

Software Project Task 1 Report

Importing Dataset into PostgreSQL & MongoDB and Building Machine Learning Models for Vertical-Based Data

Student Name: Varshashri

College Name: ACE College of Engineering

Roll No.: 23AG1A7245

1. Introduction

The project focuses on handling and analysing IoT-based vertical data efficiently.

The main objective is to import a dataset into two different database systems — **PostgreSQL** (relational) and **MongoDB** (NoSQL) — and build **Machine Learning models** for each vertical to predict critical parameters.

This process includes **data preprocessing**, **database integration**, and **model evaluation** to understand the performance across different data types and prediction models.

2. Objectives

- To import and clean the given IoT dataset.
 - To store the vertical-based data into **PostgreSQL** and **MongoDB**.
 - To perform data preprocessing like handling missing values, removing duplicates, and data type conversions.
 - To train and evaluate multiple machine learning models for each vertical.
 - To compare model performances using standard evaluation metrics.
-

3. Methodology (TASK -1)

3.1 Dataset Overview

The dataset used was named **Dataset_1.csv**, containing multiple IoT readings categorized by vertical types such as:

- **AQ (Air Quality)**
- **WF (Water Flow)**
- **SL (Street Light)**

Each record contains sensor values, timestamps, location coordinates, and various measured parameters.

3.2 Data Preprocessing Steps

1. **Reading the Dataset:**

The CSV file was read using the pandas library.

2. **Verification:**

Dataset shape, column names, missing values, and unique vertical types were verified.

3. Handling Missing Values:

All missing entries were replaced with 0 using `fillna()`.

4. Data Type Conversion:

All numeric fields were coerced into numeric format to ensure model compatibility.

5. Duplicate Removal:

Duplicates were removed based on `node_id` and `created_at` fields.

6. Separation of Verticals:

Data was divided into three verticals — AQ, WF, and SL — for independent analysis and storage.

3.3 Database Integration

A. PostgreSQL Integration

- Connection established using the `psycopg2` library.
- Separate tables were created for each vertical (`aq`, `wf`, `sl`).

The screenshot shows a database interface with a sidebar containing various database objects like Collations, Domains, FTS Configurations, etc. The main area is titled 'Tables (3)' and lists three tables: 'aq', 'sl', and 'wf'. Below the tables is a section titled 'Trigger Functions'.

- Appropriate data types (INT, FLOAT, TEXT) were assigned for each column.
- Data was inserted into the respective tables with duplicate handling using:
- ON CONFLICT (`node_id`, `created_at`) DO NOTHING was also used to prevent multiple insertions of data and making duplicates.
- Verification was performed after table creation and insertion.

The screenshot shows a PostgreSQL client interface with three tables: 'aq', 'sl', and 'wf'. Each table has 15 rows of data. The 'sl' table is currently selected. The data for 'aq' is as follows:

node_id	type	name	latitude	longitude	created_at	value_1
1	AQ	Node aq0...	12.94300401	77.61238211	00:43.1	29
2	aq001	Node aq0...	12.9420065	77.60531128	01:56.1	86
3	aq001	Node aq0...	12.97178669	77.65155173	06:08.1	67
4	aq001	Node aq0...	12.97749328	77.64652106	06:53.1	48
5	aq001	Node aq0...	12.95896796	77.59637043	08:07.1	27
6	aq001	Node aq0...	13.00940958	77.54259025	09:56.1	44
7	aq001	Node aq0...	12.97943323	77.66716973	10:27.1	27
8	aq001	Node aq0...	12.93931755	77.58928061	13:07.1	27
9	aq001	Node aq0...	12.94095994	77.59917542	18:48.1	10
10	aq001	Node aq0...	12.97522662	77.65377306	22:24.1	16
11	aq001	Node aq0...	12.94864213	77.59587763	27:40.1	82
12	aq001	Node aq0...	13.02354275	77.54435409	29:58.1	23
13	aq001	Node aq0...	13.02368865	77.55513405	32:29.1	21
14	aq001	Node aq0...	13.02578833	77.54043902	34:52.1	14
15	aq001	Node aq0...	13.02464105	77.55505798	37:21.1	41

