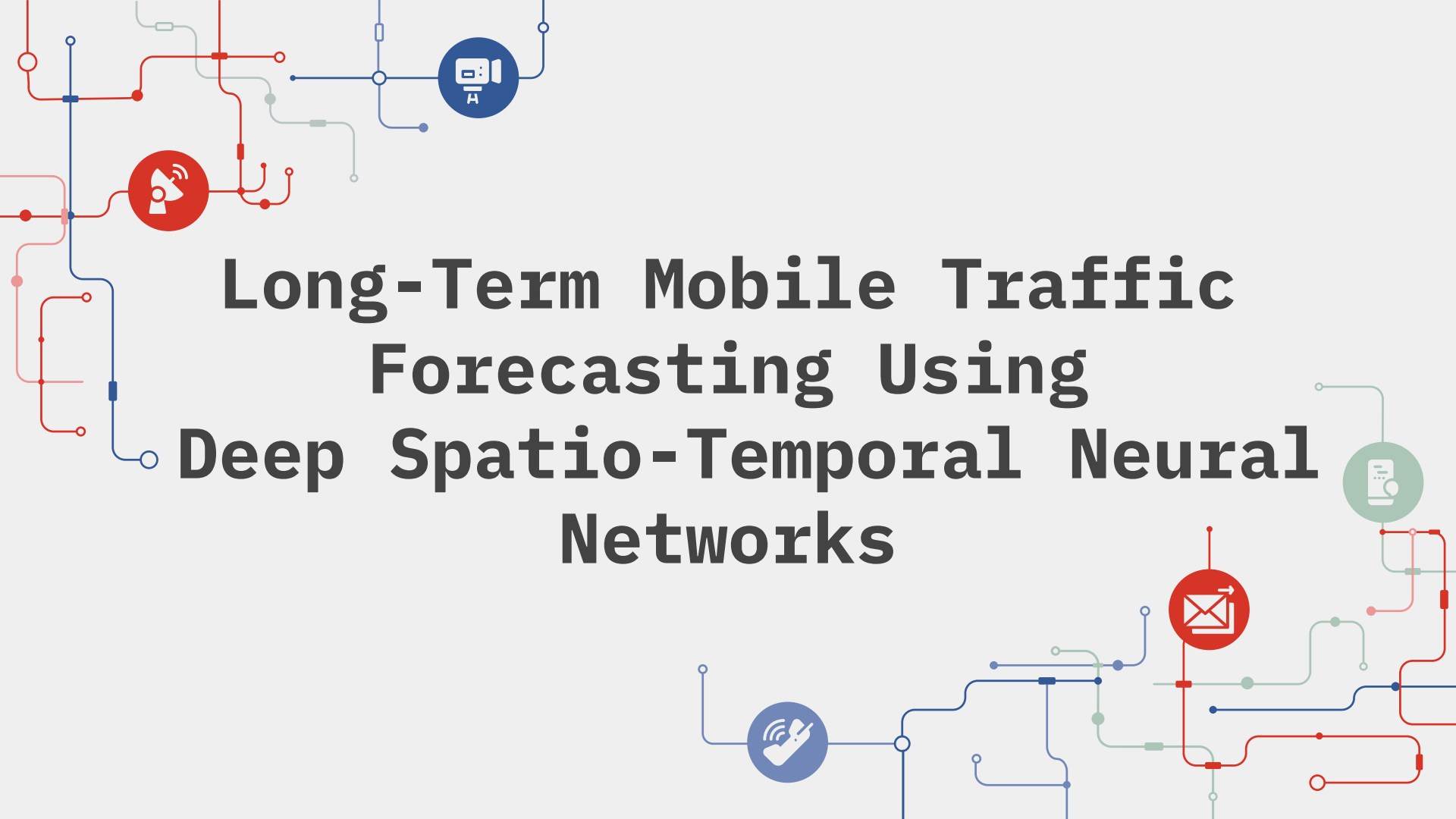
Long-Term Mobile Traffic Forecasting Using Deep Spatio-Temporal Neural Networks



Forecast future mobile traffic patterns over time in different regions of Milan and Trentino.

Mobile Traffic Dataset Overview

- **Regions & Grid Setup**:
  - **Milan**: 100 × 100 grid cells (each cell: **235m × 235m**)
  - **Trentino**: 117 × 98 grid cells (each cell: **1km × 1km**)
- **Data Format**:
  - Each data point represents **traffic volume (MB)** per cell
  - Recorded every **10 minutes**
- **Time Span**:
  - **60 days** total
  - From **Nov 1, 2013** to **Jan 1, 2014**

9901	9902	···	9999	10000
9801	···	···	9899	9900
···	···	···	···	···
101	102	···	···	200
1	2	3	···	100

Milan Grid

Nature

Telecommunications activity

The first type of dataset represents the activity of Trentino and Milan, showing all the aforementioned telecommunication events which took place within these areas. The data provides information of Telecom Italia's customers interacting with the network and of other people using it while roaming.

Telecommunications interactions

Two types of CDR datasets were also produced to measure the interaction intensity between different locations: one from a particular area (Trentino/Milan) to any of the Italian provinces and one quantifying the interactions within the city/province (e.g., Milan to Milan). Since Telecom Italia only possesses the data of its own customers, the computed interactions are only between them. This means that (at most) 34% of population's data is collected, due to Telecom Italia's market share (http://www.agcom.it/documents/10179/1734740/Studio-Ricerca+24-07-2014/5541e017-3c7a-42ff-b82f-66b460175f68?version=1.0, date of access 06/08/2014). Moreover there is no information about missed calls.

Telecommunications Activity Dataset — Field Summary

Field	Description
`square_id`	Unique ID of a 100 m × 100 m cell in the Milan / Trentino grid.
`time_interval`	Start-time of a **10-minute** window (Unix ms). End-time = `start + 600 000 ms`.
`sms_in`	Relative volume of **SMS received** in the cell during the interval.
`sms_out`	Relative volume of **SMS sent** from the cell during the interval.
`call_in`	Relative volume of **voice calls received** in the cell during the interval.
`call_out`	Relative volume of **voice calls originated** from the cell during the interval.
`internet_traffic`	Count of data-session CDRs generated in the cell during the interval.
`country_code`	ITU phone country code of the remote party (SMS / call) or data-session initiator.

Spatial Resolution

Region	Grid Dimensions (rows × cols)	Area per Cell
Milan	100 × 100	235 m × 235 m ≈ 0.055 km²
Trentino	117 × 98	1 km²

Temporal Split

Purpose	Time Range	Days	Region(s) Covered
Training	**Nov 1 – Dec 10**	40	Milan
Validation	**Dec 10 – Dec 20**	10	Milan
Testing	**Dec 20 – Dec 30**	10	Milan + Trentino

Snapshot Geometry

Feature	Value
Duration	2 months (Nov 1 – Jan 1)
Snapshot interval	Every **10 minutes**
Snapshots per day	24 h × 6 = **144**
Total days	**60**
Total snapshots	60 × 144 = **8 640**

Feature Selection Strategy 📊

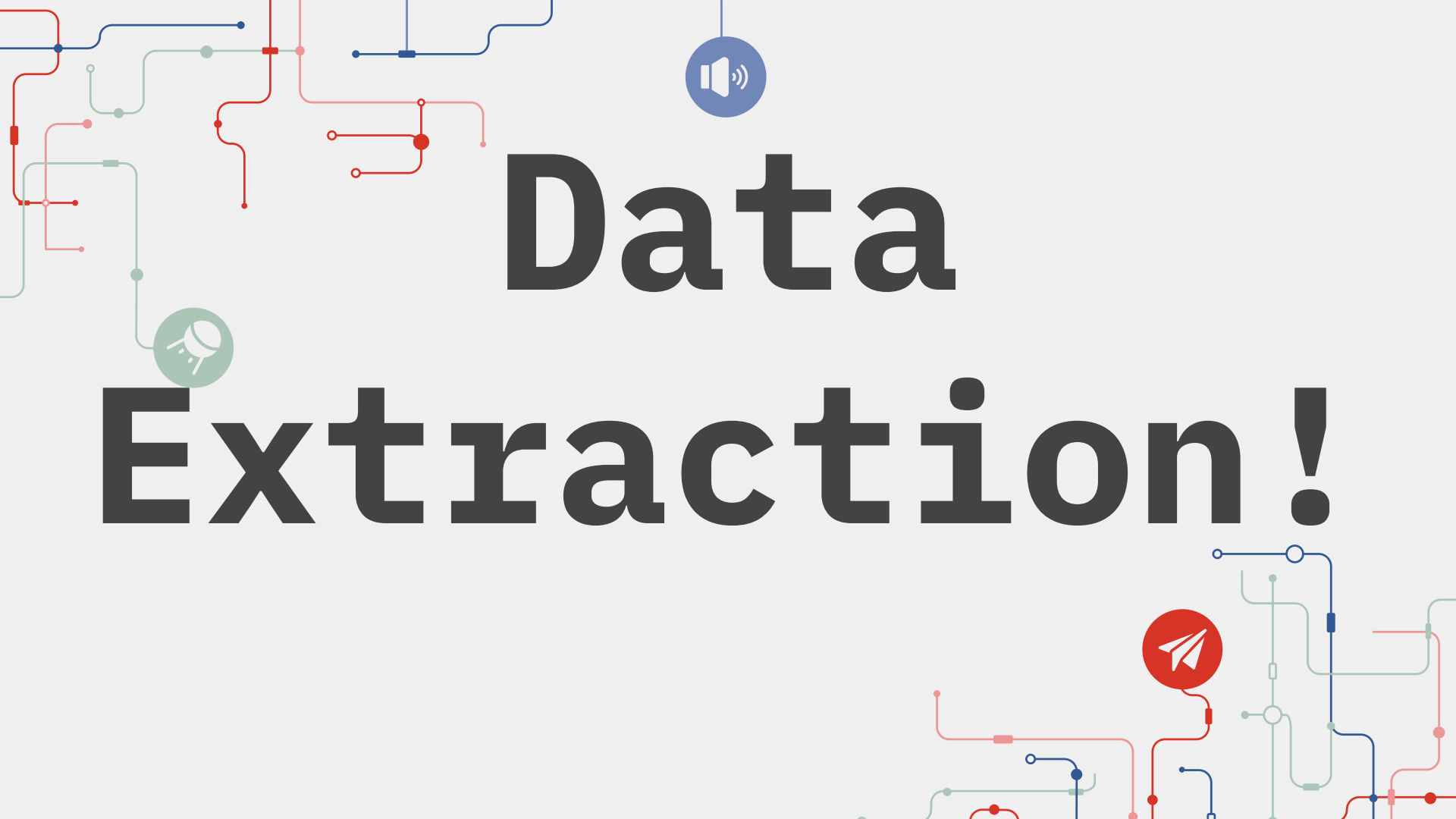
1. Core Identifiers

Column	Role
`square_id`	Spatial key – identifies each 100 m × 100 m grid cell
`time_interval`	Temporal key – 10-min start timestamp (end = start + 600 000 ms)

---

2. Candidate Telecommunication Features

Feature(s)	Issue(s) Observed	Decision
`sms_in`, `sms_out`	Paired & correlated → multicollinearity risk	**Dropped**
`call_in`, `call_out`	Same pairing issue; weaker seasonality; roaming noise	**Dropped**
`internet_traffic`	Strong daily / weekly cycles; single, self-contained metric	**Selected**


Data
Extraction!


Country ID!

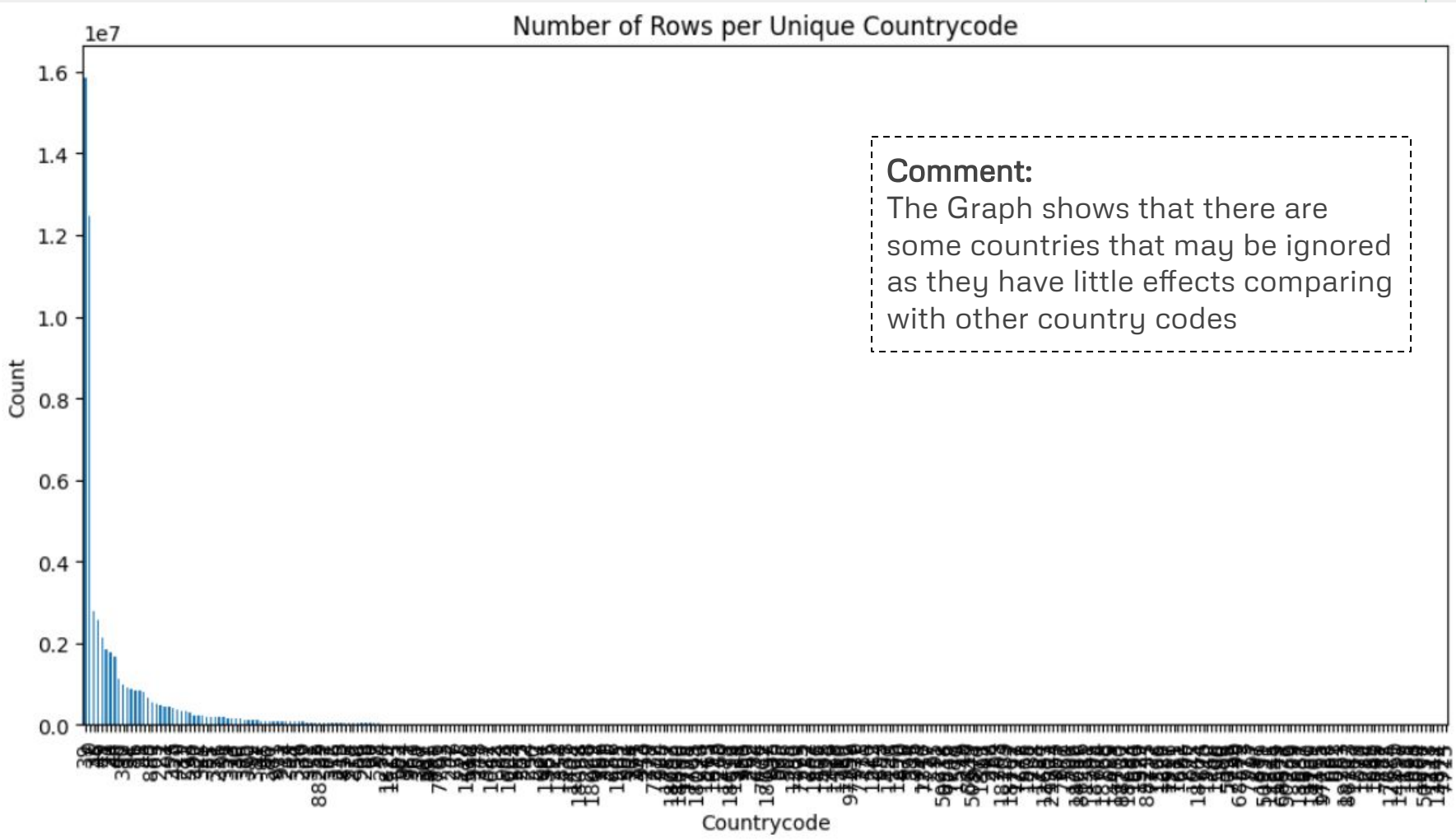
	CellID	datetime	countrycode	smsin	smsout	callin	callout	internet
0	1	1383260400000	0	0.081363	NaN	NaN	NaN	NaN
1	1	1383260400000	39	0.141864	0.156787	0.160938	0.052275	11.028366
2	1	1383261000000	0	0.136588	NaN	NaN	0.027300	NaN
3	1	1383261000000	33	NaN	NaN	NaN	NaN	0.026137
4	1	1383261000000	39	0.278452	0.119926	0.188777	0.133637	11.100963

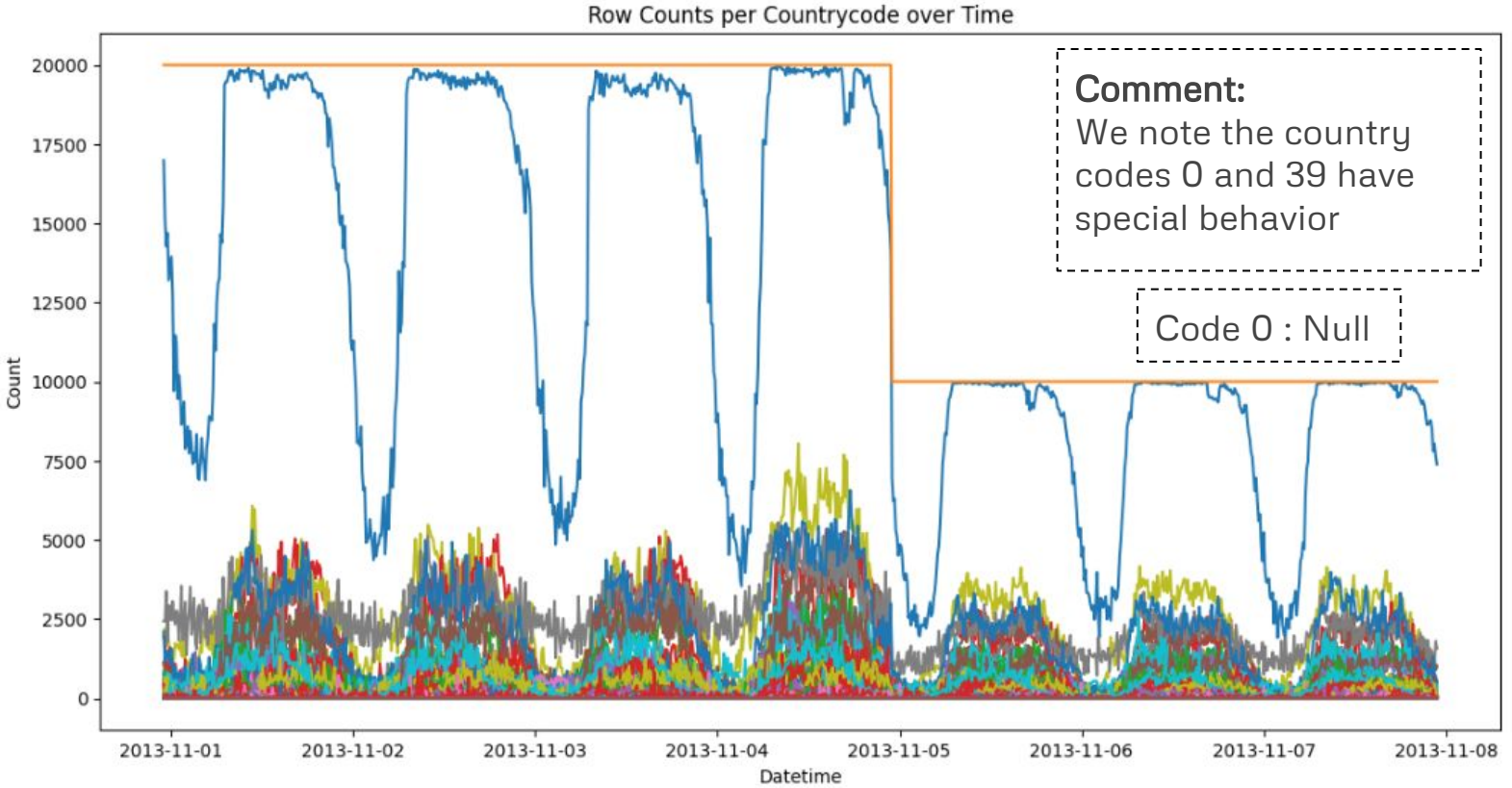
Country code: the phone country code of the nation.

Number of unique country codes: **326**

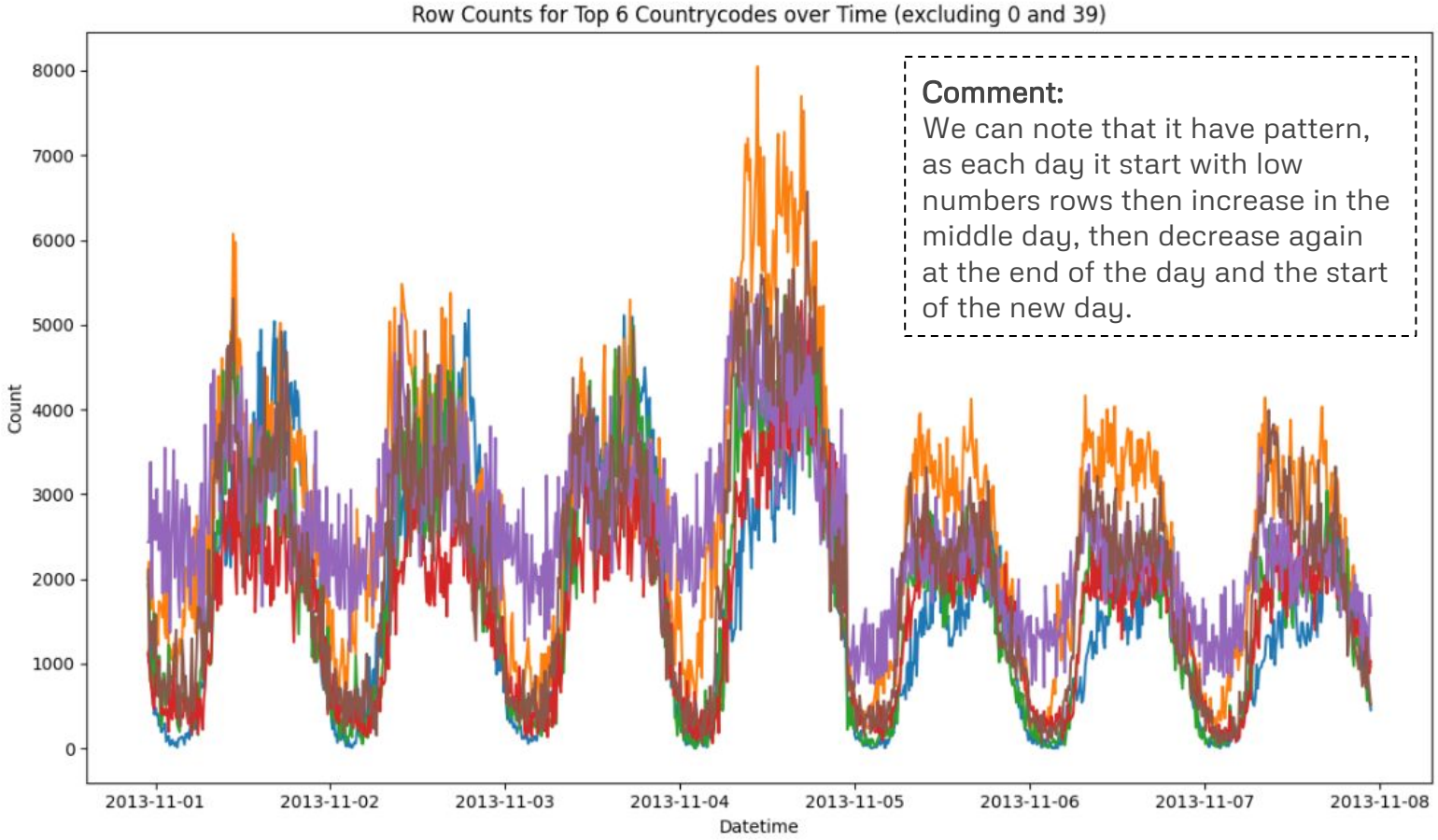
International Calls & Roaming:
Even if the physical network or cell towers are in Italy, the phone calls or messages might involve numbers from many different countries due to roaming or international communications. So, the `countrycode` here refers to the originating or terminating phone number's country.

```
countrycode
39       15839751
0        12475454
33        2770562
46        2556736
49        2146999
           ...
7718            7
1441            6
12844           4
1924            4
7715            1
Name: count, Length: 326, dtype: int64
```


Number of Rows per Unique Countrycode
1e7
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Count
Countrycode
Comment:
The Graph shows that there are some countries that may be ignored as they have little effects comparing with other country codes



Code 39 has uniform pattern as it is for the original people


Row Counts for Top 6 Countrycodes over Time (excluding 0 and 39)
Count
Datetime
8000
7000
6000
5000
4000
3000
2000
1000
0
2013-11-01
2013-11-02
2013-11-03
2013-11-04
2013-11-05
2013-11-06
2013-11-07
2013-11-08
Countrycode
Countrycode 20
Countrycode 33
Countrycode 41
Countrycode 44
Countrycode 46
Countrycode 49
Comment:
We can note that it have pattern, as each day it start with low numbers rows then increase in the middle day, then decrease again at the end of the day and the start of the new day.

