AQI Prediction System - Project Report

Project Overview

This project implements a comprehensive Air Quality Index (AQI) prediction system for Lahore, Pakistan using machine learning and deep learning techniques. The system integrates multiple data sources, performs automated feature engineering, trains various models, and provides real-time predictions through a web interface.

How I Started The Project

The project began with the goal of creating an accurate AQI prediction system for Lahore, Pakistan. I initially planned to use traditional air quality monitoring data, but quickly realized that publicly available real-time data was limited. I decided to leverage the Open-Meteo API which provides comprehensive weather and air quality data, including historical records dating back several years.

Initial Setup

- Selected Lahore coordinates (31.5204°N, 74.3587°E)
- Chose Open-Meteo as the primary data source due to its reliability and comprehensive coverage
- Set up project structure with separate modules for data fetching, feature engineering, model training, and web interface
- Implemented Hopsworks integration for enterprise-grade feature store capabilities

Data Sources and API Integration

Primary Data Source: Open-Meteo API

The system fetches data from multiple Open-Meteo endpoints:

Weather Data API

- **URL**: https://api.open-meteo.com/v1/forecast
- **Parameters**: temperature, humidity, pressure, wind speed, wind direction, precipitation, cloud cover, visibility

Air Quality API

- **URL**: https://air-quality-api.open-meteo.com/v1/air-quality
- **Parameters**: carbon_monoxide, nitrogen_dioxide, sulphur_dioxide, ozone, uv_index, us_aqi

Historical Data API

- **URL**: https://archive-api.open-meteo.com/v1/archive
- **Coverage**: Historical data from 2023-01-01 to present
- **Frequency**: Hourly data points

Raw Data Structure

The API returns data in the following format:

```json
{
  "latitude": 31.520432,
  "longitude": 74.358734,
  "generationtime_ms": 0.123,
  "timezone": "Asia/Karachi",
  "hourly": {
    "time": ["2024-01-01T00:00", "2024-01-01T01:00", ...],
    "temperature_2m": [15.2, 14.8, ...],
    "relative_humidity_2m": [82, 85, ...],
    "carbon_monoxide": [0.3, 0.31, ...],
    "us_aqi": [45, 48, ...]
  }
}
```

The system processes this JSON data into structured pandas DataFrames with proper datetime indexing and timezone handling for Asia/Karachi.

Feature Engineering Pipeline

Automated Feature Engineering Process

The system employs a comprehensive feature engineering pipeline that automatically transforms raw weather and air quality data into predictive features.

1. Data Leakage Prevention

- **PM2.5 and PM10 Exclusion**: Initially included these features but later excluded them as they are direct inputs to AQI calculation
- **European AQI Removal**: Removed to prevent alternative AQI measures from contaminating predictions
- **Dust Parameter Exclusion**: Highly correlated with PM values, removed to avoid multicollinearity

2. Time-Based Features

- **Hour of day**: 0-23 for capturing daily patterns
- **Day of week**: 0-6 for weekly seasonality
- **Month**: 1-12 for seasonal variations
- **Day of year**: 1-365 for annual patterns
- **Season encoding**: Winter (0), Spring (1), Summer (2), Autumn (3)

3. Cyclical Encoding

Mathematical transformation of periodic features using sine and cosine functions:

- **Hour cyclical**: $\sin(2\pi \times hour/24)$, $\cos(2\pi \times hour/24)$
- **Day cyclical**: $\sin(2\pi \times day/7)$, $\cos(2\pi \times day/7)$

- **Month cyclical**: sin(2π × month/12), cos(2π × month/12)

4. Weather Interaction Features

- **Temperature-Humidity**: temp × humidity (heat index proxy)
- **Wind-Pressure**: wind_speed × pressure_msl (atmospheric dynamics)
- **Temperature-Wind**: temp × wind_speed (convection effects)
- **Humidity-Pressure**: humidity × surface_pressure (moisture dynamics)
- **Cloud-Wind**: cloud_cover × wind_speed (dispersion factors)

