COLLEGE CODE : 1133

COLLEGE NAME : VELAMMAL INSTITUTE OF TECHNOLOGY

DEPARTMENT : ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

STUDENT NM-ID : aut113323aia48

ROLL NO : 113323243097

DATE : 03.05.2025

TECHNOLOGY-PROJECT NAME : PRODUCTION YIELD ANALYSIS

SUBMITTED BY :

1) **SANJAY S**
2) **SHESHANTH R**
3) **HARIHARA SUBRAMANIAN K**
4) **SHRIRAM S S**
5) **MUTHU KRISHNAN R**

INDEX OF THE DOCUMENTS

Phase 5: Project Demonstration & Documentation

Title: Production Yield Analysis

Abstract:
The *Production Yield Analysis* project focuses on enhancing quality monitoring in manufacturing by leveraging data analytics to track yield efficiency across production lines. This phase encapsulates the completion and implementation of the solution, which aggregates simulated production data and applies computational logic to visualize trends and deviations. The system allows stakeholders to assess line-wise performance and supports early detection of quality drops. This documentation outlines the demonstration, final system features, revisions made, and project handover.

---

1. Project Demonstration

Overview:

The completed system was presented to highlight its role in yield tracking and defect reduction. It was designed to evaluate day-to-day operations across multiple lines in a simulated industrial environment.

Demonstration Details:

- **Live Metrics Panel:** Displayed key production indicators like total units, defect count, and yield rate.
- **Line Comparison Dashboard:** Showed how each production line performed via distinct visual formats.
- **Graphical Output:** Yield fluctuations were shown using trend charts and density maps.
- **Interactive Reports:** Enabled real-time inspection of production logs and calculated summaries.
- **Accuracy Verification:** Yield percentages were validated against sample data for consistency.

Outcome:

The demonstration proved the solution's effectiveness in visualizing production patterns and providing actionable insights.

---

2. Project Documentation

Overview:

Comprehensive documentation was compiled covering both functional behavior and backend logic, ensuring clarity for future maintenance or extension.

Documentation Sections:

- **Workflow Blueprint:** Illustrated how raw production data flows into analysis and visualization modules.

- **Script Structure:** Explained core Python modules responsible for data aggregation and chart generation.
- **User Instructions:** Described steps to run the system and interpret output results for operational staff.
- **Technical Guide:** Provided references for customizing system parameters or updating line configurations.
- **Verification Notes:** Captured outcomes from various tests ensuring accurate data handling and presentation.

Outcome:
The structured documentation provides transparency in design and facilitates ease of use and training.

---

3. Feedback and Final Adjustments
Overview:
Following the prototype walkthrough, reviewers shared suggestions that guided the final system enhancements.
Steps:

- **Feedback Compilation:** Gathered input from users regarding interface clarity and output usability.
- **Refinements:** Adjusted visual color schemes, improved date labels, and optimized the responsiveness of charts.
- **Extended Testing:** Validated enhancements using alternative production data to ensure robustness.

Outcome:
Final system tuning based on feedback led to better usability and a cleaner, more readable dashboard.

---

4. Final Project Report Submission
Overview:
The final submission encapsulated the design journey, code architecture, and performance evaluation of the system.
Report Sections:

- **Project Objective:** Reiterated the aim to analyze yield and support quality assurance through data analysis.
- **Implementation Stages:** Detailed development from simulation logic to final UI visualization.
- **Issue Handling:** Described how issues like inconsistent data and low-frequency spikes were tackled.
- **Result Analysis:** Discussed insights like defect patterns and line-wise performance comparison.

Outcome:
The finalized report serves as a complete reference for evaluating the system's capabilities and potential value.

