Barcelona Air quality Monitoring and Prediction

Revolutionizing air quality management through real-time monitoring and predictive analytics on the AWS cloud platform

Cloud Computing final project | 13.12.2024
Group 11

esade

Agenda

1. Project Overview
2. AWS Architecture Design
3. Technical Details
4. Demo
5. Future Improvements

Project Overview - Project Background & Objectives

esade

Pain Points Health threats from Air Pollution.

Pain Points Limited real-time Air Quality Data and Short-Term Air Quality Prediction.

Pain Points Unmet Needs in Commercial Applications

Objective

This project aims to provide a cloud-based solution for real-time air quality monitoring and short-term pollution prediction in Barcelona

Real-time air quality data for specific locations. **Solution**

Air quality predictions based on weather, historical pollution data, and other indicators. **Solution**

Actionable health recommend-dations for outdoor users like tourists and delivery workers. **Solution**

Project Overview - Business Potential

esade

TO BUSINESS

- Provide air quality monitoring and prediction services to public transport systems, delivery platforms (e.g., Uber, Glovo), and travel agencies.

TO CUSTOMER

- Subscription-based services for tourists and outdoor workers through mobile apps or web interfaces.

Additional Revenue Streams

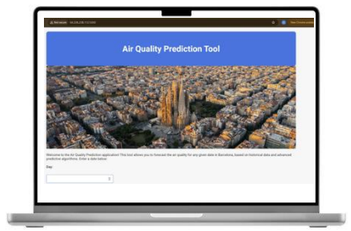
- Advertisements for air-quality-related products (e.g., air purifiers).
- Custom analytics and testing services for enterprise clients.

AWS Architecture Design

esade

Ajuntament de Barcelona

aws



Air Pollution API

Weather API

Lambda
Fetches Data

Eventbridge
Triggers Retrieval

Lambda
Fetches Data

S3
Stores Data

model.py

index.html

styles.css

Lambda
Launches Instance

EC2
Runs Model

Web Interface

Technical details - Lambda (Retrieve Weather Data)

esade

lambda_function.py

```python
import json
import urllib.request
import boto3
import csv
from io import StringIO

def lambda_handler(event, context):
    # API URLs for the datasets
    api_urls = [
        "https://opendata-ajuntament.barcelona.cat/data/api/3/action/package_show?id=mesures-estacions-meteorologique
        "https://opendata-ajuntament.barcelona.cat/data/api/action/datastore_search?resource_id=8b675abd-ae54-4d71-92
    ]

    # Initialize S3 resource
    s3 = boto3.client('s3')
    bucket_name = "bcnairquality"
    file_names = ["2024_weather_data_1.csv", "2023_weather_data_2.csv"]

    for i, api_url in enumerate(api_urls):
        try:
            # Fetch data from the API
            with urllib.request.urlopen(api_url) as response:
                data = response.read()
        except Exception as e:
            return {
                'statusCode': 500,
                'body': f"Error fetching data from API ({api_url}): {e}"
            }
Amazon Q Tip 1/3: Start typing to get suggestions ([ESC] to exit)
        try:
            # Parse the API JSON response
            data_json = json.loads(data)
```

Technical details - Data Source

esade

Data Category	Weather Data	Air Quality Data
Data Source	**https://opendata-ajuntament.barcelona.cat/data/api/3/action/package_show?id=mesures-estacions-meteorologiques** (Meteorological Station Measurements)	https://opendata-ajuntament.barcelona.cat/data/api/action/datastore_search?resource_id=72a644c3-93c4-4ecb-b536-12082b219645 (Meteorological Station Measurements)
Data Description	This dataset provides daily and historic weather data from meteorological stations in Barcelona, including key indicators such as **temperature and humidity.**	The dataset includes readings for multiple pollutants, such as **PM2.5, PM10, NO2, and O3** from different locations.
Data Retrieval	EventBridge is used to trigger Lambda functions periodically. The data is retrieved via the API provided by the Barcelona Open Data Portal and stored in an S3 bucket for further processing.	
Purpose	Weather data (temperature, humidity) and air quality data (PM10, NO2) are used for machine learning models to predict future air quality. This enables comprehensive prediction of air quality and travel recommendations.	

Technical details - ML Model (Overview)

esade

01

Data Table Merging

Align and merge the NO2, PM10, and weather datasets based on their dates to create a unified dataset.