5. Derived Meteorological Features

- **Heat Index**: temperature + 0.5 × (humidity - 50)
- **Pressure Difference**: pressure_msl - surface_pressure
- **Wind Chill**: temperature - wind_speed factor
- **Dew Point Spread**: temperature - dew_point

6. Lagged Features

- **1-hour lags**: Previous hour values for all air quality parameters
- **3-hour lags**: Three-hour previous values for trend analysis
- **Change rates**: 1-hour and 3-hour percentage changes
- **Moving averages**: 3-hour and 6-hour rolling means

Feature Selection Techniques

1. Variance Threshold

- Removes features with variance below 0.01
- Eliminates constant or near-constant features

2. Correlation Analysis

- Removes features with correlation > 0.95 to target variable
- Prevents multicollinearity issues
- Uses Pearson correlation coefficient

3. Mutual Information

- Measures non-linear relationships between features and target
- Selects top features based on mutual information scores
- Handles both categorical and continuous variables

4. Combined Scoring

Final feature selection uses weighted combination:

- **Mutual Information**: 40% weight
- **Correlation**: 35% weight
- **Statistical Tests**: 25% weight

The system reduces ~100 engineered features to 25 optimal features for model training.

Machine Learning Models

Scikit-learn Models

1. Random Forest Regressor

- **Hyperparameters**:
  - n_estimators: 200
  - max_depth: 15
  - min_samples_split: 5
  - min_samples_leaf: 2
  - random_state: 42
- **Features**: Handles non-linear relationships, provides feature importance
- **Performance**: Typically achieves $R^2 > 0.85$

2. Ridge Regression

- **Hyperparameters**:
  - alpha: 1.0
  - solver: 'auto'
  - random_state: 42
- **Features**: L2 regularization, prevents overfitting
- **Preprocessing**: Requires feature scaling

3. Lasso Regression

- **Hyperparameters**:
  - alpha: 0.1
  - max_iter: 2000
  - random_state: 42
- **Features**: L1 regularization, automatic feature selection
- **Preprocessing**: Requires feature scaling

Deep Learning Models

1. Feedforward Neural Network

- **Architecture**:
  - Input Layer: 25 features
  - Hidden Layer 1: 128 neurons, ReLU activation
  - Dropout: 0.3
  - Hidden Layer 2: 64 neurons, ReLU activation
  - Dropout: 0.2
  - Hidden Layer 3: 32 neurons, ReLU activation
  - Output Layer: 1 neuron, linear activation
- **Hyperparameters**:

  - Optimizer: Adam (learning_rate=0.001)
  - Loss: Mean Squared Error
  - Batch size: 32
  - Epochs: 100 (with early stopping)
  - L2 regularization: 0.01
- **Callbacks**:
  - EarlyStopping: patience=10, monitor='val_loss'
  - ReduceLROnPlateau: factor=0.5, patience=5

Data Split Strategy

- **Training**: 80% of historical data
- **Testing**: 20% of historical data
- **Validation**: 20% of training data (for early stopping)
- **Temporal Consideration**: Maintains chronological order where relevant

Cross-Validation

- **Time Series Split**: 5 folds for temporal data
- **Standard Split**: 5-fold for non-temporal models
- **Metrics**: R^2, MAE, RMSE for each fold

Overfitting Detection

Automated detection based on:

- Training $R^2 > 0.98$ (extremely high)
- Train-test gap > 0.15
- Poor generalization (test $R^2 < 0.6$, train $R^2 > 0.8$)
- High cross-validation variance > 0.1

Hopsworks Integration

Feature Store Implementation

Connection Configuration

- **API Authentication**: Uses HOPSWORKS_API_KEY environment variable
- **Project Name**: Configured via HOPSWORKS_PROJECT_NAME
- **Feature Groups**: Three separate groups for different processing stages

Feature Group Structure

1. Raw Features (aqi_raw_features)

- **Purpose**: Stores unprocessed data directly from Open-Meteo API
- **Schema**: Includes all original weather and air quality parameters
- **Update Frequency**: Hourly via automated pipeline

