Hong Kong Air Quality Health Index Analysis

This notebook loads hourly Air Quality Health Index (AQHI) data from the Hong Kong Environmental Protection Department, cleans and reshapes the data, performs basic exploratory analysis, and builds a simple classification model to predict AQHI risk categories.

The AQHI data files are monthly CSVs (English version). Each file contains notes in the first few lines, then a header row followed by hourly values for each monitoring station.

In [3]:
```
import pandas as pd
import numpy as np
from io import StringIO
from pathlib import Path
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
```

Helper functions

In [4]:
```
def load_aqhi_csv(path):
    '''Load and clean a monthly AQHI CSV file.

    The function skips the notes at the top of the file, fills missing da
    removes asterisks from values, converts numeric fields, and returns a
    '''
    with open(path, 'r', encoding='utf-8-sig') as f:
        lines = f.readlines()
    header_idx = next(i for i,line in enumerate(lines) if line.strip().st
    csv_data = ''.join(lines[header_idx:])
    df = pd.read_csv(StringIO(csv_data))
    for col in df.columns[2:]:
        df[col] = df[col].astype(str).str.replace('*','', regex=False)
        df[col] = pd.to_numeric(df[col], errors='coerce')
    df['Date'] = df['Date'].ffill()
    df['Hour'] = pd.to_numeric(df['Hour'], errors='coerce')
    df['Hour'] = df['Hour'].ffill()
    df = df.dropna(subset=['Hour'])
    df['Hour'] = df['Hour'].astype(int)
    hour_offset = (df['Hour'] - 1) % 24
    df['Datetime'] = pd.to_datetime(df['Date']) + pd.to_timedelta(hour_of
    return df
```

Load and combine monthly data

In [5]:
```
# Specify the directory containing downloaded CSV files
# By default, data files should be in the same folder as this notebook

data_dir = Path('.')
monthly_files = ['202501_Eng.csv','202502_Eng.csv','202503_Eng.csv','2025

monthly_dfs = []
for fname in monthly_files:
    file_path = data_dir / fname
    if file_path.exists():
        df_month = load_aqhi_csv(file_path)
        monthly_dfs.append(df_month)
    else:
        print(f"Warning: {fname} not found. Skipping.")

all_df = pd.concat(monthly_dfs, ignore_index=True)
print('Combined shape:', all_df.shape)
all_df.head()
```

```
Combined shape: (3775, 21)
```

Out[5]:

	Date	Hour	Central/Western	Southern	Eastern	Kwun Tong	Sham Shui Po	Kwai Chung	Tsuen Wan
0	2025-01-01	1	6.0	6	6.0	6.0	5.0	5.0	5
1	2025-01-01	2	7.0	6	5.0	5.0	5.0	5.0	5
2	2025-01-01	3	7.0	5	5.0	5.0	5.0	5.0	5
3	2025-01-01	4	6.0	5	5.0	5.0	5.0	5.0	5
4	2025-01-01	5	5.0	5	5.0	5.0	5.0	5.0	5

5 rows × 21 columns

Reshape to long format

In [6]:
```
long_df = all_df.melt(id_vars=['Datetime','Date','Hour'], var_name='Stati
long_df = long_df.dropna(subset=['AQHI'])
print('Long format shape:', long_df.shape)
long_df.head()
```

```
Long format shape: (67837, 5)
```

Out[6]:

