

Use Case – Sea-cargo traffic analysis with AI “Google Colab”

Instructions: Using ChatGPT and Colab to analyze sea-cargo traffic and create explainable visualizations for cargo shipping trends and anomalies.

Overview

Here we propose a use case for cargo traffic data analysis using generative AI tools. The data contains the traffic of cargo vessels in Baltic sea ports within 12 months.

Data details:

Port arrivals of cargo ships to the following countries: Sweden, Finland, Estonia, Latvia, Lithuania, Poland, during the period of time: 2021-01-01 00:00' UTC and '2022-04-31 23:59' UTC.

Cargo Vessels type: all cargo and all tankers.

Ship Dimension: >= 65 meter length.

Fields for each arrival event: Port ID, Port name, LOCODE, MMSI, IMO, Vessel name, Vessel destination, Vessel type, Time of arrival, Time of departure.

Number of events: 87624.

Steps

Step 1: Prompt to understand preliminary Data Analysis

ChatGPT prompt:

“You are a maritime cargo shipping expert. Summarize this dataset for me. What trends or anomalies do you see?”

Answer highlights:

Short Sea Shipping, Regional Trade and energy and bulk logistics corridors were correctly identified. Conforms stable, contract-based flows, not spot-market volatility.

Identified Data Inconsistency, Arrival and departure destination fields are not harmonized.

AI Recommended Next Steps:



1. Data engineering pass
2. Port & destination normalization

Manual data validation confirmed issues in data:

- Data in certain columns are not normalized i.e. differs by some chars. For example destination port value can be "LVVNT" or "LV VNT" or "VENTSPILS" for the same port in different events.
- For Some events data column "vesselDestinationArrival" contains two-port route, non-normalized value, for example "SE STO FI SKV".

Screenshots:

portName	portLocode	portArrival	portDeparture	vesselMMSI	vesselIMO	vesselName	vesselDestinationArrival	vesselDestinationDeparture
Ventspils	LVVNT	2021-01-01 01:38:44	2021-01-01 17:12:02	245286000	9207039	CORAL IVORY	LV VNT	SE KOG
Ventspils	LVVNT	2021-01-01 02:27:45	2021-01-01 20:40:16	276848000	9148843	CRYSTALWATER	VENTSPILS	PALJASSARE
Jakobstad	FIPRS	2021-01-01 02:28:19	2021-01-02 19:37:21	244621000	9201827	LADY CARINA	PIETARSAARI	EMDEN VIA NOK
Brofjorden	SEBRO	2021-01-01 03:02:17	2021-01-02 03:06:55	255623000	9260407	AMARANT	SE BRO	SE BRO
Göteborg	SEGOT	2021-01-01 03:22:11	2021-01-01 15:30:01	246694000	9284647	BIT OKTANIA	SEGOT	SEKIT
Klaipeda	LTKLJ	2021-01-01 04:24:44	2021-01-01 18:12:31	209467000	9483671	X PRESS AGILITY	LTKLJ	NLRMT

Very short dwell time:

28190	Helsinki	FIHEL	2021-01-01 04:41:31	2021-01-01 18:12:24	255806258	9344710	VIOLETTA	FIHEL	FIKTK
28820	Swinoujscie	PLSWI	2021-01-01 05:37:52	2021-01-01 05:37:52	261037120	9131199	ANGON	SZCZECIN	SZCZECIN
27850	Tolkkinen	FITOK	2021-01-01 06:29:12	2021-01-01 23:33:22	230957000	9267560	SUULA	SE STO FI SKV	FI SKV SE STO

Answer screenshot:



5. Recommended Next Steps

If your objective is analytical or decision-support oriented, I recommend:

1. **Data engineering pass**
 - Clean quoting, encoding, column names
2. **Port & destination normalization**
 - Map all destinations to UN/LOCODE
3. **Derived KPIs**
 - Average dwell time by port
 - Vessel turnaround frequency
 - Tanker vs cargo ship split by port
4. **Energy corridor mapping**
 - Identify dominant tanker routes and dependencies

4.2 Time Validation

- Remove or flag:
 - Negative dwell times
 - Dwell < 30 minutes (unless anchorage)
 - Overlapping calls for same vessel

4.3 Destination Harmonization

- Map free-text destinations → UN/LOCODE
- Enforce **one canonical destination per port**

Tips:

<https://medium.com/ai-analytics-diaries/7-chatgpt-prompts-that-i-use-daily-as-a-data-analyst-714bb47839cc>



Step 2: Data Normalization

ChatGPT prompt

“Harmonize Arrival and departure destination fields”

Answer: Data harmonization objectives and samples of data transformation. What will be changed.

Removes spaces in UN/LOCODE (SE GOT → SEGOT), Easy to extend with additional ports, keeps raw columns for traceability.

Tips:

[Add helpful tips or best practices here]

Step 3: Creating data processing script code for Colab (Python)

Next we load data to Colab, adding code, running code boxes.

