## EXECUTIVE SUMMARY

This report documents the development of a complete machine learning pipeline for predicting Air Quality Index (AQI) in Karachi, Pakistan. The system addresses critical environmental monitoring needs through automated data collection, advanced ML modeling, and real-time visualization.

**KEY ACHIEVEMENTS:** • End-to-end automated pipeline from data collection to dashboard • 3-day AQI forecasts with 45% accuracy ($R^2$ score) • Real-time monitoring with freshness indicators • Production deployment with GitHub Actions CI/CD • Interactive dashboard with health recommendations

## PROBLEM STATEMENT

Karachi, Pakistan's largest city with over 20 million residents, faces severe air pollution issues with AQI regularly exceeding 150 (Unhealthy levels). However, the city lacks:

1. REAL-TIME PREDICTIVE SYSTEMS • Current monitoring only shows historical data • No future predictions for planning purposes
2. MULTI-DAY FORECASTS • No reliable 3-day AQI predictions • Limited ability to plan outdoor activities
3. AUTOMATED PIPELINES • Manual data processing limits scalability • No continuous model improvement
4. HEALTH ALERTS • No proactive warning system for hazardous conditions • Limited public awareness tools

TECHNICAL REQUIREMENTS: • Collect and process real-time pollution data • Generate accurate 3-day AQI forecasts • Automate the entire ML pipeline • Provide accessible visualization for citizens • Implement production-grade reliability

## SOLUTION ARCHITECTURE

TECHNOLOGY STACK: • **Backend:** Python 3.10, Scikit-learn, XGBoost, Prophet • **Database:** MongoDB Atlas (Feature Store + Model Registry) • **CI/CD:** GitHub Actions with scheduled automation • **Dashboard:** Streamlit with Plotly visualizations • **Infrastructure:** Cloud-based, fully automated pipeline

HYBRID ML APPROACH: Three-layer forecasting system:

1. ML MODELS (60% weight) → 3-hour recursive Random Forest/XGBoost → $R^2 = 0.63$ for 3-hour predictions
2. TIME SERIES (40% weight) → Seasonal + Exponential Smoothing + Moving Average → Captures daily/weekly patterns
3. ENSEMBLE SYSTEM → Weighted average with performance-based adjustments → Fallback mechanisms for robustness

## CRITICAL CHALLENGES & SOLUTIONS

### 1. DATA LEAKAGE ISSUE

PROBLEM: Initial models showed suspiciously high accuracy ($R^2 > 0.95$), indicating data leakage where future information was contaminating training data.

ROOT CAUSE: • Features included future AQI values through incorrect lag calculations • MongoDB collections had timestamp mismatches • Feature engineering pipeline wasn't properly time-ordered

SOLUTION IMPLEMENTED:

1. COMPLETE RESET • Deleted entire feature store and model registry • Started fresh with proper temporal validation
2. TEMPORAL VALIDATION • Implemented strict time-based train/test splits • Ensured chronological processing order
3. FEATURE AUDIT • Verified all lag features only used historical data • Fixed pipeline in features.py

RESULT: • Realistic $R^2$ scores (0.63 for 3h predictions) • Proper generalization to new data • Sustainable model performance

**2. 72-HOUR FORECASTING PERFORMANCE**

PROBLEM: Direct 72-hour predictions performed poorly ($R^2 < 0.3$) due to: • High variance over longer horizons • Accumulating prediction errors • Limited feature relevance for distant future

SOLUTION: RECURSIVE APPROACH Instead of direct 72h prediction:

1. Train model for 3-hour prediction (high accuracy: $R^2 = 0.63$)
2. Use prediction as input for next 3-hour forecast
3. Recursively apply for 24 cycles → 72-hour forecast
4. Ensemble with time series models for stability

IMPROVEMENT ACHIEVED: • 3h accuracy: $R^2 = 0.63$ • 72h accuracy: $R^2 = 0.42$ (40% improvement over direct prediction) • More stable predictions with error bounding • Better correlation with actual trends

**3. CI/CD PIPELINE ISSUES**

PROBLEM: GitHub Actions failures due to: • Python import path mismatches • Missing dependencies in cloud environment • Environment variable propagation issues • Timeout during model training

SOLUTION IMPLEMENTED:

KEY FIXES IN YML:

1. PATH RESOLUTION export PYTHONPATH="(pwd)"
2. DEPENDENCY MANAGEMENT Explicit pip install with version specifications
3. TIMEOUT HANDLING Increased to 60 minutes for full pipeline
4. ERROR RECOVERY Fallback mechanisms and graceful degradation
5. ENVIRONMENT SETUP Proper .env loading and variable validation

RESULT: • 95% pipeline success rate • Automated error recovery • Consistent deployment across environments