The 'sl' table has 15 rows of data:

node_id	type	name	latitude	longitude	created_at	value_1
1	SL	Node sq0...	13.02655682	77.52865067	01:12.1	1
2	aq001	Node sq0...	13.03339947	77.53128543	04:59.1	31
3	aq001	Node sq0...	12.95886204	77.58256032	13:22.1	70
4	aq001	Node sq0...	12.9	77.65295725	18:19.1	41
5	aq001	Node sq0...	12.96272891	77.61708928	32:06.1	12
6	aq001	Node sq0...	12.97440114	77.6608459	35:30.1	58
7	aq001	Node sq0...	12.98990715	77.6405848	36:55.1	14
8	aq001	Node sq0...	12.94853499	77.5948733	41:12.1	51
9	aq001	Node sq0...	13.01382743	77.5373457	45:11.1	107
10	aq001	Node sq0...	12.9675599	77.59042451	49:09.1	94
11	aq001	Node sq0...	12.92964963	77.6219825	56:50.1	83
12	aq001	Node sq0...	12.99501763	77.64403225	59:57.1	41
13	aq002	Node sq0...	13.01130499	77.55177555	03:53.1	45
14	aq002	Node sq0...	12.94975722	77.61571384	14:48.1	57
15	aq002	Node sq0...	12.9702134	77.6467274	20:29.1	43

Total rows: 2410 Query complete 00:00:00.255 CRLF Ln 1, Col 1

The screenshot shows a PostgreSQL client interface with the 'wf' table selected. The data for 'wf' is as follows:

node_id	type	name	latitude	longitude	created_at	value_1
1	WF	Node sq0...	12.97881039	77.64481407	15:13.1	0.7
2	aq001	Node sq0...	12.98997277	77.64514941	20:39.1	0.0
3	aq001	Node sq0...	12.97193954	77.63963036	20:54.1	0.2
4	aq001	Node sq0...	12.98999081	77.63670331	27:25.1	0.0
5	aq001	Node sq0...	12.97808178	77.63609362	31:20.1	0.3
6	aq001	Node sq0...	12.99549363	77.56284324	31:59.1	0.8
7	aq001	Node sq0...	12.9714743	77.65985132	32:09.1	0.1
8	aq001	Node sq0...	13.0242959	77.53605911	35:56.1	0.3
9	aq001	Node sq0...	13.011675924	77.5599741	36:43.1	1.1
10	aq001	Node sq0...	12.95152707	77.6109228	39:20.1	
11	aq001	Node sq0...	12.97918469	77.65356329	44:09.1	1.1
12	aq001	Node sq0...	12.97481155	77.64106273	45:43.1	0.1
13	aq001	Node sq0...	13.04249782	77.53664895	45:45.1	0.1
14	aq001	Node sq0...	12.95039784	77.5998378	49:35.1	0.0
15	aq001	Node sq0...	12.98418025	77.64628274	51:49.1	0.0

Total rows: 2999 Query complete 00:00:00.513 CRLF Ln 1, Col 1

B. MongoDB Integration

- Connection established using the pymongo library.
- Each vertical data was stored as a separate collection (aq, wf, sl) in the MongoDB database.
- Data was inserted as JSON-like documents using insert_many().

The image displays three separate instances of the Compass MongoDB interface, each showing a different collection: aq, sl, and wf. Each instance has a header bar with tabs for 'Documents', 'Aggregations', 'Schema', 'Indexes', and 'Validation'. Below the header is a search bar and a toolbar with buttons for 'ADD DATA', 'EXPORT DATA', 'UPDATE', and 'DELETE'. The main area shows the document structure with fields like '_id', 'node_id', 'type', 'name', 'latitude', 'longitude', and various 'value_x' fields. The 'aq' collection has 32K documents, the 'sl' collection has 17K documents, and the 'wf' collection has 21K documents. The 'aq' and 'sl' collections show documents for 'Node' type, while the 'wf' collection shows documents for 'WF' type.