Code	Country / Region	Code	Country / Region	Code	Country / Region	Code	Country / Region	Code	Country / Region	Code	Country / Region	Code	Country / Region	Code	Country / Region	Code	Country / Region
0	Unknown / Local	39	Italy	90	Turkey	220	Gambia	370	Lithuania	500	Falkland Islands	590	Guadeloupe (France)	7700	Special code	1214	Canada	18686	Special code
1	USA, Canada (NANP)	40	Romania	91	India	221	Senegal	371	Latvia	501	Belize	591	Bolivia	7701	Special code	1246	Barbados	18762	Special code
7	Russia, Kazakhstan	41	Switzerland	92	Pakistan	222	Mauritania	372	Estonia	502	Guatemala	592	Guyana	7702	Special code	1250	Canada	18763	Special code
20	Egypt	43	Austria	93	Afghanistan	223	Mali	373	Moldova	503	El Salvador	593	Ecuador	7705	Special code	1264	Anguilla	1926	Special code
27	South Africa	44	United Kingdom	94	Sri Lanka	224	Guinea	374	Armenia	504	Honduras	594	French Guiana	7707	Special code	1284	Antigua and Barbuda	1938	Special code
30	Greece	45	Denmark	95	Myanmar (Burma)	225	Ivory Coast	375	Belarus	505	Nicaragua	595	Paraguay	7711	Special code	1289	Special code	1705	Special code
31	Netherlands	46	Sweden	98	Iran	226	Burkina Faso	376	Andorra	506	Costa Rica	596	Martinique (France)	7712	Special code	1306	Special code	1709	Special code
32	Belgium	47	Norway	212	Morocco	227	Niger	377	Monaco	507	Panama	597	Suriname	7713	Special code	1340	U.S. Virgin Islands	1721	Special code
33	France	48	Poland	213	Algeria	228	Togo	378	San Marino	508	Saint Pierre and Miquelon	598	Uruguay	7722	Special code	1438	Bermuda	1758	Special code
34	Spain	49	Germany	216	Tunisia	229	Benin	379	Vatican City (rare)	509	Haiti	599	Caribbean Netherlands	7725	Special code	1441	Bermuda	1778	Special code
35	(No longer used)	51	Peru	218	Libya	230	Mauritius	380	Ukraine	510	Special code	7700	Special code	7727	Special code	1787	Special code	264	Namibia
36	Hungary	52	Mexico	219	(Not assigned)	231	Liberia	381	Serbia	700	Special code	7715	Special code	7731	Special code	1799	Special code	265	Malawi
37	(No longer used)	53	Cuba	220	Gambia	232	Sierra Leone	382	Montenegro	800	Toll-free numbers	7732	Special code	7741	Special code	1800	Special code	266	Lesotho
38	(Reserved)	54	Argentina	221	Senegal	233	Ghana	385	Croatia	801	Toll-free numbers	7742	Special code	7751	Special code	1809	Special code	267	Botswana
39	Italy	55	Brazil	222	Mauritania	234	Nigeria	386	Slovenia	802	Toll-free numbers	7755	Special code	7761	Special code	1819	Special code	268	Eswatini (Swaziland)
40	Romania	56	Chile	223	Mali	235	Chad	387	Bosnia and Herzegovina	803	Toll-free numbers	7775	Special code	7778	Special code	1829	Special code	269	Comoros
41	Switzerland	57	Colombia	224	Guinea	236	Central African Republic	389	North Macedonia	804	Toll-free numbers	7788	Special code	7799	Special code	1849	Special code	290	Saint Helena
42	(Not assigned)	58	Venezuela	225	Ivory Coast	237	Cameroon	420	Czech Republic	805	Toll-free	7800	Special code	7820	Special code	1850	Special code	291	Eritrea

These "600+" codes?

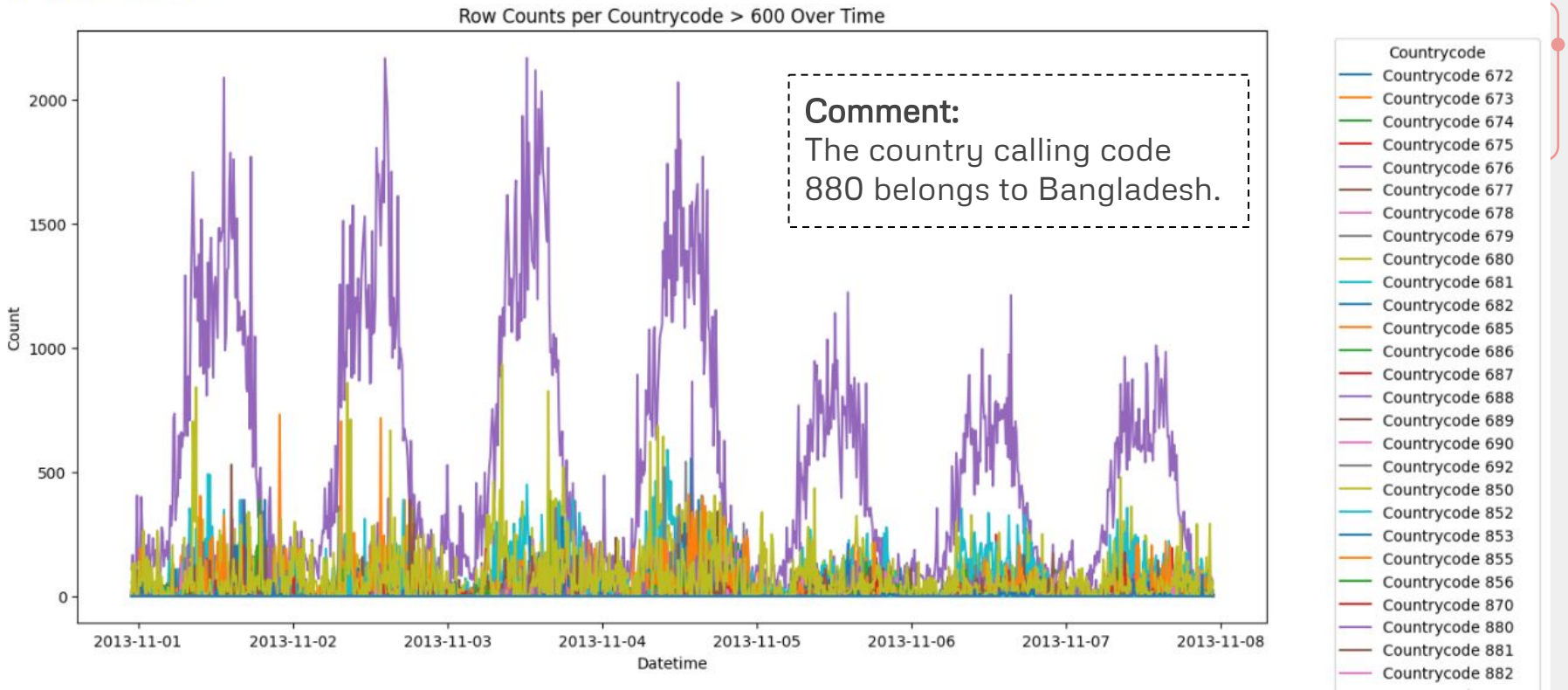
These codes do not represent individual countries but rather **special telecommunication services or regions.** They are part of the ITU-T E.164 numbering plan, reserved for services rather than geographic nations.

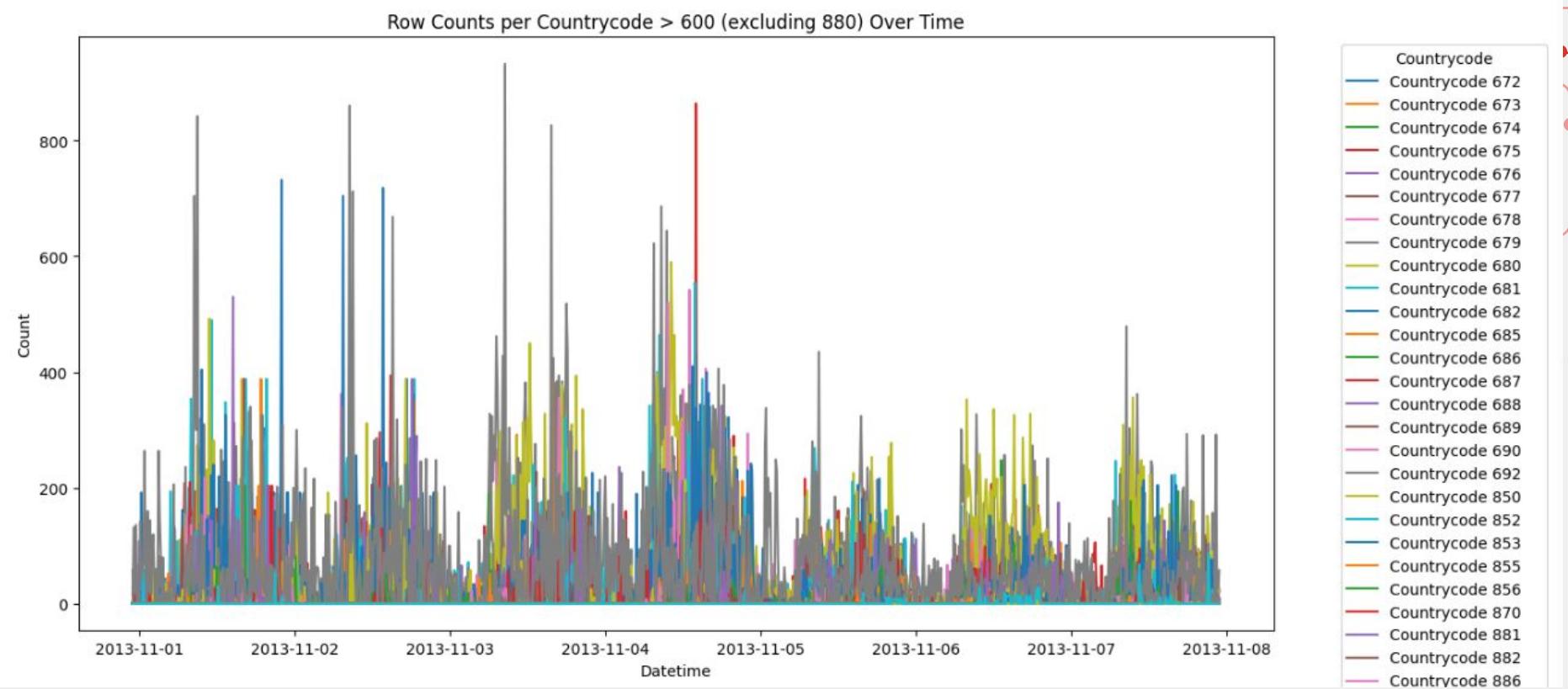
Code	Description
870	Inmarsat (satellite phones)
878	Universal Personal Telecommunications (UPT) service
881	Global Mobile Satellite System (GMSS) services
882	International Networks
883	International Networks

Satellite Networks

- Codes like **870**, **871**, **872**, **873**, **874**, **875**, and **876** are assigned to **satellite telephone services**.
    - For example, **870** is for **Inmarsat**, a global satellite communications provider, used for ships, aircraft, and remote locations.
    - Other satellite codes serve specific satellite operators or constellations.
- These are not tied to a geographic country but to mobile or fixed satellite networks.

```
Countrycode with highest total rows > 600: 880 (Count: 554878)
<ipython-input-13-697cf457e1f7>:26: UserWarning: Tight layout not applied. The bottom and top margins cannot be made large enough to accommodate all
  plt.tight_layout()
```



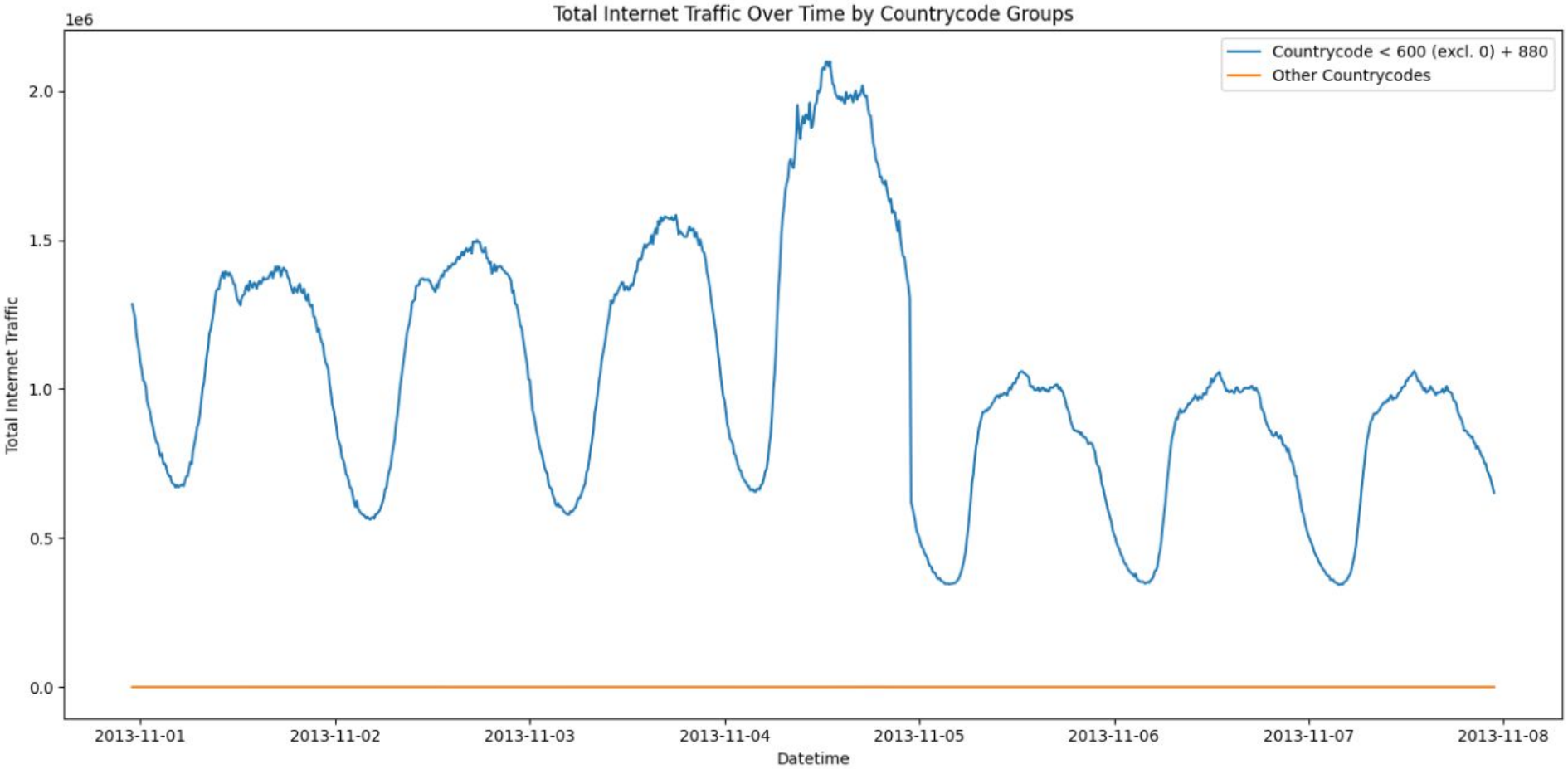

Row Counts per Countrycode > 600 (excluding 880) Over Time
Count
0
200
400
600
800
2013-11-01
2013-11-02
2013-11-03
2013-11-04
2013-11-05
2013-11-06
2013-11-07
2013-11-08
Datetime
Countrycode
Countrycode 672
Countrycode 673
Countrycode 674
Countrycode 675
Countrycode 676
Countrycode 677
Countrycode 678
Countrycode 679
Countrycode 680
Countrycode 681
Countrycode 682
Countrycode 685
Countrycode 686
Countrycode 687
Countrycode 688
Countrycode 689
Countrycode 690
Countrycode 692
Countrycode 850
Countrycode 852
Countrycode 853
Countrycode 855
Countrycode 856
Countrycode 870
Countrycode 881
Countrycode 882
Countrycode 886

What to do?!

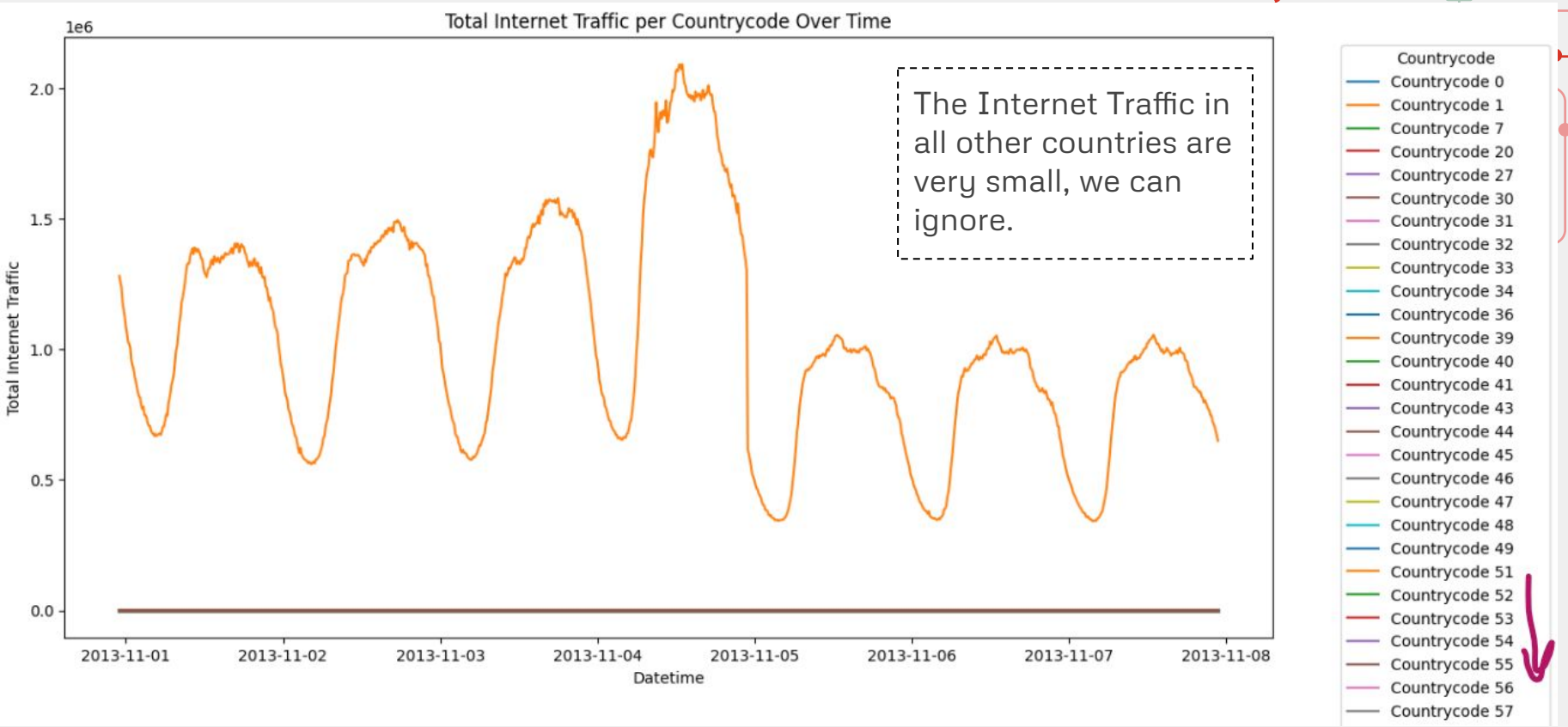
Non-representative traffic patterns:
Traffic from satellite phones or toll-free lines may have very different usage patterns – bursts, irregular spikes, or very low/high volumes that distort trends.

Since these codes don't correspond to Italian users or normal country-specific telecom traffic, their inclusion could skew your model with data that does not reflect real user behavior in Italy.

Take the IDs less than 600



Take the IDs less than 600



Take only 39 ID

After cleaning the data:

1. Drop any country ID other 39
2. Drop unneeded columns
3. Filter only my 10*10 cells decided in previous phase

First 5 rows:

	CellID	datetime	internet
0	4546	2013-11-01 00:00:00	251.868143
1	4546	2013-11-01 00:10:00	275.656908
2	4546	2013-11-01 00:20:00	263.815989
3	4546	2013-11-01 00:30:00	284.824317
4	4546	2013-11-01 00:40:00	337.275423

I found that there is no any nun or missing values

So No need to handle missing data

	Null Count	Null %
CellID	0	0.0
datetime	0	0.0
internet	0	0.0

```
                     CellID    internet
datetime
2013-11-01 00:00:00    4546  503.736286
2013-11-01 00:10:00    4546  551.313816
2013-11-01 00:20:00    4546  527.631979
2013-11-01 00:30:00    4546  569.648635
2013-11-01 00:40:00    4546  674.550847
New shape: (820200, 2)
```

```
[30] print(df_grouped.tail(10))
```

```
                     CellID    internet
datetime
2013-12-27 21:20:00    5455  81.046695
2013-12-27 21:30:00    5455  63.275906
2013-12-27 21:40:00    5455  66.324451
2013-12-27 21:50:00    5455  64.688647
2013-12-27 22:00:00    5455  76.709334
2013-12-27 22:10:00    5455  88.085613
2013-12-27 22:20:00    5455  68.009826
2013-12-27 22:30:00    5455  73.264230
2013-12-27 22:40:00    5455  69.719986
2013-12-27 22:50:00    5455  69.075578
```

As we can note that my final dataset contain (820200, 2)

each day: 144 time stamp

last day in my data was 27-12-2013 at 22:50

so the total time stamps we have should be

144*(30[mounth'11]+27[mounth'12])= 8208 - 6

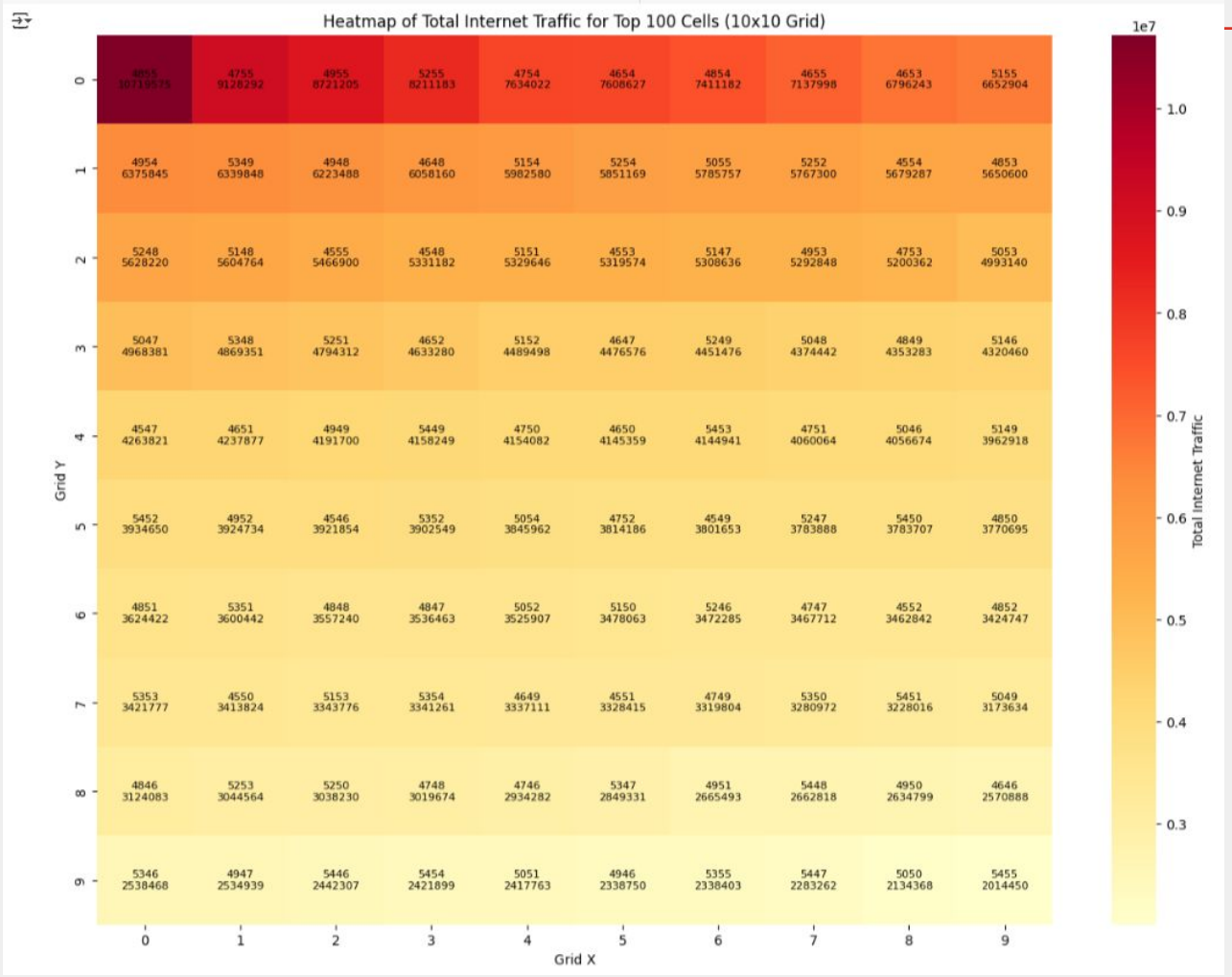
8202 * 100 [cells] = **820200**

Start with
01-11-2013 00:00:00

End with
17-12-2013 22:50:00

(820200,3)


Choose Cell!