ChatGPT prompt

Create a Python script that corrects destinations in these 2 columns,

"vesselDestinationArrival", "vesselDestinationDeparture", to UN/LOCODE:

- Map free-text destinations to UN/LOCODE
- remove spaces in UN/LOCODE

Answer: Python script code. Destination_normalization.py

```
# Auto-generated UN/LOCODE mapping  
# Do not edit manually
```

```
PORT_NAME_TO_UNLOCODE = {  
    'BROFJORDEN': 'SEBRO',  
    'GOTHENBURG': 'SEGOT',  
    'JAKOBSTAD': 'FIJAK',  
    'VENTSPILS': 'LVVEN',
```



Next Prompt:

write python script that creates a Python dictionary data structure to map UN/LOCODE from "portName" to "portLocode" by checking all items in csv data.

Uploading data to Collab,

- Create new code block, and insert code from ChatGPT
- Click on Folder icon , create "ais" folder under /content path and upload data file (PRJ896.csv)
- Edited script variables `INPUT_CSV = "/content/ais/PRJ896.csv"`
- `OUTPUT_PY = "/content/ais/port_name_to_unlocode.py"` which will contain `PORT_NAME_TO_UNLOCODE = {}` structure.
-



CO AIS.ipynb ☆ ☁
File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all ▾

```
[ ] ▶ import pandas as pd
import unicodedata
import re
from collections import defaultdict

# -----
# CONFIGURATION
# -----

INPUT_CSV = "/content/ais/PRJ896.csv"
PORT_NAME_COL = "portName"
PORT_LOCODE_COL = "portLocode"
OUTPUT_PY = "/content/ais/port_name_to_unlocode.py"
DICT_NAME = "PORT_NAME_TO_UNLOCODE"

# -----
# NORMALIZATION FUNCTIONS
# -----
```

CO AIS.ipynb ☆ ☁
File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all ▾

RAM
Disk

Table of contents

- Task
- Load CSV
- Summary:
- + Section

```
[ ] ▶ import pandas as pd
import unicodedata
import re
from collections import defaultdict

# -----
# CONFIGURATION
# -----

INPUT_CSV = "/content/ais/PRJ896.csv"
PORT_NAME_COL = "portName"
PORT_LOCODE_COL = "portLocode"
OUTPUT_PY = "/content/ais/port_name_to_unlocode.py"
DICT_NAME = "PORT_NAME_TO_UNLOCODE"

# -----
# NORMALIZATION FUNCTIONS
# -----

def normalize_port_name(name: str) -> str | None:
    """
    Normalize port names:
    - Remove accents
    - Uppercase
    - Strip spaces
    """
    if not isinstance(name, str):
        return None

    name = unicodedata.normalize("NFKD", name)
    name = name.encode("ASCII", "ignore").decode("ASCII")
```

Files

- bin
- boot
- content
 - .config
 - .ipynb_checkpoints
 - ais
 - PRJ896.csv
 - sample_data
 - README.md
 - anscombe.json
 - california_housing_test.csv
 - california_housing_train.csv
 - mnist_test.csv
 - mnist_train_small.csv
 - datalab
 - dev

86.53 GB available

Python 3



The screenshot shows a JupyterLab interface for a file named 'AIS.ipynb'. The top navigation bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below this is a search bar for 'Commands' and buttons for '+ Code', '+ Text', and 'Run all'. A sidebar on the left contains several icons: a hamburger menu, a search icon, a code icon, a key icon, a folder icon (highlighted with a yellow circle), and a grid icon. The main area displays a 'Table of contents' sidebar with the following items: 'Task', 'Load CSV', 'Summary:', and '+ Section'. A small '[]' icon is visible to the right of the sidebar.

```
import pandas as pd
import unicodedata
import re
from collections import defaultdict

# -----
# CONFIGURATION
# -----

INPUT_CSV = "/content/ais/PR3896.csv"
PORT_NAME_COL = "portname"
PORT_LOCODE_COL = "portlocode"
OUTPUT_PY = "/content/ais/port_name_to_unlocode.py"
DICT_NAME = "PORT_NAME_TO_UNLOCODE"

# -----
# NORMALIZATION FUNCTIONS
# -----

def normalize_port_name(name: str) -> str | None:
    """
    Normalize port names:
    - Remove accents
    - Uppercase
    - Strip spaces
    """
    if not isinstance(name, str):
        return None

    name = unicodedata.normalize("NFKD", name)
    name = name.encode("ASCII", "ignore").decode("ASCII")
```

Change variable INPUT_CSV and OUTPUT_PY .

