EV Charging Station Demand Forecasting System (1).docx

My Files

My Files

Indian Institute of Technology Jodhpur

Document Details

Submission ID

trn:oid:::29334:121561235

Submission Date

Nov 16, 2025, 11:54 AM GMT+5:30

Download Date

Nov 16, 2025, 11:59 AM GMT+5:30

File Name

EV Charging Station Demand Forecasting System (1).docx

File Size

1.6 MB

14 Pages

4,177 Words

24,752 Characters

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

EV Charging Station Demand Forecasting System

A Time Series Analysis Project Report

Abstract

The number of electric vehicles (EVs) on the road has been rising steadily, and this growth is starting to put noticeable pressure on city power systems, especially at public and workplace charging points. When nobody plans ahead for the load, some hours end up overcrowded with vehicles waiting to charge, while other periods remain almost empty. In this project, I look at how to predict this short-term charging demand using standard time series tools, with particular attention to a SARIMAX-type model.

For the analysis, I rely on real session data from the Adaptive Charging Network (ACN). The raw JSON logs are first transformed into a tabular format and then combined into an hourly series of energy delivered at the site. The data is then analyzed to identify trends and daily seasonal patterns. A SARIMAX model is trained on historical data and evaluated on a hold-out period. The model's performance is examined using MAE and RMSE, while the behaviour of MAPE is discussed in the presence of near-zero demand values.

In addition to the statistical analysis, a small FastAPI-based backend is developed to automate the pipeline: ingesting data, aggregating it, computing diagnostics, and serving forecasts to a frontend. The study shows that daily patterns in charging demand can be captured reasonably well by a seasonal time series model, and that such forecasts can support planning and management decisions in a smart-city context.

Keywords: Electric vehicles, charging demand, time series analysis, SARIMAX, forecasting, FastAPI, ACN dataset.

1. Introduction / Background

The increasing interest in electric mobility is one of the visible components of the broader smart-city vision. As more drivers move towards electric vehicles, the pressure on local distribution networks does not only depend on how many vehicles exist, but also on *when* and *where* they are charged. Public and workplace charging stations thus play a crucial role in daily travel patterns and in overall energy planning.

This project grew out of a desire to work on a time series problem that is connected to real infrastructure rather than purely synthetic or financial data. An EV charging site is a good real-world example of how everyday habits show up as numbers in a dataset. Office hours, commute patterns, and weekend plans all get reflected in when and how people plug in their vehicles. The picture is far from regular: on certain days the chargers are almost always occupied, on others they remain idle for long stretches; some drivers stay long enough for a full charge, while others just "sip" a few units of energy and leave. This blend of predictable routine and unpredictable behaviour makes the underlying time series both challenging and appealing to study.

If we look at the problem from the operator's side, a few practical questions immediately appear: at what times will most vehicles arrive, how much energy will they collectively draw, and will the current infrastructure be able to handle that load comfortably? A reasonable forecast can support decisions about how much capacity to keep available, when to carry out maintenance, and, in more advanced setups, how to adjust prices over the day. In the absence of such forecasts, many decisions are based mainly on gut feeling or past habits, which can easily fail when usage patterns slowly shift.

The project therefore concentrates on predicting hourly energy demand at a single EV charging location over short horizons. I rely on real ACN session logs, which contain information such as when a vehicle connects and disconnects and how many kilowatt-hours are delivered during that session. After basic cleaning and aggregation, these records are turned into a regular hourly time series, which is then modelled using a SARIMAX framework.The report is laid out in a straightforward way. Section 2 summarises the main concepts from time series analysis that are used later, with an emphasis on seasonal ARIMA-type models. Section 3 lists the specific aims of the work, and Section 4 briefly surveys earlier studies on EV charging demand and energy forecasting. In Section 5, I state the research questions and the associated hypotheses. Section 6 explains the dataset and the overall methodology, including both the statistical steps and the software implementation. Section 7 discusses the main empirical results, and Section 8 closes the report with a short summary, key limitations, and ideas for extending the work. The appendix includes small code fragments and example data to show how the system was implemented in practice.