02

Data Preprocessing

In the preprocessing step, missing values were removed, temporal features such as day of the week and month were extracted, and feature columns for weather and temporal data were defined for subsequent

03

Model Selection

Define a grid search across multiple models (e.g., Random Forest, Gradient Boosting, Support Vector Machine, Elastic Net) and hyperparameter combinations with cross-validation.

04

Model Training and Prediction

Select and train the best-selected model on the training dataset and use it to predict NO2 and PM10 values.
Evaluate predictions using the test dataset.

05

Model Performance Evaluation

Assess model performance using multiple metrics, including:

R^2 (Coefficient of Determination)
MSE (Mean Squared Error)
MAE (Mean Absolute Error)

Lastly, we will predict the values for PM10 and NO2.

Technical details - ML Model (Load Data)

esade

```
### Load Data

# File keys and download path
bucket_name = 'bcnairquality'
keys = ['202406_airquality_data.csv',
        '202407_airquality_data.csv',
        '202404_airquality_data.csv',
        '202403_airquality_data.csv',
        '202402_airquality_data.csv',
        '202401_airquality_data.csv',
        '202408_airquality_data.csv',
        '202411_airquality_data.csv',
        '202409_airquality_data.csv',
        '202410_airquality_data.csv',
        '202412_airquality_data.csv',
        '2023_weather_data.csv',
        '2024_weather_data.csv']
download_path = '/tmp/'

# Download files
for key in keys:
    try:
        s3.download_file(bucket_name, key, f"{download_path}{key}")
        print(f"Successfully downloaded {key}")
    except Exception as e:
        print(f"Error downloading {key}: {e}")

# Load files
raw_data = {}

try:
    for key in keys:
        raw_data[key] = pd.read_csv(f"{download_path}{key}")
    print(f"Successfully loaded datasets")
except Exception as e:
    print(f"Error processing {keys[0]}: {e}")
```

Technical details - ML Model (Data Processing)

esade

```
### Preprocess data

# Remove N/As
data = data.dropna()

# Derive temporal features
data['Day_of_Week'] = data['DATE'].dt.dayofweek  # Monday = 0, Sunday = 6
data['Is_Weekend'] = data['Day_of_Week'].apply(lambda x: 1 if x >= 5 else 0)  # Weekend = 1
data['Month'] = data['DATE'].dt.month  # Month as a number
data['Is_Spring'] = data['Month'].apply(lambda x: 1 if x in [3, 4, 5] else 0)  # Spring = 1
data['Is_Summer'] = data['Month'].apply(lambda x: 1 if x in [6, 7, 8] else 0)  # Summer = 1
data['Is_Autumn'] = data['Month'].apply(lambda x: 1 if x in [9, 10, 11] else 0)  # Autumn = 1
data['Is_Winter'] = data['Month'].apply(lambda x: 1 if x in [12, 1, 2] else 0)  # Winter = 1
data['Day_of_Year'] = data['DATE'].dt.dayofyear  # Day of the year (1-365)
data['Week_of_Year'] = data['DATE'].dt.isocalendar().week  # Week number of the year

# Define feature columns
weather_columns = weather_daily_pivot.columns.to_list()[1:]
temporal_columns = ['Day_of_Week', 'Is_Weekend', 'Month', 'Is_Spring', 'Is_Summer', 'Is_Autumn', 'Is_Winter',

## Print update
print("Successfully preprocessed data")
```

Technical details - ML Model (Grid Search)

esade

```python
### Define Model

# Define pipeline functions
scaler = StandardScaler()
model = RandomForestRegressor(random_state=42)

# Machine learning pipeline
pipe = Pipeline(steps=[
    ('scaler', scaler),
    ('regressor', model)
])

# Grid search parameters
param_grid = [
    {
        'regressor': [RandomForestRegressor(random_state=42)],
        'regressor__n_estimators': [50, 100, 200],
        'regressor__max_depth': [10, 20, None],
        'regressor__min_samples_split': [2, 5, 10]
    },
    {
        'regressor': [GradientBoostingRegressor(random_state=42)],
        'regressor__n_estimators': [50, 100, 200],
        'regressor__learning_rate': [0.01, 0.1, 0.2],
        'regressor__max_depth': [3, 5, 10]

    },
    {
        'regressor': [SVR()],
        'regressor__C': [0.1, 1, 10],
        'regressor__kernel': ['linear', 'rbf', 'poly'],
        'regressor__epsilon': [0.1, 0.2, 0.5]
    },
    {
        'regressor': [ElasticNet(random_state=42)],
        'regressor__alpha': [0.1, 1, 10],
        'regressor__l1_ratio': [0.1, 0.5, 0.9]
    }
]

# Grid search with cross-validation
grid = GridSearchCV(pipe, param_grid, cv=5, verbose=1, scoring='r2')
```