---

5. Handover of the Project and Future Development

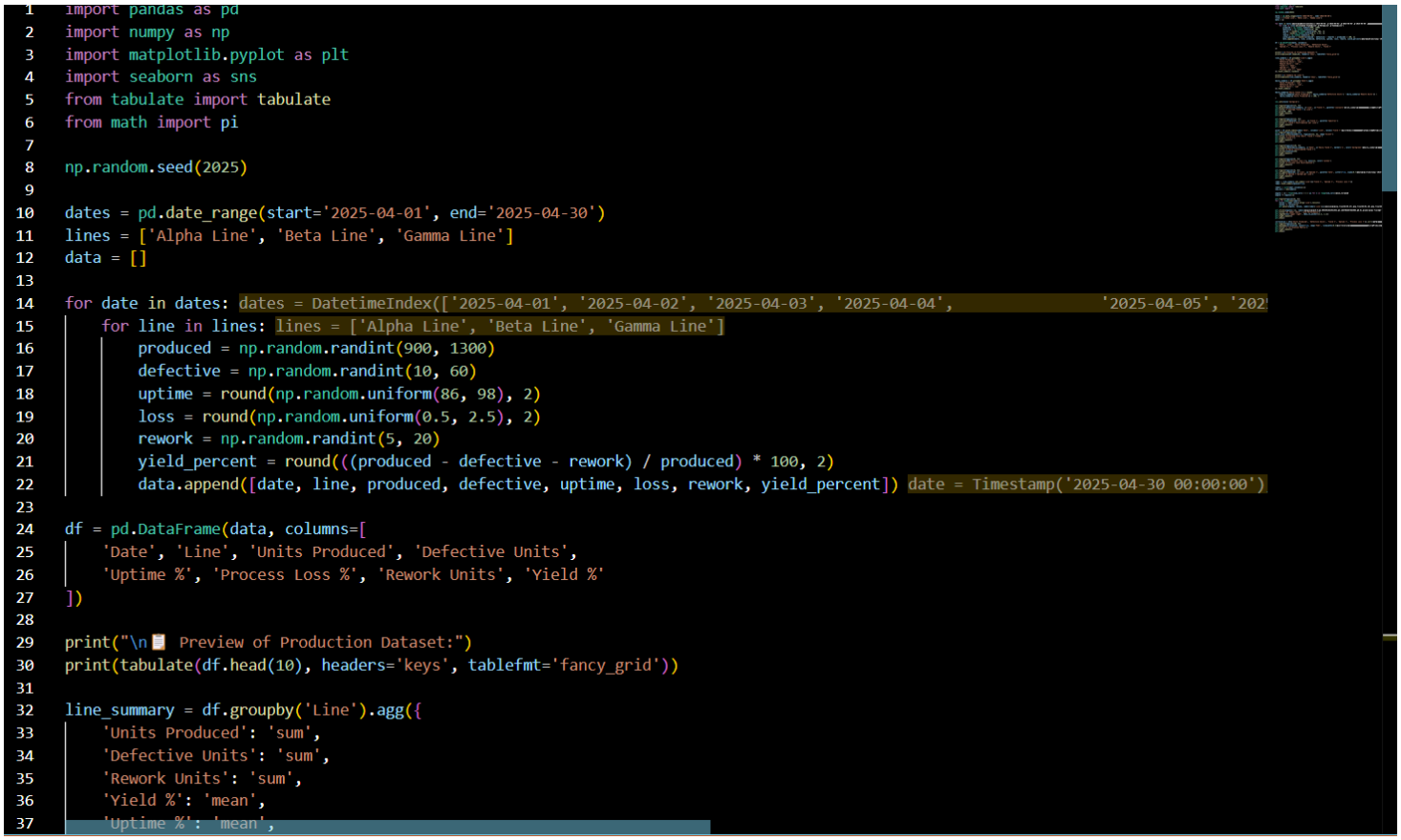
Overview:
The project has reached its handover stage, equipped for deployment or further development as required.

Handover Details:

- **Enhancement Suggestions:** Recommend incorporating anomaly detection, SMS/email alerts, and real-time sensors.
- **Expansion Readiness:** The system's flexible structure allows easy scaling to include more production units or plants.

