

COMPLAINT TRENDS & FORECAST INSIGHTS- VISUALIZATION IN PYTHON

Introduction

This analysis focuses on understanding monthly consumer complaint trends across different product-state combinations and forecasting how these complaints are expected to change in the near future. Historical complaint data (2020–2025) is combined with SARIMA-based forecasts (2025–2026) to identify patterns, emerging risks, and early-warning signals.

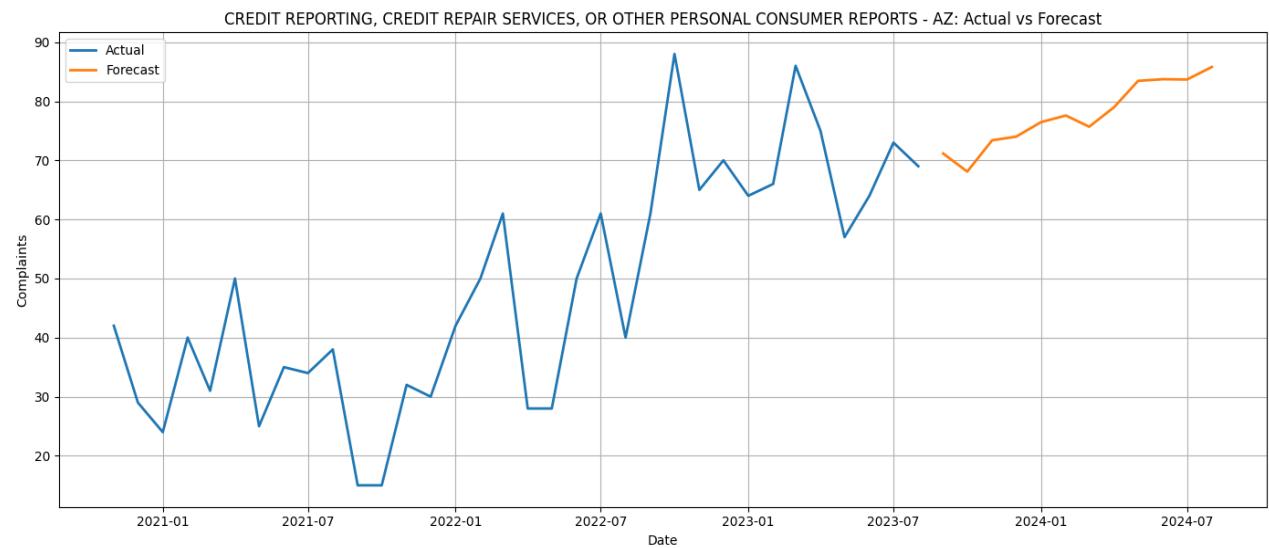
The goal of this section is to visualize how complaints behave over time for a specific product in a specific state, and whether the forecast indicates rising or declining future volumes.