Heatmap of Total Internet Traffic for Top 100 Cells (10x10 Grid)
Grid Y
Grid X
Total Internet Traffic
1e7
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
4855 10719575
4755 9128292
4955 8721205
5255 8211183
4754 7634022
4654 7608627
4854 7411182
4655 7137998
4653 6796243
5155 6652904
4954 6375845
5349 6339848
4948 6223488
4648 6058160
5154 5982580
5254 5851169
5055 5785757
5252 5767300
4554 5679287
4853 5650600
5248 5628220
5148 5604764
4555 5466900
4548 5331182
5151 5329646
4553 5319574
5147 5308636
4953 5292848
4753 5200362
5053 4993140
5047 4968381
5348 4869351
5251 4794312
4652 4633280
5152 4489498
4647 4476576
5249 4451476
5048 4374442
4849 4353283
5146 4320460
4547 4263821
4651 4237877
4949 4191700
5449 4158249
4750 4154082
4650 4145359
5453 4144941
4751 4060064
5046 4056674
5149 3962918
5452 3934650
4952 3924734
4546 3921854
5352 3902549
5054 3845962
4752 3814186
4549 3801653
5247 3783888
5450 3783707
4850 3770695
4851 3624422
5351 3600442
4848 3557240
4847 3536463
5052 3525907
5150 3478063
5246 3472285
4747 3467712
4552 3462842
4852 3424747
5353 3421777
4550 3413824
5153 3343776
5354 3341261
4649 3337111
4551 3328415
4749 3319804
5350 3280972
5451 3228016
5049 3173634
4846 3124083
5253 3044564
5250 3038230
4748 3019674
4746 2934282
5347 2849331
4951 2665493
5448 2662818
4950 2634799
4646 2570888
5346 2538468
4947 2534939
5446 2442307
5454 2421899
5051 2417763
4946 2338750
5355 2338403
5447 2283262
5050 2134368
5455 2014450

Criteria:

1. Choose a cell with **high total internet traffic** to ensure enough data points and signal strength for modeling.
2. stable patterns improve forecast accuracy.
3. Prefer a cell with **minimal missing data or nulls** in the `internet` column.
4. Look for cells with interesting **daily/weekly traffic patterns** (clear peaks and troughs) — valuable for testing forecasting methods.
5. Avoid cells with extreme outliers or irregular spikes
6. plan specific anomaly analysis.

Outlier detection: using **seasonal** decomposition and **z-score**

STL (Seasonal-Trend decomposition using Loess) for detecting **seasonality** strength

CellID	TotalTraffic	NullPct	CoefVar	OutlierCount	Seasonality
4855	1.071957e+07	0.0	0.598233	247	0.737028
4755	9.128292e+06	0.0	0.513722	214	0.766600
4955	8.721205e+06	0.0	0.916392	265	0.702517
5255	8.211183e+06	0.0	0.684926	230	0.742158
4754	7.634022e+06	0.0	0.508736	232	0.756138
4654	7.608627e+06	0.0	0.537992	243	0.800738
4854	7.411182e+06	0.0	0.528924	209	0.701841
4655	7.137998e+06	0.0	0.509919	253	0.819756
4653	6.796243e+06	0.0	0.571995	255	0.790454
5155	6.652904e+06	0.0	0.752899	263	0.702996

1. CoefVar (Stability / Variability) Range from ~0.5 to 0.92.

Lower values (~0.5) mean stable traffic patterns; easier to forecast.

Higher values (~0.9) imply more volatility or noise, which may complicate modeling but might be interesting for anomaly detection.

2. OutlierCount (Irregular Spikes) Wide range outliers detected.

Cells with low outlier counts suggest smoother traffic, better for classic forecasting.

Higher outliers might indicate anomalies, bursts, or noisy data.

3. Seasonality (Daily Pattern Strength) Values ~0.76 to 0.88 indicate strong daily/weekly cycles.

Strong seasonality supports time series models that leverage periodic patterns (ARIMA with seasonality, SARIMA, LSTM with cyclical inputs).

```python
# New outlier detection function using seasonal decomposition and z-score
def count_outliers_using_stl(x, period=144):
    try:
        # Drop missing values
        #x = x.dropna()

        if len(x) < period * 2:  # Ensure enough data for decomposition
            return np.nan

        # Apply seasonal decomposition (STL)
        stl = STL(x, period=period, robust=True)
        result = stl.fit()

        # Extract the residual component
        residuals = result.resid

        # Apply z-score to the residuals
        z_scores = zscore(residuals)

        # Identify outliers as those with z-score greater than 3 or less than -3
        outliers = np.sum(np.abs(z_scores) > 3)  # You can adjust the threshold if needed

        return outliers
    except Exception as e:
        return np.nan

# Apply the outlier detection with STL and z-score
outlier_counts = grouped['internet'].apply(count_outliers_using_stl)
```

```python
# Define STL-based seasonality strength function
def stl_seasonality_strength(x):
    try:
        #x = x.dropna()
        if len(x) < 7 * 144:  # at least two full daily cycles for stability
            return np.nan
        stl = STL(x, period=144, robust=True)
        res = stl.fit()
        seasonal_var = np.var(res.seasonal)
        resid_var = np.var(res.resid)
        # Seasonality strength ratio: higher means stronger seasonality
        return seasonal_var / (seasonal_var + resid_var)
    except Exception as e:
        return np.nan

# Replace previous seasonality with STL-based seasonality strength
seasonality = grouped['internet'].apply(stl_seasonality_strength)
```

```python
# Assign weights to each metric based on importance (you can adjust these)
weights = {
    'TotalTraffic': 0.3,
    'NullPct': 0.0,
    'CoefVar': 0.25,
    'OutlierCount': 0.2,
    'Seasonality': 0.25
}

# Calculate weighted score
metrics_norm['Score'] = (
    metrics_norm['TotalTraffic'] * weights['TotalTraffic'] +
    metrics_norm['NullPct'] * weights['NullPct'] +
    metrics_norm['CoefVar'] * weights['CoefVar'] +
    metrics_norm['OutlierCount'] * weights['OutlierCount'] +
    metrics_norm['Seasonality'] * weights['Seasonality']
)

# Rank cells by descending score
metrics_norm = metrics_norm.sort_values(by='Score', ascending=False)

print(metrics_norm[['Score']].head(10))  # Top 10 best cells
```

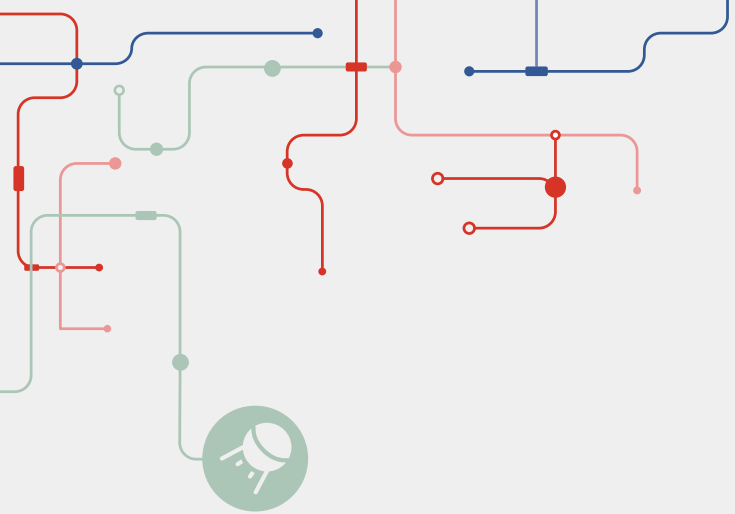
```
            Score
CellID
4755     0.760353
4855     0.704367
4648     0.681833
4754     0.679354
4654     0.676411
4655     0.673469
5151     0.660493
4555     0.660077
4854     0.657946
5349     0.652281
```

Order The Metrics:

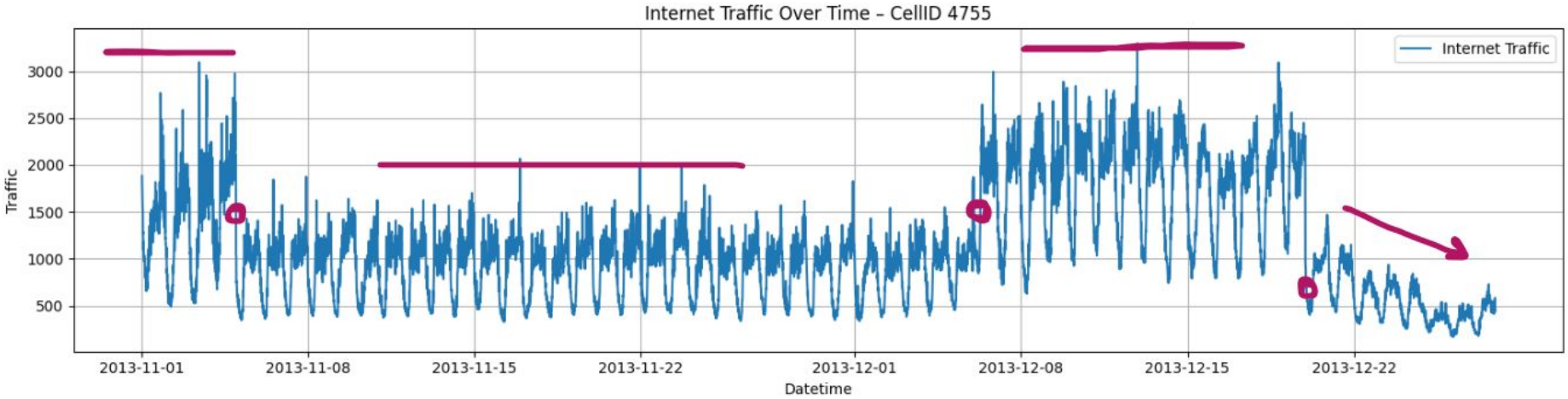
1. **TotalTraffic** : Shows actual usage – high traffic = high activity = valuable region.
2. **Seasonality** : Shows how predictable and structured the cell's usage is.
3. **CoefVar** : Measures consistency: low variation means stable performance.
4. **OutlierCount** : Measures how often behavior breaks from the expected pattern.
5. **NullPct** : have no nulls.

Work only on Cell ID **4755**

```
                        internet
datetime
2013-11-01 00:00:00  1882.835224
2013-11-01 00:10:00  1792.001814
2013-11-01 00:20:00  1721.562928
2013-11-01 00:30:00  1608.051995
2013-11-01 00:40:00  1533.467322
(8202, 1)
```



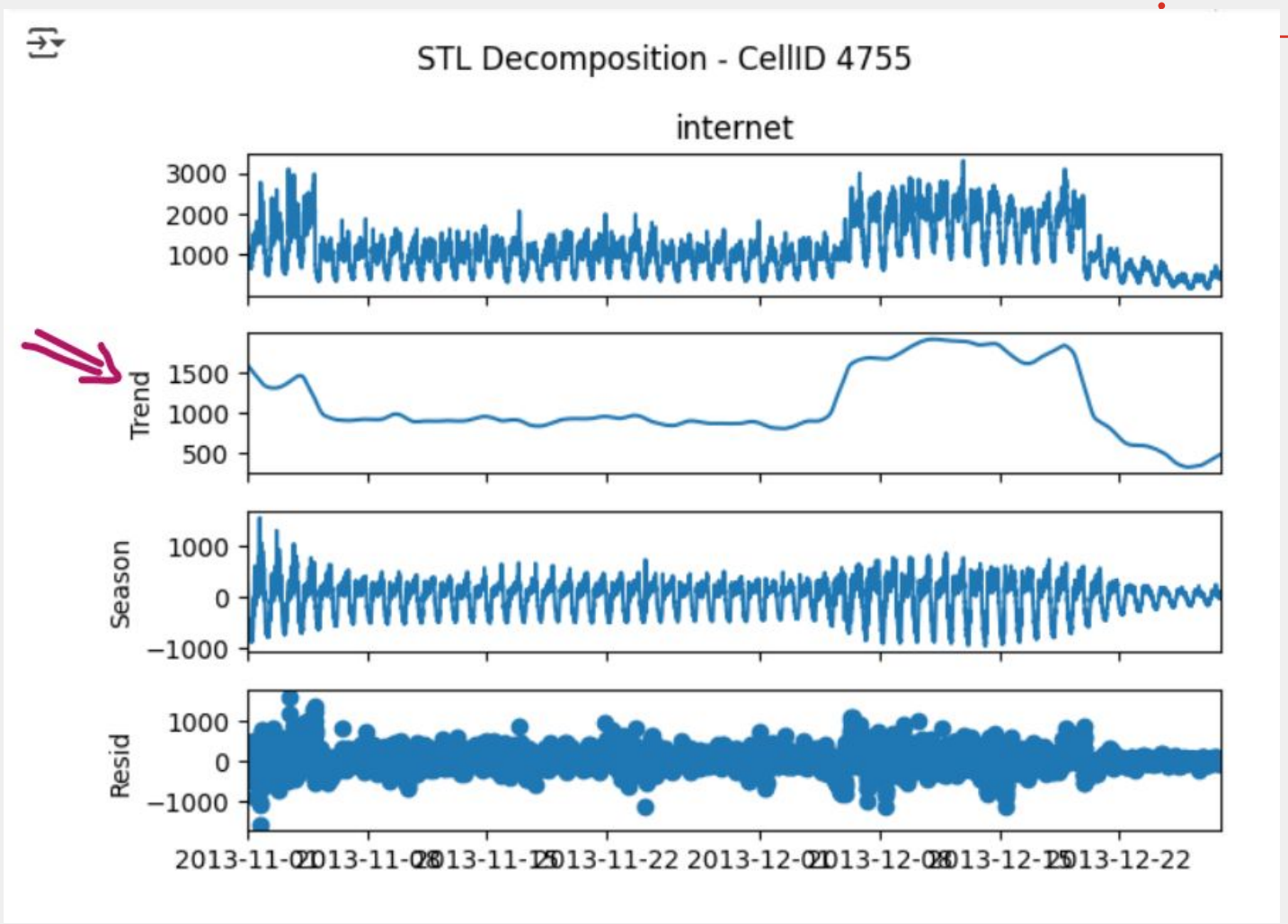

RCA
Root Cause Analysis!



Before going to handle the **outliers**
We can note that we have **"Abrupt Changes"** in my data

"An **abrupt change** is a sudden, sharp deviation in the behavior of a time series that violates the expected pattern of the data."

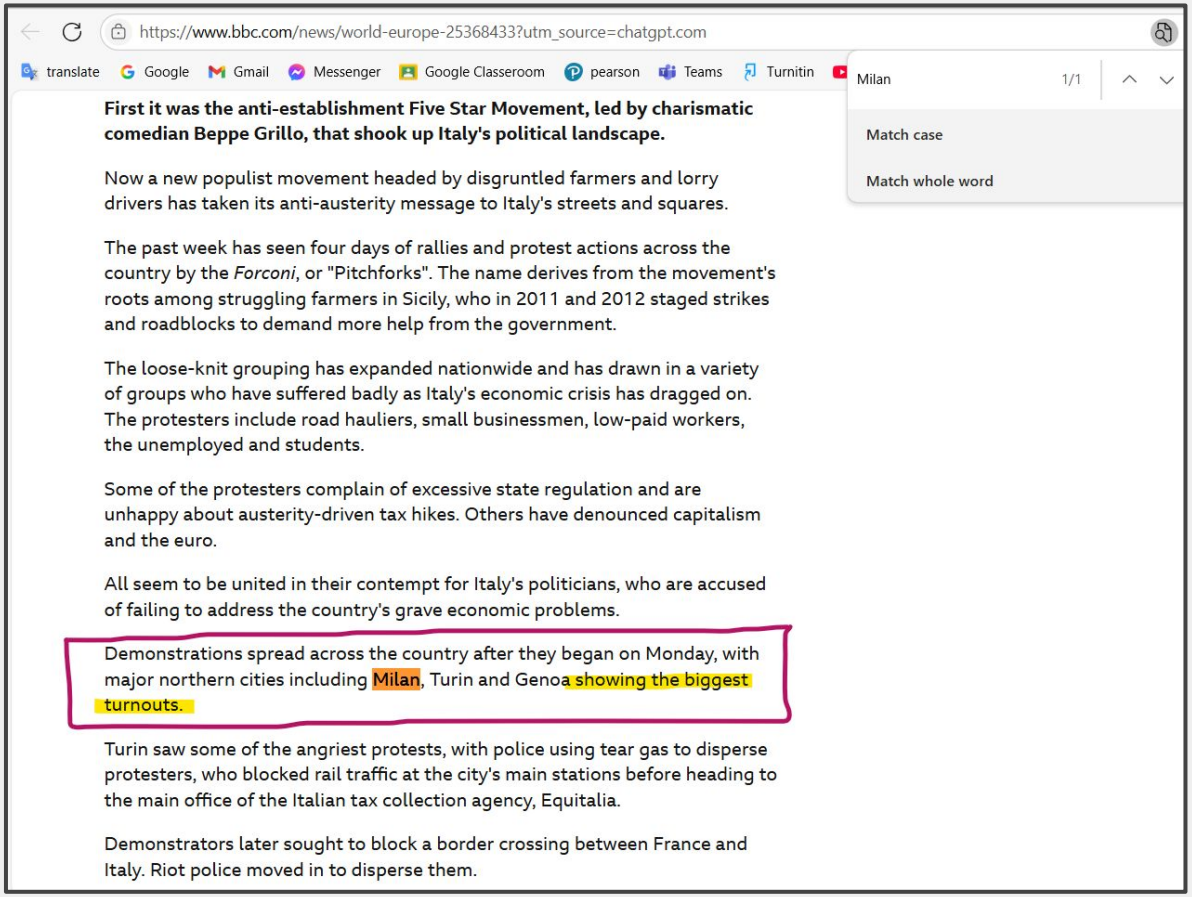
What Happen in this duration [from 6-12-2013 till 21-12-2013]?


STL Decomposition - CellID 4755
internet
3000
2000
1000
Trend
1500
1000
500
Season
1000
0
−1000
Resid
1000
0
−1000
2013-11-01
2013-11-08
2013-11-15
2013-11-22
2013-12-01
2013-12-08
2013-12-15
2013-12-22

What Happen in this duration [from 6-12-2013 till 21-12-2013]?

The **Pitchfork protests** began on **December 9, 2013** , and continued for several days, with varying intensity across different regions. In some areas, the protests lasted **until December 18, 2013** . The movement was characterized by its decentralized nature, with different groups organizing actions in their respective regions.

In **Milan** , protesters participated in **nationwide strikes** and demonstrations organized by the **Pitchfork movement** . The protests in Milan were part of a larger series of actions that included roadblocks, sit-ins, and other forms of civil disobedience in major northern cities such as Turin and Genoa. **These actions were aimed at expressing dissatisfaction with the government's austerity measures, high taxation, and economic policies.**



https://www.bbc.com/news/world-europe-25368433?utm_source=chatgpt.com

https://www.bbc.com/news/world-europe-25368433?utm_source=chatgpt.com

translate Google

First it was th
comedian Be

Now a new po
drivers has ta

The past wee
country by th
roots among
and roadbloc

The loose-kni
of groups wh
The protester
the unemploy

Some of the p
unhappy abou
and the euro.

All seem to b
of failing to a

Demonstratio
major northe
turnouts.

Turin saw son
protesters, wh
the main offic

Demonstrato
Italy. Riot pol

General News | Politics | Business | Science&Technology | LifeStyle + | Sport | Vatican | World | Photo | Other +

TRENDING malaria case • growth • Pope Francis • Colombia • dioxins alarm • vaccines

ANSA.it › English › Latest News › Anti-austerity 'Pitchfork' protests spread

Anti-austerity 'Pitchfork' protests spread

Grillo tells police 'don't protect politicians'

Redazione ANSA
ROME
10 December 2013
20:14
NEWS

Suggerisci
Facebook
Twitter
Altri
A+ A A-
Stampa
Scrivi alla redazione

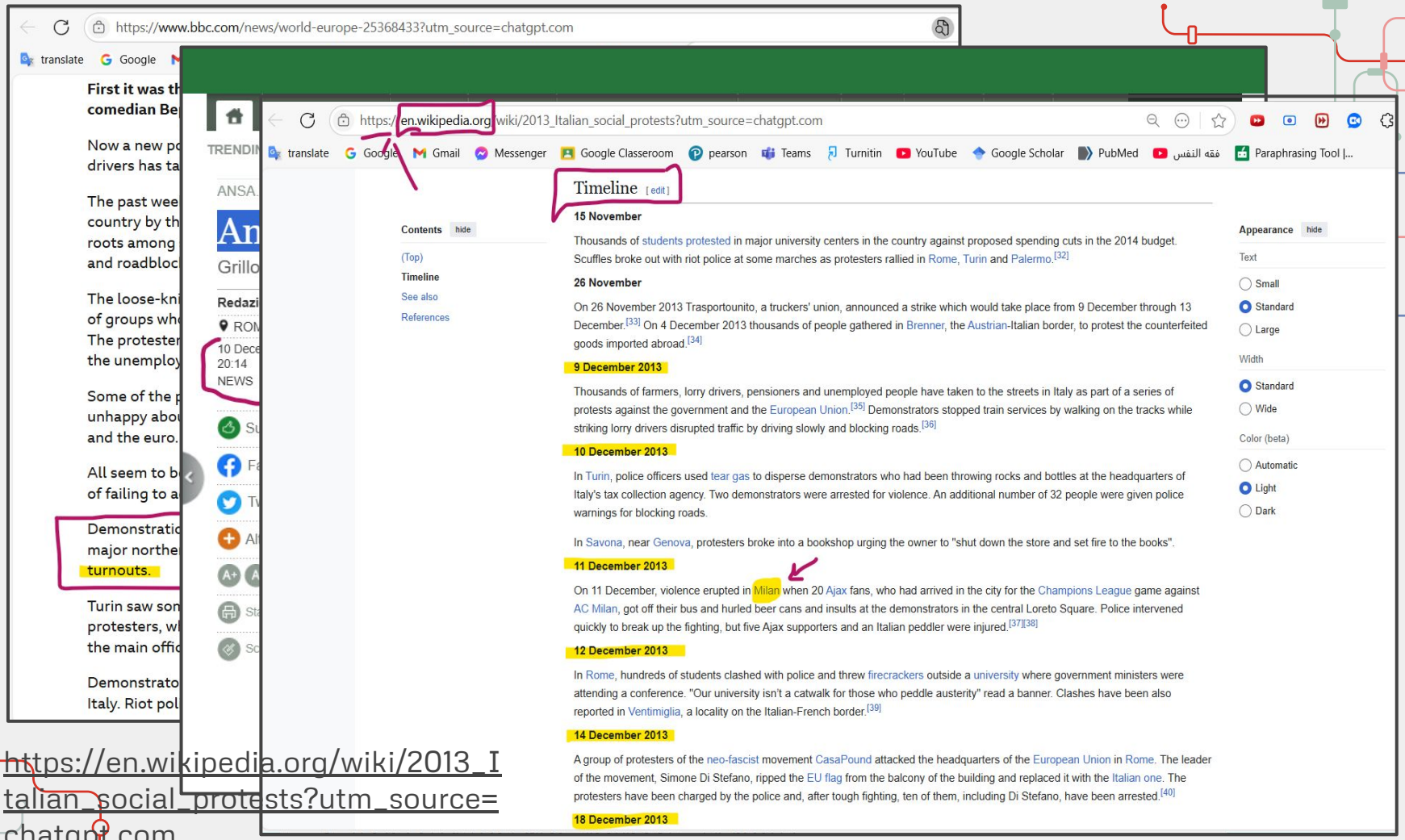
- RIPRODUZIONE RISERVATA CLICK TO ENLARGE

(By Christopher Livesay) (ANSA) - Rome, December 10 - Thousands of protesters joined truckers in Italy on Tuesday as they entered a second day of strikes as part of the so-called Pitchfork (Forconi) Movement, blocking or slowing transportation in a number of cities in Italy. The nationwide wildcat strike protesting heavy taxes,

LATEST NEWS

14:23 Court OKs gay's adoption of child born via surrogacy
13:32 Generali posts Q1 profit of 1.2 bn
13:16 Milan shop puts up sign saying Israelis not welcome
12:19 Security boosted at Rome ghetto after Washington attack
12:06 Not recognizing intending mother unconstitutional -top court
10:35 PNA working to arrange meeting with pope
10:04 I stand with Israel after Washington attack - Tajani

https://www.ansa.it/english/news/2013/12/10/Anti-austerity-Pitchfork-protests-spread_20ce1ed3-2180-4ecd-8fdd-e8c2e2774e27.html?utm_source=chatgpt.com



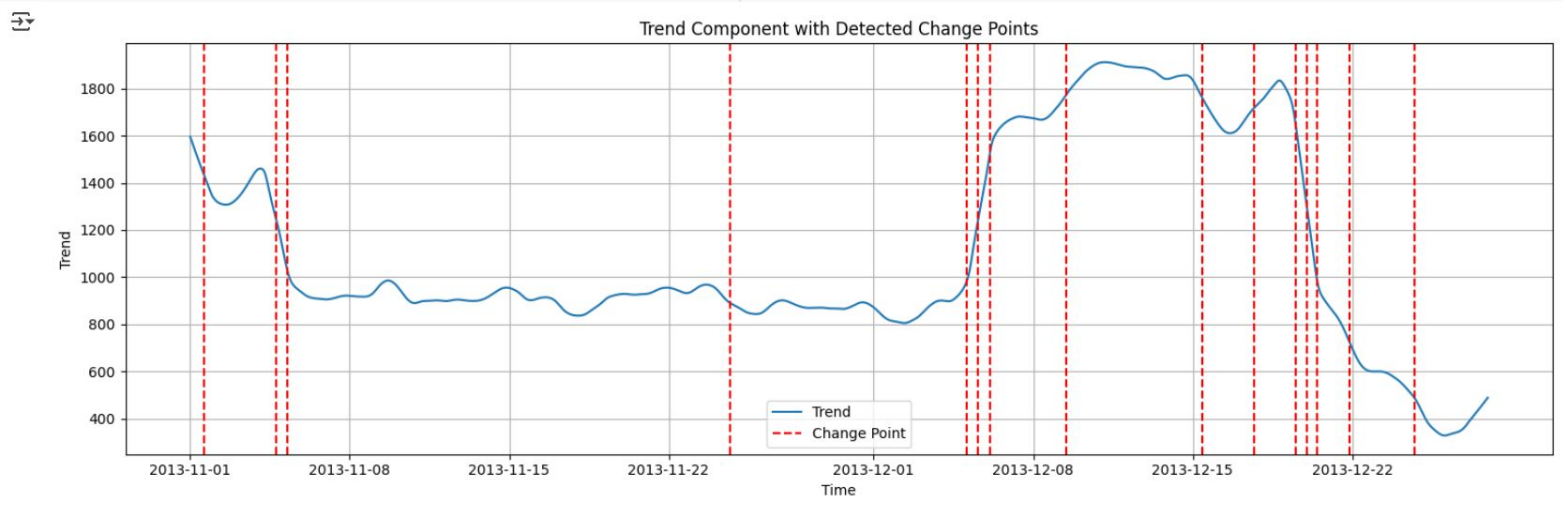
https://en.wikipedia.org/wiki/2013_Italian_social_protests?utm_source=chatgpt.com

🧠 Goal:
Detect points in a time series where the statistical properties (e.g., mean, variance, distribution) change – these are called change points.