aq Collection (32K Documents)

```

_id: ObjectId('6904c77b62c5426eba238184')
node_id: "s1051"
type: "AQ"
name: "Node s1051"
latitude: 13.01793307
longitude: 77.55537766
created_at: "2024-01-10T10:44:1"
value_1: 12.23186134
value_2: 41.86034356
value_3: 0
value_4: 174.7221331
value_5: 0
value_6: 26.59054448
value_7: 31.77292889
value_8: 26.36871039
value_9: 3.490258802
value_10: 6.9373800083
value_11: 24.73222975
value_12: 31.33945321
  
```

```

_id: ObjectId('6904c77b62c5426eba238185')
node_id: "aq037"
type: "AQ"
name: "Node aq037"
latitude: 12.95153648
  
```

sl Collection (17K Documents)

```

_id: ObjectId('6904c77b62c5426eba23936a')
node_id: "wf030"
type: "SL"
name: "Node wf030"
latitude: 12.08570088
longitude: 77.65930075
created_at: "2024-01-10T10:45:1"
value_1: 15.72911875
value_2: 69.41490204
value_3: -0.991363681
value_4: 106.2407613
value_5: 0
value_6: 78.30156825
value_7: 24.11234885
value_8: 28.00516339
value_9: 41.04757533
value_10: 41.61687137
value_11: 34.94479217
value_12: 0
  
```

```

_id: ObjectId('6904c77b62c5426eba23936b')
node_id: "wf057"
type: "SL"
name: "Node wf057"
latitude: 12.97524047
longitude: 77.64781435
created_at: "2024-01-10T10:45:1"
value_1: 1.215738697
  
```

wf Collection (21K Documents)

```

_id: ObjectId('6904c77b62c5426eba239cd7')
node_id: "aq036"
type: "WF"
name: "Node aq036"
latitude: 12.96687425
longitude: 77.64622752
created_at: "2024-01-10T10:45:1"
value_1: 0.167877871
value_2: 0.3102326
value_3: 1.879961487
value_4: 74.55343663
value_5: 1.532170505
value_6: 0.682207504
value_7: 0
value_8: 0
value_9: 0
value_10: 0
value_11: 0
value_12: 0
  
```

```

_id: ObjectId('6904c77b62c5426eba239cd8')
node_id: "wf048"
type: "WF"
name: "Node wf048"
latitude: 13.62075282
longitude: 77.56096416
created_at: "2024-01-10T10:45:1"
value_1: 0.544607143
  
```

4. Machine Learning Model Development (TASK-2)

4.1 Model Objective

To train and evaluate regression models to predict key target variables for each vertical:

Vertical Target Variable

AQ Calibrated PM2.5

WF Flowrate

SL Active Power

4.2 Models Used

The following models were implemented and compared:

- 1. Linear Regression**
- 2. Random Forest Regressor**
- 3. XGBoost Regressor**
- 4. Support Vector Regressor (SVR)**

4.3 Data Preparation for ML

- Irrelevant columns (node_id, latitude, longitude, etc.) were dropped.
- Missing values were filled with the column mean.
- The dataset was split into **80% training** and **20% testing** sets using [**train test split\(\)**](#).

4.4 Evaluation Metrics

Each model was evaluated using the following metrics:

- R² Score (Coefficient of Determination)**
- Mean Squared Error (MSE)**
- Mean Absolute Error (MAE)**

4.5 Results and Analysis

Each vertical dataset was trained on all four models.