.

---

2. Theoretical Concept

Time series analysis is concerned with observations that arrive in sequence, where the timing and ordering of points carries important information. Consecutive values are usually not independent of each other. In the context of EV charging, this implies that the energy drawn in a particular hour is influenced by what happened in the hours just before it, rather than being a fresh, unrelated number every time. Instead, patterns such as daily commuting cycles, office hours, and evening charging routines can create systematic behaviour in the series.

A central concept is **stationarity**. A time series is called stationary if its statistical properties (mean, variance, autocorrelation structure) do not change with time. Many real datasets are not stationary because of long-term trends, seasonal effects, or structural breaks. In such cases, transformations like differencing are used to remove trends and stabilize the series.
The **Autoregressive Integrated Moving Average (ARIMA)** model is a classical framework combining three elements:

- **Autoregressive (AR)** part: the current value depends on a linear combination of past values.
- **Integrated (I)** part: differencing operations applied to make the series stationary.
- **Moving Average (MA)** part: the current value depends on past forecast errors.

An ARIMA model is usually denoted as ARIMA(p, d, q), where:

- p is the order of the autoregressive component,
- d is the number of differences,
- q is the order of the moving average component.

In many applications, including energy systems, there is strong **seasonality**. For EV charging, this often appears as a daily cycle—demand rises and falls in a pattern linked to human routines. To incorporate such repeated patterns explicitly, the ARIMA framework is extended to **seasonal ARIMA (SARIMA)** and further to **SARIMAX** when exogenous variables are included.
A SARIMAX model is written as:
SARIMAX(p,d,q)×(P,D,Q)S\text{SARIMAX}(p, d, q) \times (P, D, Q)_SSARIMAX(p,d,q)×(P,D,Q)S

where (P,D,Q)(P, D, Q)(P,D,Q) and SSS correspond to seasonal autoregressive order, seasonal differencing order, seasonal moving average order, and seasonal period respectively. In this project, the seasonal period SSS is taken as 24 for hourly data to represent daily seasonality.
The idea of **autocorrelation** and **partial autocorrelation** is also important:

- The autocorrelation function (ACF) shows how correlated the series is with itself at different lags.
- The partial autocorrelation function (PACF) isolates the direct effect at a given lag, controlling for the intermediate lags.

ACF and PACF plots are typically used in model identification to choose preliminary values of p and q. The seasonal components P and Q can also be informed by repeating patterns in these plots at multiples of the seasonal lag.
This theoretical framework provides the basis for the modelling choices in this study: assessing stationarity, identifying orders, and fitting a SARIMAX model to capture both short-term dependencies and daily seasonality in EV charging demand.

---

3. Objective of the Paper

The overall aim of this project is to understand and forecast short-term demand at an electric vehicle charging site using time series analysis. More specifically, the objectives are:

1. **To explore temporal patterns** in EV charging energy demand, including daily cycles and overall trends, using real data from the Adaptive Charging Network (ACN).
2. To design and fit a SARIMAX-type model that can generate forecasts of hourly energy use at the chosen site.
3. To judge how well the model performs using measures such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), and to comment on why MAPE is less suitable for this particular dataset.
4. To build a small but repeatable Python pipeline, based on FastAPI and helper functions, which automates the main steps: reading in the data, cleaning it, aggregating it, running diagnostics, and producing forecasts.
5. To interpret the forecasts from an operational angle, especially in relation to capacity planning and scheduling quiet hours for maintenance at charging stations.

Taken together, these goals tie the statistical modelling side to a concrete software system, matching the time series theme of the course while keeping smart-city applications in view.