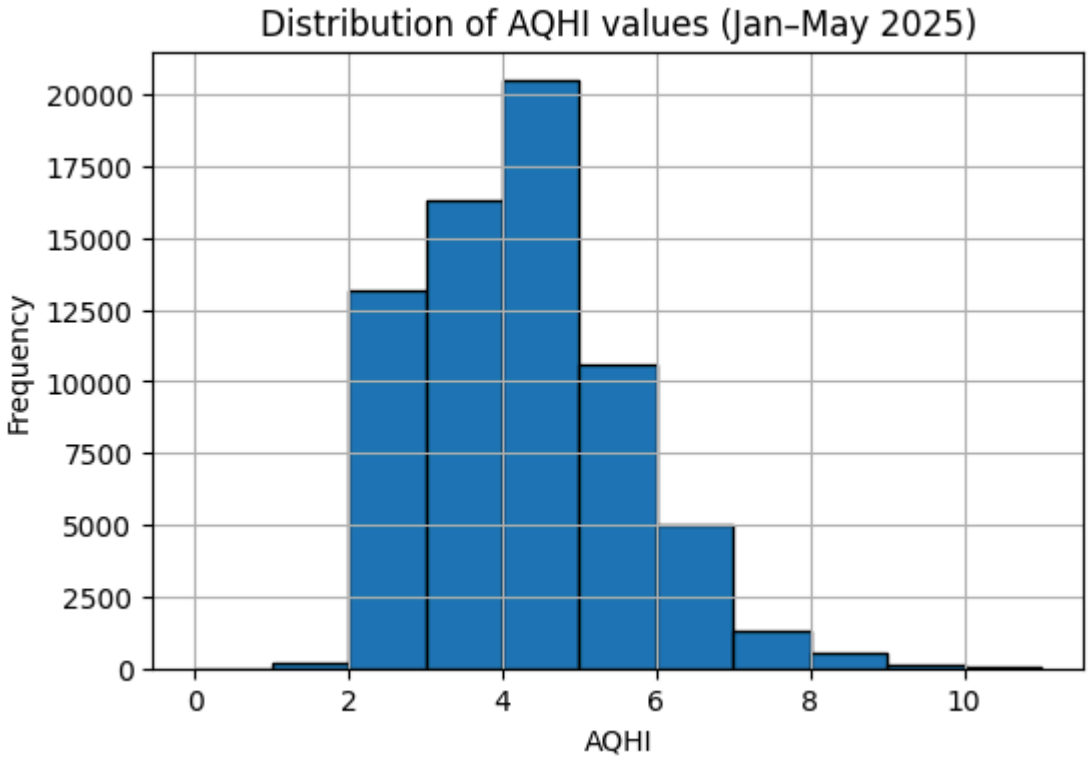
	Datetime	Date	Hour	Station	AQHI
0	2025-01-01 00:00:00	2025-01-01	1	Central/Western	6.0
1	2025-01-01 01:00:00	2025-01-01	2	Central/Western	7.0
2	2025-01-01 02:00:00	2025-01-01	3	Central/Western	7.0
3	2025-01-01 03:00:00	2025-01-01	4	Central/Western	6.0
4	2025-01-01 04:00:00	2025-01-01	5	Central/Western	5.0

Exploratory data analysis

In [7]:
```
print(long_df['AQHI'].describe())
plt.figure(figsize=(6,4))
long_df['AQHI'].hist(bins=range(0,12), edgecolor='black')
plt.title('Distribution of AQHI values (Jan–May 2025)')
plt.xlabel('AQHI')
plt.ylabel('Frequency')
plt.show()

station_mean = long_df.groupby('Station')['AQHI'].mean().sort_values(asce
print(station_mean)
```

```
count    67837.000000
mean         3.772528
std          1.351221
min          1.000000
25%          3.000000
50%          4.000000
75%          5.000000
max         10.000000
Name: AQHI, dtype: float64
```



```
Station
Eastern            3.980382
Tai Po             3.926790
Causeway Bay       3.923607
Tuen Mun           3.905951
Tseung Kwan O      3.866048
Central/Western    3.862636
Mong Kok           3.842259
Central            3.806255
Southern           3.782781
Kwun Tong          3.782286
Sham Shui Po       3.764113
North              3.721485
Kwai Chung         3.719247
Tap Mun            3.696189
Tung Chung         3.668790
Sha Tin            3.668081
Yuen Long          3.541744
Tsuen Wan          3.446623
Name: AQHI, dtype: float64
```

Feature engineering and classification model

In [8]:

```
def categorize(aqhi):
    if aqhi <= 3:
        return 'Low'
    elif aqhi <= 6:
        return 'Moderate'
    elif aqhi <= 8:
        return 'High'
    elif aqhi <= 10:
        return 'Very High'
    else:
        return 'Serious'

long_df['RiskCategory'] = long_df['AQHI'].apply(categorize)

long_df['Hour_of_Day'] = long_df['Datetime'].dt.hour
long_df['Day_of_Week'] = long_df['Datetime'].dt.dayofweek

X = long_df[['Station','Hour_of_Day','Day_of_Week']]
y = long_df['RiskCategory']

preprocess = ColumnTransformer([
    ('cat', OneHotEncoder(handle_unknown='ignore'), ['Station'])
], remainder='passthrough')

clf = Pipeline(steps=[('preprocess', preprocess),
                      ('model', RandomForestClassifier(n_estimators=200, ra

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
print('Accuracy:', accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.5351808176100629
              precision    recall  f1-score   support

        High       0.00      0.00      0.00       553
         Low       0.49      0.39      0.43      8902
    Moderate       0.56      0.69      0.61     10837
   Very High       0.00      0.00      0.00        60

    accuracy                           0.54     20352
   macro avg       0.26      0.27      0.26     20352
weighted avg       0.51      0.54      0.52     20352
```

```
/opt/anaconda3/envs/COM6005/lib/python3.10/site-packages/sklearn/metrics/_
classification.py:1731: UndefinedMetricWarning: Precision is ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape
[0])
/opt/anaconda3/envs/COM6005/lib/python3.10/site-packages/sklearn/metrics/_
classification.py:1731: UndefinedMetricWarning: Precision is ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape
[0])
/opt/anaconda3/envs/COM6005/lib/python3.10/site-packages/sklearn/metrics/_
classification.py:1731: UndefinedMetricWarning: Precision is ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape
[0])
```