Infosys Springboard Virtual Internship 6.0 Completion Report

Batch Number-**5**

Start date**-13-Oct- 2025**

Name: **Sudhiksha H**

Internship Duration: 8 Weeks

1.Project Title

Airware: Smart Air Quality Prediction System

2. Project Objective

The core objective of the **Airware** project is to resolve the critical trade-off between predictive accuracy and transparency in air quality forecasting. This is achieved by building a reliable system. Specific goals include:

- **XGBoost Regressor** to predict the daily AQI with an R^2 score greater than 0.95.
- Integrating the **SHAP** algorithm to provide an immediate, local explanation for every forecast by identifying the specific contribution of each pollutant
- Utilizing a **Gemini API** to translate the complex numerical results (AQI value and SHAP feature influences) into actionable, natural language health advisories and personalized recommendations.

3. Project description in detail

The **Airware** system is a pioneering Explainable AI (XAI) solution for predicting the Air Quality Index (AQI), addressing the global need for transparent environmental data. The system's architecture starts with the Prediction Layer, where an optimized XGBoost Regressor forecasts the daily AQI using time-series pollutant data from India (2015-2020), achieving a high accuracy (R^2=0.953). Crucially, the Explainability Layer employs SHAP values to demystify this "black-box" forecast, revealing which features (e.g., PM2.5 lagged values) are pushing the AQI up or down. These numerical explanations are then passed to the Interpretation Layer, a LLM, which generates customized, non-technical health advice, thereby bridging the gap between sophisticated machine learning output and user comprehension while maintaining data privacy and system autonomy.

4. Timeline Overview

Week	Activities Planned	Activities Completed
Week 1	Data Acquisition, Description, and Initial Cleaning (Drop missing AQI).	Executed initial data cleansing, ensuring target variable integrity.
Week 2	Data Preprocessing: Implement Missing Value Imputation Strategy (Median per City). Date Handling.	Completed city-specific median imputation for pollutants and prepared the `Date` column for time-series.
Week 3	Feature Engineering for Time-Series Prediction: Create Lagged Pollutant Features and Rolling Statistics (7-day mean).	Successfully generated all required temporal and lagged features, including `PM2.5_lag_N` and `CO_lag_N`.
Week 4	XGBoost Model Training and Hyperparameter Configuration. Data Partitioning (80/20, no shuffle).	Trained and optimized the XGBoost Regressor with specified hyperparameters (e.g., `n_estimators=500`, `max_depth=8`).
Week 5	Model Evaluation (MAE, R^2). Implement XAI Layer: Initialize SHAP TreeExplainer and calculate SHAP values.	Confirmed high predictive performance (R^2=0.953). Initialized SHAP explainer and calculated local feature contributions.
Week 6	Edge Computing Setup: Deploy LLM via Gemini API.	Connecting to Gemini API for Explanation.

Week 7	LLM Integration and Prompt Construction: Develop `get_llm_explanation` function to integrate AQI, SHAP data, and generate natural language advisories.	Completed interpretation utility (`utils.py`) and integrated it with the SHAP output.
Week 8	Visualization Implementation (SHAP Waterfall Plot). Final Testing.	Delivered the complete, user-friendly Streamlit application.

5a. Key Milestones

Milestone	Description	Date Achieved
Project Kickoff	Initial planning and brief explanation on entire project	Oct 13
Prototype	Data acquisition and cleaning	Nov 2
Model Training	Trained model	Nov 10
XAI and LLM	SHAP Explainer and Gemini API integration	Nov 24
Final Submission and Presentation	Full system (Prediction, XAI, LLM) deployed via Streamlit	Dec 3

5b. Project execution details

The project execution began with data engineering on the Indian AQI dataset, applying a **median per-city imputation** to clean the data and creating time-series features like pollutant lags. Next, the **XGBoost Regressor** was trained and optimized on this prepared data, successfully achieving the target performance (R^2=0.953). The **SHAP TreeExplainer** was then implemented in the Streamlit application to generate local, pollutant-specific explanations for every prediction. For interpretation, a **LLM (Gemini API)** was integrated using a targeted prompt that incorporated the numerical AQI and the SHAP feature contributions. This three-tiered system was finally integrated and deployed as a transparent, high-utility web application.