---

4. Literature Review

Work on modelling EV charging behaviour has grown alongside the wider adoption of electric vehicles. A recurring observation in the literature is that demand at charging stations is highly uneven over the day: usage typically spikes around the times when people arrive at or leave work, while late-night and early-morning periods are much quieter. Weekends often show a different pattern again, reflecting more flexible travel plans.

Traditional time series models such as ARIMA and SARIMA have been used for many energy-related forecasting tasks, including grid-level load, building consumption, and renewable

generation. These models exploit the fact that the present value in a series is linked to its past values and past errors, and they tend to work well when the system has stable, repeating patterns.

SARIMAX extends this family by allowing additional explanatory variables—like temperature, fuel prices, or holiday indicators—to be included on the right-hand side. Several studies report that such external factors can improve forecasts, especially when demand is sensitive to weather or calendar effects. In practice, however, collecting clean and consistent external data is often non-trivial.

For EV charging, prior research often notes strong daily cycles associated with commuting, large variation in how long sessions last and how much energy is delivered, and the coexistence of short opportunistic charges and long-stay sessions.

- pronounced daily cycles linked to commuting,
- variability in session length and energy delivered,
- the presence of both “opportunistic” short stays and long parked charging sessions.

A separate question in the literature is how to measure forecast quality. MAPE is frequently reported because it is scale-free and easy to interpret as a percentage, but it becomes unstable when the true values are very small or zero: even a tiny absolute deviation can lead to an extremely large percentage error. For time series with many low-usage hours, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are typically more informative.

In this project I follow that practice: a SARIMAX-type model (without exogenous regressors, due to data limitations) is fitted, and MAE and RMSE are used as the primary evaluation metrics. MAPE is still computed, but mainly to illustrate and discuss its limitations for this dataset.

---

5. Research Questions and Hypothesis

Based on the literature and the initial exploration of the ACN dataset, the following research questions are formulated:

Research Question 1:

Can hourly EV charging energy demand at a given site be modeled as a seasonal time series with a clear daily pattern?

Research Question 2:

Is a SARIMAX-type model able to produce reasonably accurate short-term forecasts (e.g., 48 hours ahead) for this demand?

Research Question 3:

Do the forecasts provide information that could be useful for operational decisions, such as anticipating peaks or planning low-demand maintenance slots?

From these research questions, the study proposes the following hypotheses:

- **H01:** The hourly energy demand series exhibits statistically significant daily seasonality that can be captured by a SARIMAX structure.
- **H02:** A SARIMAX model trained on historical ACN data can achieve low MAE and RMSE for 48-hour ahead forecasts on a hold-out test period.

- **H03:** Although the model is statistical and relatively simple, its forecasts can contribute to better understanding and management of charging station operations.

The rest of the report is framed around testing these hypotheses through data analysis and modeling.

---

6. Data and Methodology

6.1 Data Description

The empirical analysis is based on the Adaptive Charging Network (ACN) dataset. In this dataset, each row represents a single charging session and carries information such as the cluster and site identifiers, the start and end times of the session, the energy delivered in kilowatt-hours, the station and parking space identifiers, and some optional user-related fields.

A short illustrative JSON fragment for one session is shown below:

(you can keep the JSON example itself as it is)

:
{
"clusterID": "0001",
"connectionTime": "Wed, 05 Sep 2018 11:04:13 GMT",
"disconnectTime": "Wed, 05 Sep 2018 19:09:35 GMT",
"doneChargingTime": null,
"kWhDelivered": 9.583,
"sessionID": "1_1_179_800_2018-09-05 11:04:12.876087",
"siteID": "0001",
"spaceID": "AG-3F32",
"stationID": "1-1-179-800",
"timezone": "America/Los_Angeles",
"userID": null,
"userInputs": null
}

In some cases, userInputs is a list containing additional fields such as WhPerMile, kWhRequested, milesRequested, minutesAvailable, and a paymentRequired flag. For example:

"userInputs": [
{
"WhPerMile": 400,
"kWhRequested": 8.0,
"milesRequested": 20,
"minutesAvailable": 277,
"paymentRequired": true,
"requestedDeparture": "Wed, 05 Sep 2018 15:45:09 GMT",
"userID": 333
}
]

For the time series modelling task, the most important variables are the connection time and the energy delivered (kWhDelivered). The study period used in the analysis covers multiple months, with thousands of sessions, and the goal is to transform these individual sessions into a regular hourly time series representing total energy delivered per hour at the site.