**4. MODEL REGISTRY & FEATURE STORE MANAGEMENT**

PROBLEM: Versioning chaos with: • Multiple collections without clear naming • No production/staging separation • Missing metadata for model comparison • Feature drift detection absent

SOLUTION: MONGODB MANAGER CLASS

IMPLEMENTED FEATURES: • Feature versioning with hashing • Model promotion/demotion workflows • Performance tracking with A/B testing • Data retention policies (30-day cleanup) • Collection standardization

CODE STRUCTURE: class MongoDBManager: • Unified interface for all database operations • Automatic versioning and tracking • Production-ready error handling

## SYSTEM COMPONENTS

A. DATA PIPELINE

1. COLLECTION • Source: Open-Meteo API • Frequency: 45-day historical + 3-hour incremental • Location: Karachi (24.8607° N, 67.0011° E)
2. STORAGE • Database: MongoDB Atlas • Records: 9,111 AQI measurements • Collections: aqi_measurements, aqi_features_simple
3. FEATURES ENGINEERED • Targets: 3h/6h/24h future AQI • Temporal: hour, day_of_week, month, is_peak_hour • Lag Features: 1h, 3h, 6h, 24h AQI values • Rolling Statistics: 3h/6h averages and std dev
4. VALIDATION • Temporal splits only • Leakage prevention protocols • Cross-validation with time series

B. MODEL PIPELINE

1. TRAINING SCHEDULE • Daily automated runs (2 AM UTC) • Weekly full retraining (Sunday 5 AM UTC)
2. MODEL TYPES • Random Forest: Baseline ML model • XGBoost: Gradient boosting for better accuracy • Prophet: Facebook's time series algorithm • Seasonal Models: For daily/weekly patterns
3. MODEL REGISTRY • Versioned storage with metadata • Performance metrics ($R^2$, MAE, RMSE) • Automatic promotion based on thresholds
4. DEPLOYMENT • Automated promotion to production • A/B testing capability • Rollback mechanisms

C. PREDICTION PIPELINE

1. GENERATION FREQUENCY • 3-day forecasts every 3 hours • Real-time updates with freshness tracking
2. ENSEMBLE METHOD • 60% ML + 40% Time Series weighted average • Performance-based weight adjustments • Fallback to individual models if needed
3. STORAGE STRUCTURE • ml_recursive_forecasts: ML model predictions • timeseries_forecasts_3day: Time series predictions • ensemble_forecasts_3day: Final ensemble predictions
4. FRESHNESS MANAGEMENT • <3-hour old: "Fresh" (Green indicator) • 3-6 hours: "Stale" (Yellow indicator) • >6 hours: "Outdated" (Red indicator)

D. DASHBOARD PIPELINE

1. VISUALIZATION PLATFORM • Framework: Streamlit • Auto-refresh: Every 5 minutes • Real-time updates

2. DASHBOARD PAGES • Current AQI: Real-time air quality with health recommendations • EDA Analysis: Exploratory data analysis with visualizations • 3-Day Forecast: Predictions from all models • Feature Importance: Model interpretability • Model Performance: Metrics and comparison • System Status: Pipeline health monitoring
3. ALERT SYSTEM • Hazardous AQI detection (AQI > 300) • Health warnings and precautions • System status indicators
4. MONITORING FEATURES • Freshness indicators for all predictions • Pipeline execution status • Error logging and reporting

## PERFORMANCE METRICS

MODEL PERFORMANCE:

| Model | $R^2$ Score | MAE | RMSE | Horizon | Status |
|---|---|---|---|---|---|
| 3h ML Model | 0.63 | 5.6 | 7.2 | 3 hours | Production |
| 72h Recursive | 0.42 | 8.9 | 11.4 | 3 days | Production |
| Time Series | 0.38 | 9.3 | 12.1 | 3 days | Backup |
| Ensemble | 0.45 | 8.1 | 10.8 | 3 days | Primary |

SYSTEM PERFORMANCE:

• DATA FRESHNESS: <3 hours for "fresh" predictions

• PIPELINE RELIABILITY: 95% successful automated runs

• DASHBOARD UPTIME: 24/7 with auto-recovery

• DATA VOLUME: 9,111 records covering 45+ days

• UPDATE FREQUENCY: Hourly features, Daily training

• STORAGE EFFICIENCY: 30-day retention with automatic cleanup

**ACCURACY BREAKDOWN:**

1. SHORT-TERM (3-hour): • $R^2$: 0.63 (Good predictive power) • MAE: ±5.6 AQI points • Useful for immediate planning
2. MEDIUM-TERM (24-hour): • $R^2$: 0.51 (Moderate accuracy) • MAE: ±7.8 AQI points • Suitable for daily planning
3. LONG-TERM (72-hour): • $R^2$: 0.42 (Acceptable for trends) • MAE: ±8.9 AQI points • Best used for trend awareness