Result: script.py

Sample result:



```
# -----  
# EXECUTION  
# -----  
  
if __name__ == "__main__":  
    port_mapping, port_conflicts = build_port_mapping(INPUT_CSV)  
  
    print(f"Unique port mappings created: {len(port_mapping)}")  
  
    if port_conflicts:  
        print("\nWARNING: Conflicting UN/LOCODEs found for the same port name:")  
        for port, locodes in port_conflicts.items():  
            print(f"  {port}: {' '.join(sorted(locodes))}")  
    else:  
        print("No conflicts detected.")  
  
    write_python_dict(port_mapping, OUTPUT_PY, DICT_NAME)  
  
    print(f"Mapping written to {OUTPUT_PY}")  
    print(f"Total unique mappings: {len(mapping)}")  
  
    if conflicts:  
        print("\nWARNING: Conflicting mappings detected:")  
        for port, locodes in conflicts.items():  
            print(f"  {port}: {sorted(locodes)}")
```

Script result

Unique port mappings created: 124

Data after script execution:



vessel	Destination	Departure	processed_vessel	Destination	Departure
		SE KOG			SEKOG
		PALJASSARE			EEPAS
	EMDEN VIA NOK			EMDEN VIA NOK	
		SE BRO			SEBRO
		SEPIT			SEPIT
		NLRTM			NLRTM
	FIRAU DEBRV			FIRAU DEBRV	
	PL GDN SE NRK			PLGDN SENRK	
		FIKTK			FIKTK
		SZCZECIN			PLSZZ
	FI SKV SE STO			FISKV SESTO	
	SEGOT PEC 62 552			SEGOTPEC 62552	
		SEGOT FIINK			SEGOT FIINK

Tips:

[Add helpful tips or best practices here]

Step 4: Creating Time-based anomaly detection code

Prompt: propose python code code for Time-based anomaly filtering

Answer: puthon code: apply_time_anomaly_flags



```
def apply_time_anomaly_flags(df):
    """
    Adds time-based anomaly flags to a port-call dataframe.
    Does not remove records.
    """

    df = df.copy()

    # Ensure datetime
    df["arrival_time_utc"] = pd.to_datetime(df["arrival_time_utc"], utc=True, err
    df["departure_time_utc"] = pd.to_datetime(df["departure_time_utc"], utc=True,

    # Compute dwell time
    df["port_dwell_hours"] = (
        df["departure_time_utc"] - df["arrival_time_utc"]
    ).dt.total_seconds() / 3600

    # Initialize flag column
    df["time_anomaly_flag"] = None

    # -----
    # Rule 1: Negative dwell time
    # -----
    df.loc[df["port_dwell_hours"] < 0, "time_anomaly_flag"] = "NEGATIVE_DWELL"

    # -----
    # Rule 2: Too short dwell
    # -----
```



1. Anomaly Types Covered

This proposal detects and flags:

1. Negative dwell time
2. Unrealistically short port stays
3. Unrealistically long port stays
4. Overlapping port calls for the same vessel
5. Chronologically out-of-order calls

All anomalies are **flagged, not deleted**.

2. Assumptions

- DataFrame contains:
 - imo_number
 - arrival_time_utc
 - departure_time_utc
 - Times are UTC or convertible to UTC
 - One row = one port call
-

3. Configuration (Business Rules)

python

```
MIN_DWELL_HOURS = 0.5      # < 30 minutes → anomaly (unless anchorage)
MAX_DWELL_HOURS = 336     # > 14 days → anomaly
ALLOWABLE_OVERLAP_MIN = 5 # minutes tolerance for AIS noise
```

Usage:



5. Example Usage

```
python

df = pd.read_csv("port_calls_cleaned.csv")

df_flagged = apply_time_anomaly_flags(df)

# Review anomalies only
anomalies = df_flagged[df_flagged["time_anomaly_flag"].notna()]
```

Tips:

[Add helpful tips or best practices here]

Step 5: Time-based anomaly filtering and visualization

Prompt:

provide python script to visualize analysis for port_dwell_hours, time_anomaly_flag in anomalies data.

Answer:

time_anomalies_script.py

Below is a clear, reusable Python script to visualize port dwell time and time-based anomalies from your dataset.



```
# -----  
# CONFIGURATION  
# -----  
  
INPUT_CSV = "port_calls_with_time_anomalies.csv"  
  
DWELL_COL = "port_dwell_hours"  
FLAG_COL = "time_anomaly_flag"  
  
# -----  
# LOAD DATA  
# -----  
  
df = pd.read_csv(INPUT_CSV)
```

```
# Drop rows without dwell time  
df = df[df[DWELL_COL].notna()]
```

We add code to Colab and adapt to use data from previous box:



```
[ ] df_flagged = apply_time_anomaly_flags(df)

# Review anomalies only
anomalies = df_flagged[df_flagged["time_anomaly_flag"].notna()]

[ ] import pandas as pd
import matplotlib.pyplot as plt

# Drop rows without dwell time
df_dwell = df_flagged[df_flagged["port_dwell_hours"].notna()]

# -----
# 1. Distribution of Port Dwell Hours
# -----
```

Tips:

[Add helpful tips or best practices here]

Cargo-traffic anomalies visualization

[List any optional features or advanced settings]