6.2 Preprocessing and Aggregation

The preprocessing pipeline performs the following steps:

1. **Flattening JSON:**
   The raw JSON data is flattened into a tabular structure using pandas.json_normalize, so that nested userInputs fields become separate columns when present.
2. **Timestamp Handling:**
   The `connectionTime` string is converted into a proper datetime variable, taking the recorded timezone into account. Any rows for which this conversion fails, or where the timestamp is missing, are removed from the dataset.
3. **Selecting Site and Energy Columns:**
   To keep the study focused, I restrict attention to a single site (for example, `siteID = "0001"`). The energy measure is derived from the `kWhDelivered` field, which is explicitly cast to a numeric type before further processing.
4. **Hourly Aggregation:**
   The session-level data is then aggregated into hourly buckets to build the time series. For each hour, the pipeline computes total energy in kWh and the number of sessions. Additional variables such as mean WhPerMile or number of paid sessions can be aggregated, but they are not used directly in the SARIMAX model in this report.

The aggregation logic is implemented by a helper function in the backend. A simplified version is shown below:

```
def build_hourly_daily(raw: pd.DataFrame, site: Optional[str], freq: str = "H") -> pd.DataFrame:
    if raw is None or raw.empty:
        return pd.DataFrame(columns=["energy_kwh", "sessions"])

    df = raw.copy()

    # 1) Timestamp
    ts_col = "connectionTime"
    df["ts"] = pd.to_datetime(df[ts_col], errors="coerce")
    df = df.dropna(subset=["ts"])
    if df.empty:
        return pd.DataFrame(columns=["energy_kwh", "sessions"])

    # 2) Energy in kWh
    s = pd.to_numeric(df["kWhDelivered"], errors="coerce")
    df["energy_kwh"] = s.fillna(0.0)

    # 3) Filter by site if specified
    if site is not None and "siteID" in df.columns:
        df = df[df["siteID"] == site]

    if df.empty:
```

```
    return pd.DataFrame(columns=["energy_kwh", "sessions"])

  # 4) Build frame with one row per session
  agg_df = pd.DataFrame({
    "ts": df["ts"],
    "energy_kwh": df["energy_kwh"],
    "sessions": 1
  })

  # 5) Aggregate per time bucket
  out = (
    agg_df
    .set_index("ts")
    .groupby(pd.Grouper(freq=freq))
    .agg({"energy_kwh": "sum", "sessions": "sum"})
    .sort_index()
  )
  out.index.name = "ts"
  return out
```

This function takes the raw sessions and returns a time-indexed DataFrame with the columns energy_kwh and sessions at the desired frequency (hourly in this case).

6.3 Model Specification and Estimation

Once the hourly series is constructed, the methodology follows standard time series steps:

- **Exploratory plots** of the hourly energy series to visually inspect trends and daily patterns.
- **Stationarity checks** using plots and differencing. In practice, first-order differencing combined with seasonal differencing (period 24) was considered.
- **ACF and PACF diagnostics** to identify plausible orders for the AR and MA components.