## AUTOMATION SCHEDULE

| Component | Frequency | Time (UTC) | Karachi Time | Purpose |
|---|---|---|---|---|
| Data Collection | Every 3 hours | 00,03,06,09,12,15,18,21 | 05,08,11,14,17,20,23,02 | Incremental data updates |
| Feature Engineering | Hourly | 00 min every hour | +5 hours | Feature store updates |
| Model Training | Daily | 02:00 | 07:00 | Model retraining |
| Full Pipeline | Weekly (Sunday) | 05:00 | 10:00 | Complete system refresh |
| Dashboard Refresh | Continuous | Every 5 minutes | Real-time | Real-time updates |
| Alert Checks | With every run | During execution | During execution | Hazardous AQI detection |

SCHEDULE DESIGN PRINCIPLES:

1. EFFICIENCY: Minimal resource usage with smart scheduling
2. RELIABILITY: Overlap prevention and error recovery
3. FRESHNESS: Regular updates for accurate predictions
4. MAINTAINABILITY: Clear schedule for debugging

## KEY INNOVATIONS

1. FRESHNESS-BASED PREDICTION SYSTEM

PROBLEM ADDRESSED: Stale predictions lose utility and can mislead users.

INNOVATION: • Real-time freshness indicators with color coding • Automatic prediction regeneration when stale • User awareness of prediction reliability

IMPACT: • Users know prediction reliability instantly • Automatic updates maintain accuracy • Trust in system predictions increases

2. RECURSIVE FORECASTING APPROACH

PROBLEM ADDRESSED: Long-horizon prediction inaccuracy in direct modeling.

INNOVATION: • 3h recursive approach with error bounding • Cascade predictions with confidence intervals • Ensemble stabilization at each step

IMPACT: • 40% improvement over direct 72h prediction • More stable long-term forecasts • Better error estimation

3. MONGODB AS UNIFIED STORE

PROBLEM ADDRESSED: Multiple storage systems creating complexity and sync issues.

INNOVATION: • Single database for features, models, and predictions • Versioned collections with metadata • Unified query interface

IMPACT: • Simplified architecture • Consistent data management • Easy backups and recovery

4. PRODUCTION-READY CI/CD PIPELINE

PROBLEM ADDRESSED: Manual deployment causing inconsistencies and errors.

INNOVATION: • GitHub Actions with scheduled automation • Environment-aware configuration • Automated testing and validation

IMPACT: • Consistent deployments • Reduced human error • Scalable operations

## TECHNICAL IMPLEMENTATION DETAILS

DATABASE SCHEMA:

COLLECTIONS:

1. aqi_measurements • Raw AQI data from Open-Meteo • 9,111+ records with timestamps
2. aqi_features_simple • Engineered features with targets • 3h/6h/24h prediction targets
3. models / model_registry • Trained model versions • Performance metrics and metadata
4. ml_recursive_forecasts • ML model predictions (3-day horizon)
5. timeseries_forecasts_3day • Time series model predictions
6. ensemble_forecasts_3day • Final ensemble predictions (primary)

KEY ALGORITHMS:

1. FEATURE ENGINEERING: • Temporal features extraction • Lag feature calculation • Rolling statistics computation
2. MODEL TRAINING: • Random Forest with hyperparameter tuning • XGBoost for improved accuracy • Prophet for time series patterns
3. ENSEMBLE METHOD: • Weighted average based on recent performance • Fallback to individual models • Confidence interval calculation

## CONCLUSION

The AQI Karachi Prediction System successfully addresses a critical environmental monitoring need through innovative technical solutions. By combining machine learning, time series analysis, and production engineering, the system provides:

✓ACCURATE PREDICTIONS: 3-day forecasts with 45% accuracy

✓ REAL-TIME MONITORING: Continuous updates with freshness tracking

✓ PRODUCTION RELIABILITY: Automated pipeline with 95% success rate

✓ USER-FRIENDLY INTERFACE: Interactive dashboard with health guidance

✓ SCALABLE ARCHITECTURE: Cloud-based design for future growth

This project demonstrates how technology can address real-world environmental challenges, providing citizens with actionable air quality information while establishing a framework for scalable environmental monitoring systems.

## APPENDICES

APPENDIX A: PROJECT LINKS • GitHub Repository: https://github.com/AjiyaAnwar/aqi-Karachi •

APPENDIX B: TECHNICAL SPECIFICATIONS • Python Version: 3.10.12 • MongoDB Version: 6.0+ • Streamlit Version: 1.28.0 • Primary APIs: Open-Meteo Air Quality API

APPENDIX C: PERFORMANCE DATA • Training Data Size: 7,000+ samples • Feature Count: 12 engineered features • Model Training Time: ~45 seconds • Prediction Generation: ~10 seconds