Prompt:

provide a Python script to visualize analysis for port_dwell_hours, time_anomaly_flag



```
[ ] ▶ import pandas as pd
import matplotlib.pyplot as plt

# Drop rows without dwell time
df_dwelling = df_flagged[df_flagged["port_dwell_hours"].notna()]

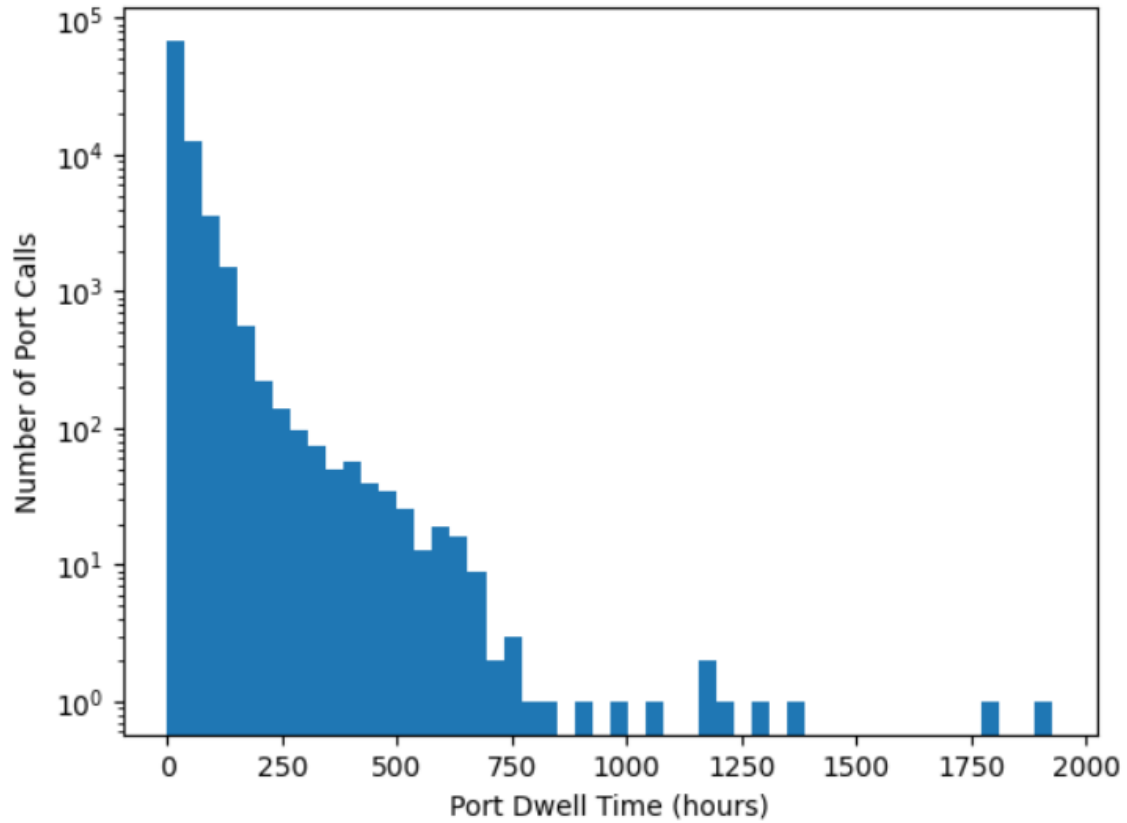
# -----
# 1. Distribution of Port Dwell Hours
# -----

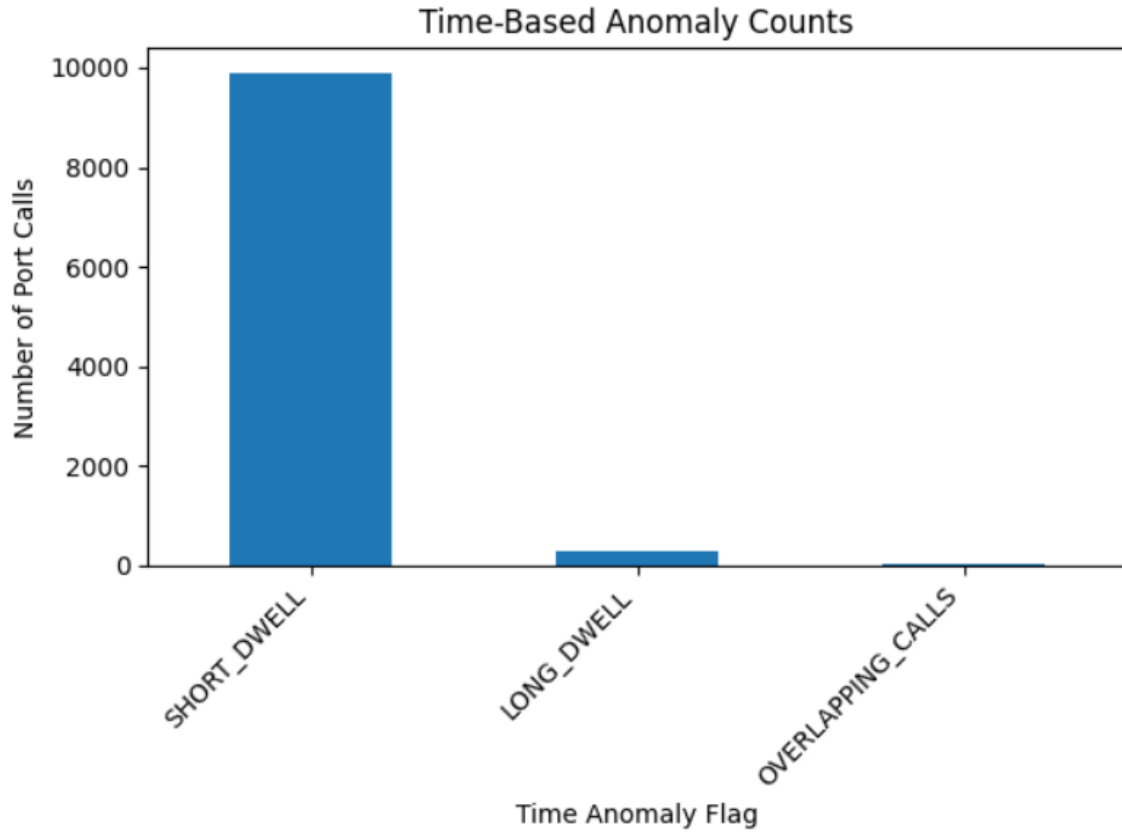
plt.figure()
plt.hist(df_dwelling["port_dwell_hours"], bins=50)
plt.xlabel("Port Dwell Time (hours)")
plt.ylabel("Number of Port Calls")
plt.title("Distribution of Port Dwell Time")
plt.yscale("log") # useful for heavy-tailed dwell distributions
plt.show()
```