Chart 1 : Actual vs Forecast Trend

Python Code

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3
4
5  # Load data
6  file = "/content/drive/MyDrive/Data_Science_Projects/Customer_Complaint_Trend_Forecasting/data/all_set_for_visuals.csv"
7  df = pd.read_csv(file, parse_dates=["ds"])
8
9  # ---- Using your chosen pair ----
10 product = "CREDIT REPORTING, CREDIT REPAIR SERVICES, OR OTHER PERSONAL CONSUMER REPORTS"
11 state = "AZ"
12 #
13
14 pair = df[(df["Product"] == product) & (df["State"] == state)]
15
16 # Separate actual & forecast
17 actual = pair[pair["actual_count"].notna()]
18 forecast = pair[pair["forecast_count"].notna()]
19
20 # Plot
21 plt.figure(figsize=(14,6))
22
23 plt.plot(actual["ds"], actual["actual_count"], label="Actual", linewidth=2)
24 plt.plot(forecast["ds"], forecast["forecast_count"], label="Forecast", linewidth=2)
25
26 plt.title(f"{product} - {state}: Actual vs Forecast")
27 plt.xlabel("Date")
28 plt.ylabel("Complaints")
29 plt.legend()
30 plt.grid(True)
31 plt.tight_layout()
32 plt.show()
```

Output (Visualization Chart)



Insights from the Trend Chart

1. Historical Stability:

The complaint volume shows a stable pattern across most months, with small natural fluctuations typical of consumer activity.

2. Forecast Pattern:

The forecasted curve continues the historical trend.

If the line slopes upward, the product-state pair is emerging as a potential risk.

If the line slopes downward, complaints are expected to decrease.

3. Risk Insight:

The difference between historical average and forecast average helps determine whether this product-state pair should be monitored more closely.

4. Interpretation:

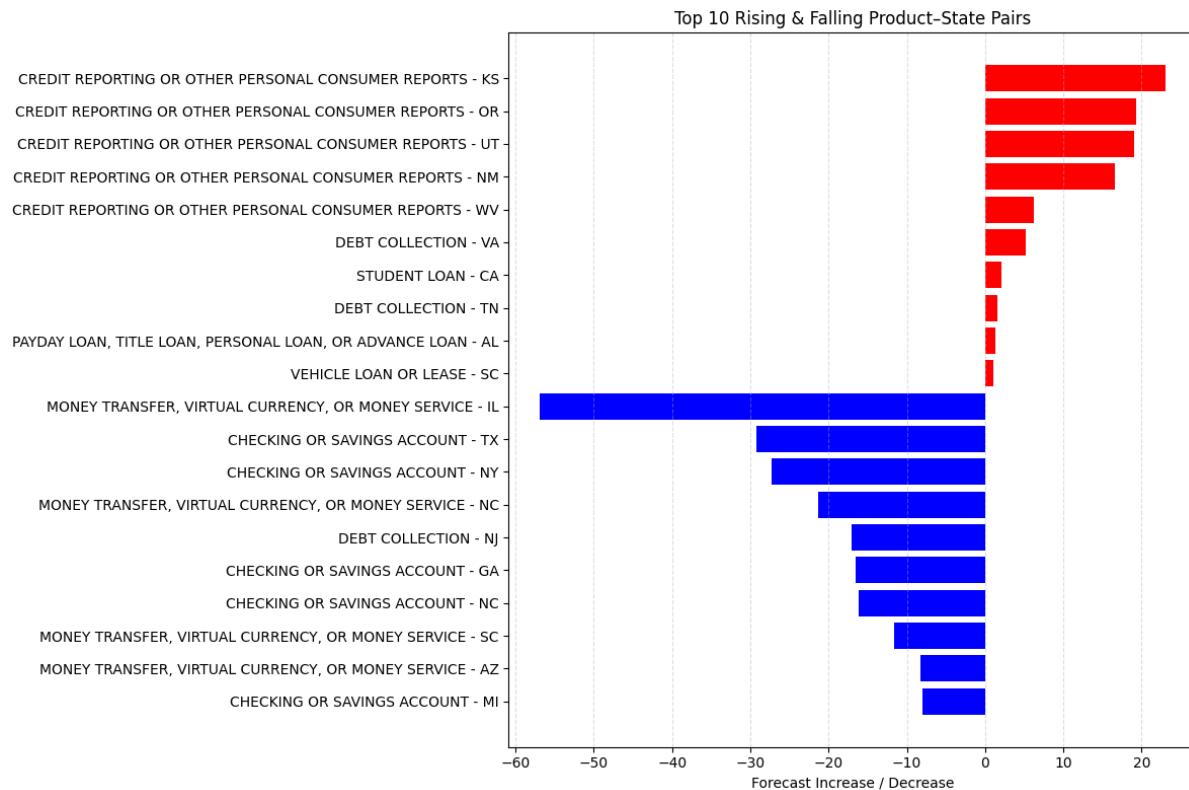
This selected product state pair shows slightly similar trend to the historical data

Chart 2: Top 10 Rising and Falling Product-State Pairs

Python code

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 # Load dataset
5 df = pd.read_csv("/content/drive/MyDrive/Data_Science_Projects/customer_complaint_trend_forecasting/data/all_set_for_visuals.csv")
6 df["ds"] = pd.to_datetime(df["ds"])
7 df["year"] = df["ds"].dt.year
8
9 # --- HISTORICAL MEAN (2024-2025) ---
10 hist = (
11     df[(df["actual_count"].notna()) & (df["year"].isin([2024, 2025]))]
12     .groupby(["Product", "State"])["actual_count"]
13     .mean()
14     .reset_index()
15     .rename(columns={"actual_count": "hist_mean"})
16 )
17
18 # --- FORECAST MEAN (2026) ---
19 fut = (
20     df[(df["forecast_count"].notna()) & (df["year"] == 2026)]
21     .groupby(["Product", "State"])["forecast_count"]
22     .mean()
23     .reset_index()
24     .rename(columns={"forecast_count": "future_mean"})
25 )
26
27 # --- MERGE ---
28 risk = pd.merge(hist, fut, on=["Product", "state"], how="inner")
29 risk["Risk_Score"] = risk["future_mean"] - risk["hist_mean"]
30
31 # --- Split top & bottom 10 ---
32 top10_risky = risk.sort_values("Risk_Score", ascending=False).head(10)
33 top10_safe = risk.sort_values("Risk_Score", ascending=True).head(10)
34
35 # --- Combine for plotting ---
36 viz = pd.concat([top10_risky, top10_safe], axis=0)
37
38 # --- Label column ---
39 viz["Label"] = viz["Product"] + " - " + viz["State"]
40
41 # --- Plot ---
42 plt.figure(figsize=(12, 8))
43 colors = viz["Risk_Score"].apply(lambda x: "red" if x > 0 else "blue")
44
45 plt.barch(viz["Label"], viz["Risk_Score"], color=colors)
46
47 plt.xlabel("Forecast Increase / Decrease")
48 plt.title("Top 10 Rising & Falling Product-State Pairs")
49 plt.gca().invert_yaxis() # Highest risk on top
50 plt.grid(axis='x', linestyle='--', alpha=0.4)
51
52 plt.tight_layout()
53 plt.show()
```