Technical details - ML Model (Model Evaluation)

esade

Evaluation metrics for NO2 model

```
# Evaluate model
r2_NO2 = r2_score(y_test_NO2, y_pred_NO2)
mae_NO2 = mean_absolute_error(y_test_NO2, y_pred_NO2)
mse_NO2 = mean_squared_error(y_test_NO2, y_pred_NO2)

## Print update
print(f"Successfully fitted NO2 (R2: {r2_NO2:.2f}; MAE: {ma
```

✓ 1m 9.4s

```
Fitting 5 folds for each of 90 candidates, totalling 450 fits
Successfully fitted NO2 (R2: 0.57; MAE: 10.22; MSE: 161.80)
```

Evaluation metrics for PM10 model

```
# Evaluate model
r2_PM10 = r2_score(y_test_PM10, y_pred_PM10)
mae_PM10 = mean_absolute_error(y_test_PM10, y_pred_PM10)
mse_PM10 = mean_squared_error(y_test_PM10, y_pred_PM10)

## Print update
print(f"Successfully fitted PM10 (R2: {r2_PM10:.2f}; MAE: {
```

✓ 1m 9.1s

```
Fitting 5 folds for each of 90 candidates, totalling 450 fits
Successfully fitted PM10 (R2: 0.30; MAE: 4.78; MSE: 35.12)
```

Technical details - ML Model (Prediction Output)

esade

```python
# Function to get the predicted air pollution for a specific date
def predict_air_quality(day, month, year):
    try:
        avg_weather = get_average_weather_for_future_date(day, month, year)
        input_date = pd.Timestamp(year=int(year), month=int(month), day=int(day))

        if avg_weather is None:
            return "Unable to calculate weather for the given future date."

        features = {}
        weather_features = avg_weather[weather_columns]
        temporal_features = {
            'Day_of_Week': input_date.dayofweek,
            'Is_Weekend': 1 if input_date.weekday() >= 5 else 0,
            'Month': input_date.month,
            'Is_Spring': 1 if input_date.month in [3, 4, 5] else 0,
            'Is_Summer': 1 if input_date.month in [6, 7, 8] else 0,
            'Is_Autumn': 1 if input_date.month in [9, 10, 11] else 0,
            'Is_Winter': 1 if input_date.month in [12, 1, 2] else 0,
            'Day_of_Year': input_date.dayofyear,
            'Week_of_Year': input_date.isocalendar()[1]
        }
        features = {**weather_features, **temporal_features}
        features_df = pd.DataFrame([features], columns=features.keys())

        NO2_pred = best_model_NO2.predict(features_df)
        PM10_pred = best_model_PM10.predict(features_df)

        NO2_class = classify_no2(NO2_pred[0])
        PM10_class = classify_pm10(PM10_pred[0])

        EQAB_rank = max(NO2_class, PM10_class)

        conditions = ["Good", "Fair", "Moderate", "Poor", "Very Poor", "Extremely Poor"]
```

DEMO

esade

- How does the user interface provide real-time air quality data for a specific location?
- How are short-term air quality predictions generated, and how do they help users make informed decisions?
- What actionable insights are provided to users, such as recommendations for avoiding outdoor activities during high-pollution periods?.