✅ Uses the **PELT** (Pruned Exact Linear Time) algorithm to find **change points** :

- model="**rbf**" tells it to use the **RBF** (**Radial Basis Function**) kernel, which is useful when the changes are **non-linear** (not just mean/variance).

```
Detected change points at indices: [85, 540, 615, 3410, 4910, 4980, 5055, 5535, 6395, 6725, 6990, 7055, 7120, 7330, 7735, 8202]
```



How i will deal with the Differencing in the Trend?

1. **Flag** and model separately
- Create a binary flag feature marking that period as “special” or “anomalous”
- Train model to recognize and handle this condition differently

2. **Segment** your modeling
- Build separate models for “normal” vs “unusual” periods (regime-switching)
- Combine predictions based on context or external indicators

3. Data transformation
- Apply **smoothing** , winsorizing, or capping to reduce extreme effects but keep information
- Use log transforms to reduce range skewness

Event is a one-time anomaly

Bagging = Bootstrap Aggregating.
It's an ensemble technique where you:

- Create **multiple training subsets** by sampling (with replacement) from your dataset.
- Train a **model on each subset** .
- Aggregate their predictions (e.g., averaging for regression, voting for classification).
- **Goal**: **Reduce variance** and improve generalization.

Anomaly Detection

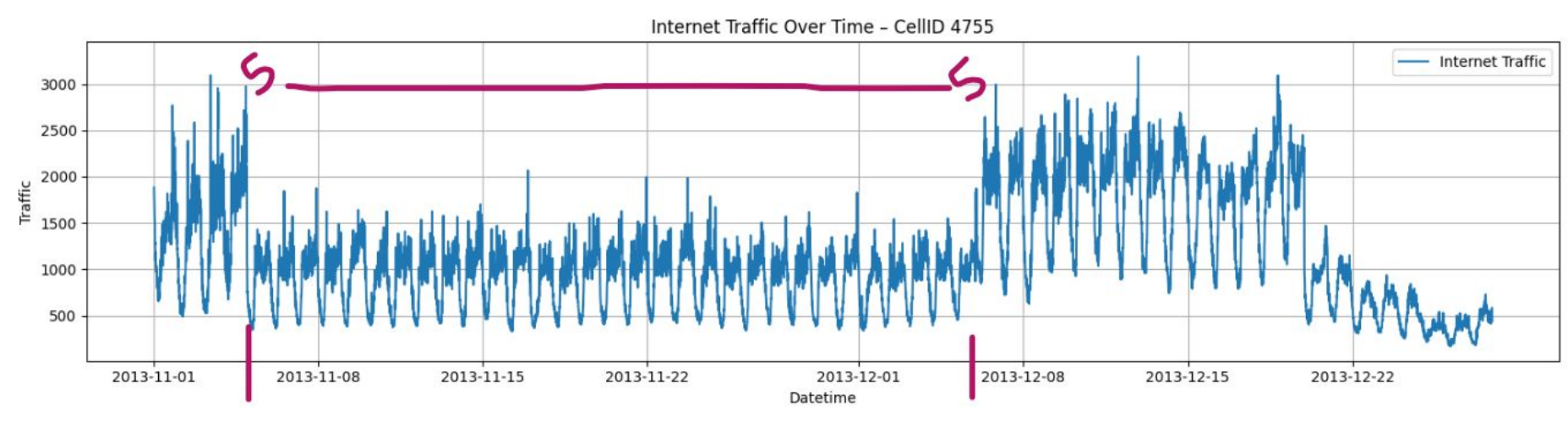
1. Contextual anomalies
   - Data points anomalous in a specific context or timeframe
   - High traffic during unusual hours

2. **Collective anomalies**
   - A sequence or group of points that **together** are anomalous
   - **Traffic pattern changes over several days**

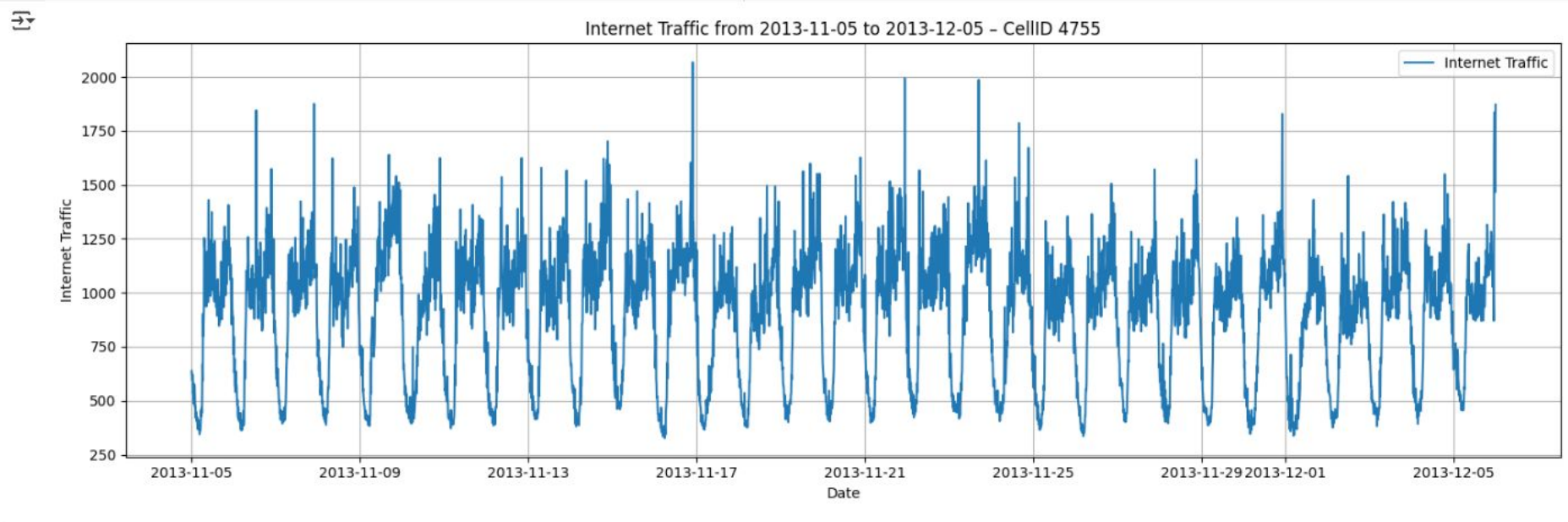
Anomaly Severity

- Measuring how serious or impactful anomalies are

🔪 Segment the Time Series

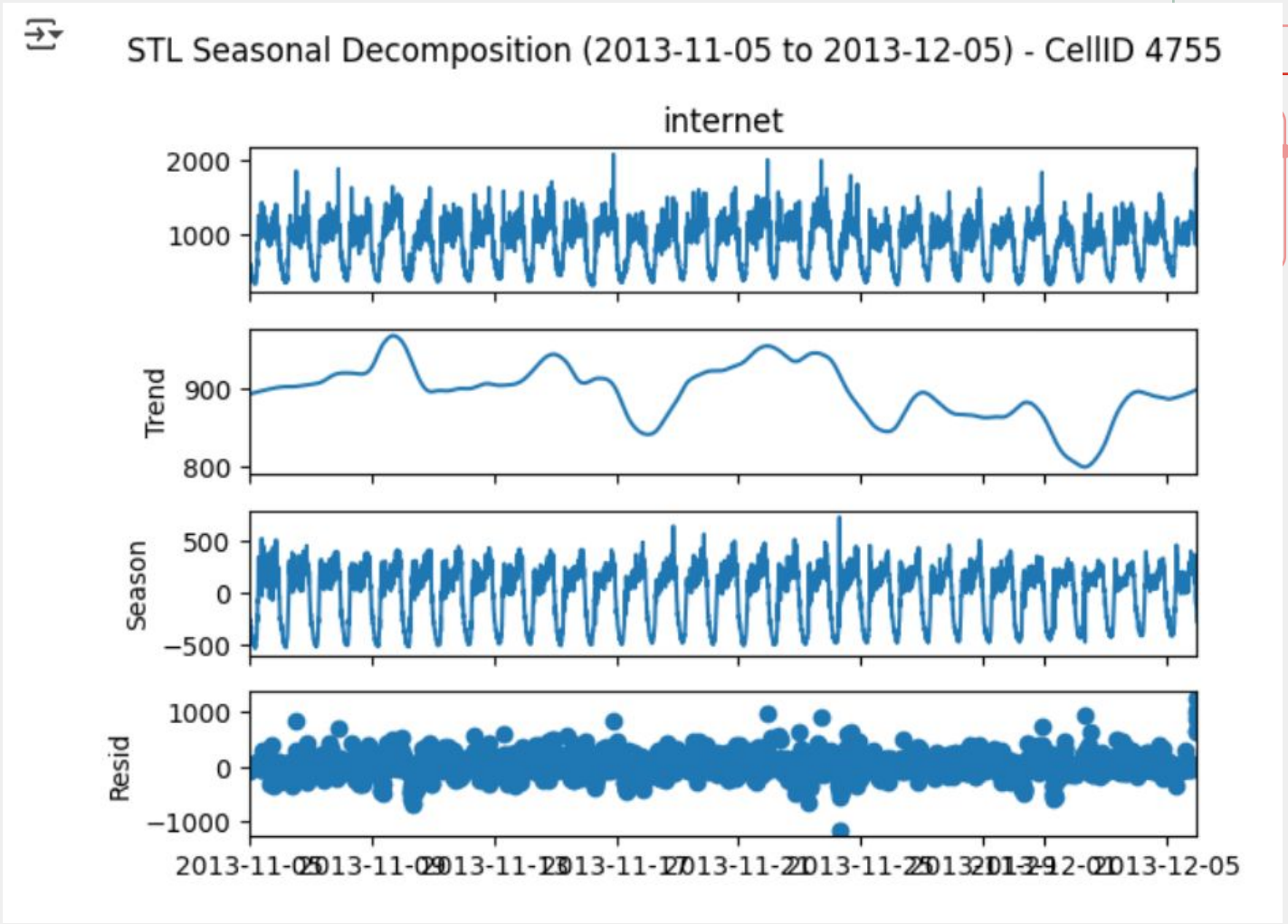


🔪 Segment the Time Series

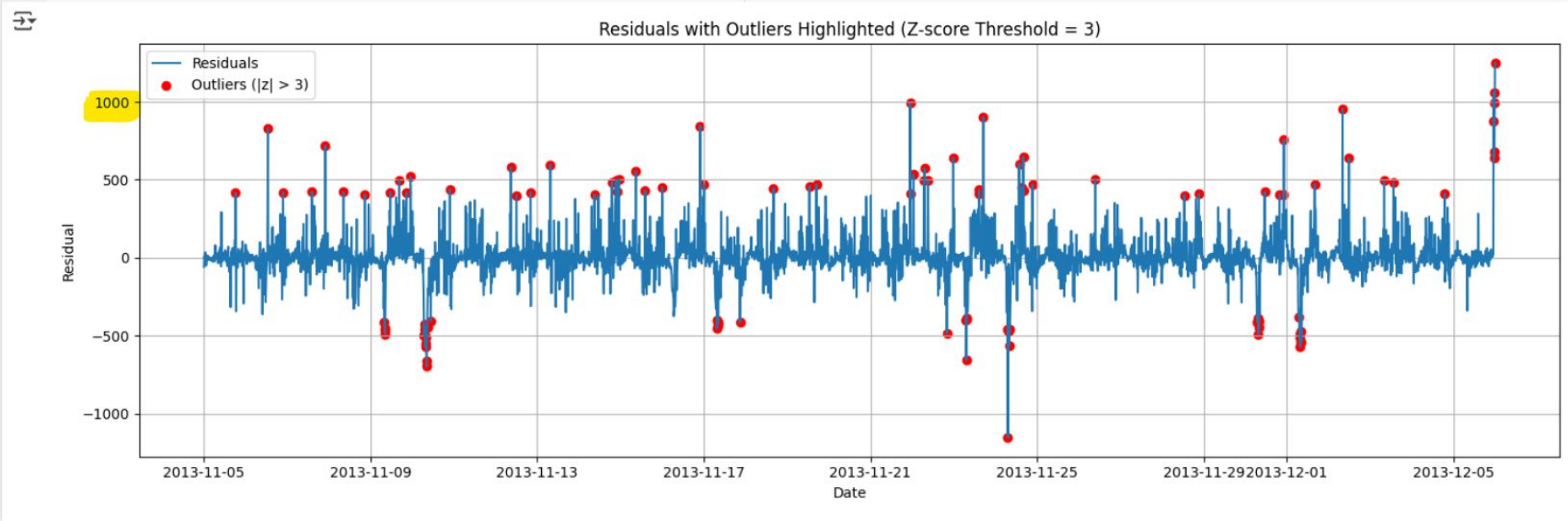


Handle Outliers

Apply STL Seasonal Decompose after cut sample from the dataset



Apply Z-score on the Residuals to detect the outliers

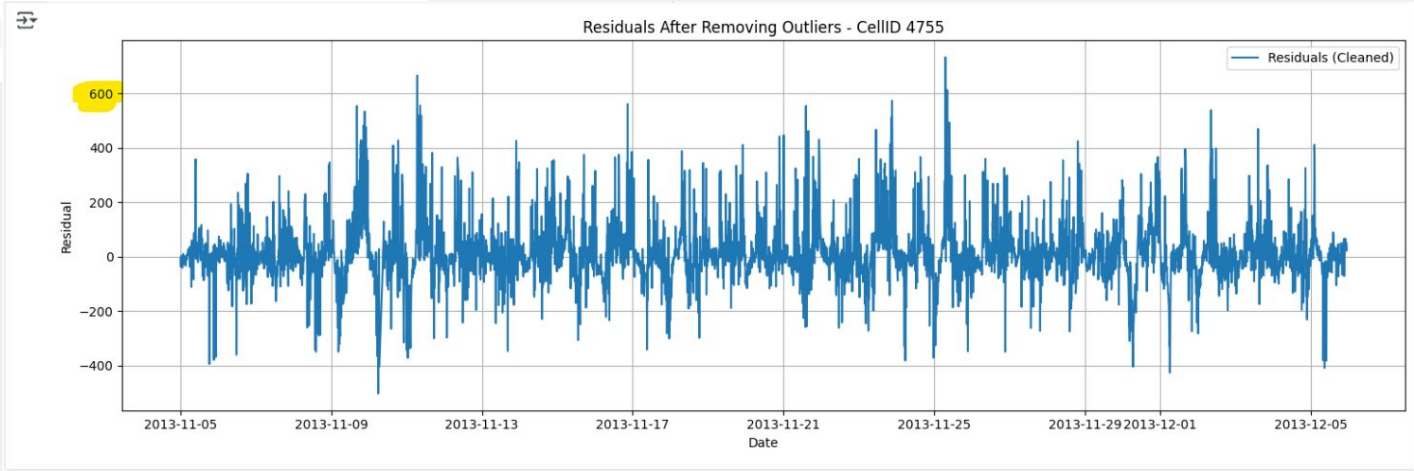


```
Number of outliers detected: 110
```

datetime	Residual	Z_score
2013-11-05 18:10:00	415.165198	3.117215
2013-11-06 12:50:00	828.808623	6.306942
2013-11-06 21:30:00	420.802654	3.160687
2013-11-07 14:10:00	423.878022	3.184402
2013-11-07 21:50:00	718.281116	5.454632
...	...	...
2013-12-05 23:10:00	1058.476187	8.077977
2013-12-05 23:20:00	639.930340	4.850446
2013-12-05 23:30:00	681.493664	5.170953
2013-12-05 23:40:00	995.110420	7.589345
2013-12-05 23:50:00	1251.184072	9.564004

[110 rows x 2 columns]

```
Original data shape: (4464, 1)
Cleaned data shape: (4354, 1)
```



Stationarity check (ADF Test)

Stationarity check (ADF Test)

```
ADF Statistic: -12.156229187388181
p-value: 1.5298503292283665e-22
```

$p < 0.05 \rightarrow$ Stationary (good) ✓

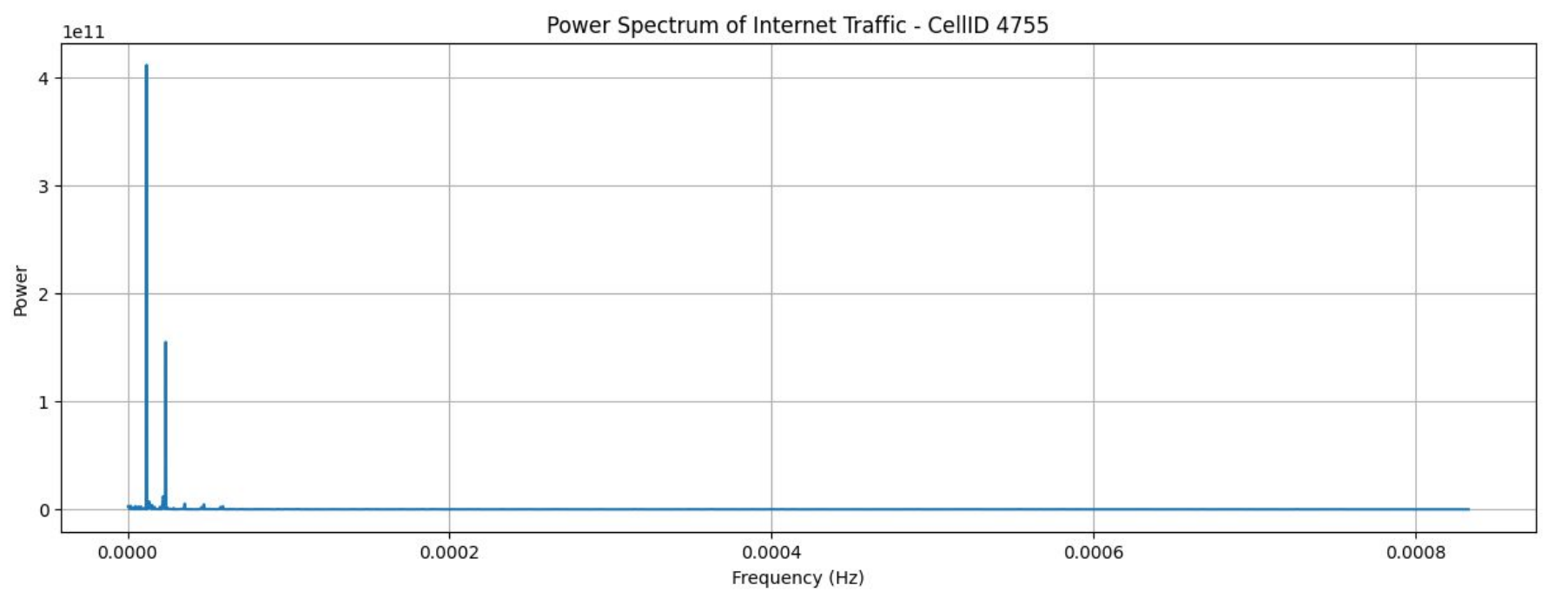
$p \geq 0.05 \rightarrow$ Non-stationary $\rightarrow$ will need differencing

Stationary = A time series whose mean, variance, and autocorrelation structure do not change over time.