The forecasting metric chosen is hourly energy (kWh). The final model used in the study is a SARIMAX model with the structure:

- Non-seasonal part: (p, d, q) = (1, 1, 1)
- Seasonal part: (P, D, Q, S) = (1, 1, 1, 24)

The dataset is split into a training and test set. The training set covers the initial months, while the last portion (e.g., the final weeks) serves as a test period for validation. The backend code exposes a /forecast endpoint that automates fitting and forecasting. A simplified version of the core logic is as follows:

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

def run_sarimax_forecast(series: pd.Series, horizon: int = 48):
  y = series.dropna()
  train = y.iloc[:-horizon]
  test = y.iloc[-horizon:]

  model = SARIMAX(
    train,
```

```
        order=(1, 1, 1),
        seasonal_order=(1, 1, 1, 24),
        enforce_stationarity=False,
        enforce_invertibility=False,
    )
    result = model.fit(disp=False)

    # Validation forecast
    fc_test = result.get_forecast(steps=len(test))
    test_pred = fc_test.predicted_mean

    # Future forecast
    fc_future = result.get_forecast(steps=horizon)
    future_mean = fc_future.predicted_mean
    future_ci = fc_future.conf_int(alpha=0.2)

    return result, test, test_pred, future_mean, future_ci
```

Error metrics are then computed using NumPy:

```
def evaluate_np(y_true: np.ndarray, y_pred: np.ndarray) -> dict[str, float]:
    mae = float(np.mean(np.abs(y_true - y_pred)))
    rmse = float(np.sqrt(np.mean((y_true - y_pred) ** 2)))
    denom = np.maximum(1e-8, np.abs(y_true))
    mape = float(np.mean(np.abs((y_true - y_pred) / denom)) * 100.0)
    return {"MAE": mae, "RMSE": rmse, "MAPE": mape}
```

These functions form the core of the modeling methodology, with the FastAPI layer acting as a wrapper around them.

---

7. Result Analysis

After aggregating the data and fitting the SARIMAX model, the forecasts were evaluated on a held-out test period. The main findings from the analysis are summarized below.

7.1 Error Metrics

The model achieved an approximate **Mean Absolute Error (MAE)** of 2.6 kWh and a **Root Mean Squared Error (RMSE)** of about 3.7 kWh on hourly forecasts. These values are reasonable when compared to the overall range of hourly energy consumption in the dataset. The model therefore captures a substantial portion of the systematic variation in the series.

The **Mean Absolute Percentage Error (MAPE)**, however, turned out to be extremely high in numerical value. The main reason is that several hourly observations in the test period are very close to zero. In these hours, even a small absolute difference between forecast and actual demand translates into a very large percentage error. This behaviour is consistent with known limitations of MAPE in datasets where the target can be near zero.

Given this, MAE and RMSE are considered more informative in this context. The combination of these two metrics indicates that, in absolute terms, the forecast errors are not excessive relative to the scale of hourly consumption.

7.2 Visual Fit and Seasonal Patterns

Plotting the actual and predicted series shows that the model follows the daily cycles reasonably well. Peaks in demand during typical office hours are often captured, and troughs during late-

night or very early morning hours are also reflected. There are some mismatches on individual days, especially when demand spikes or drops unexpectedly, but the broad pattern aligns with what is seen in the test data.
Residual analysis does not reveal strong remaining autocorrelation, suggesting that the chosen model structure captures much of the dependence in the data. However, there is still some variability that may be related to factors not included in the model, such as specific events, weather, or holidays.

7.3 Operational Interpretation

From an operational standpoint, an hourly forecast with MAE on the order of a few kWh can still be very useful. For example:

- If the forecast for a particular morning indicates a sharp increase in expected demand compared to the previous day, the operator could ensure that all chargers are available and functional during that period.
- If the forecasts indicate that demand will stay very low over a stretch of night-time hours, that interval could be used for maintenance, software upgrades, or other interventions that require chargers to be temporarily offline, with little inconvenience to users.
- In a more mature deployment, the same forecasts might be linked to simple dynamic-pricing schemes, where tariffs change slightly across the day to encourage users to shift some charging to less congested periods.

While this project does not implement such control strategies, the results imply that even a relatively simple SARIMAX model can act as a useful building block for more advanced operational decision systems.

---

8. Conclusion

In summary, the work presented here uses time series tools to predict hourly energy usage at an EV charging location, drawing on real data from the Adaptive Charging Network. The main aim was to uncover the daily structure in the series and to see how far a SARIMAX model can go in producing reliable short-term forecasts.

The findings are broadly in line with the initial expectations. The first hypothesis is supported: the series shows clear day-to-day periodicity, and the model is able to reproduce this pattern. The second hypothesis is only partly met: the SARIMAX specification achieves reasonably low MAE and RMSE values for a 48-hour forecast horizon, but performance drops somewhat when the demand behaves irregularly. The third hypothesis, concerning direct operational impact, is harder to evaluate within the scope of this project, yet the results indicate that the forecasts could act as useful inputs for planning and management decisions.

There are also several clear limitations to keep in mind:

- Because many hourly values in the series are very small, MAPE behaves poorly and is not a reliable indicator of forecast quality in this context.

- Potentially important external drivers—such as weather, holidays, or special events—are not included in the model, even though they might help explain part of the remaining variability.
- The analysis is restricted to a single site and to one primary outcome (energy), whereas a realistic deployment would need to support several stations and possibly other performance measures.

Looking ahead, it would be natural to experiment with hybrid models that combine SARIMAX with machine-learning components, to bring in external variables where available, and to adapt the current pipeline so that it can serve multiple charging locations simultaneously.

Nevertheless, this project demonstrates that classical time series tools, supported by a simple API-based implementation, can contribute meaningfully to understanding and forecasting EV charging demand in a smart-city context.

References

Acemoglu, D., & Restrepo, P. (2018). *Demographics and automation* (No. w24421). National Bureau of Economic Research.
Manjhi, G., & Mehra, M. K. (2019). A dynamic analysis of special interest politics and electoral competition. *Dynamic Games and Applications*, 9(1), 142–164.
Rogoff, K. (2002). Dornbusch's overshooting model after twenty-five years. *IMF Staff Papers*, 49(1), 1–34.
Adaptive Charging Network (ACN). (2019). *EV charging dataset*. California Institute of Technology.
(Additional references you actually used in your course/literature review can be added here in APA style.)

Appendix

Appendix A – Sample Raw Data Record

A simplified ACN JSON object representing one charging session:

{
"clusterID": "0001",
"connectionTime": "Wed, 05 Sep 2018 11:04:13 GMT",
"disconnectTime": "Wed, 05 Sep 2018 19:09:35 GMT",
"doneChargingTime": null,
"kWhDelivered": 9.583,
"sessionID": "1_1_179_800_2018-09-05 11:04:12.876087",
"siteID": "0001",
"spaceID": "AG-3F32",
"stationID": "1-1-179-800",
"timezone": "America/Los_Angeles",
"userID": null,
"userInputs": null
}

Another record with userInputs:

{

