IE7374 Machine Learning Operations (MLOps)

Project Scoping

Ozone Level Detection

Team Members:

- Ramana Siddanth Emani
- Sameera Mandalika
- Vivekbhai Ashokbhai Radadiya
- Bhuvan Channagiri
- Nikita Vinod Mandal

---

1. Introduction

This project focuses on the analysis and prediction of ozone levels to improve air quality management in the Houston, Galveston, and Brazoria areas. Ozone, a major component of smog, is known for its harmful effects on human health and the environment. Elevated ozone levels can lead to a variety of respiratory issues, including difficulty breathing, increased susceptibility to respiratory infections, and aggravation of chronic respiratory diseases such as asthma. Additionally, prolonged exposure to high ozone levels can result in reduced lung function and long-term lung damage. The environmental impact of elevated ozone levels is equally significant. Ozone can cause damage to crop, forests, and other vegetation, leading to reduced agricultural productivity and compromised natural ecosystems. Sensitive plant species can experience stunted growth, leaf damage, and increased vulnerability to disease and pests. The cumulative effect of these impacts can disrupt ecological balances and reduce biodiversity.

Given the importance of maintaining optimal air quality for public health and environmental sustainability, our project aims to develop an advanced machine learning model capable of accurately forecasting peak ozone levels. This model will analyze historical atmospheric data, including meteorological factors and pollutant concentrations, to predict future ozone levels with high precision. By providing accurate and timely predictions, the model will enable environmental agencies and public health officials to implement proactive measures to mitigate the adverse effects of high ozone levels. For instance, predictions of elevated ozone levels can trigger health advisories, encouraging vulnerable populations to take precautionary measures such as staying indoors or limiting outdoor activities. Additionally, the model can inform regulatory actions, such as temporary emission reductions, to prevent ozone levels from reaching harmful thresholds.

To achieve these goals, we will employ sophisticated prediction techniques, leveraging machine learning algorithms that can capture complex patterns and relationships within the data. The model will be trained on a comprehensive dataset that includes various atmospheric variables, ensuring it can generalize well to different scenarios and conditions.

Ultimately, our project strives to create a robust and reliable tool for air quality management. By enhancing the ability to predict and respond to ozone pollution, we aim to contribute to the well-being of communities and the preservation of natural ecosystems in the Houston, Galveston, and Brazoria areas. This initiative aligns with broader efforts to address air pollution and promote sustainable development, reflecting our commitment to leveraging technology for the betterment of society and the environment.

2. Dataset Information

The dataset includes records from 1998 to 2004 for eight-hour and one-hour peak ozone levels. It contains 72 features and 2536 instances, focusing on atmospheric conditions such as temperature, wind speed, and humidity.

- **Data Card:**
  - I. **Size:** 2536 instances, 72 features
  - II. **Format:** Multivariate, Sequential, Time-Series
  - III. **Data Types:** Numerical values (e.g., temperature, wind speed, humidity)
  - IV. **Collection Period:** 1998 – 2004
  - V. **Target Variable:** Ozone_Level
  - VI. **License:** Public domain, available for research purposes

- **Data Sources: Ozone Level Detection - UCI Machine Learning Repository**

- **Data Rights and Privacy:** The dataset is publicly available and can be used for research purposes. It adheres to general data protection principles but does not have specific privacy concerns.

3. Data Planning and Splits

Data preprocessing and planning involve several detailed steps:

Handling Missing Values: Identify missing values in the dataset. Impute missing values using techniques such as mean, median, or mode substitution. For more complex scenarios, advanced imputation methods like k-NN or regression imputation will be used.

Normalizing Features: Apply normalization techniques to ensure all features contribute equally to the model. Common methods include Min-Max Scaling and Z-score normalization.

Feature Engineering: Create new features based on domain knowledge, such as interaction terms, temporal features (e.g., time of day, season), and derived metrics.