Seasonality check (FFT)

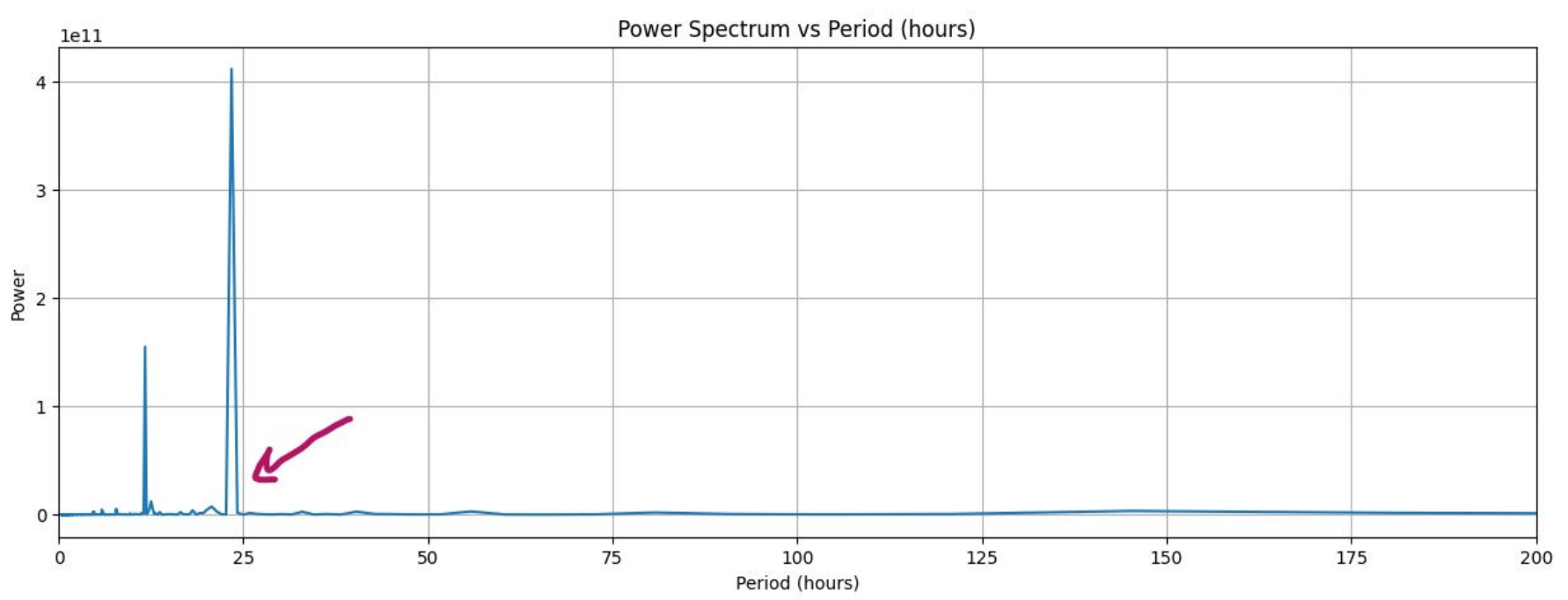
Using the **Fourier Transform (FT)** for detecting **seasonality**

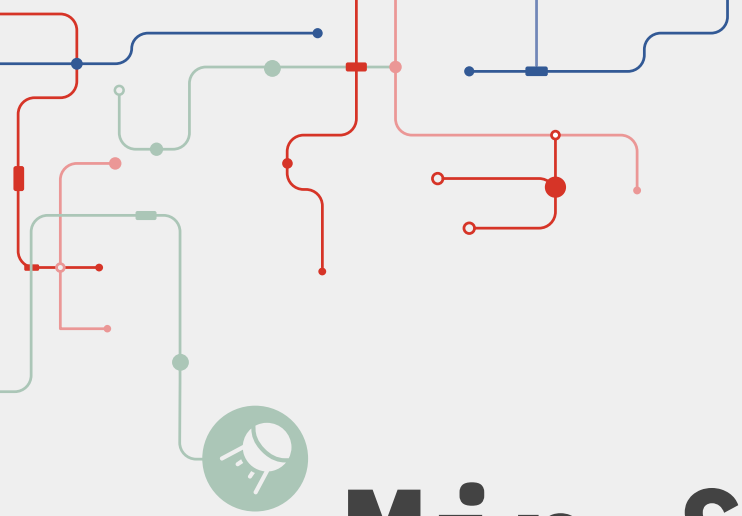
In the frequency domain, these **cycles appear** as **peaks** at specific frequencies.



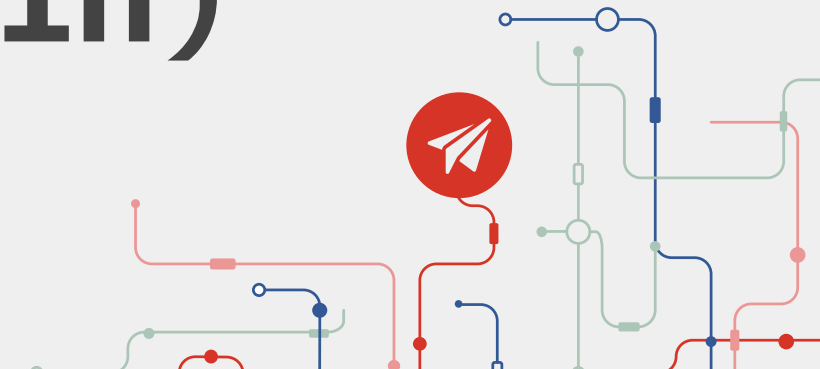
Peaks near **24 hours** (~1 day) → **strong daily seasonality** .

Peaks near 168 hours (~7 days) → weekly seasonality.






Min & Max History (Train)



Choose the **Minimum** and the **maximum history** used in train:

MIN

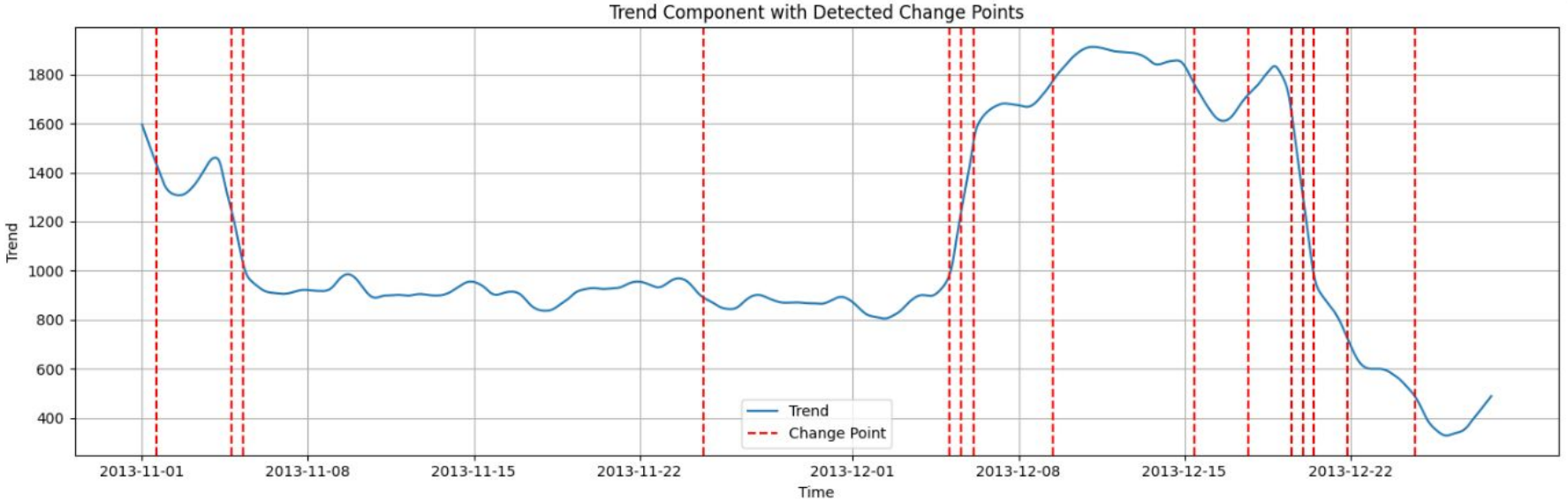
- Less than this risks insufficient data to learn seasonality and model parameters reliably.

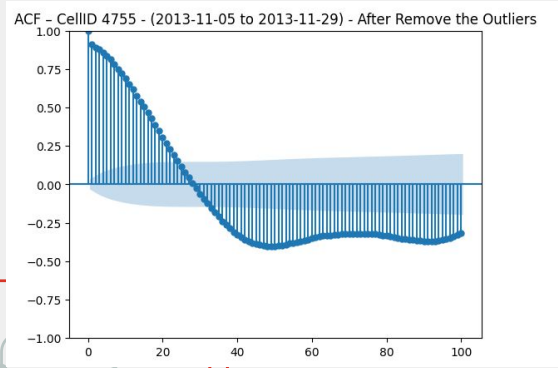
MAX

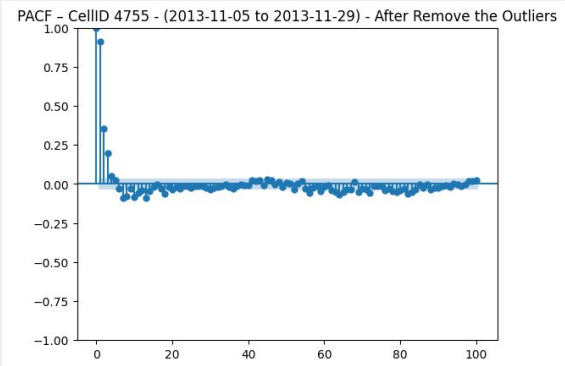
- Too long can include outdated data
- Since my seasonal period is 144 (1 day), minimum should be at least one or two full seasonal cycles.

1. Using Autocorrelation and Partial Autocorrelation
   - the lag after which ACF/PACF values drop below significance (confidence bounds).
   - That lag indicates the effective memory of my series – We want at least that many points in training to capture dependencies.

2. Statistical Change Point Detection
   - Training windows can be chosen to stay within these “stable” segments.


Trend Component with Detected Change Points
Trend
Time
Trend
Change Point
1800
1600
1400
1200
1000
800
600
400
2013-11-01
2013-11-08
2013-11-15
2013-11-22
2013-12-01
2013-12-08
2013-12-15
2013-12-22


ACF – CellID 4755 - (2013-11-05 to 2013-11-29) - After Remove the Outliers
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
20
40
60
80
100


PACF – CellID 4755 - (2013-11-05 to 2013-11-29) - After Remove the Outliers
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
20
40
60
80
100

✅ Interpretation of ACF & PACF

1. **ACF (Autocorrelation Function):**
   - Shows **strong autocorrelation** up to about **lag 30**, and then **slowly decays**.
   - **Significant positive values** up to **lag 25–30**, then oscillates below significance.
   - Implies that past **values up to 30 lags** ago have a **statistically significant impact** on the current value.
2. **PACF (Partial Autocorrelation Function):**
   - Shows a sharp drop after lag 2 (**first few lags are significant**, especially lag 1 and 2).
   - ~~PACF is useful for identifying the order of AR terms – suggests AR(2) model.~~

Minimum History (to capture dependencies):

30 lags based on the **ACF decay**
30 × 10 min = 300 minutes = 5 hours

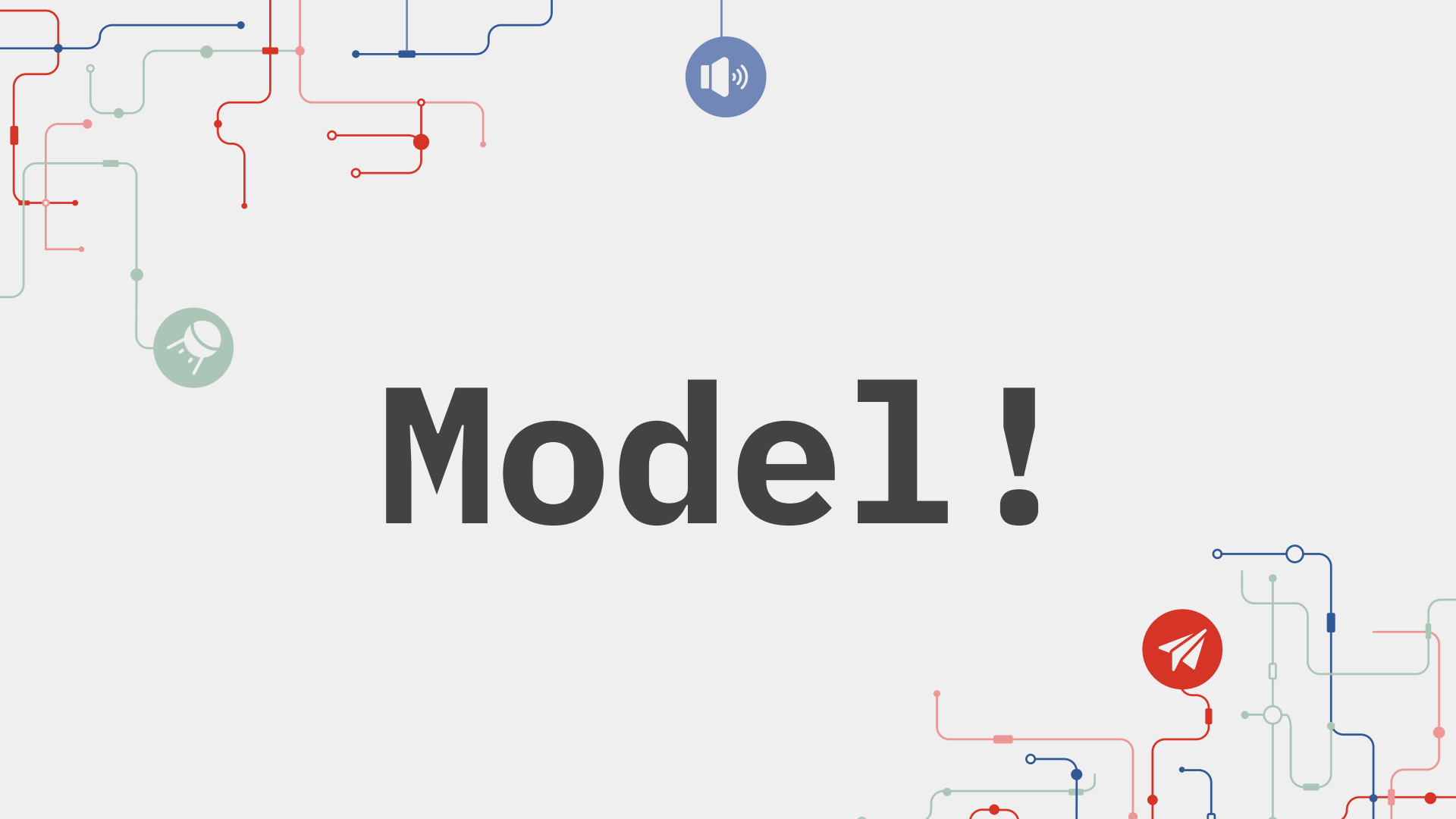
So we need **at least 5 hours** of past data in training.

Maximum History (to capture seasonality):

as my FFT indicate a **daily seasonality** = 24 hours
In 10-minute resolution, that equals:
24 hours × 60 / 10 = 144 lags

To confidently model seasonal and trend patterns, and avoid overfitting:
Capture **multiple cycles** – often **3** to 5 full **cycles**.
144 × 3 = **432 lags**
to
144 × 5 = 720 lags

432 lags


Model!

Why use harmonic terms?

1. Real-world seasonality is rarely a pure sine wave.
2. Adding multiple harmonics lets you approximate complex seasonal shapes like sharp peaks, asymmetric cycles, or multiple peaks per period.
3. Harmonics form the basis of Fourier series, which represent periodic signals as sums of sine and cosine terms.

Mathematical form:

For a seasonality with fundamental period T, the harmonic seasonal component S_t can be modeled as:

$$S_t = \sum_{k=1}^{K} \left[a_k \cos\left(\frac{2\pi k t}{T}\right) + b_k \sin\left(\frac{2\pi k t}{T}\right) \right]$$

Where:

- K = number of harmonics included
- a_k, b_k = coefficients for each harmonic
- t = time index
- T = fundamental seasonal period (e.g., 24 hours)

Harmonic trend:

But within this upward trend, there are slow oscillations or waves that last weeks or months (**longer cycles than typical “seasonality”**).

These are **not abrupt jumps** , but **gentle waves** riding on top of the trend.

Harmonic trend lets your model capture these smooth, **long-term cyclical variations** naturally.

Instead of modeling trend as a simple linear or polynomial function:

$$\text{Trend}(t) = \beta_0 + \beta_1 t$$

We add a **slow periodic component** using sine and cosine waves (harmonics):

$$\text{Harmonic Trend}(t) = \beta_0 + \beta_1 t + \sum_{k=1}^{K} \left(a_k \cos\left(\frac{2\pi k t}{T}\right) + b_k \sin\left(\frac{2\pi k t}{T}\right) \right)$$

Where:

- $\beta_0 + \beta_1 t$ is the usual linear trend
- The sum of sine and cosine terms adds **slow oscillations** with period T (e.g., monthly or quarterly cycles)
- K controls how many harmonics you include (more harmonics capture more detail)

Models like **Prophet** model seasonality with Fourier terms (**harmonics**) AND allow the trend to have changepoints and smooth variations – sometimes including harmonic terms to capture cyclical trends.

I will build my Model Based on these points:

hybrid **Seasonal ARIMA** model enriched with **harmonic trend** and **seasonality** components, integrating classical tools (**ACF, PACF** , **stationarity tests**) and modern insights (**FFT-based seasonality** , **harmonic features**), plus grounding in **historical mean**

univariate SARIMAX modeling.

📊 Model Design Overview

1. Seasonality Detection
I used Fast Fourier Transform (FFT) to analyze the power spectrum of the signal.

This analysis revealed strong periodic components near 24 hours (1 day), indicating a clear daily seasonality in the data.

This seasonal behavior will be explicitly modeled using harmonic functions.

2. Stationarity Check
Conducted the Augmented Dickey-Fuller (ADF) test to assess stationarity:

ADF Statistic: -11.03

p-value: 5.61e-20

The extremely low p-value confirms that the series is strongly stationary, allowing the use of ARIMA-based models without differencing.

3. Modeling Approach
I will use a Seasonal ARIMA (SARIMA) model, augmented with additional components for improved interpretability and accuracy:

✔ Classical Time Series Techniques:
Seasonal ARIMA: To capture both short-term and seasonal autoregressive and moving average behaviors.

Autocorrelation (ACF) and Partial Autocorrelation (PACF) plots: Used to determine the appropriate AR (p) and MA (q) terms for the model.

Historical Mean Component: Incorporated as a baseline reference for modeling and training. This is considered very important due to its stabilizing effect and contextual relevance.

✔ Harmonic Components:
Harmonic Seasonal Component (important):
Sinusoidal terms will be used to explicitly model the identified ~24-hour periodicity.

Harmonic Trend Component (important):
Low-frequency harmonics will be used to model gradual trend changes that might not be fully captured by ARIMA alone.

4. Modern + Classical Integration
This model intentionally combines:

Classical statistical models (SARIMA, ACF/PACF, stationarity testing),

With modern signal processing insights (FFT for frequency domain analysis, harmonic modeling),

And a data-driven baseline (historical mean-based correction and adjustment).

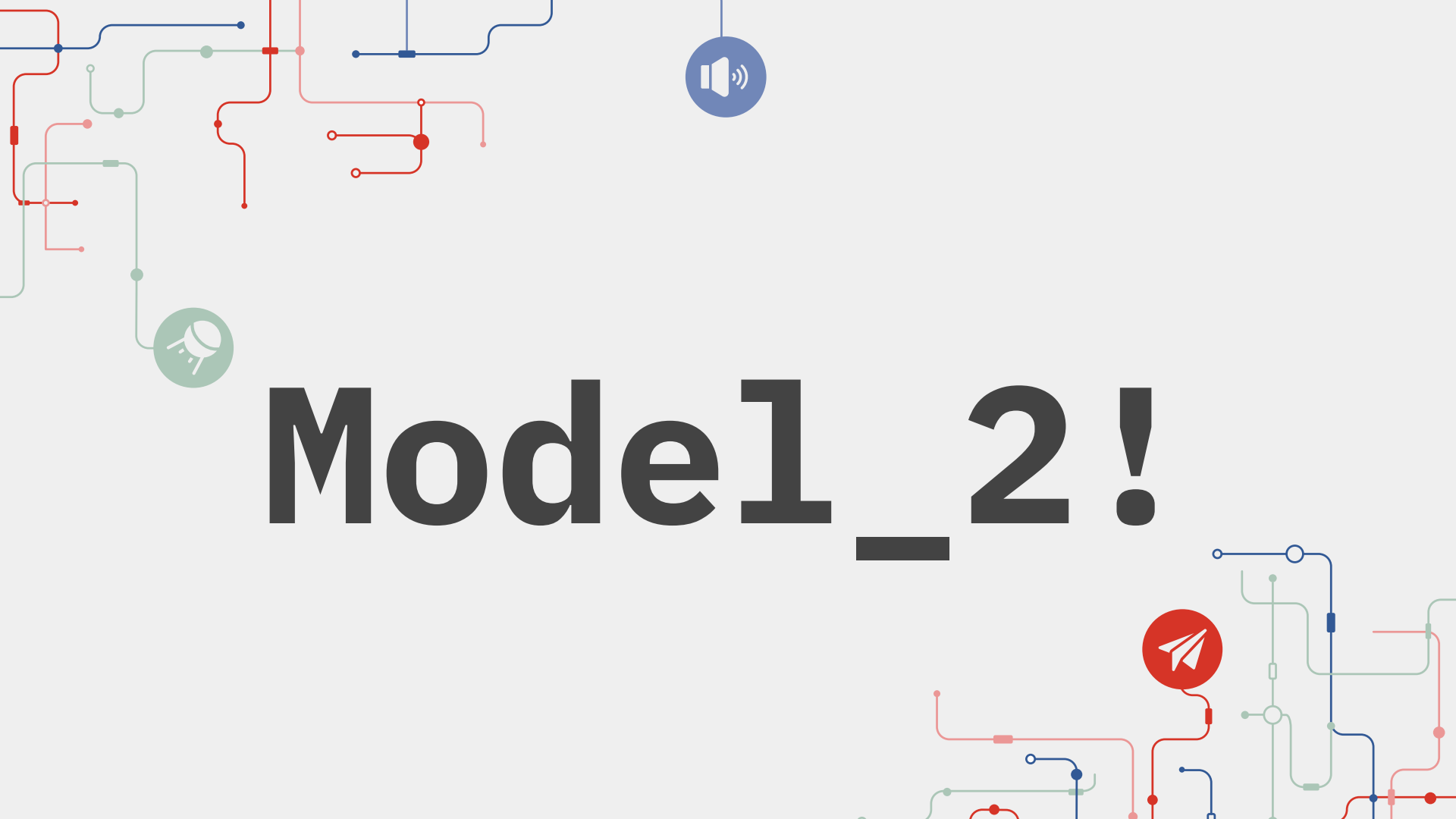
This hybrid approach ensures interpretability, robustness, and adaptability to both short-term dynamics and long-term cyclic trends.

Problem with the outliers, as removing the outliers lead to miss some timestamps, so the SRIMAX faced problems

1. Skip the step of removing the outliers and train the dataset

2. **Interpolation**
   - What it does: Estimates missing values using **linear** or other interpolation methods **between known values** .
   - When to use: When the metric is expected to **vary smoothly over time**.
   - Best for numeric, continuous signals (like network usage, internet traffic, etc.).

```
df_filled = df.interpolate(method='linear')
```

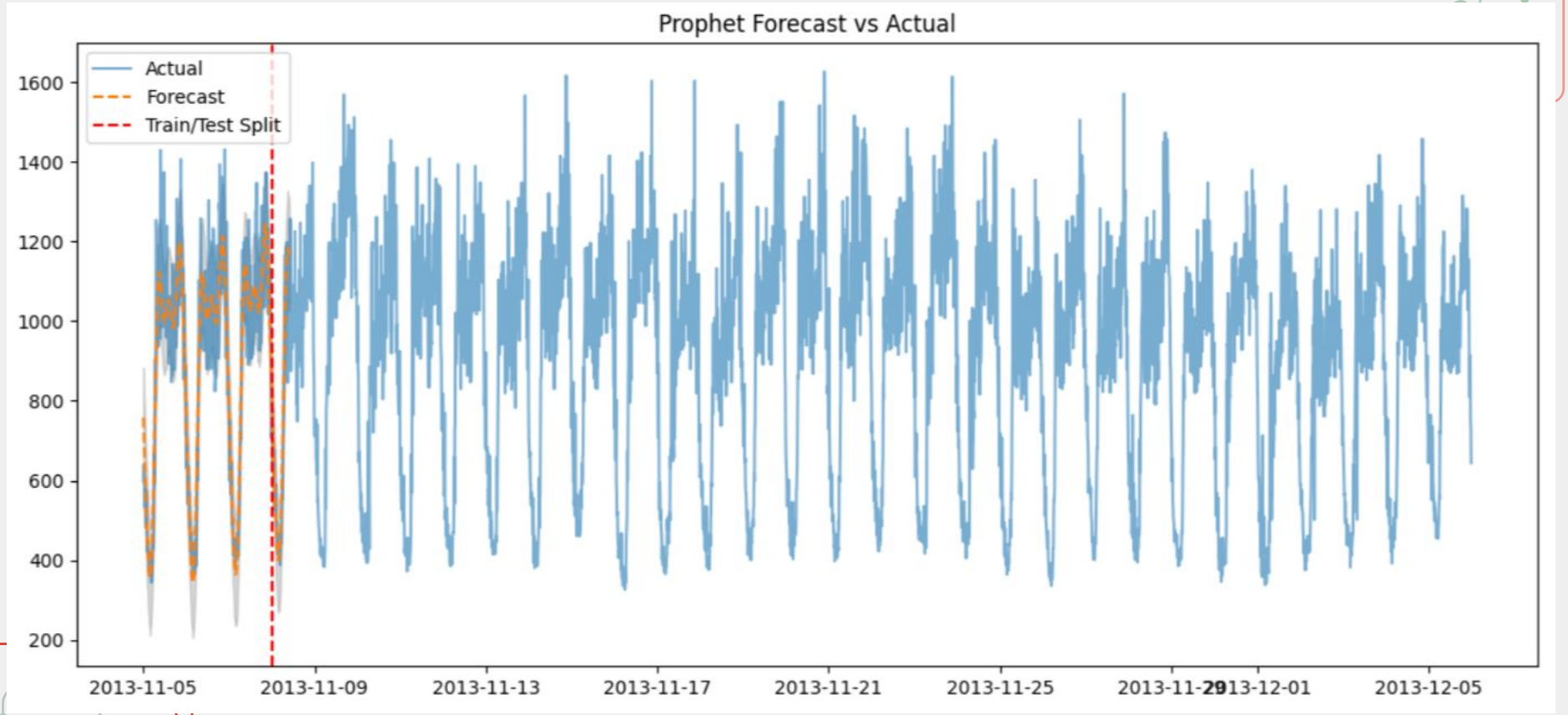

Model_2!