Outcome:
The project is ready for real-world adaptation and includes a roadmap for future improvement.

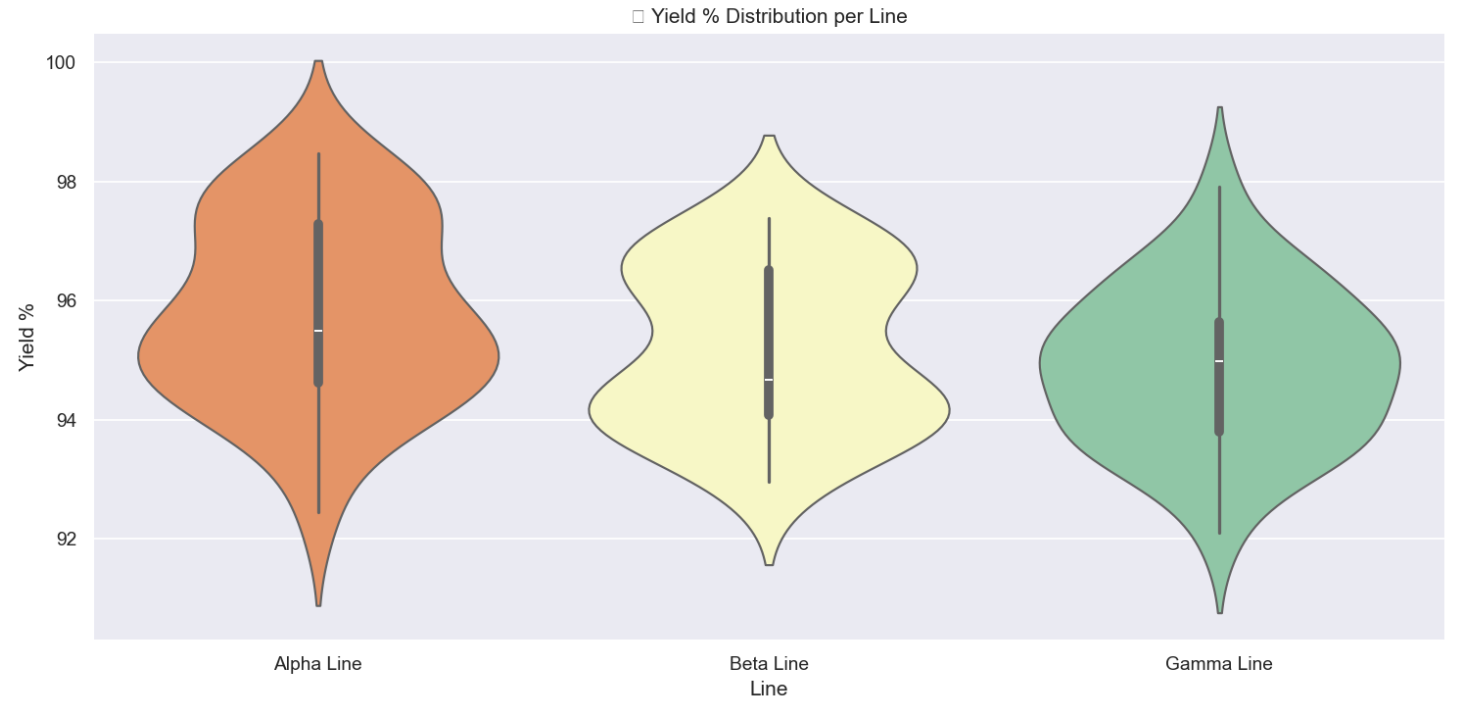
Screenshots, Code & Progress of the Project



```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from tabulate import tabulate
from math import pi

np.random.seed(2025)

dates = pd.date_range(start='2025-04-01', end='2025-04-30')
lines = ['Alpha Line', 'Beta Line', 'Gamma Line']
data = []

for date in dates: dates = DatetimeIndex(['2025-04-01', '2025-04-02', '2025-04-03', '2025-04-04',                '2025-04-05', '202
    for line in lines: lines = ['Alpha Line', 'Beta Line', 'Gamma Line']
        produced = np.random.randint(900, 1300)
        defective = np.random.randint(10, 60)
        uptime = round(np.random.uniform(86, 98), 2)
        loss = round(np.random.uniform(0.5, 2.5), 2)
        rework = np.random.randint(5, 20)
        yield_percent = round(((produced - defective - rework) / produced) * 100, 2)
        data.append([date, line, produced, defective, uptime, loss, rework, yield_percent]) date = Timestamp('2025-04-30 00:00:00')

df = pd.DataFrame(data, columns=[
    'Date', 'Line', 'Units Produced', 'Defective Units',
    'Uptime %', 'Process Loss %', 'Rework Units', 'Yield %'
])

print("\n Preview of Production Dataset:")
print(tabulate(df.head(10), headers='keys', tablefmt='fancy_grid'))

line_summary = df.groupby('Line').agg({
    'Units Produced': 'sum',
    'Defective Units': 'sum',
    'Rework Units': 'sum',
    'Yield %': 'mean',
    'Uptime %': 'mean',
```

```
    'Process Loss %': 'mean'
}).reset_index().round(2)

print("\n📊 Summary by Line:")
print(tabulate(line_summary, headers='keys', tablefmt='fancy_grid'))

daily_summary = df.groupby('Date').agg({
    'Units Produced': 'sum',
    'Defective Units': 'sum',
    'Rework Units': 'sum'
}).reset_index()

daily_summary['Daily Yield %'] = round(