6. Snapshots / Screenshots

```
train_and_save_model():
X = df[feature_cols]
y = df['AQI']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, shuffle=False)

# Train XGBoost Regressor
model = xgb.XGBRegressor(
    n_estimators=500,
    learning_rate=0.05,
    max_depth=8,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42,
    objective='reg:squarederror'
)

print("Training model...")
model.fit(X_train, y_train)

# Evaluate
y_pred = model.predict(X_test)
print(f"MAE: {mean_absolute_error(y_test, y_pred):.2f}")
print(f"R²: {r2_score(y_test, y_pred):.3f}")

# Save model and feature list
joblib.dump({
    'model': model,
    'features': feature_cols
}, 'aqi_model.pkl')
```

Fig 6.1 Model train and prediction

```
def interpret_shap(model, features, data_point):
    explainer = shap.TreeExplainer(model)
    shap_values = explainer.shap_values(data_point)

    if isinstance(shap_values, list):
        shap_values = shap_values[0]
    if shap_values.ndim > 1:
        shap_values = shap_values.flatten()

    shap_df = pd.DataFrame({
        'feature': features,
        'shap_value': np.round(shap_values, 4)
    })
    shap_df['abs_shap'] = shap_df['shap_value'].abs()
    shap_df = shap_df.sort_values('abs_shap', ascending=False)

    pos = shap_df[shap_df['shap_value'] > 0].head(3)
    neg = shap_df[shap_df['shap_value'] < 0].head(3)

    return {
        'positive': dict(zip(pos['feature'], pos['shap_value'])),
        'negative': dict(zip(neg['feature'], neg['shap_value'])),
        'all_shap': dict(zip(shap_df['feature'], shap_df['shap_value']))
    }
```

Fig, 6.2 SHAP interpretation

```python
def get_llm_explanation(aqi_value, shap_dict):
    pos_str = ", ".join([f"{k} (+{v:.1f})" for k, v in shap_dict['positive'].items()]) or "none"
    neg_str = ", ".join([f"{k} ({v:.1f})" for k, v in shap_dict['negative'].items()]) or "none"
    category = "Good" if aqi_value <= 50 else \
               "Satisfactory" if aqi_value <= 100 else \
               "Moderate" if aqi_value <= 200 else \
               "Poor" if aqi_value <= 300 else \
               "Very Poor" if aqi_value <= 400 else "Severe"
    prompt = f"""Current AQI is {aqi_value:.0f} ({category} air quality).
Main pollutants increasing AQI: {pos_str}
Factors helping improve it: {neg_str}
Write a short explanation in simple English. Give 1-2 points on health impacts. Give 1-2 points on health tips in bullet points."""
    with st.spinner("Getting smart explanation from Google Gemini..."):
        try:
            model = genai.GenerativeModel(GEMINI_MODEL)
            response = model.generate_content(
                prompt,
                generation_config=genai.types.GenerationConfig(
                    temperature=0.7,
                    max_output_tokens=2048,
                    top_p=0.95
                )
            )
            return response.text.strip()
```