1. seasonal_terms
2. trend_terms
3. harmonic_regressors
4. train_historical_mean
5. Add regressors
6. Prophet modeling
7. Train the model
8. Forecast

```
train_size = 432
test_size = 60
```

cell_4755

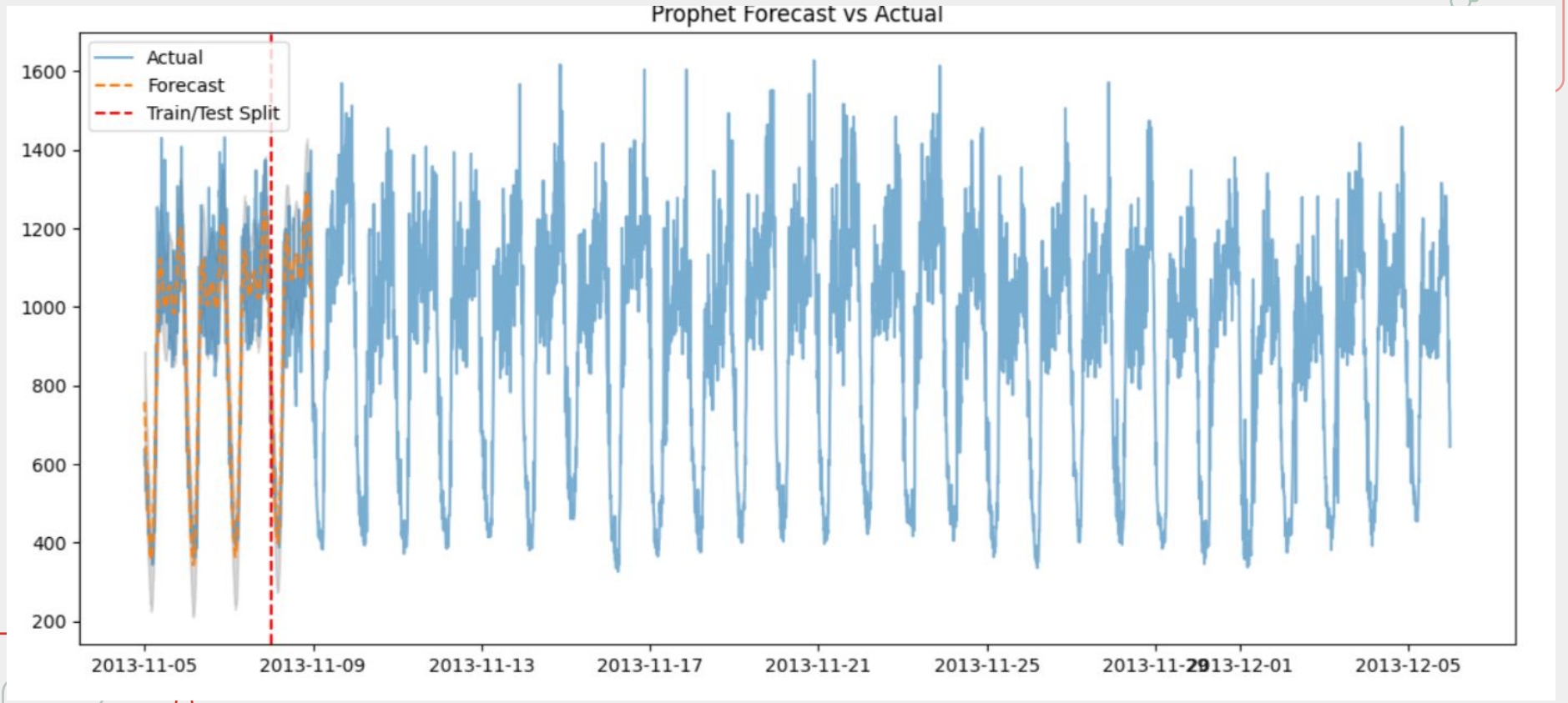
```
Custom Accuracy: 89.30%
```



```
train_size = 432
test_size = 144
```

cell_4755

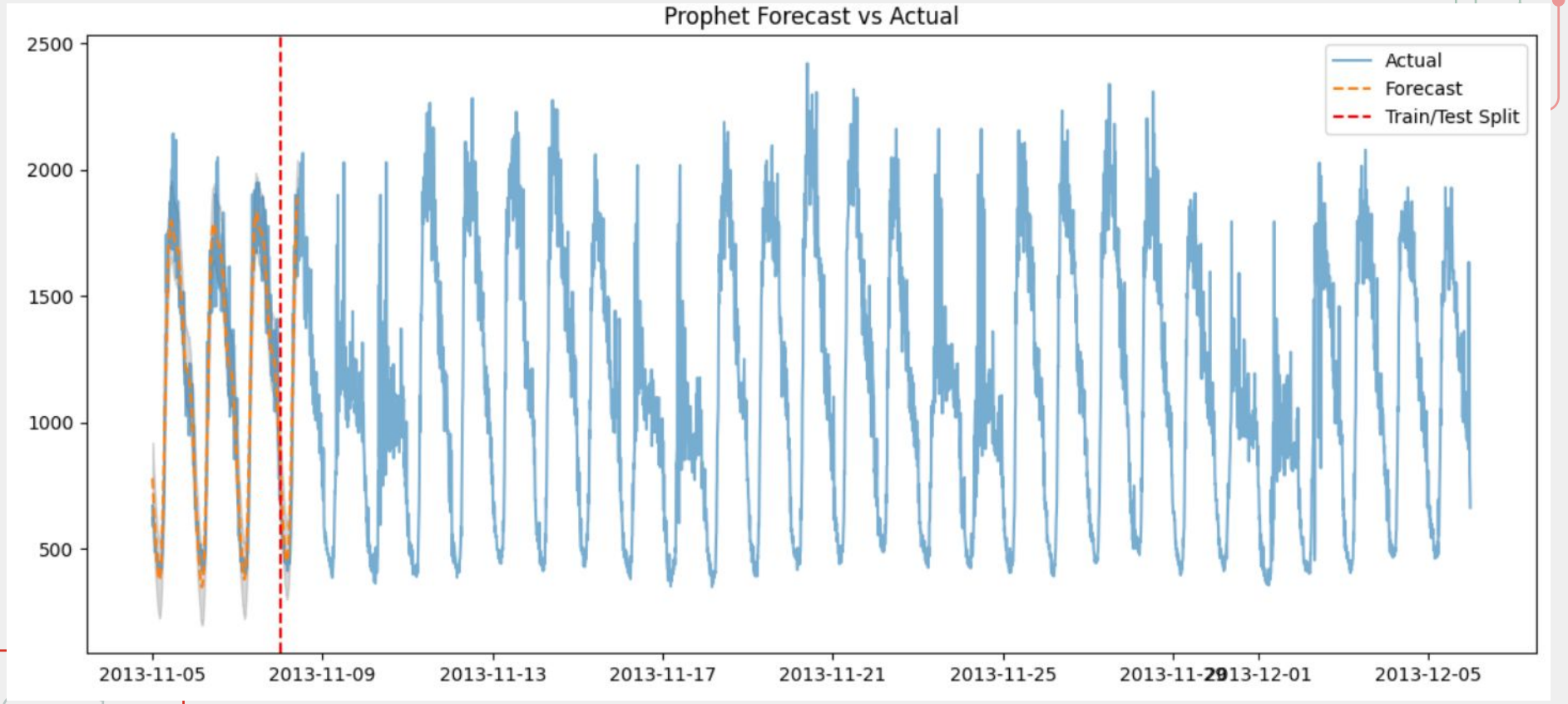
```
Custom Accuracy: 88.93%
```



```
train_size = 432
test_size = 60
```

cell_4855

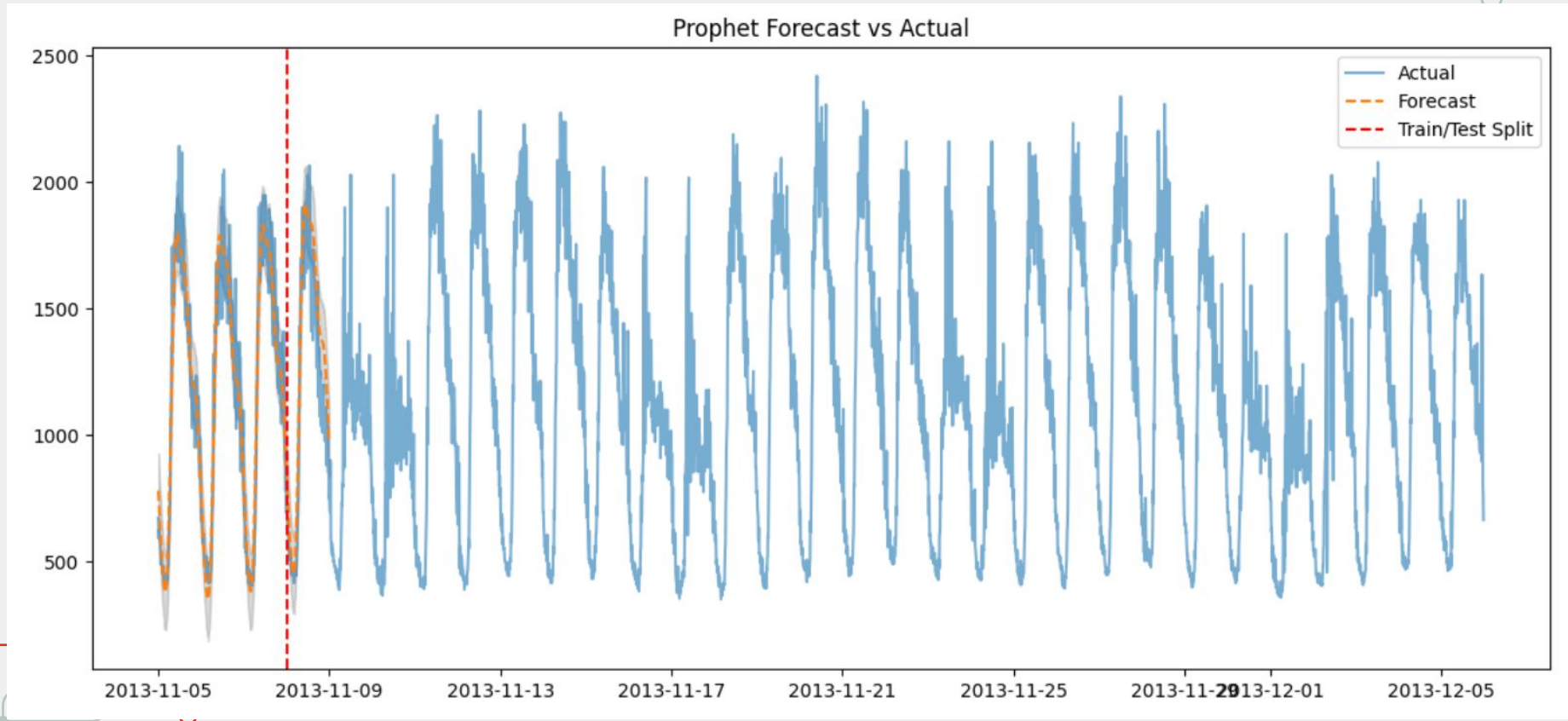
```
Custom Accuracy: 86.19%
```

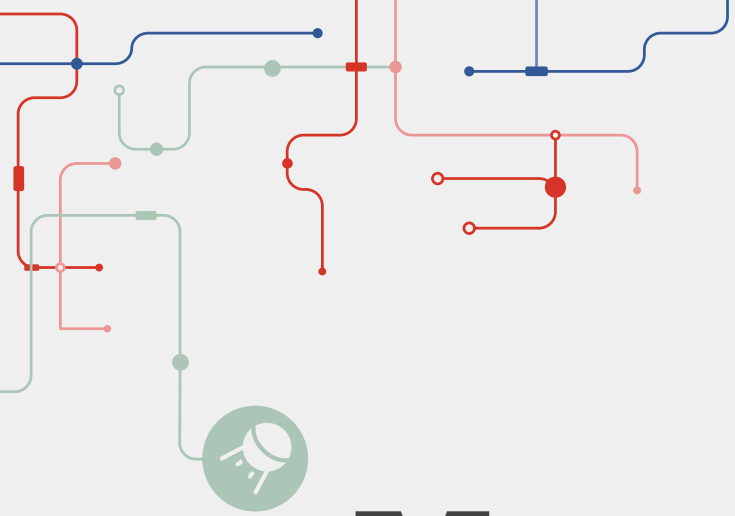


```
train_size = 432
test_size = 144
```

cell_4855

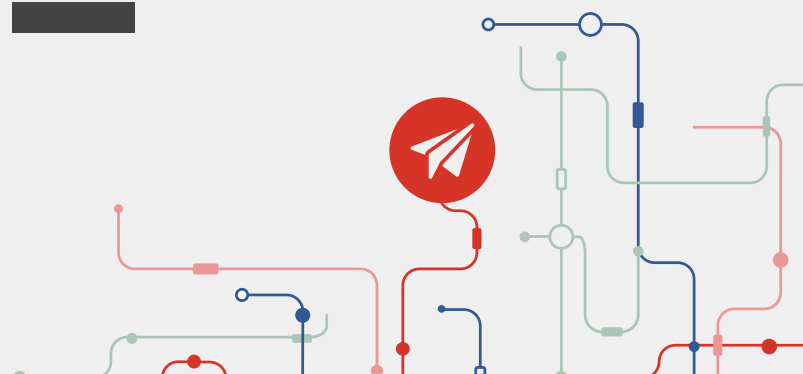
Custom Accuracy: 84.88%






Model_3!

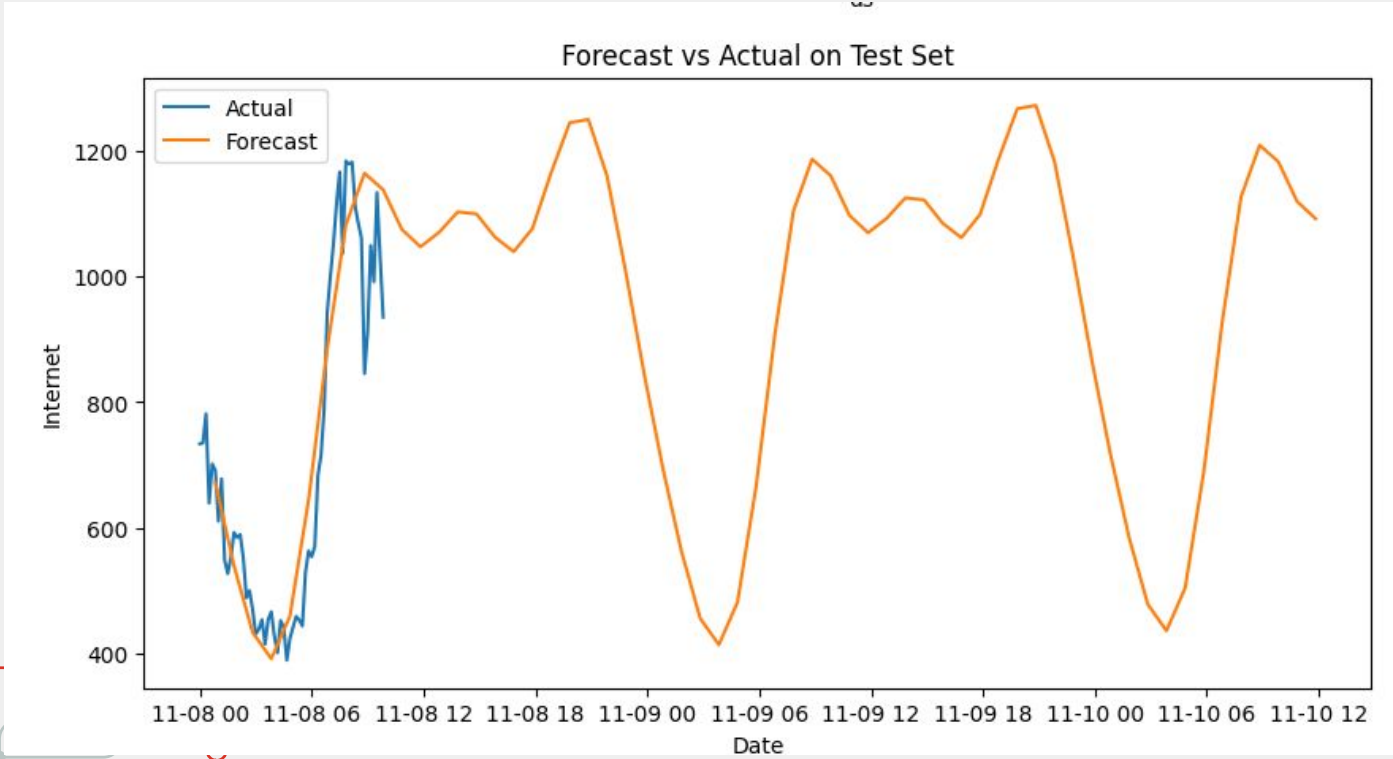
Basic Prophet



```
future = model.make_future_dataframe(periods=test_size, freq='H')  #  ('H' for hourly)
```

```
Accuracy: 86.73%
```

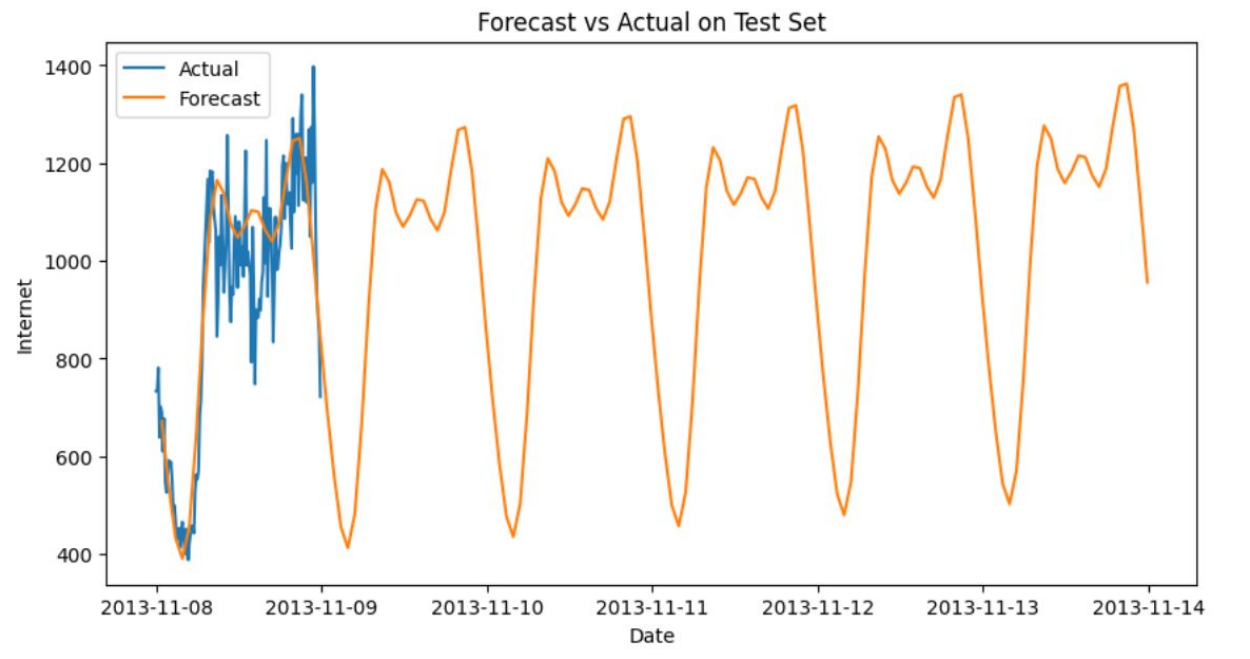
```
train_size = 432
#train_size = 4300
test_size = 60
```

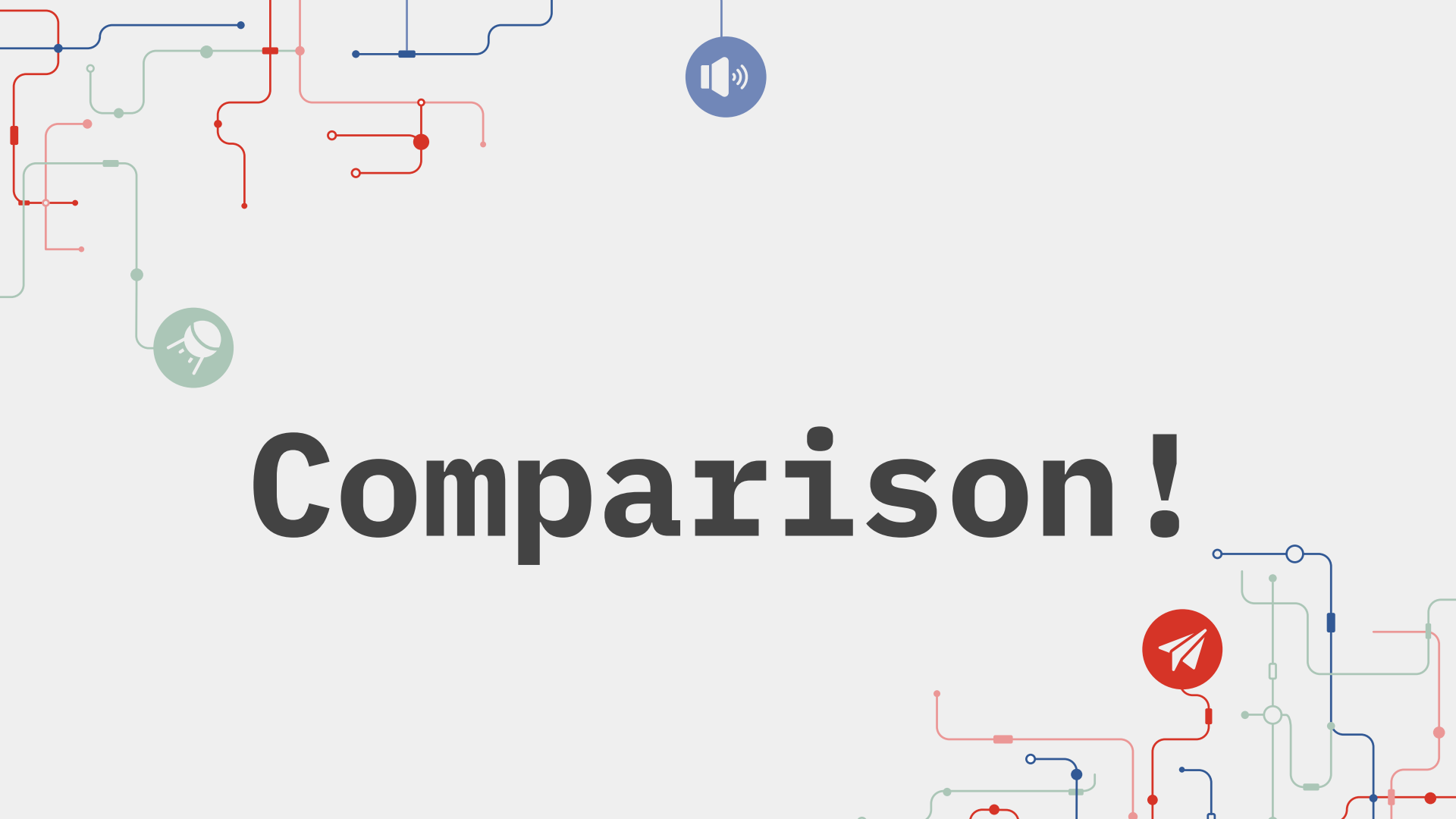


```
future = model.make_future_dataframe(periods=test_size, freq='H')  #  ('H' for hourly)
```