Output(Visualization chart)



Insights from the Ranking Chart

1. Clear Identification of Risky Hotspots:

The highest-risk product–state combinations (e.g., Credit reporting or other personal consumer reports – KS, Debt collection – VA) show a large positive jump in forecasted complaints compared to their historical levels. These pairs deserve priority monitoring.

2. Emerging Systemic Issues:

Many risky pairs belong to categories like Credit reporting or other personal consumer report

Their risk scores suggest underlying systemic issues in dispute handling or credit-related errors.

3. Improving Product–State Pairs:

The bottom part of the chart highlights combinations where complaint volume is expected to decline. These represent successful operational performance or improved customer experience.

4. Strategic Use:

This chart answers:

“Which product–state combinations require immediate attention, and which ones are improving?”

5. Business Value:

Helps allocate resources to genuinely risky areas.

Supports state-wise priority planning.

Enables product managers to track improvement impact.

Chart 3: Root Cause for Risky Pairs

Python code

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3
4  # Load data
5  df = pd.read_csv("/content/drive/MyDrive/Data_Science_Projects/Customer_Complaint_Trend_Forecasting/data/all_set_for_visuals.csv")
6
7  # Convert ds to datetime
8  df["ds"] = pd.to_datetime(df["ds"])
9
10 # -----
11 # ① Compute Historical Mean per pair
12 # -----
13 hist = (
14     df[df["actual_count"].notna()]
15     .groupby(["Product", "State"])["actual_count"]
16     .mean()
17     .reset_index()
18     .rename(columns={"actual_count": "historical_mean"})
19 )
20
21 # -----
22 # ② Compute Forecast Mean per pair
23 # -----
24 fc = (
25     df[df["forecast_count"].notna()]
26     .groupby(["Product", "State"])["forecast_count"]
27     .mean()
28     .reset_index()
29     .rename(columns={"forecast_count": "forecast_mean"})
30 )
31
32 # -----
33 # ③ Merge and compute ORIGINAL risk_score
34 # -----
35 risk = pd.merge(hist, fc, on=["Product", "State"], how="inner")
36 risk["risk_score"] = risk["forecast_mean"] - risk["historical_mean"]
37
```