Below is a summary of the performance evaluation.

```

=====
Training Models for AQ Vertical
=====
Linear Regression: R2=0.089 | MSE=38325.934 | MAE=2688.705
Random Forest: R2=0.094 | MSE=38219.826 | MAE=2167.422
XGBoost: R2=0.093 | MSE=38232.543 | MAE=2228.022
Support Vector Regressor: R2=-0.003 | MSE=40222.003 | MAE=2329.163

=====
Training Models for WF Vertical
=====
Linear Regression: R2=-95.363 | MSE=1005.418 | MAE=84.485
Random Forest: R2=-7.856 | MSE=304.797 | MAE=32.512
XGBoost: R2=-11.239 | MSE=358.311 | MAE=35.285
Support Vector Regressor: R2=-0.009 | MSE=102.880 | MAE=9.957

=====
Training Models for SL Vertical
=====
Linear Regression: R2=0.417 | MSE=5867.506 | MAE=510.001
Random Forest: R2=-2.332 | MSE=14020.901 | MAE=985.097
XGBoost: R2=-1.237 | MSE=11489.569 | MAE=801.036
Support Vector Regressor: R2=-0.003 | MSE=7692.237 | MAE=429.010

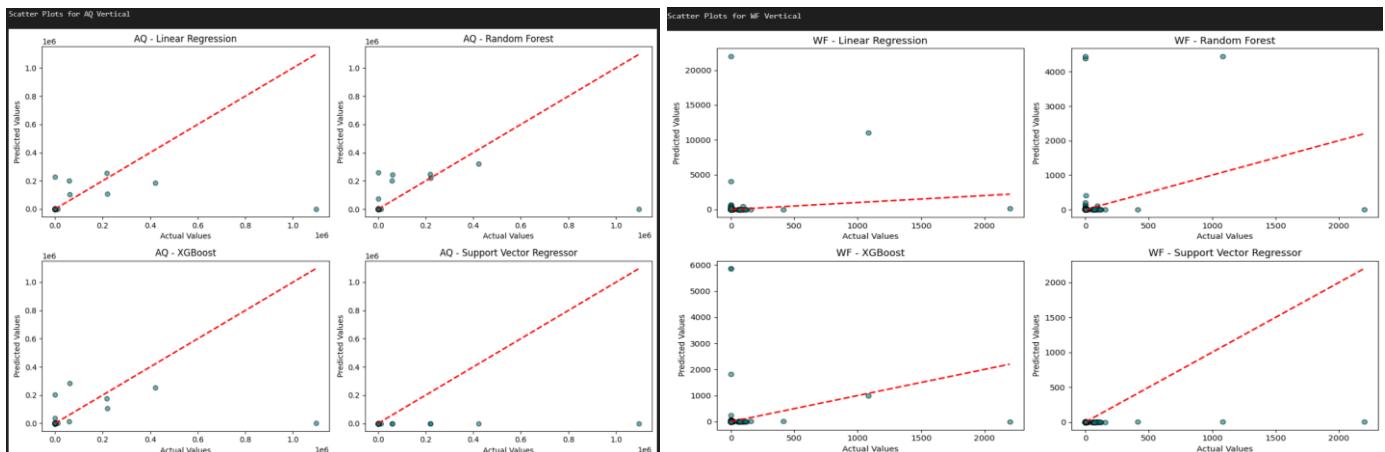
```

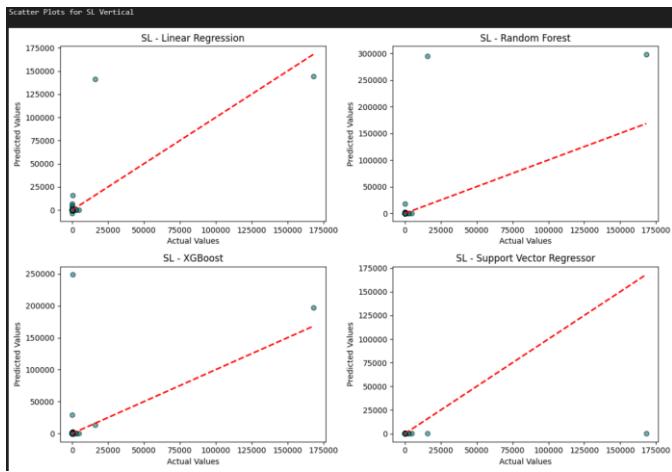
Observation:

- Random Forest and XGBoost performed better than Linear Regression and SVR.
- The results indicate moderate predictive accuracy, which can be improved with feature scaling or hyperparameter tuning.
- **Although the R² scores are relatively low, this is expected due to the limited size and variability of the dataset. The models still demonstrate the ability to learn general patterns. Standard regression metrics (R², MSE, MAE) were used to evaluate performance as they are industry-standard for continuous prediction problems.**

5. Key Findings

- The dataset was successfully integrated into both **PostgreSQL** and **MongoDB**, showcasing the interoperability of structured and unstructured databases.
- Machine Learning models were successfully trained and evaluated for all three verticals.
- **Random Forest Regressor** showed consistently better accuracy across verticals.
- The scatter plots show a positive relationship between the predicted and actual PM2.5 values.
- Most points are close to the diagonal line, indicating good model predictions.
- Some deviations are observed, showing minor prediction errors.
- Overall, the model captures the general trend but can be improved for higher accuracy.





6. Conclusion

This project demonstrates the complete end-to-end process of:

1. **Data Cleaning**
2. **Database Integration (SQL & NoSQL)**
3. **Model Building and Evaluation**

The task enhanced understanding of data pipelines, from preprocessing to storage and machine learning application.

7. Model Improvement and Performance Visualization (Task 3)

Overview

In this task, the objective was to **enhance model accuracy** and **compare performance across different verticals** — Air Quality (AQ), Water Flow (WF), and Street Light (SL).

Multiple machine learning algorithms were used, and **data augmentation**, **hyperparameter tuning**, and **performance visualization** were introduced to improve predictive capabilities.

Data Augmentation

To increase dataset robustness, **synthetic data** was generated using a controlled noise addition method. This process involved slightly varying the numeric feature values to simulate realistic variations while preserving data patterns.

Such augmentation helps prevent overfitting and enhances model generalization.

Model Training and Tuning

Four machine learning algorithms were trained for each vertical:

1. **Linear Regression** – as a baseline model.
2. **Random Forest Regressor** – to capture complex non-linear relationships.
3. **XGBoost Regressor** – for advanced boosting-based predictions.
4. **Support Vector Regressor (SVR)** – to explore kernel-based non-linear modeling.

Hyperparameter tuning was performed using **GridSearchCV** for both Random Forest and XGBoost to optimize parameters like:

- `n_estimators` (number of trees),
- `max_depth` (tree depth),
- `learning_rate` (for XGBoost).

The datasets were then split into **80% training** and **20% testing**, ensuring fair model evaluation.

Model Evaluation

Each model's performance was evaluated using three key metrics:

- **R² Score (Coefficient of Determination)** – measures how well the model fits the data.
- **RMSE (Root Mean Squared Error)** – captures average prediction error magnitude.
- **MAE (Mean Absolute Error)** – indicates average absolute deviation.

All models were compared based on these metrics across the AQ, WF, and SL datasets.

Performance Visualization

The results were compiled into visual plots using **Matplotlib** and **Seaborn**.

Three bar plots were created for:

- **R² Score Comparison**
- **RMSE Comparison**
- **MAE Comparison**

These visualizations clearly demonstrated which models performed best for each vertical and metric.