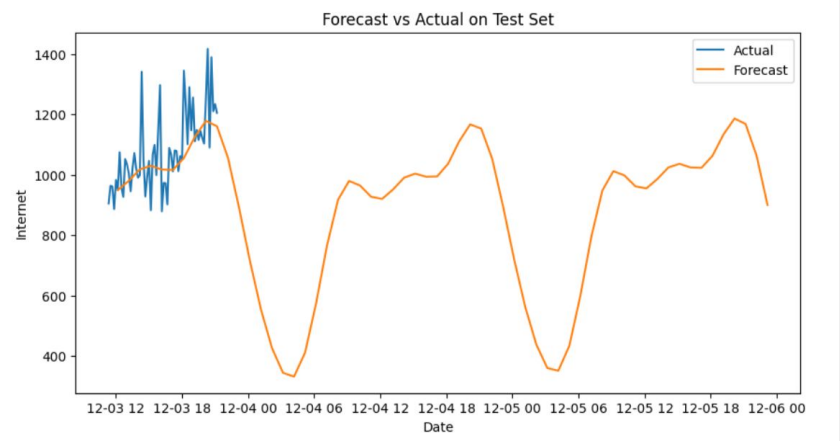
```
Accuracy: 87.13%
```

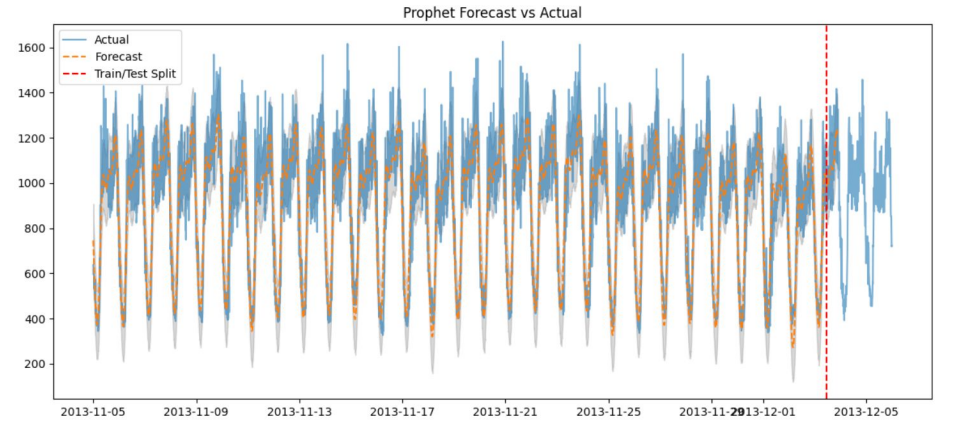
```
train_size = 432
#train_size = 4300
#test_size = 60
test_size = 144
```




Comparison!

train_size = 4100
test_size = 60

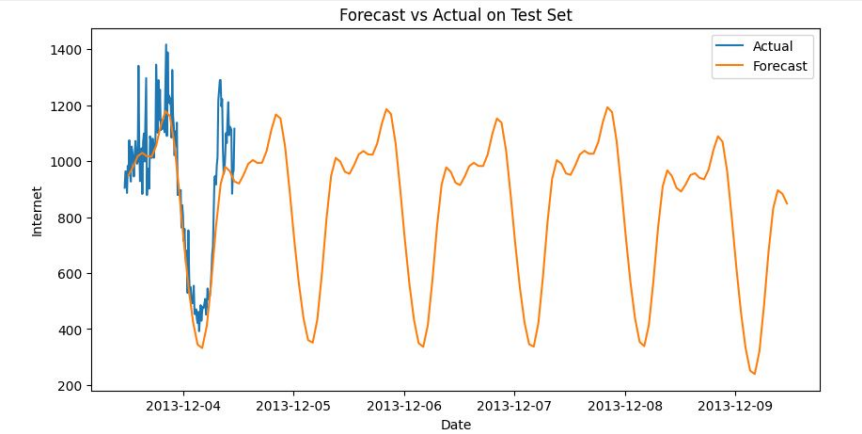


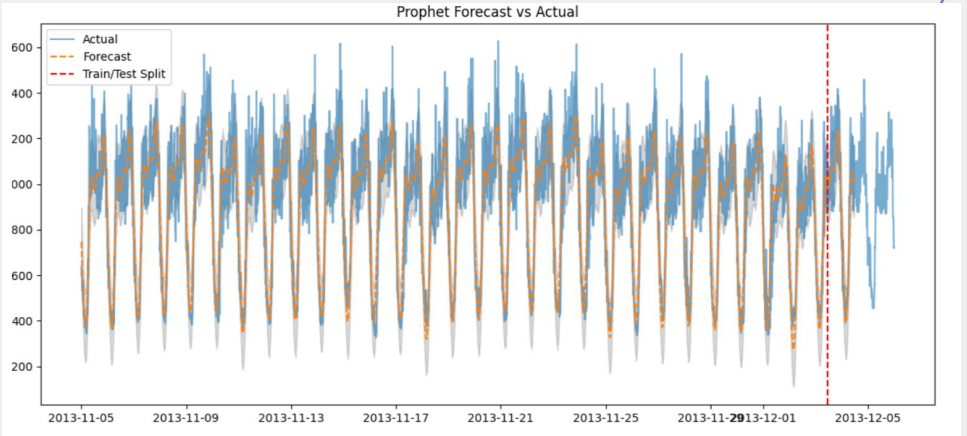


Accuracy: 93.00%

Custom Accuracy: 93.24%

train_size = 4100
test_size = 144

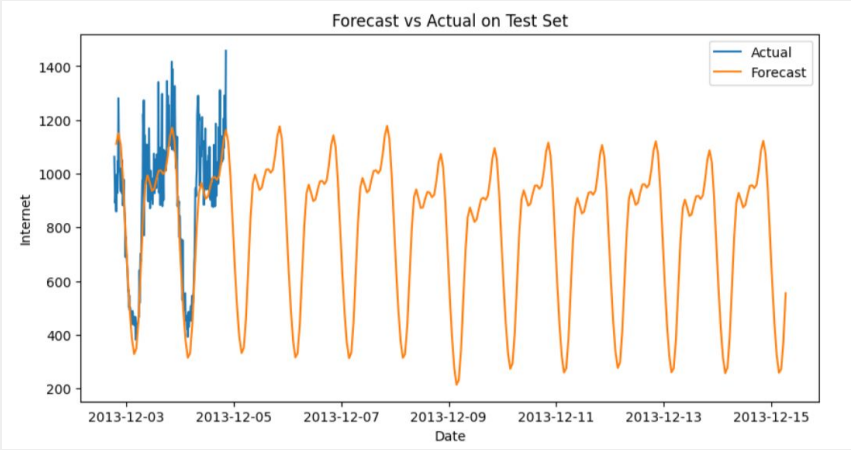


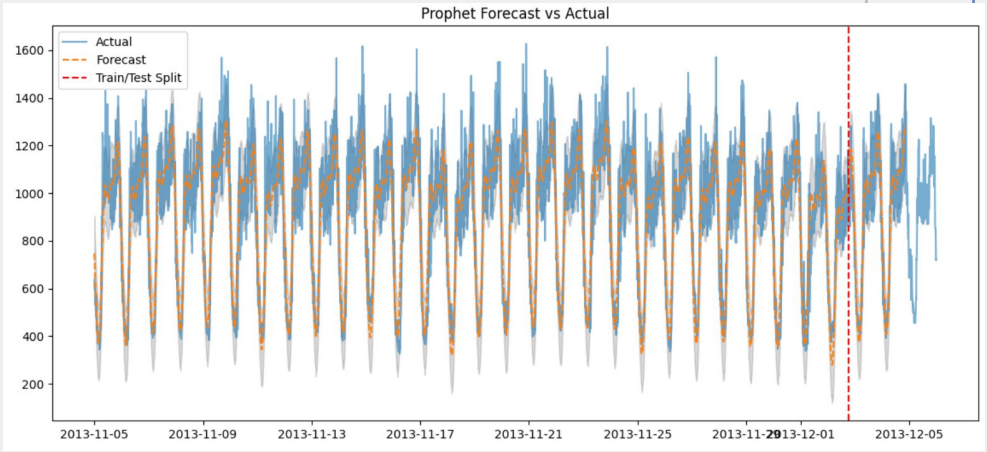


Accuracy: 89.10%

Custom Accuracy: 92.30%

train_size = 4000
test_size = 300





Accuracy: 90.76%

Custom Accuracy: 90.76%

Model_4!

ARIMA +
Seasonality +
Trend +
Mean History

📊 Model Summary: ARIMA with Fourier Terms + Mean History

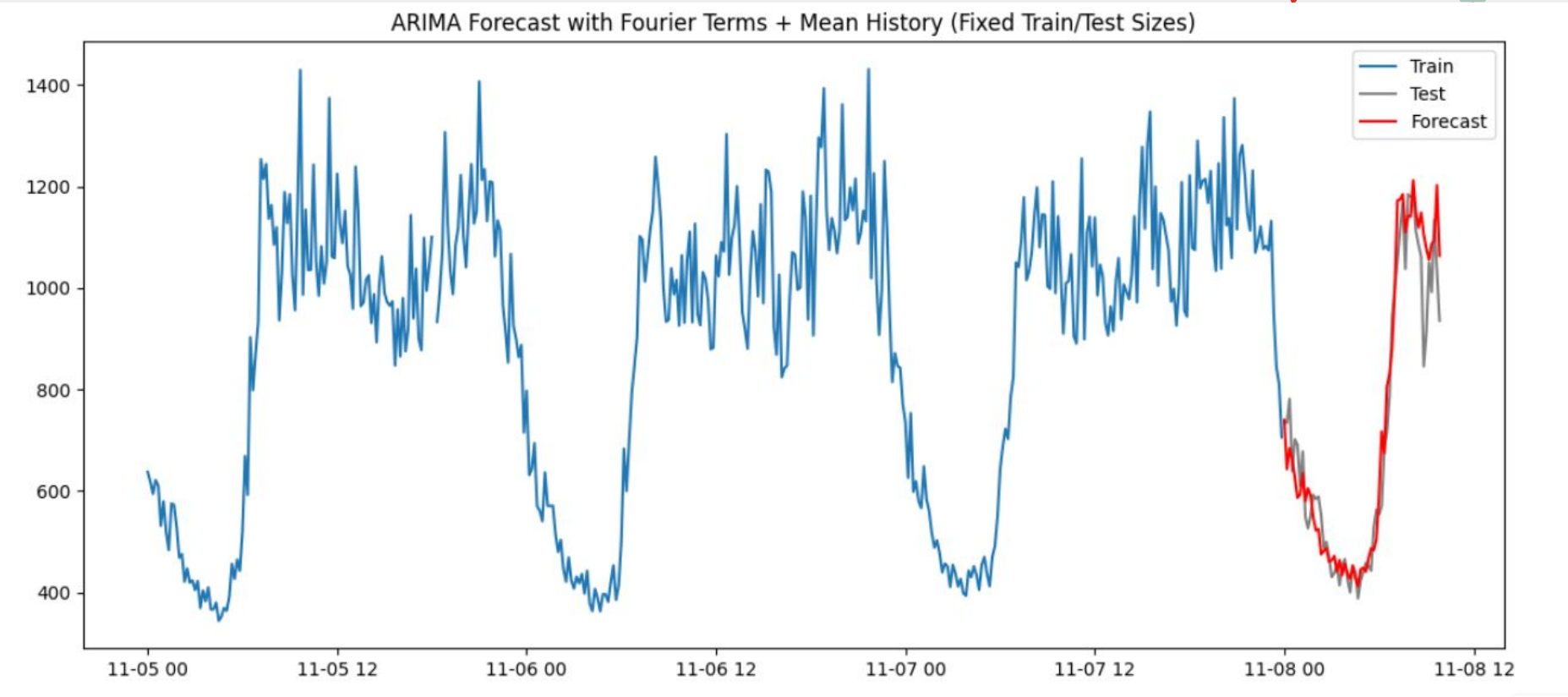
using a **hybrid** approach combining **ARIMA**, **Fourier seasonal** /**trend** terms, and **historical** time-based **averages**.

SARIMAX (ARIMA with **exogenous** variables)

Exogenous Inputs : Fourier terms for **seasonality** & **trend** + **mean** internet usage by time-of-day

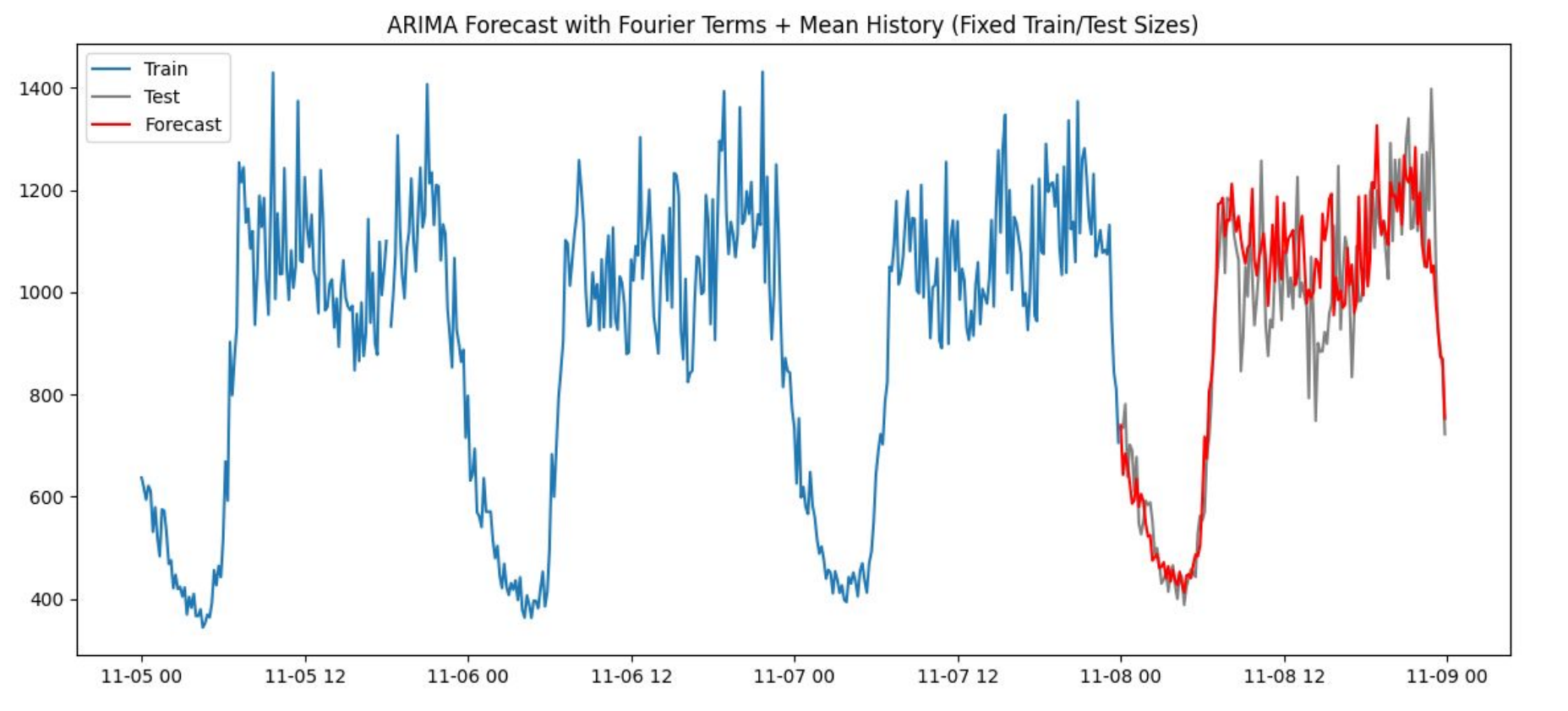
🕑 **Mean History** Component:
Calculated average usage per 10-min period only from training set
Used as an additional **exogenous** feature to capture daily repetitive patterns

🧠 **Fourier** Terms:
Used to capture **periodic patterns** in the data.
Seasonal Terms: 3 harmonics (sin/cos) for daily cycle
Trend Terms: 2 harmonics for overall trend



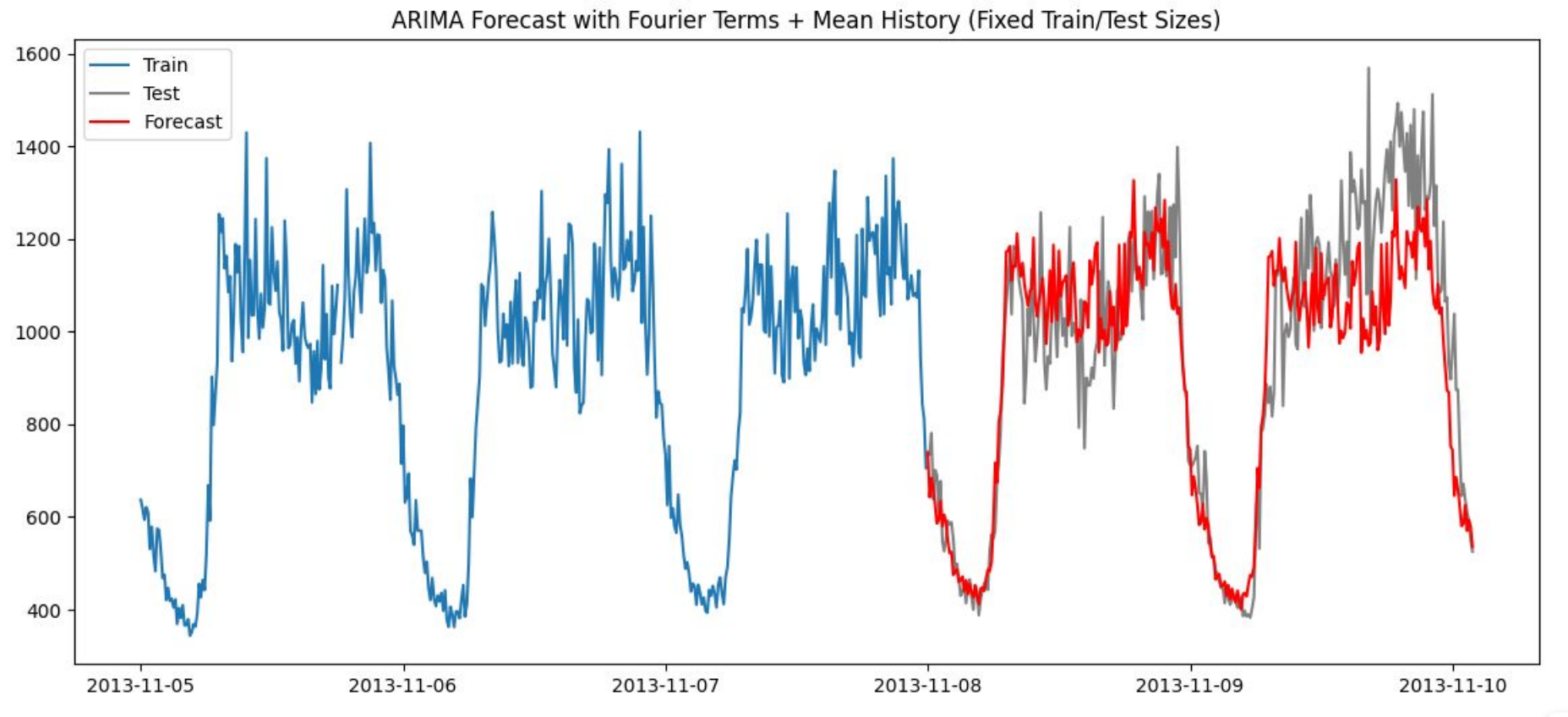
Accuracy: 92.71%

```
train_size = 432
test_size = 60
```



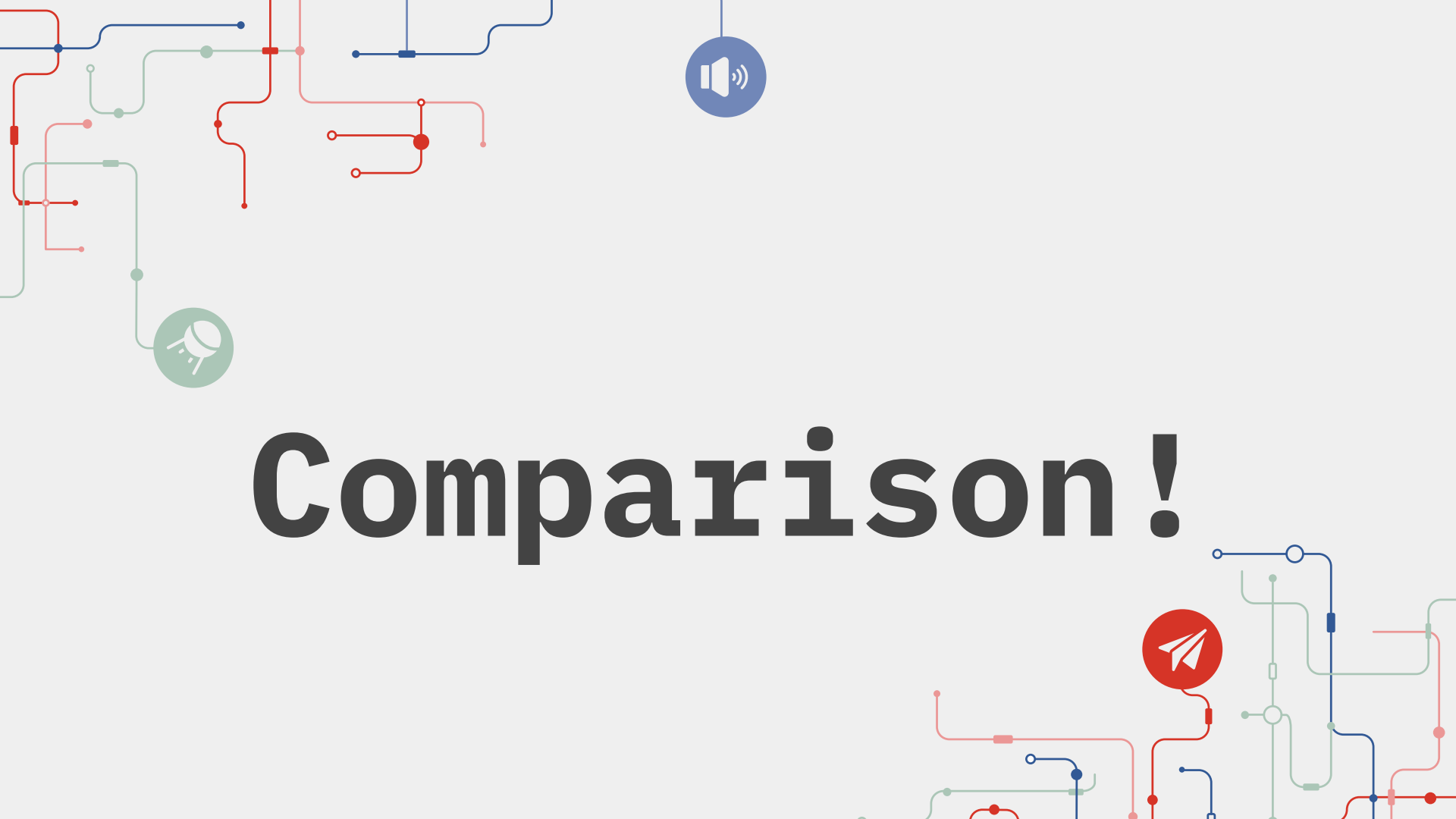
Accuracy: 91.34%

```
train_size = 432
test_size = 144
```