```

38 # -----
39 # ❸ Pick Top 10 RISING pairs (highest positive score)
40 # -----
41 top10_risky = risk.nlargest(10, "risk_score")
42 print("\n== TOP 10 RISKY PAIRS (Original Formula) ==")
43 print(top10_risky)
44
45 # -----
46 # ❹ Extract only these pairs for root-cause visualization
47 #
48 pairs = list(zip(top10_risky["Product"], top10_risky["state"]))
49
50 root_df = df[
51     (df[["Product", "State"]].
52      .apply(tuple, axis=1).
53      .isin(pairs))
54 ]
55
56 # Use only historical rows (because forecast rows have null root-cause)
57 root_df = root_df[root_df["actual_count"].notna()]
58
59 # -----
60 # ❺ Group root causes
61 #
62 root_count = (
63     root_df.groupby([ "Product", "State", "topic_interpretation"]).
64     .size().
65     .reset_index(name="count")
66 )
67
68 # -----
69 # ❻ PLOT - Grouped Bar Chart of Root Causes for Top 10 Rising Pairs
70 #
71
72 plt.figure(figsize=(16, 9))
73
74 # Create a combined key for x-axis

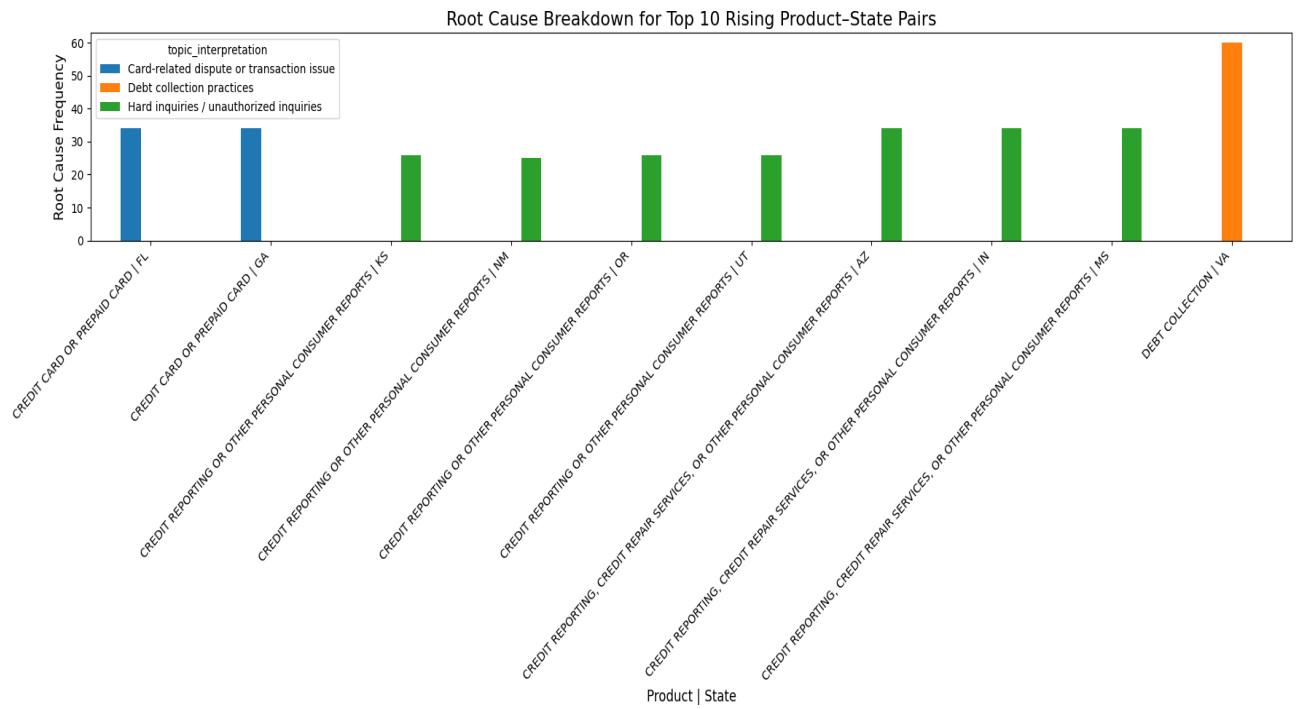
```

```

74 # Create a combined key for x-axis
75 root_count["Pair"] = root_count["Product"] + " | " + root_count["State"]
76
77 # Pivot to create grouped bars
78 pivot_df = root_count.pivot(index="Pair", columns="topic_interpretation", values="count").fillna(0)
79
80 pivot_df.plot(kind="bar", figsize=(18, 8))
81
82 plt.title("Root Cause Breakdown for Top 10 Rising Product-State Pairs", fontsize=16)
83 plt.xlabel("Product | State", fontsize=12)
84 plt.ylabel("Root Cause Frequency", fontsize=12)
85 plt.xticks(rotation=45, ha="right")
86 plt.tight_layout()
87 plt.show()

```

Output(Visualization Chart)



Insights from the Root Cause Breakdown

1. Dominant Root Causes Stand Out Clearly:

Complaint clusters such as:

Credit Card or Prepaid card ,Debt Collection, Credit reporting appear most frequently among rising pairs.

These issues are likely major contributors to the forecasted increase.

2. Category-Specific Problems Emerge:

Risky credit-related products (e.g., Credit Card or Prepaid card ,Debt Collection) show strong associations with disputes regarding:

Hard inquiries or transaction issue, Debt collection practice. This ties forecasted growth directly to known pain points.