Data Splitting: Split the dataset into training (70%), validation (15%), and test (15%) sets. Stratified sampling will be used to maintain the distribution of the target variable across splits, ensuring comprehensive model evaluation.

4. GitHub Repository

This repository provides access to the project's source code, workflow definitions, and documentation, ensuring transparency and facilitating collaboration among team members and stakeholders. Detailed instructions for accessing the full dataset and logs, stored securely in Google Cloud Storage (GCS), are provided in the README.md file.

Link: **https://github.com/SiddanthEmani/Ozone_Level_Detection**

Folder Structure:

dags/: Contains Directed Acyclic Graph (DAG) definitions for Apache Airflow, organizing the workflow steps for data preprocessing and model training.
src/: Source code, including scripts for data preprocessing, feature engineering, and model training.
tests/: Unit and integration tests to ensure the correctness of the pipeline components.
.dvc/: DVC (Data Version Control) files for tracking data versions and pipeline stages.
README.md: Essential project information, setup instructions, and usage guidelines.
requirements.txt: List of required Python packages for the project.

5. Data Storage

Data and logs will be stored in Google Cloud Storage (GCS) to ensure scalability, security, and efficient data management. The GitHub repository will include only the source code and synthetic sample data for demonstration purposes. Detailed instructions for accessing the full dataset from GCS will be provided in the README.md file, ensuring that sensitive data is managed appropriately, and storage practices comply with best standards.

Data and logs for the "Ozone Level Detection" project are stored securely in Google Cloud Storage (GCS). This approach ensures scalability, security, and efficient data management. The use of GCS allows for the following benefits:

Scalability: GCS can handle large volumes of data, making it suitable for storing extensive datasets and logs generated during the project.

Security: GCS provides robust security features, including encryption at rest and in transit, access control through IAM (Identity and Access Management), and audit logging to monitor data access.

Efficiency: Storing data in GCS allows for seamless integration with other Google Cloud services, such as BigQuery for data analysis, AI Platform for model training, and Cloud Functions for serverless processing.

Accessing the Data:

To access the full dataset and logs, team members can follow the instructions provided in the README.md file of the GitHub repository. These instructions include details on setting up Google Cloud SDK, obtaining the necessary permissions, and downloading or uploading data to the GCS bucket.

GCS Bucket Structure:

raw_data/: Contains raw data files as received from data sources.

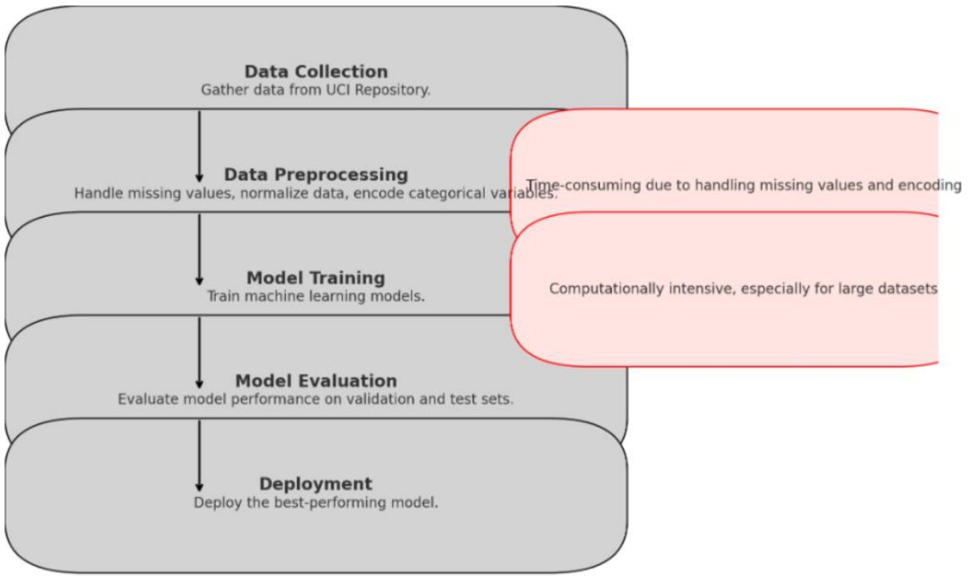
processed_data/: Contains processed data files ready for analysis or model training.

logs/: Stores log files generated during data processing and model training.

models/: Stores trained models and model artifacts for deployment.

