de la solution, étape par étape :

- Création de l'environnement personnalisé.
- Chargement, exploration et analyse des données.
- Traitement des données.
- Recherche des hyperparamètres et entraînement du modèle.
- Évaluation du modèle.
- Déploiement du modèle sur Azure ML Model.
- Création et paramétrage du endpoint, ainsi que son test.
- Développement de l'interface utilisateur Streamlit et mise à jour du endpoint.

Ce notebook constitue une documentation interactive permettant de comprendre facilement chaque composante de la solution.

Script Python : `data_exploration_to_deployment.py`

Conversion structurée du notebook en script Python, organisé sous forme de pipeline modulaire avec des fonctions bien définies, compilées dans un fichier `main.py`. Ce script suit les étapes de création de l'environnement jusqu'au déploiement et paramétrage du endpoint. Les indicateurs de suivi permettent de visualiser l'avancement des différentes étapes directement dans le terminal.

Autres fichiers et scripts associés :

- **score.py** : Utilisé pour la configuration et le scoring du endpoint.
- **app.py** : Gère l'interface utilisateur Streamlit, connectée au endpoint.
- **test_input.json** : Fichier d'exemples de données pour tester le endpoint.
- **Dossier pipeline** : Contient le code d'un pipeline Azure (non fonctionnel) avec les jobs nécessaires pour automatiser les différentes étapes.

Nos résultats démontrent une solution robuste et bien documentée, permettant à la fois une compréhension globale et une utilisation pratique en production. Cette organisation modulaire facilite également les améliorations futures et l'intégration de nouvelles fonctionnalités.

Difficultés rencontrées

Intégration d'Application Insights

Malgré l'activation d'Application Insights sur le endpoint et la vérification de la connectivité via les fichiers de configuration JSON, aucune métrique n'a été récupérée dans l'interface KQL. Plusieurs solutions ont été testées pour résoudre ce problème, notamment :

- Passage à un environnement curated.
- Création d'un nouveau endpoint.
- Configuration via CLI.
- Modification de la région des compute instances.

Malheureusement, aucune de ces approches n'a abouti.

Problème de pipeline Azure

Lors de la création d'un pipeline dans Azure Jobs, bien que l'image Docker soit correctement construite, un problème de souscription empêchait les opérations (comme la récupération de données) à cause de droits IAM insuffisants sur les compute instances. Malgré l'ajout des rôles Contributeur, Owner, et d'autres préconisés dans la documentation Azure, le problème persiste.

En attendant une résolution, le pipeline se limite à un script Python combinant toutes les étapes (environnement, déploiement, configuration du endpoint) dans un fichier principal, réduisant sa modularité.

Améliorations potentielles

Résolution des problèmes techniques :

- Résoudre les difficultés liées à Application Insights pour améliorer la visibilité sur le monitoring du endpoint.
- Corriger les erreurs du pipeline Azure pour une meilleure modularité et flexibilité.

Optimisation du modèle :

Bien que le modèle Random Forest optimisé avec Optuna atteigne une précision de 96 %, des techniques avancées de machine learning pourraient être explorées pour résoudre les déséquilibres dans la distribution des classes.

Sécurisation des informations sensibles :

Éviter de stocker la clé primaire et l'URL REST directement dans le script Streamlit en utilisant un fichier de configuration, des arguments, ou des solutions de stockage sécurisé proposées par Azure.

Adoption des bonnes pratiques FinOps :

Mettre en place un suivi des coûts pour optimiser l'utilisation des ressources, notamment les compute instances et les clusters. Cela serait essentiel pour anticiper les évolutions nécessitant davantage de ressources.

Ces améliorations visent à renforcer la robustesse, la sécurité et l'efficacité globale de la solution, tout en assurant une meilleure maîtrise des coûts.

Conclusion

Malgré les difficultés rencontrées au cours de ce projet, notamment le temps consacré à résoudre certains problèmes techniques qui, dans certains cas, n'ont pas abouti à une solution définitive, nous avons réussi à livrer une solution fonctionnelle, performante et interactive. Grâce à l'application Streamlit, reliée à notre endpoint déployé sur Azure Machine Learning, nous avons pu proposer une interface utilisateur conviviale permettant d'effectuer des prédictions en temps réel sur la qualité de l'air.

La réalisation de ce projet sur Azure a également été une opportunité précieuse pour renforcer nos compétences existantes sur cette plateforme dans le domaine de la data science. En explorant les fonctionnalités clés d'Azure Machine Learning, nous avons abordé des aspects fondamentaux tels que la préparation des données, la construction et l'optimisation de modèles, le suivi des expérimentations, et le déploiement de solutions prêtes pour la production. Ce projet s'inscrit parfaitement dans les apprentissages théoriques et pratiques proposés par la certification DP-100 de Microsoft Learn, consolidant ainsi notre expertise dans la mise en œuvre de pipelines d'apprentissage automatique complets et professionnels.