3. Operational Value:

Helps prioritize which root causes to fix first in high-risk states, Supports root-cause–driven incident reduction planning, Allows regulators and product teams to link forecasted spikes to clear behavioral patterns.

4. Actionable Outcome:

This chart answers:

“What problems are driving the increase in complaints for the most risky product–state pairs?”

Chart 4: State-wise Historical Complaint

Python Code

```
1 import pandas as pd
2 import plotly.graph_objects as go
3
4 # Load your data
5 df = pd.read_csv("/content/drive/MyDrive/Data_Science_Projects/customer_complaint_Trend_Forecasting/data/all_set_for_visuals.csv")
6
7 # Keep only historical (actual_count)
8 hist = df[df["actual_count"].notna()]
9
10 # Aggregate
11 state_summary = (
12     hist.groupby("state")["actual_count"]
13     .sum()
14     .reset_index()
15     .rename(columns={"actual_count": "total_complaints"})
16 )
17
18 # State center coordinates (needed for labels)
19 state_centers = {
20     "AL": [32.806671, -86.791130], "AK": [61.370716, -152.404419],
21     "AZ": [33.729759, -111.431221], "AR": [34.969704, -92.373123],
22     "CA": [36.116203, -119.681564], "CO": [39.059811, -105.311104],
23     "CT": [41.597782, -72.755371], "DE": [38.897438, -77.026817],
24     "DE": [39.318523, -75.507141], "FL": [27.766279, -81.686783],
25     "GA": [33.040619, -83.643074], "HI": [21.094318, -157.498337],
26     "IA": [42.011539, -93.210526], "ID": [44.240459, -114.478828],
27     "IL": [40.349457, -88.986137], "IN": [39.849426, -86.258278],
28     "KS": [38.526600, -96.726486], "KY": [37.668140, -84.670067],
29     "LA": [31.169546, -91.867805], "MA": [42.230171, -71.530106],
30     "MD": [39.063946, -76.802101], "ME": [44.693947, -69.381927],
31     "MI": [43.326618, -84.536095], "MN": [45.694454, -93.900192],
32     "MO": [38.456085, -92.288368], "MS": [32.741646, -89.678696],
33     "MT": [46.921925, -110.454353], "NC": [35.630066, -79.806419],
34     "ND": [47.528912, -99.784012], "NE": [41.125370, -98.268082],
35     "NH": [43.452492, -71.563896], "NJ": [40.298904, -74.521011],
36     "NM": [34.840515, -106.248482], "NV": [38.313515, -117.055374],
```