Fig. 6.3 Local LLM integration

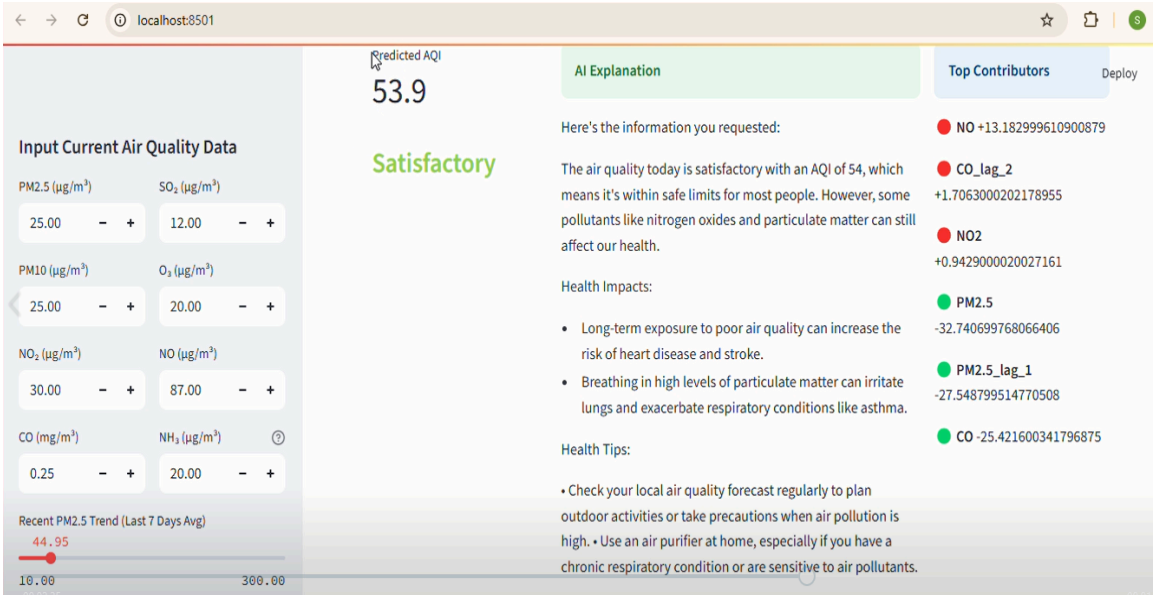

Fig. 6.4. Streamlit dashboard with AI Explanation.