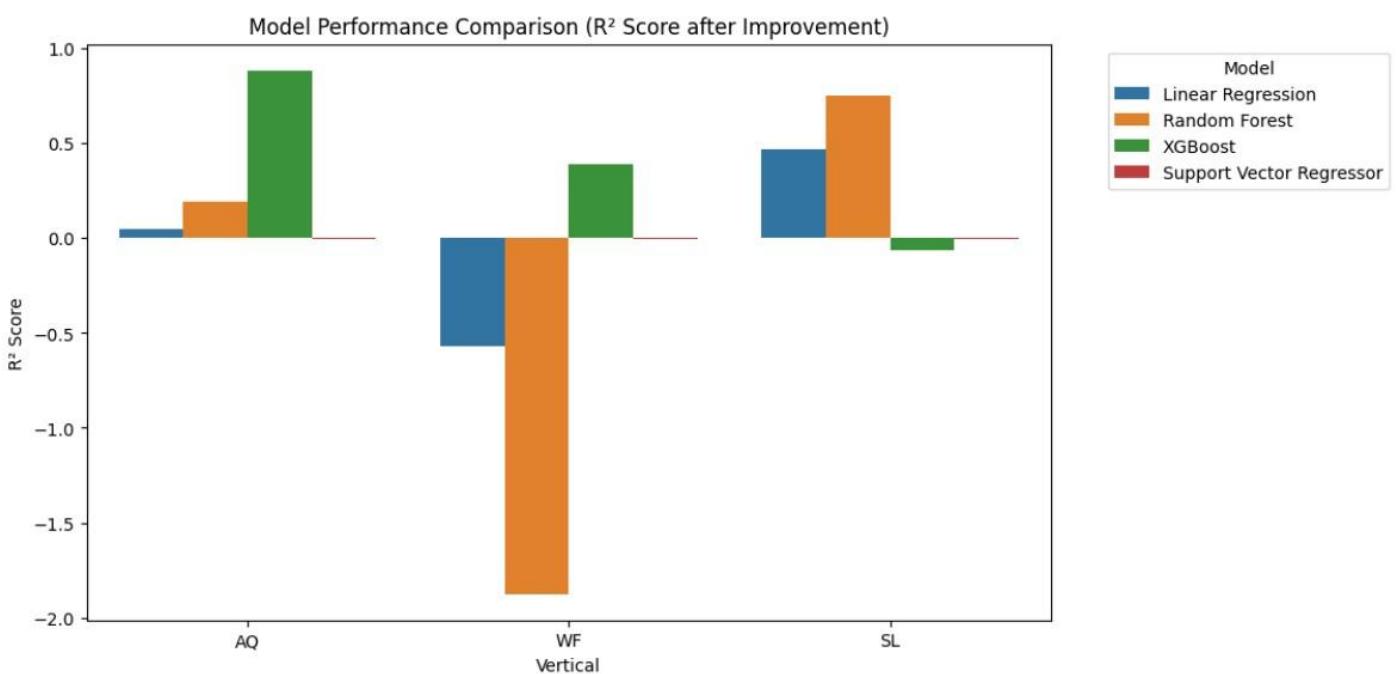
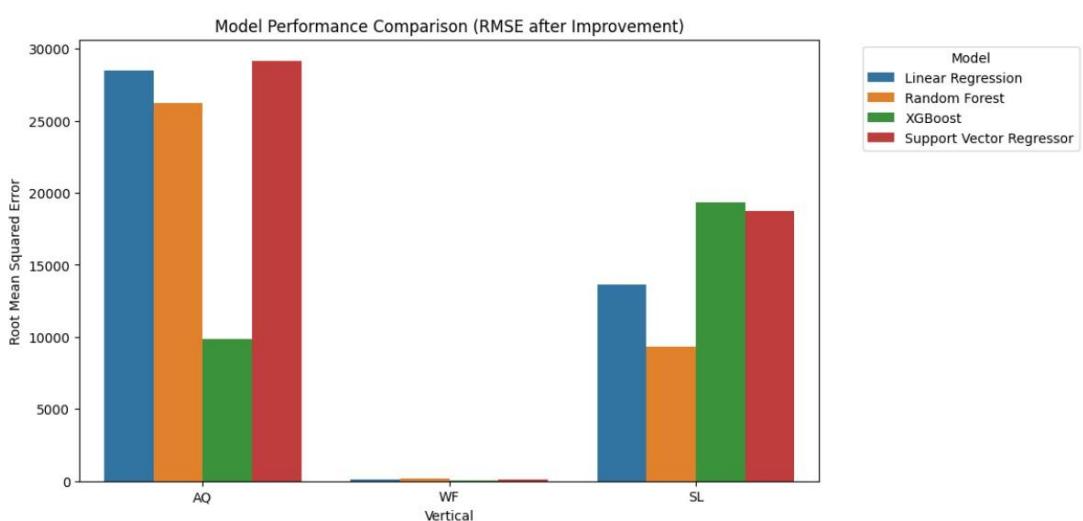
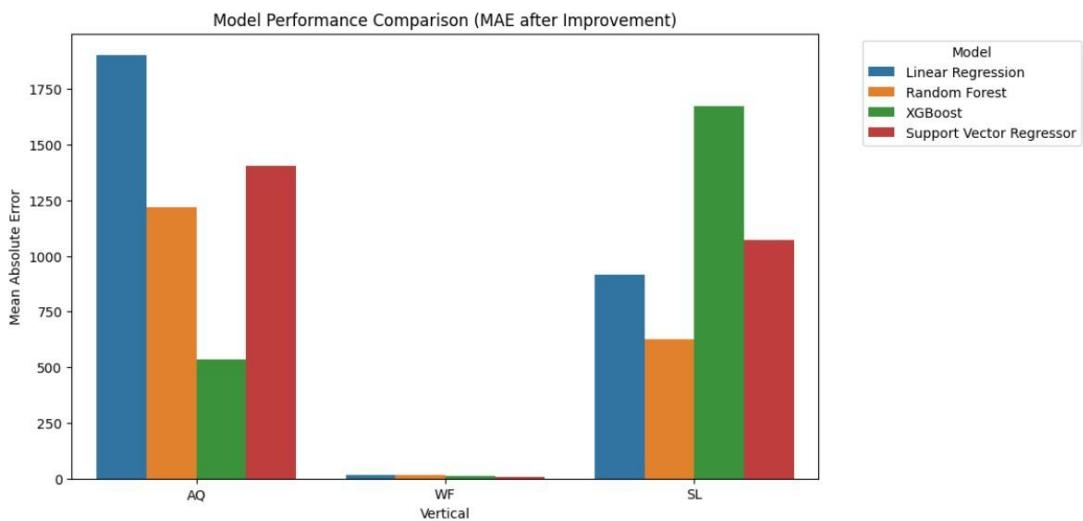
Results Summary