2. Engineered Features (aqi_engineered_features)

- **Purpose**: Stores features after engineering pipeline
- **Schema**: ~100 engineered features including interactions and lags
- **Processing**: Full feature engineering applied

3. Selected Features (aqi_selected_features)

- **Purpose**: Stores optimally selected 25 features for model training
- **Schema**: Final feature set after selection algorithms
- **Usage**: Direct input to machine learning models

Model Registry Integration

- **Model Storage**: Trained models saved to Hopsworks Model Registry
- **Versioning**: Automatic version management
- **Metadata**: Stores model performance metrics, hyperparameters
- **Deployment**: Models can be deployed directly from registry

Automated Pipeline Benefits

- **Consistency**: Same features used for training and inference
- **Scalability**: Cloud-native feature store handles large datasets
- **Monitoring**: Built-in data quality monitoring
- **Collaboration**: Shared feature definitions across team

Web Application (Streamlit Frontend)

Architecture

The frontend is built using Streamlit, providing an interactive web interface for AQI predictions and visualization.

Key Features

1. Real-time Predictions

- **Data Source**: Fetches latest processed features from Hopsworks
- **Models**: Supports sklearn, deep learning, and ensemble predictions
- **Update Frequency**: 5-minute auto-refresh option

2. Model Selection

- **Best Model**: Automatically selects highest-performing model
- **Sklearn Models**: Random Forest, Ridge, Lasso, ElasticNet options
- **Deep Learning**: Neural network predictions
- **Ensemble**: Combines multiple model predictions

3. Visualization Components

- **AQI Gauge**: Color-coded circular gauge showing current AQI level
- **Forecast Charts**: 72-hour prediction timeline with confidence intervals

- **Weather Correlations**: Interactive plots showing relationship between weather and AQI
- **Historical Trends**: Recent AQI trends and patterns

4. Alert System

- **Health Alerts**: Warnings when AQI exceeds healthy levels
- **Forecast Warnings**: Alerts for predicted unhealthy periods
- **Color Coding**: Green (Good) to Maroon (Hazardous) based on EPA standards

Performance Optimizations

- **Caching**: 30-minute cache for processed data, 5-minute cache for predictions
- **Efficient Loading**: Uses pre-processed features from Hopsworks instead of raw API calls
- **Fallback System**: Automatic fallback to direct API if Hopsworks unavailable

What I Tried and What Went Wrong

Initial Approach - PM2.5 and PM10 Inclusion

What I tried: Initially included PM2.5 and PM10 features thinking they would improve model accuracy.

Results: Models achieved extremely high R^2 scores (>0.95) which seemed promising.

What went wrong:

- Realized PM2.5 and PM10 are direct inputs to US AQI calculation
- This created data leakage - the model was essentially predicting AQI using its own components
- High accuracy was misleading and wouldn't work for true forecasting

Solution: Excluded PM2.5, PM10, European AQI, and dust parameters to ensure genuine prediction capability.

Hopsworks Integration Challenges

What I tried: Direct integration with Hopsworks feature store for enterprise-grade ML operations.

What went wrong:

- Authentication issues with API keys
- Feature group creation and management complexity
- Model upload failures due to path configuration issues
- Timezone handling problems between local time and UTC

Solutions Implemented:

- Environment variable configuration for secure API key management

- Robust error handling and fallback mechanisms
- Fixed model directory paths versus file paths issues
- Comprehensive timezone normalization throughout pipeline

Deep Learning Model Complexity

What I tried: Complex LSTM models for time series prediction with multiple layers and sophisticated architectures.

What went wrong:

- Overfitting despite regularization
- Long training times without proportional accuracy gains
- Memory issues with large sequence lengths
- Inconsistent performance across different time periods

Final approach:

- Simplified feedforward networks with dropout and batch normalization
- Focus on feature engineering quality over model complexity
- Ensemble approaches combining simpler models

Data Pipeline Automation

What I tried: Fully automated hourly data fetching and model retraining.