By using GCS for data storage, we ensure that the project adheres to best practices for data management, providing a robust, secure, and efficient environment for handling all project-related data.

6. Flow Chart & Bottleneck Detection:


Data Collection
Gather data from UCI Repository.
Data Preprocessing
Handle missing values, normalize data, encode categorical variables.
Time-consuming due to handling missing values and encoding
Model Training
Train machine learning models.
Computationally intensive, especially for large datasets
Model Evaluation
Evaluate model performance on validation and test sets.
Deployment
Deploy the best-performing model.

Bottleneck Detection:

Data Collection: This phase can face bottlenecks due to inconsistent data formats and sources. When data comes from various systems or sources with different formats, integrating and processing it can become challenging and time-consuming. Ensuring data consistency and compatibility is crucial to avoid delays in downstream processes like model training.

Model Training: Bottlenecks during model training often stem from the high computational resources required, especially for complex models. Training these models can consume a lot of time and computational power, which can be a bottleneck in the development and deployment cycle. Techniques like distributed computing or using specialized hardware (like GPUs) are often employed to mitigate these bottlenecks.

Real-Time Predictions: For applications requiring real-time predictions, the challenge lies in maintaining low-latency processing. This becomes a bottleneck when dealing with streaming data, where predictions need to be made quickly as new data arrives. Optimization techniques such as model simplification, efficient algorithms, and infrastructure setup (like scalable cloud architectures) are used to achieve low-latency predictions.

7. Metrics, Objectives, & Business Goals

Metrics:

Mean Squared Error (MSE):

Definition: MSE measures the average squared difference between the observed (actual) values and the predicted values by the model.

Use: It quantifies the overall quality of predictions. Lower MSE indicates better model performance, with values closer to zero indicating better accuracy.

Root Mean Squared Error (RMSE):

Definition: RMSE is the square root of MSE, providing a measure of the average magnitude of the errors in the same units as the target variable.

Use: RMSE is easier to interpret than MSE since it gives error magnitude directly in the scale of the target variable. Achieving a low RMSE is typically a specific goal in predictive modeling tasks.

R^2 Score (Coefficient of Determination):

Definition: R^2 score indicates the proportion of variance in the target variable that is explained by the model. It ranges from 0 to 1, where 1 indicates a perfect fit.

Use: This metric helps assess how well the model captures the variability in the data. Higher R^2 scores indicate better model performance.

Objectives:

Achieve an RMSE below 10 ppb (parts per billion):

Specific Objective: This objective sets a clear target for model performance in terms of predicting ozone levels accurately. It defines a threshold (10 ppb) that the model needs to meet or exceed to be considered effective.

Develop a deployable model for real-time ozone level prediction:

Specific Objective: This objective focuses on the practical application of the model. It emphasizes developing a model that can process incoming data in real-time and make predictions promptly, which is crucial for operational deployment.

Business Goals:

Enhance public health safety by providing accurate ozone level predictions:

Business Impact: Accurate predictions help authorities and the public take proactive measures to mitigate health risks associated with high ozone levels, such as issuing alerts or adjusting outdoor activities.

Support environmental agencies in monitoring air quality effectively:

Business Impact: By providing reliable predictions, the model supports environmental agencies in their mission to monitor and manage air quality. This can lead to better-informed policy decisions and resource allocation.