```
  "clusterID": "0001",
  "connectionTime": "Wed, 05 Sep 2018 11:08:09 GMT",
  "disconnectTime": "Wed, 05 Sep 2018 14:09:02 GMT",
  "kWhDelivered": 7.114,
  "sessionID": "1_1_179_794_2018-09-05 11:08:08.945820",
  "siteID": "0001",
  "spaceID": "AG-3F20",
  "stationID": "1-1-179-794",
  "timezone": "America/Los_Angeles",
  "userID": "000000333",
  "userInputs": [
    {
      "WhPerMile": 400,
      "kWhRequested": 8.0,
      "milesRequested": 20,
      "minutesAvailable": 277,
      "paymentRequired": true
    }
  ]
}
```


These examples illustrate the structure that the preprocessing functions need to handle.

Appendix B – Backend Code Snippets (FastAPI)

B.1 JSON Ingestion and Flattening

```
from fastapi import FastAPI, UploadFile, File, Body
import pandas as pd
import io
import os
import json

app = FastAPI(title="EV Charging Station Demand Forecasting API")

BASE_DIR = os.path.dirname(__file__)
DATA_DIR = os.path.join(BASE_DIR, "data", "uploads")
os.makedirs(DATA_DIR, exist_ok=True)
LATEST_FILLED_CSV = os.path.join(DATA_DIR, "ACN-data-filled.csv")

def parse_acn_json(payload: dict) -> pd.DataFrame:
    if not isinstance(payload, dict) or "_items" not in payload:
        return pd.DataFrame()
    items = payload["_items"]
    df = pd.json_normalize(items, sep=".")
    return df

@app.post("/ingest-json")
async def ingest_json(payload: dict = Body(...)):
```

```
    try:
        df = parse_acn_json(payload)
    except Exception as e:
        return {"ok": False, "message": f"Failed to parse JSON: {e}"}
    if df.empty:
        return {"ok": False, "message": "No _items found in JSON payload."}
    df.to_csv(LATEST_FILLED_CSV, index=False)
    return {
        "ok": True,
        "rows": int(len(df)),
        "columns": list(df.columns),
        "message": "Flattened JSON saved to ACN-data-filled.csv"
    }
```

B.2 Aggregation Endpoint

```
@app.get("/clean")
def get_clean(site: Optional[str] = None, freq: str = "H", limit: int = 200):
    raw = pd.read_csv(LATEST_FILLED_CSV) if os.path.exists(LATEST_FILLED_CSV) else
pd.DataFrame()
    before_rows = int(len(raw))
    agg = build_hourly_daily(raw, site=site, freq=freq)
    after_rows = int(len(agg))

    if agg.empty:
        return {
            "ok": True,
            "message": f"Could not build aggregated series. Raw rows: {before_rows}.",
            "preview": []
        }

    preview = json.loads(agg.reset_index().head(limit).to_json(orient="records"))
    return {
        "ok": True,
        "message": f"Aggregated {before_rows} raw rows into {after_rows} {freq}-level buckets.",
        "preview": preview,
    }
```

B.3 Forecast Endpoint (Core Idea)

```
from pydantic import BaseModel
from statsmodels.tsa.statespace.sarimax import SARIMAX
import numpy as np

class ForecastRequest(BaseModel):
    site: Optional[str] = None
    metric: str = "energy"   # "energy" or "sessions"
    freq: str = "H"
    horizon: int = 48
    test_size: int = 48
```

```
@app.post("/forecast")
def forecast(req: ForecastRequest):
    raw = pd.read_csv(LATEST_FILLED_CSV) if os.path.exists(LATEST_FILLED_CSV) else
pd.DataFrame()
    agg = build_hourly_daily(raw, site=req.site, freq=req.freq)

    if agg.empty:
        return {"ok": False, "message": "No aggregated data available for forecasting."}

    series = agg["energy_kwh"].dropna().astype(float)
    if len(series) < 50:
        return {"ok": False, "message": "Not enough data points for forecasting."}

    y = series
    y_train = y.iloc[:-req.test_size]
    y_test = y.iloc[-req.test_size:]

    model = SARIMAX(
        y_train,
        order=(1, 1, 1),
        seasonal_order=(1, 1, 1, 24),
        enforce_stationarity=False,
        enforce_invertibility=False,
    )
    res = model.fit(disp=False)

    fc_test = res.get_forecast(steps=len(y_test))
    test_pred = fc_test.predicted_mean.values
    metrics = evaluate_np(y_test.values, test_pred)

    fc_future = res.get_forecast(steps=req.horizon)
    future_mean = fc_future.predicted_mean
    ci = fc_future.conf_int(alpha=0.2)

    return {
        "ok": True,
        "metrics": metrics,
        "horizon": req.horizon,
        "forecast": [
            {"ts": str(ts), "y_pred": float(val), "lower": float(l), "upper": float(u)}
            for (ts, val, (l, u)) in zip(
                future_mean.index, future_mean.values, ci.values
            )
        ],
    }
```

These snippets summarize the practical implementation used to support the time series analysis presented in the main body of the report.