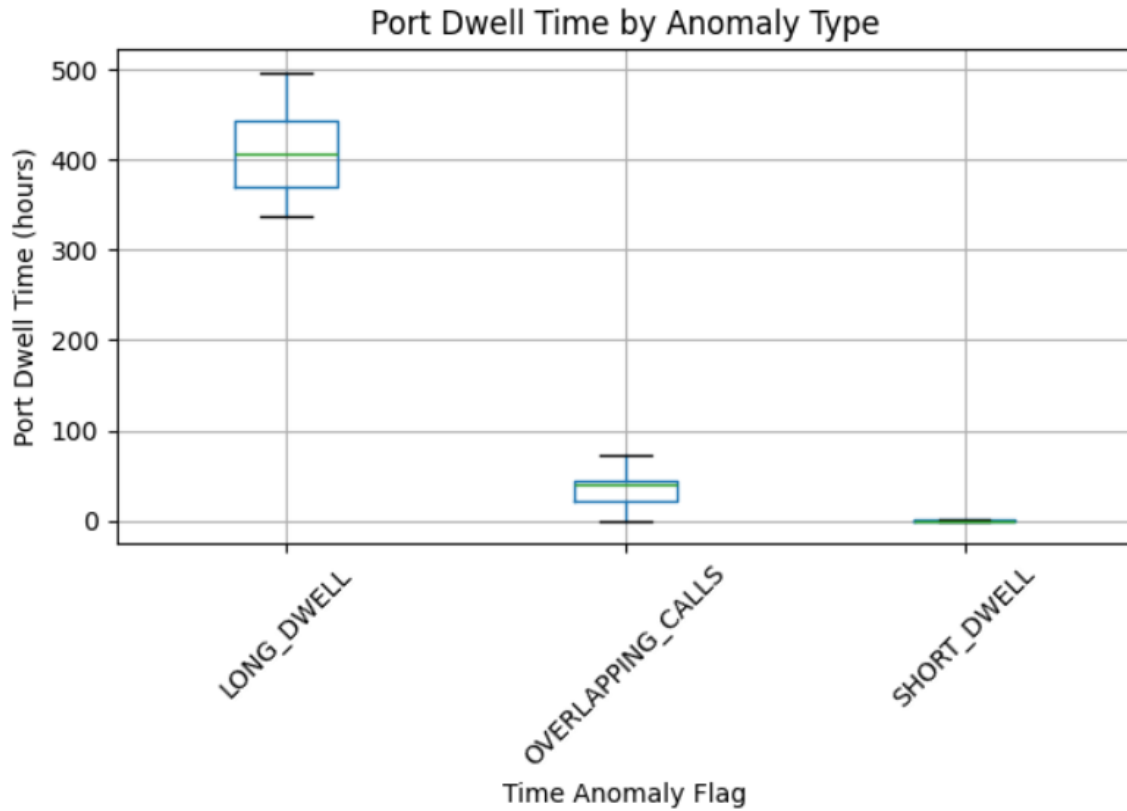
4 diagrams below: "Distribution of Port Dwell Time", "Time-Based Anomaly Counts", "Port Dwell Time by Anomaly Type"



Distribution of Port Dwell Time







Prompt:

provide python script to visualize analysis for port_dwell_hours, time_anomaly_flag related to port name.