```

44     "WI": [44.268543, -89.616508], "WV": [38.491226, -80.954456],  

45     "WY": [42.755966, -107.302490], "PR": [18.220833, -66.590149],  

46     "GU": [13.444304, 144.793732], "VI": [18.335764, -64.896335],  

47     "MP": [15.097900, 145.673900], "AS": [-14.270972, -170.132217]  

48   }  

49  

50 # Merge coordinates  

51 state_summary["lat"] = state_summary["State"].map(lambda x: state_centers.get(x, [None,None])[0])  

52 state_summary["lon"] = state_summary["State"].map(lambda x: state_centers.get(x, [None,None])[1])  

53  

54 # Create figure  

55 fig = go.Figure(data=go.Choropleth(  

56     locations=state_summary["State"],  

57     z=state_summary["total_complaints"],  

58     locationmode="USA-states",  

59     colorscale="Reds",  

60     colorbar_title="Complaints",  

61 ))  

62  

63 # Add labels  

64 fig.add_trace(go.Scattergeo(  

65     locationmode="USA-states",  

66     lon=state_summary["lon"],  

67     lat=state_summary["lat"],  

68     text=state_summary["total_complaints"].astype(int),  

69     mode="text",  

70     textfont=dict(size=10, color="black"),  

71     showlegend=False  

72 ))  

73  

74 fig.update_layout(  

75     title_text="Historical complaint volume by state (With Counts)",  

76     title_x=0.5,  

77     geo_scope="usa"  

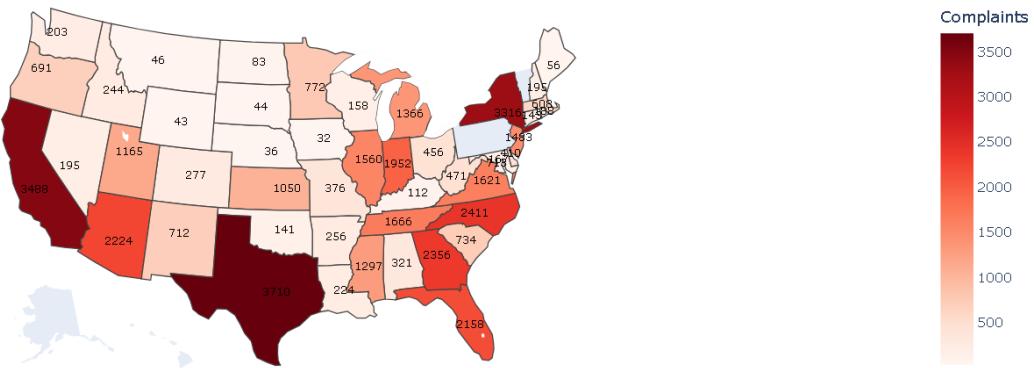
78 )  

79 fig.show()

```

Output(Visualisation Chart)

Historical Complaint Volume by State (With Counts)



States with low complaint counts may reflect either effective local operations or smaller customer footprints.

This map helps visually validate geographic imbalance in complaint distributions and can guide:

Prioritizing support resources

Monitoring high-volume regions

Identifying where targeted interventions may reduce future complaints