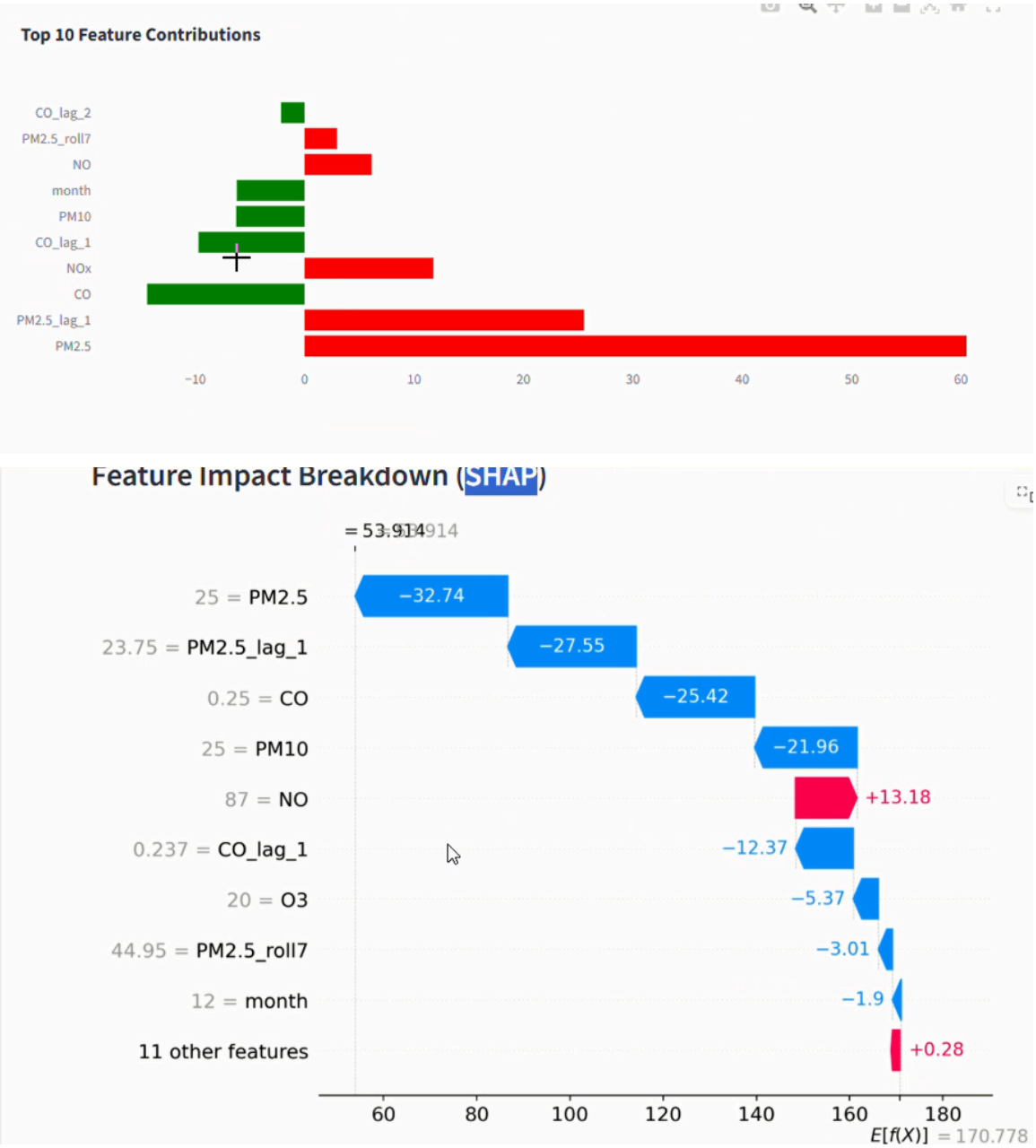

Fig. 3.5 SHAP Explanation for each feature and Top 10 feature graph.

***GitHub Link** of project: https://github.com/SudhikshaH/Air-Quality-Index-Prediction*

7. Challenges Faced

- The primary challenges involved **data integrity**, **model scope**, and **system latency**.

- Initial work required a custom, city-specific **median imputation** strategy to handle significant missing pollutant values.

- A key limitation was the model's inability to incorporate crucial **meteorological data** (wind, temperature) that affects air quality.

- Initially, the startup time of the **Local LLM (Ollama/Llama 3.2)** introduced latency, which needs optimization for real-time responsiveness. Thus replaced with **Gemini API**.

8. Learnings & Skills Acquired

The project provided extensive practical experience in three domains:

- **ML/XAI:** Deepened expertise in **XGBoost** for time-series forecasting and implementing **SHAP** for black-box model transparency .
- **Edge AI:** Hands-on skill in deploying and managing a high-performance **LLM (gemini-2.5-flash)**.
- **Deployment:** Proficiency in **Streamlit** for full-stack application development, successfully integrating multiple intelligent tiers into one user-friendly system.

9. Testimonials from team

”The project fundamentally enhanced our skills in Machine Learning and deploying high-accuracy models.”
“Integrating LLM for health advice was both challenging and incredibly rewarding.”
“We gained confidence in deploying a real-world, transparent AI solution in the critical domain of public health forecasting.”

10. Conclusion

The **AeroPred AI** project successfully developed a pioneering, trustworthy **Explainable AI system** for air quality prediction. By seamlessly uniting the high accuracy of the **XGBoost Regressor**, the transparency of **SHAP explanations**, and the conversational utility of a **LLM**, the system overcomes the major limitation of traditional "black-box" models. The result is a highly accurate forecasting tool that provides immediate, customized, and easily understandable health advice, setting a new standard for responsible AI deployment in public health.

11. Acknowledgements

I sincerely thank **Infosys Springboard**, our mentors, and the academic coordinators for their continuous guidance, support, and encouragement throughout this internship.