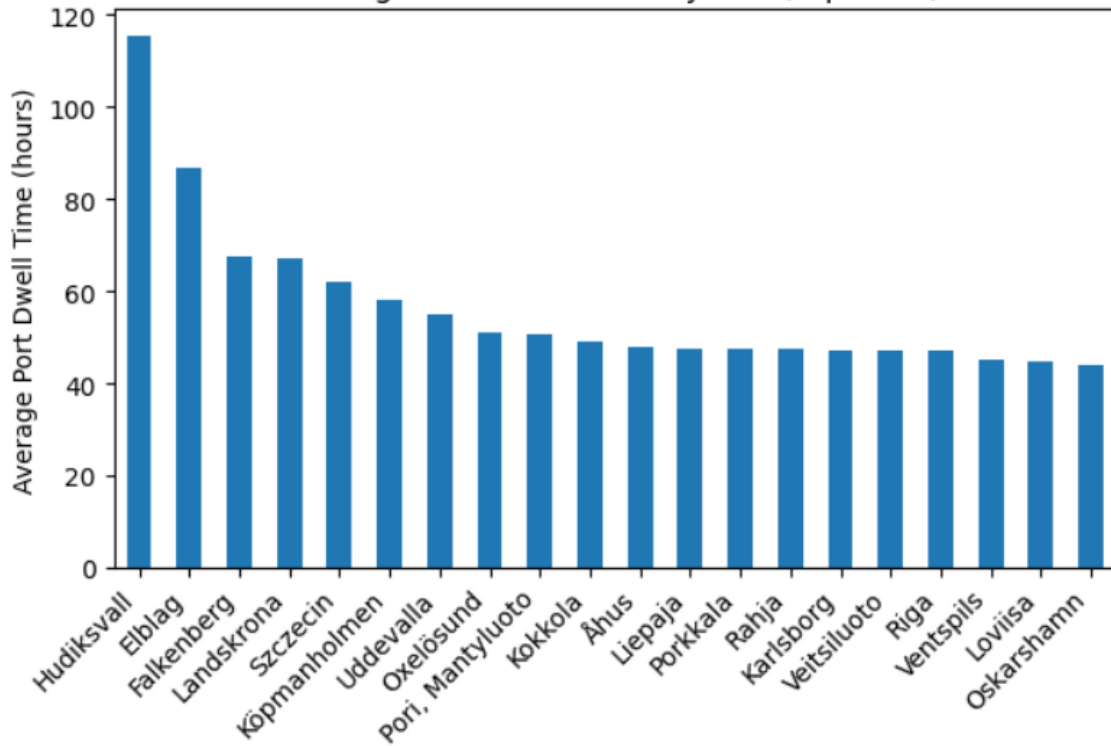


```
[ ]  
  
#import pandas as pd  
import matplotlib.pyplot as plt  
  
# -----  
# CONFIGURATION  
# -----  
  
#INPUT_CSV = "port_calls_with_time_anomalies.csv"  
  
PORT_COL = "portName"  
DWELL_COL = "port_dwell_hours"  
FLAG_COL = "time_anomaly_flag"  
  
TOP_N_PORTS = 20      # number of ports to display  
MAX_DWELL_HOURS = 500 # cap for visualization (~20 days)  
  
# -----  
# LOAD & PREP DATA  
# -----  
  
#df = pd.read_csv(INPUT_CSV)  
  
# Basic filtering  
df = df_dwell #df[df[DWELL_COL].notna()]  
df[FLAG_COL] = df[FLAG_COL].fillna("OK")  
  
# -----  
# 1. Average Port Dwell Time (Top N Ports)
```

Running code:

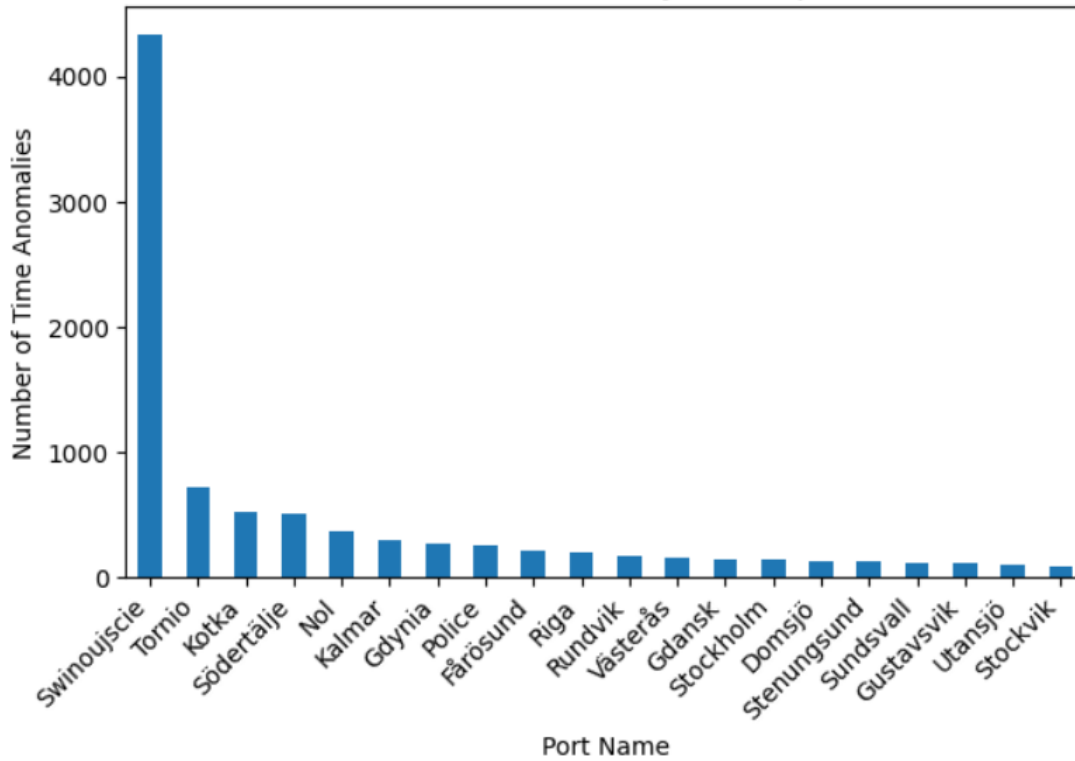


Average Port Dwell Time by Port (Top Ports)