8. Failure Analysis

i. Data Quality Issues:

- Description: Data quality issues such as missing or incorrect data can significantly impact the accuracy and reliability of the predictive model.
- Impact: Poor data quality can lead to biased or misleading insights, affecting decision-making based on model predictions.
- Mitigation Strategies:
  - Robust Data Cleaning: Implement thorough data cleaning processes to detect and handle missing values, outliers, and inconsistencies.
  - Data Validation: Use validation techniques to ensure data integrity throughout the preprocessing pipeline.
  - Data Augmentation: Where possible, augment existing data or generate synthetic data to improve model training robustness.

ii. Model Overfitting:

- Description: Overfitting occurs when a model learns noise or random fluctuations in the training data, leading to poor generalization on unseen data.
- Impact: Overfitted models may perform exceptionally well on training data but fail to generalize to new data, resulting in poor predictive performance.
- Mitigation Strategies:
  - Cross-Validation: Use techniques like k-fold cross-validation to assess model performance on multiple subsets of the data, ensuring it generalizes well.
  - Regularization: Apply regularization techniques (e.g., L1, L2 regularization) to penalize overly complex models and promote simpler, more generalizable models.
  - Dropout: Incorporate dropout regularization during model training to randomly deactivate neurons, reducing the model's reliance on specific features and improving robustness.

iii. Deployment Challenges:

- Description: Deployment challenges arise when transitioning a model from development to a production environment, especially in real-time applications.
- Impact: Inefficient deployment can lead to latency issues, scalability problems, or operational failures, hindering the model's utility in real-world scenarios.
- Mitigation Strategies:
  - Containerization (Docker): Package the model and its dependencies into containers to ensure consistency across different environments and facilitate easy deployment.
  - Orchestration (Kubernetes): Use orchestration tools like Kubernetes to manage containerized applications, ensuring scalability, resilience, and efficient resource utilization.
  - Continuous Monitoring: Implement monitoring tools and practices to track model performance, detect anomalies, and ensure timely maintenance and updates.

Importance:

Addressing these potential failure points and implementing robust mitigation strategies is crucial for the success and reliability of the predictive modeling project:

- Data Quality: Ensuring clean, reliable data is fundamental to building accurate models that reflect real-world conditions.
- Model Overfitting: Preventing overfitting ensures that models generalize well to new data, enhancing their predictive power and reliability.
- Deployment Efficiency: Efficient deployment in real-time environments ensures that the model delivers timely predictions, meeting operational needs effectively.

By proactively identifying and mitigating these challenges, stakeholders can enhance the model's performance, reliability, and impact on decision-making in air quality management and public health safety.

9. Monitoring Plan

Metrics to Monitor:

Model Accuracy: Track RMSE and R^2 score regularly to ensure the model performs as expected.

Prediction Latency: Monitor the time taken to generate predictions to ensure real-time capability.

System Uptime: Ensure the system is operational and monitor downtime to maintain reliability.

Tools:

Prometheus: For collecting and querying metrics.

Grafana: For visualizing and analyzing metrics.

ELK Stack: For logging and analyzing system logs.

Actions:

Set thresholds for each metric (e.g., RMSE < 10 ppb, latency < 1 second).

Implement automated alerts for metrics exceeding predefined thresholds, enabling timely interventions.\

10. Success and Acceptance Criteria

Success Criteria:

Accuracy: Achieve an RMSE below 10 ppb.

Reliability: Consistent model performance with an R^2 score above 0.8.

Timeliness: Real-time prediction capability with latency under 1 second.

Acceptance Criteria:

The model meets or exceeds defined accuracy and reliability thresholds.

Real-time predictions are consistently delivered within the specified latency.

The system demonstrates stability and scalability in the production environment.

11. Timeline

- Week 1-2: Data collection and preprocessing.
- Week 3-4: Model training and evaluation.
- Week 5: Model deployment and testing.
- Week 6: Monitoring setup and final adjustments.

12. Additional Information

Further details, including extended literature review and stakeholder analysis, will be included as necessary.