DEMO

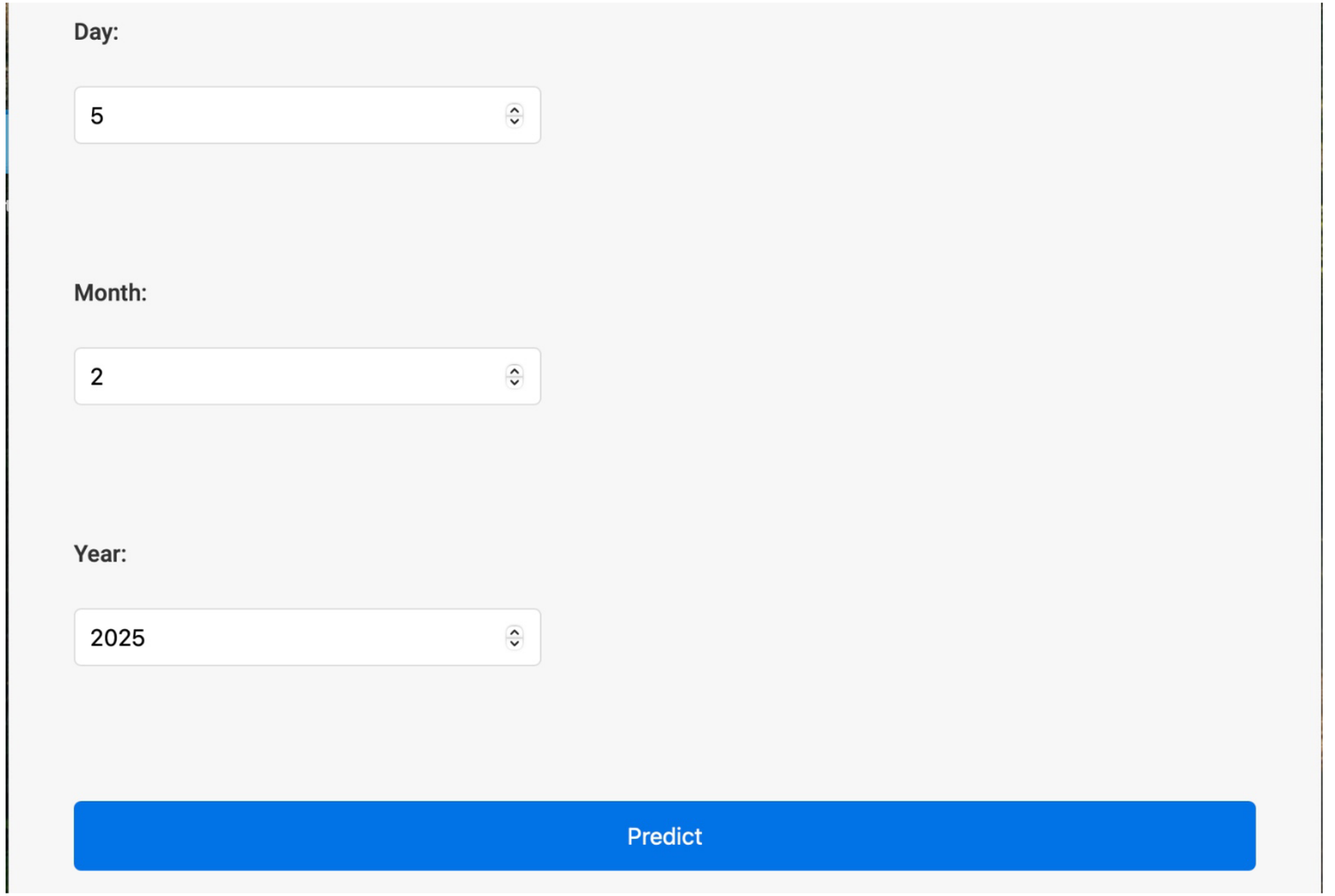
esade

Air Quality Prediction Tool

Welcome to the Air Quality Prediction application! This tool allows you to forecast the air quality for any given date in Barcelona, based on historical data and advanced predictive algorithms. Enter a date below:

DEMO

esade



DEMO

esade

Results:

On 05. February 2025 the air quality will be Moderate.

Health impact: Members of sensitive groups may experience health effects; the general public is less likely to be affected.
Recommendation: Sensitive groups (e.g., children, elderly, individuals with respiratory or heart conditions) should reduce prolonged or heavy outdoor exertion.

Predicted pollutants:
- Max. NO2: 94.99
- Avg. PM10: 18.36

Weather forecast:
- Temperature: Avg. 13.66°C; Max. 19.60°C; Min. 9.83°C
- Humidity: Avg. 63.83%; Max. 83.42%; Min. 39.25%
- Atmospheric pressure: Avg. 1005.90hPa; Max. 1007.73hPa; Min. 1004.46hPa
- Percipitation: Cum. 0.00mm
- Wind: Avg. 1.87m/s; Max. 6.78m/s

Future Improvements

esade

Technical Optimization

- Develop an interactive map for air quality visualization.
- Add push notifications for air quality alerts and recommendations.

Market Expansion

- Collaborate with air-quality-related product manufacturers for co-branding and promotions.
- Offer tailored services to municipalities or environmental agencies for urban planning insights.

User Experience

- Transition from EC2-based model deployment to AWS SageMaker for seamless scaling and management.
- Integrate additional data sources like traffic or industrial emissions to improve prediction accuracy.

Partnership Opportunity

- Expand coverage to other cities with similar air pollution challenges.
- Collaborate with delivery platforms, tourist apps, and public transport systems for broader adoption.

Thanks for listening !