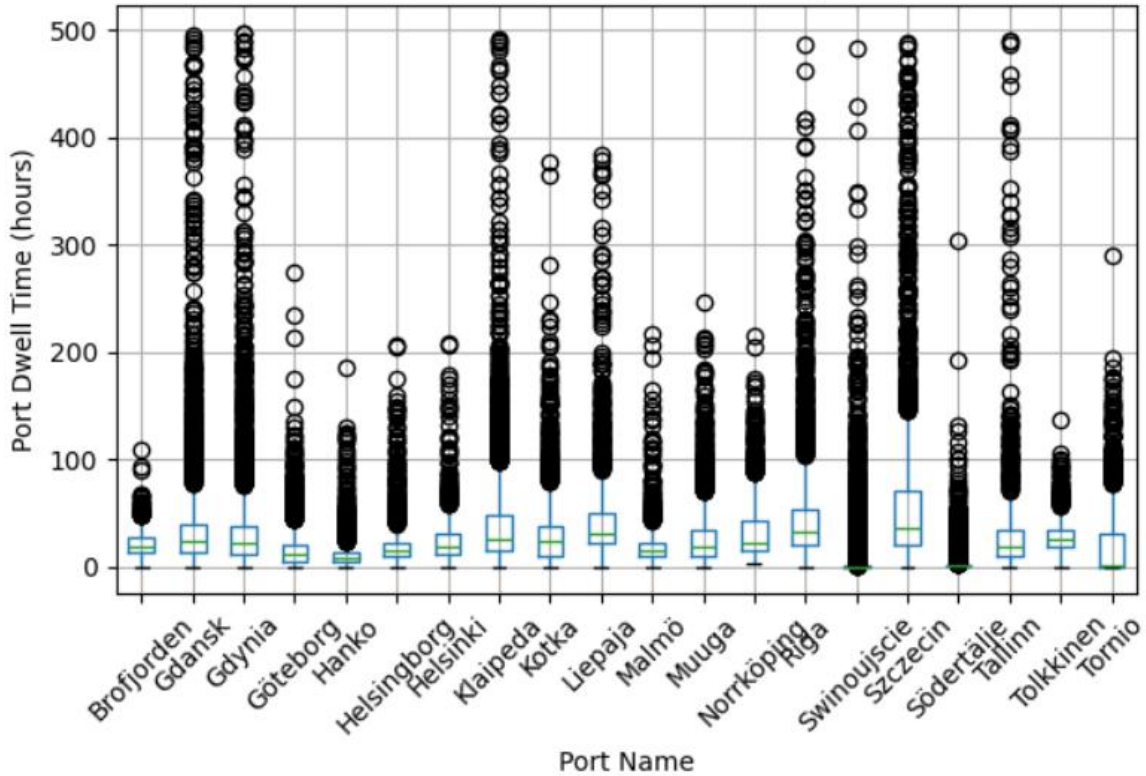


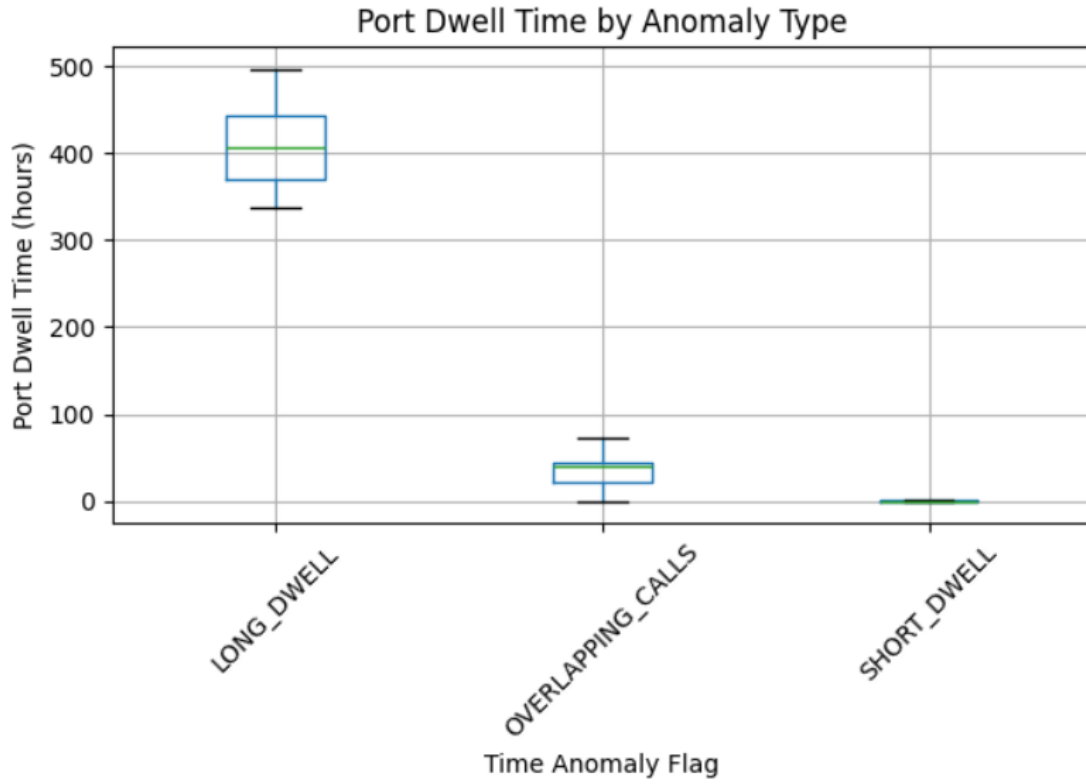
Time-Based Anomalies by Port (Top Ports)





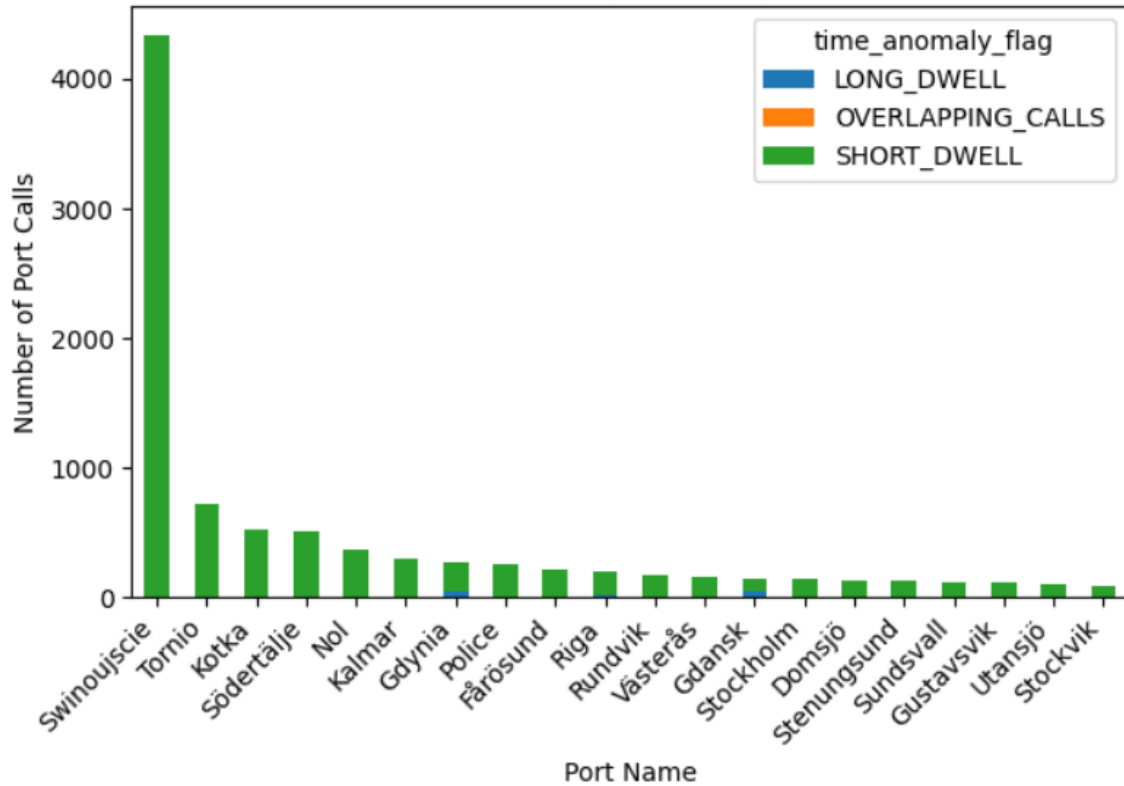
Port Dwell Time Distribution by Port







Time Anomaly Profile by Port



Resources

[Add links to guides, videos, or documentation]