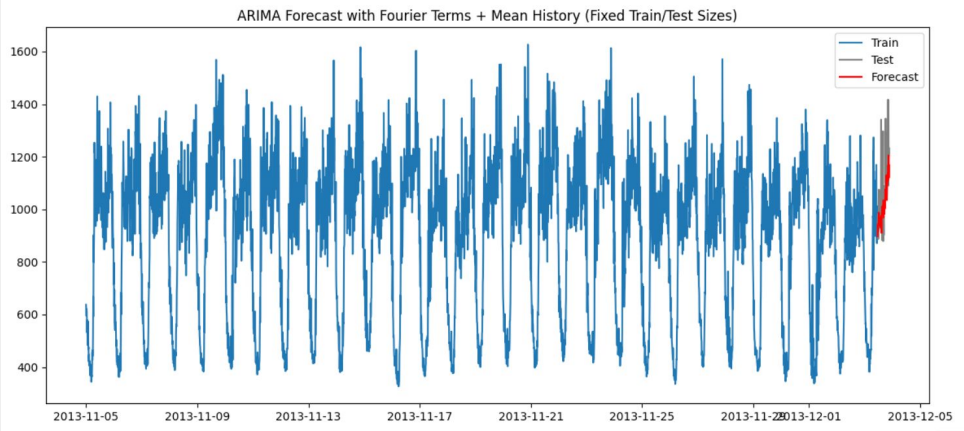


Accuracy: 89.86%

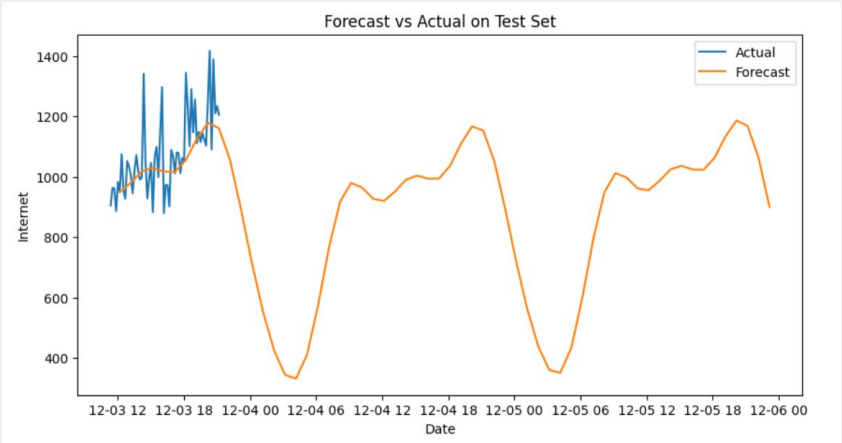
```
train_size = 432
test_size = 300
```

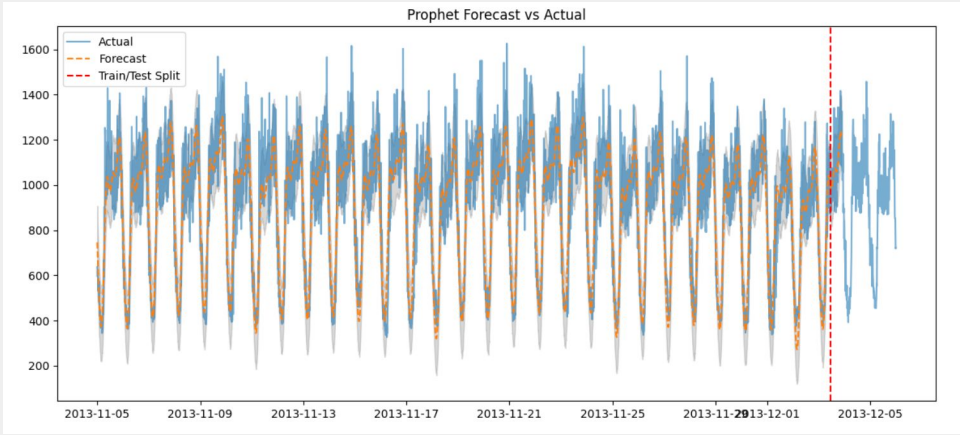

Comparison!

train_size = 4100
test_size = 60



Accuracy: 93.65%

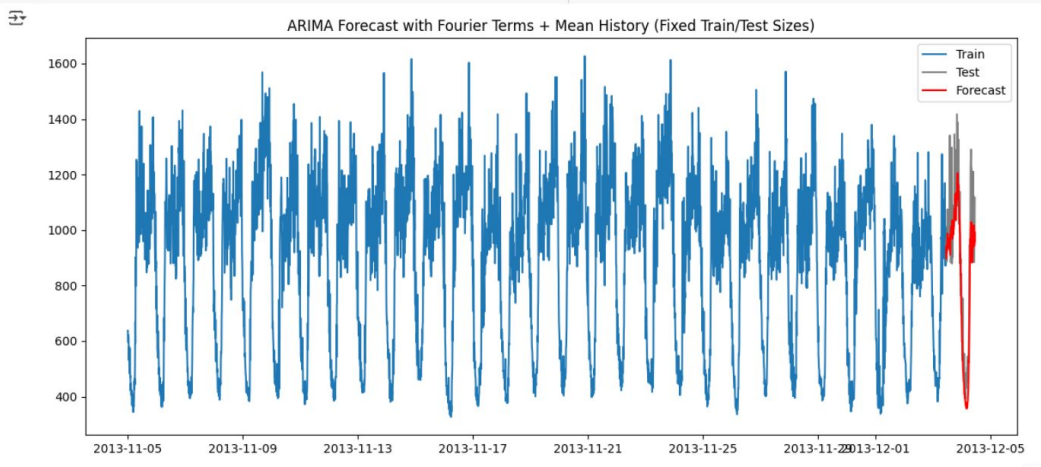




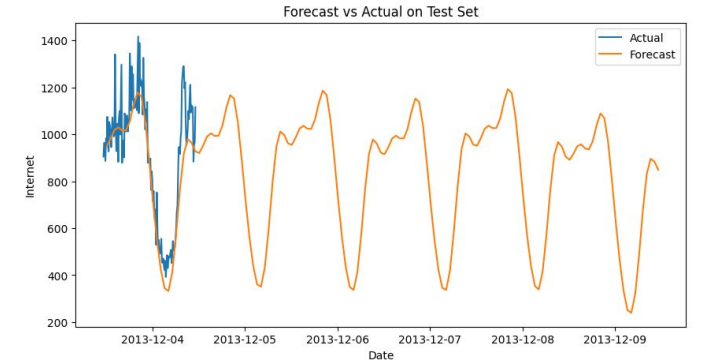
Accuracy: 93.00%

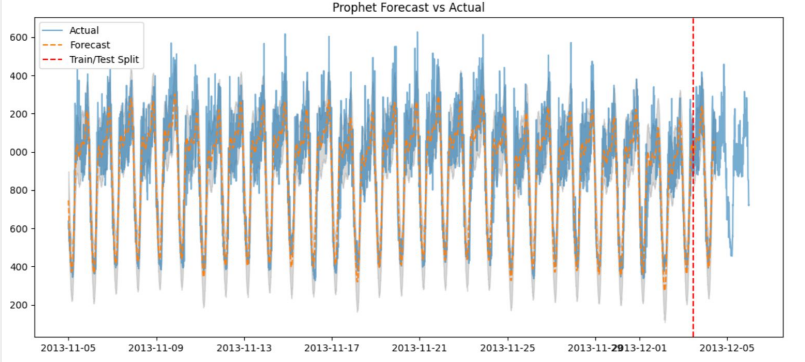
Custom Accuracy: 93.24%

train_size = 4100
test_size = 144



Accuracy: 92.83%

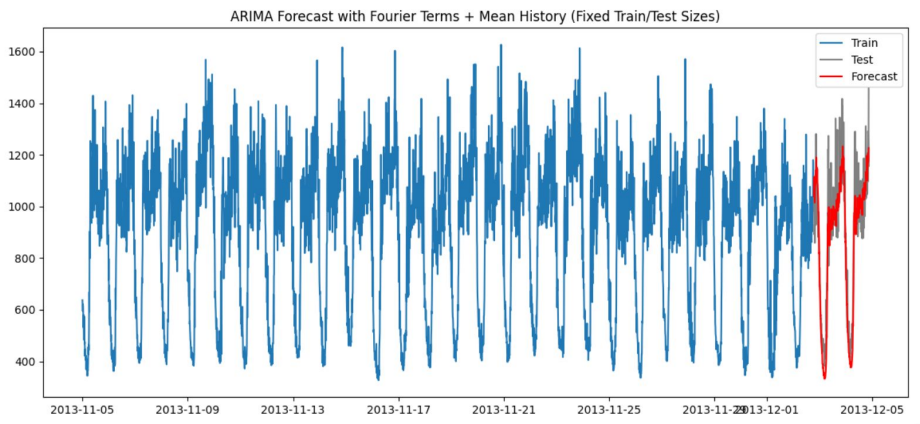




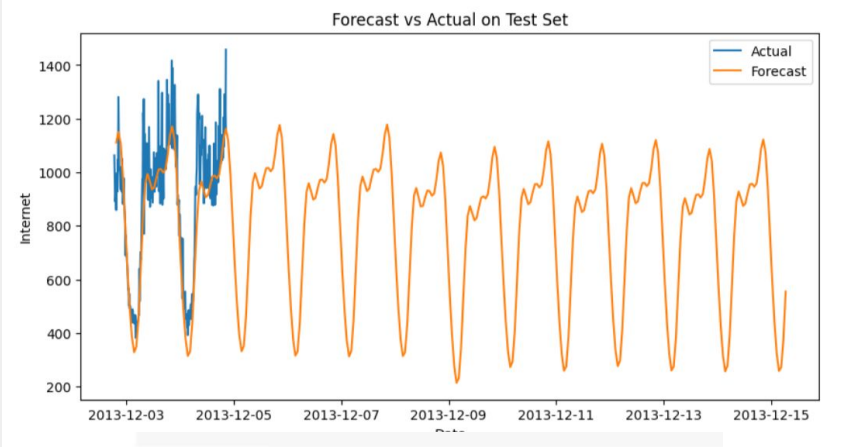
Accuracy: 89.10%

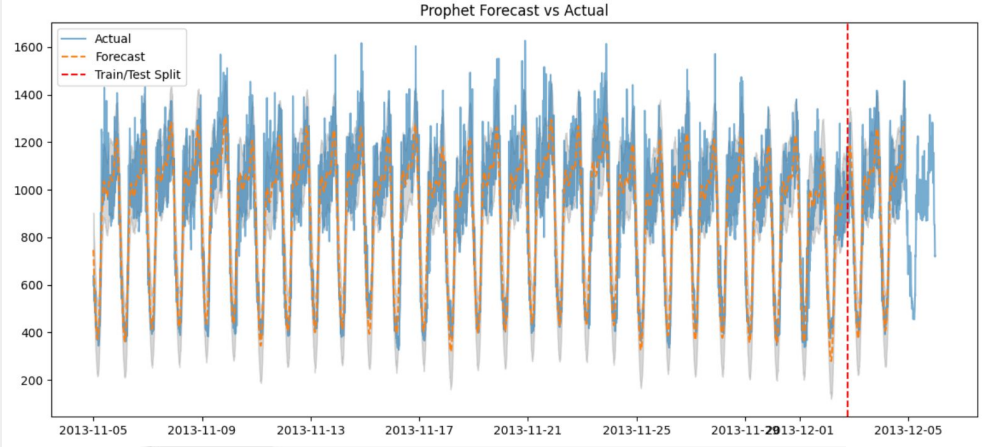
Custom Accuracy: 92.30%

train_size = 4000
test_size = 300



Accuracy: 90.68%

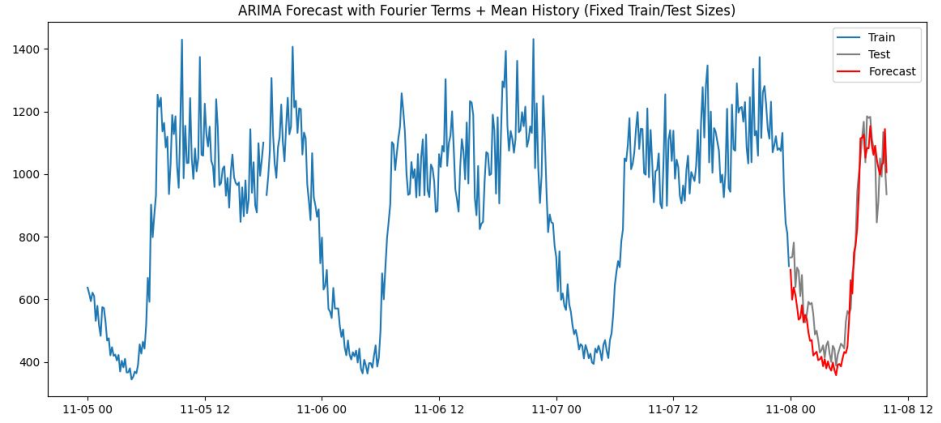




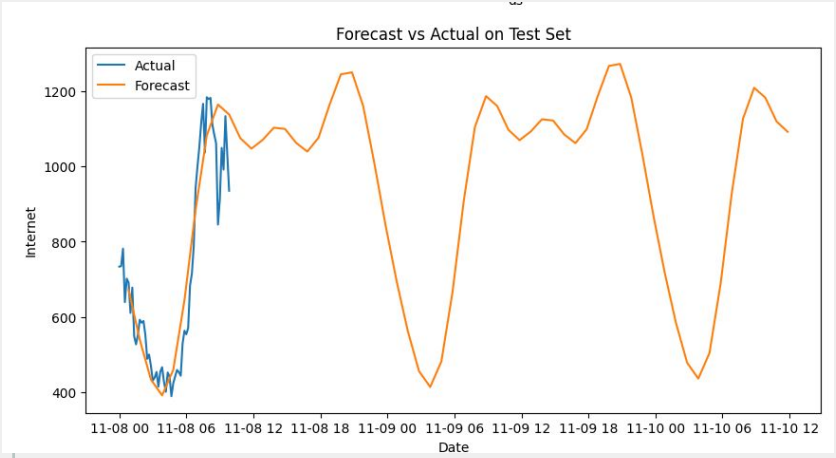
Accuracy: 90.76%

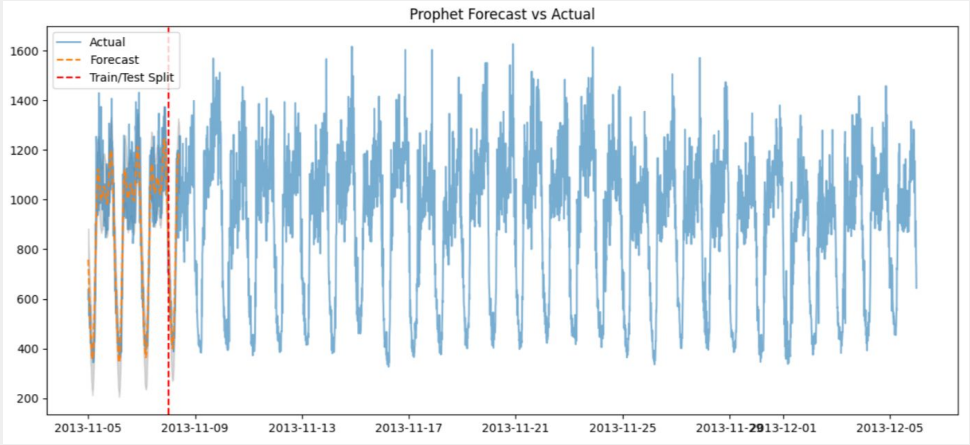
Custom Accuracy: 90.76%

```
train_size = 432
test_size = 60
```




```
Accuracy: 90.04%
```





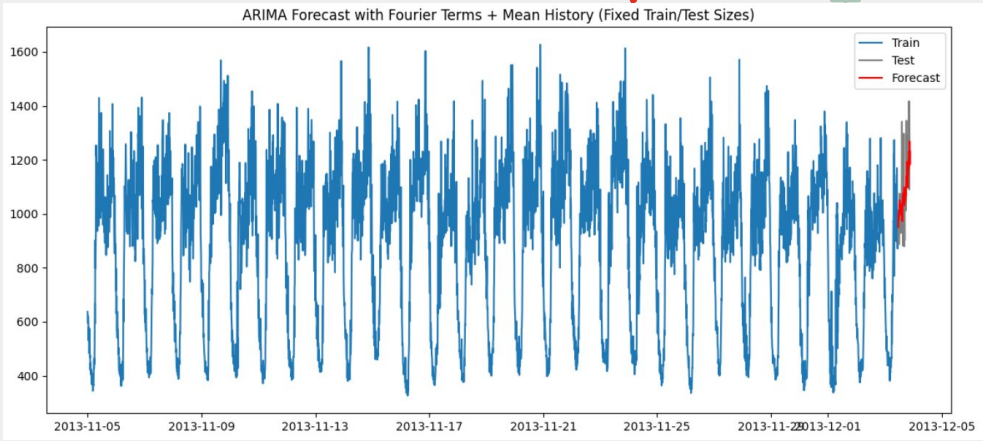

```
Accuracy: 86.73%
```

```
Custom Accuracy: 89.30%
```

How change the **train size** will affect:

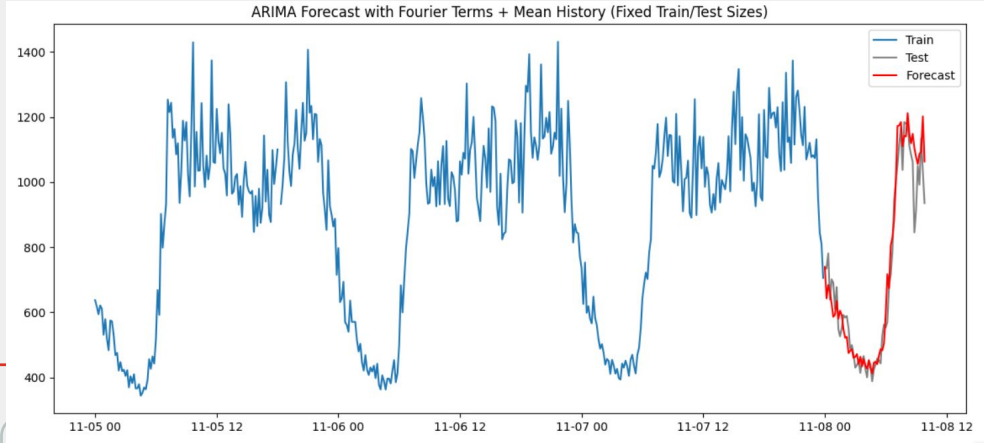
test_size = **60**

train_size = **432** ⬇️ | train_size = **4100** ↗️



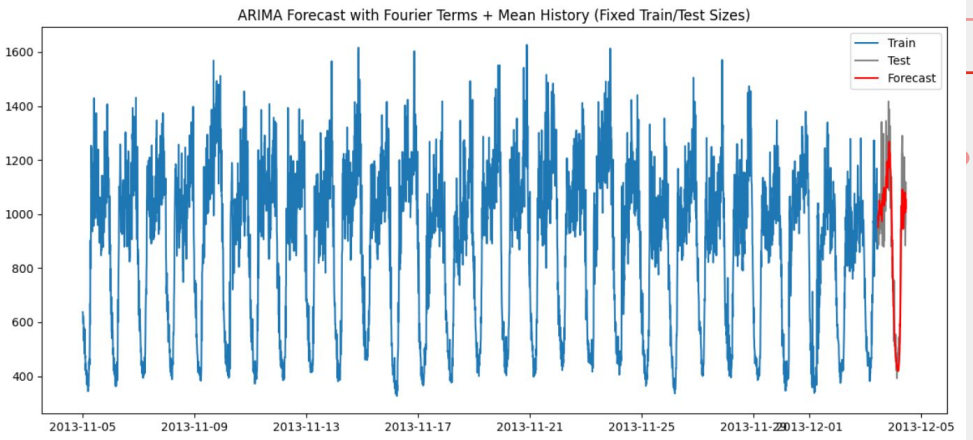
Accuracy: 92.71%

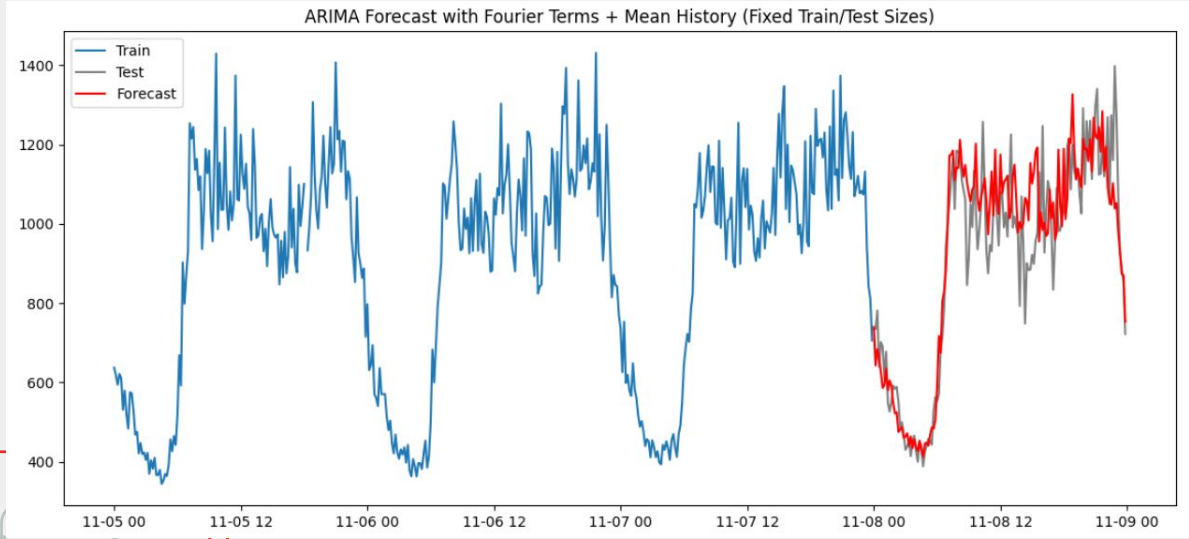
Accuracy: 93.65%



test_size = **144**

train_size = **432** ⬇️ | train_size = **4100** ↗️



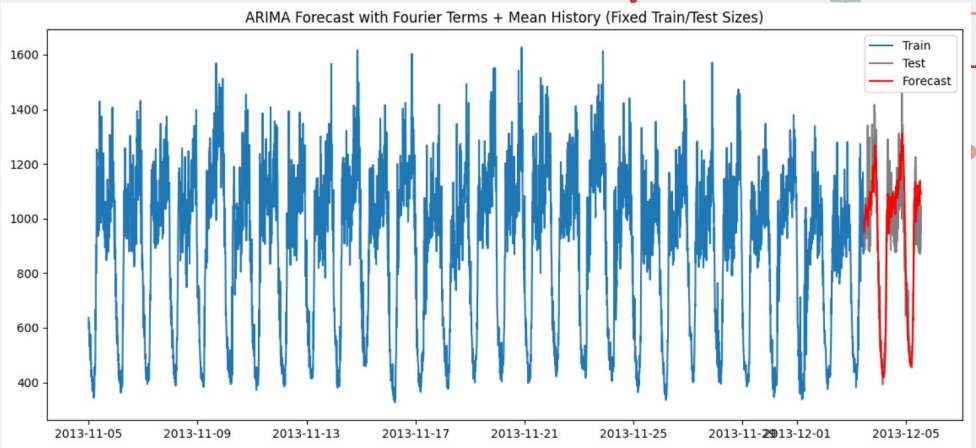


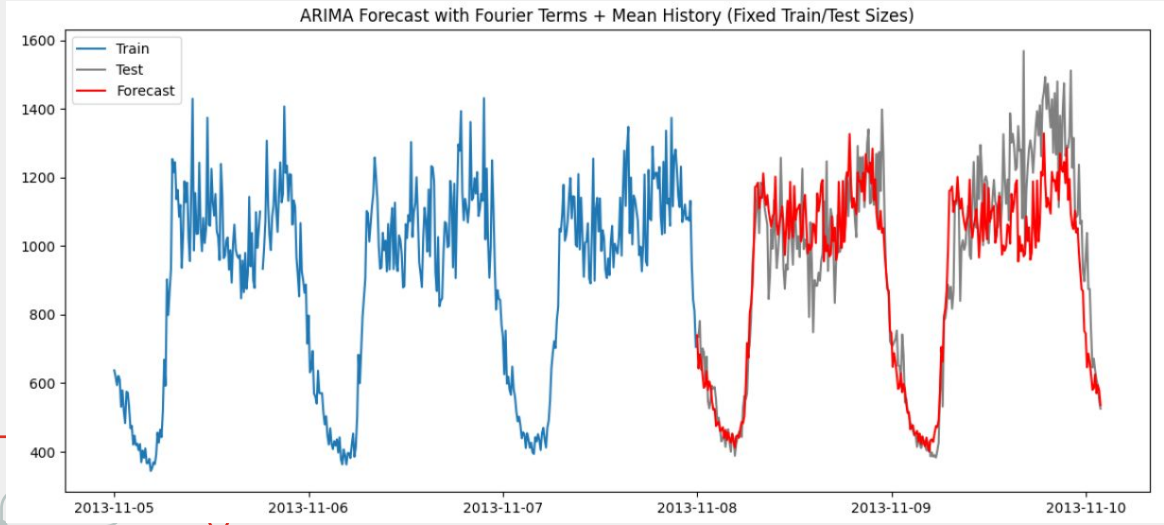
Accuracy: 92.83%

Accuracy: 91.34%

test_size = **300**

train_size = **432** ⬇️ | train_size = **4100** ↗️





Accuracy: 91.02%

Accuracy: 89.86%

Thanks. .