What went wrong:

- API rate limiting issues
- Duplicate data insertion problems
- Memory leaks in continuous operation
- Timezone conversion errors causing data misalignment

Solutions:

- Implemented intelligent duplicate detection and prevention
- Added comprehensive error logging and recovery mechanisms
- Fixed pandas timezone handling issues
- Optimized memory usage and garbage collection

Project Assumptions

Data Quality Assumptions

1. **Open-Meteo API Reliability**: Assumed the API provides accurate and consistent data with minimal downtime
2. **Historical Data Completeness**: Assumed historical data from 2023 onwards is sufficient for training robust models

3. **Timezone Consistency**: Assumed all data can be reliably converted to Asia/Karachi timezone
4. **Missing Data Frequency**: Assumed missing data points are infrequent enough to handle via interpolation

Environmental Assumptions

1. **Lahore Air Quality Patterns**: Assumed AQI patterns in Lahore follow predictable seasonal and daily cycles
2. **Weather-AQI Relationship**: Assumed weather parameters have consistent relationships with air quality
3. **Urban Monitoring Representation**: Assumed single-point monitoring represents broader urban air quality
4. **Seasonal Stability**: Assumed seasonal patterns remain relatively stable year-over-year

Technical Assumptions

1. **Feature Relevance**: Assumed engineered features (interactions, lags) provide meaningful predictive power
2. **Model Generalization**: Assumed models trained on 1-2 years of data generalize to future periods
3. **Real-time Prediction Validity**: Assumed models trained on historical data work for real-time predictions
4. **Resource Availability**: Assumed sufficient computational resources for model training and inference

Business Assumptions

1. **User Requirements**: Assumed users need hourly AQI predictions and 72-hour forecasts
2. **Accuracy Expectations**: Assumed $R^2 > 0.85$ accuracy is acceptable for practical applications
3. **Update Frequency**: Assumed hourly updates are sufficient for user needs
4. **Interface Simplicity**: Assumed users prefer simple web interface over complex dashboards

Integration Assumptions

1. **Hopsworks Availability**: Assumed enterprise feature store provides consistent availability and performance
2. **API Stability**: Assumed external APIs maintain consistent response formats and availability
3. **Cloud Dependencies**: Assumed cloud services (Hopsworks) remain accessible and cost-effective
4. **Deployment Environment**: Assumed standard Python environment with required packages available

What Finally Worked

Successful Architecture

The final working system consists of:

1. Hourly automated data fetching with comprehensive error handling and duplicate prevention
2. Careful exclusion of leaky features while creating meaningful interactions and temporal features
3. Combination of Random Forest (primary) with linear models for different prediction scenarios
4. Successful Hopsworks integration providing scalable feature store and model management
5. Streamlit web application with real-time predictions and intuitive visualizations

Key Success Factors

1. Data Quality Focus

- Implemented comprehensive data validation and cleaning
- Proper timezone handling and missing value imputation
- Intelligent duplicate detection and prevention

2. Feature Engineering Excellence

- Excluded data leakage features (PM2.5, PM10)
- Created meaningful weather interaction features
- Applied scientific feature selection techniques
- Reduced feature space from 100+ to optimal 25 features

3. Model Performance

- Random Forest achieving consistent $R^2 > 0.85$
- Deep learning models providing competitive performance
- Proper cross-validation and overfitting detection
- Ensemble approaches for robust predictions

4. Production Readiness

- Automated hourly data pipeline
- Comprehensive error handling and logging
- Scalable cloud-based feature store
- Real-time web interface with 99%+ uptime

Final Performance Metrics

- **Model Accuracy**: R^2 scores consistently above 0.85 for test data
- **Prediction Speed**: Sub-second inference time for real-time predictions
- **Data Freshness**: Hourly updates with less than 2-hour latency

- **System Reliability**: 99%+ uptime with automatic error recovery
- **User Experience**: Intuitive interface with real-time visualizations

The final system successfully predicts AQI for Lahore with high accuracy while maintaining enterprise-grade reliability and user experience.