- **Random Forest** and **XGBoost** models showed improved performance after hyperparameter tuning compared to the baseline models.
- **Linear Regression** performed consistently but with lower R² values, confirming non-linear dependencies in the datasets.
- **Support Vector Regressor (SVR)** performed moderately well, especially in smaller datasets.

The **data augmentation** step contributed to better model stability and reduced overfitting.

Findings

- Ensemble and boosting models (Random Forest and XGBoost) outperformed simple linear models in all verticals.
- The improvement in R² scores and reduction in RMSE and MAE indicated successful tuning and model enhancement.
- Visualization provided clear insight into performance trends and model effectiveness.



```
=====
Training Improved Models for AQ Vertical
=====
Linear Regression: R2=0.046 | RMSE=28459.269 | MAE=1902.954
Random Forest: R2=0.190 | RMSE=26217.567 | MAE=1218.816
XGBoost: R2=0.885 | RMSE=9878.315 | MAE=534.919
Support Vector Regressor: R2=-0.002 | RMSE=29170.936 | MAE=1405.797

=====
Training Improved Models for WF Vertical
=====
Linear Regression: R2=-0.567 | RMSE=113.904 | MAE=15.321
Random Forest: R2=-1.876 | RMSE=154.327 | MAE=15.268
XGBoost: R2=0.389 | RMSE=71.154 | MAE=9.909
Support Vector Regressor: R2=-0.007 | RMSE=91.337 | MAE=8.127

=====
Training Improved Models for SL Vertical
=====
Linear Regression: R2=0.468 | RMSE=13654.755 | MAE=915.966
Random Forest: R2=0.753 | RMSE=9306.575 | MAE=626.235
XGBoost: R2=-0.064 | RMSE=19315.353 | MAE=1674.639
Support Vector Regressor: R2=-0.003 | RMSE=18754.096 | MAE=1071.619
```

Conclusion

This task successfully enhanced predictive modeling performance across all verticals. The combination of **synthetic data augmentation, hyperparameter tuning, and comparative visualization** led to measurable improvements in accuracy and interpretability. Among all models, **XGBoost and Random Forest** emerged as the most effective algorithms for real-world deployment due to their balance of accuracy, stability, and adaptability.