    ((daily_summary['Units Produced'] - daily_summary['Defective Units'] - daily_summary['Rework Units']) /
     daily_summary['Units Produced']) * 100, 2
)

sns.set(style='darkgrid')

plt.figure(figsize=(12, 6))
sns.barplot(data=line_summary, x='Line', y='Yield %', palette='coolwarm') line_summary =          Line  Units Produced  Defective U
plt.title('📦 Average Yield % by Line')
plt.ylim(85, 100)
plt.tight_layout()
plt.show()

plt.figure(figsize=(12, 6))
sns.violinplot(data=df, x='Line', y='Yield %', palette='Spectral')
plt.title('🎻 Yield % Distribution per Line')
plt.tight_layout()
plt.show()

pivot = df.pivot_table(index='Date', columns='Line', values='Yield %') pivot = Line        Alpha Line  Beta Line  Gamma Line Date
plt.figure(figsize=(12, 6))
pivot.plot.area(stacked=True, figsize=(12, 6), cmap='Accent')
plt.title('🌈 Stacked Area Chart: Yield % Trends')
plt.ylabel('Yield %')
plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 5))
sns.lineplot(data=daily_summary, x='Date', y='Daily Yield %', marker='o', color='darkgreen') daily_summary =          Date  Units P
plt.title('📈 Daily Consolidated Yield %')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

plt.figure(figsize=(8, 5))
sns.histplot(df['Process Loss %'], bins=12, color='salmon')
plt.title('📉 Process Loss Distribution')
plt.tight_layout()
plt.show()

plt.figure(figsize=(8, 6))
sns.stripplot(data=df, x='Line', y='Uptime %', palette='Set2', jitter=True, alpha=0.7) data = [[Timestamp('2025-04-01 00:00:00'), '
plt.title('🔧 Uptime % Spread per Line')
plt.tight_layout()
plt.show()

radar = line_summary.set_index('Line')[['Yield %', 'Uptime %', 'Process Loss %']]
radar.reset_index(inplace=True)

labels = list(radar.columns[1:])
num_vars = len(labels)

angles = [n / float(num_vars) * 2 * pi for n in range(num_vars)] num_vars = 3
angles += angles[:1]

plt.figure(figsize=(8, 8))
for i in range(len(radar)):
    values = radar.loc[i].drop('Line').tolist()
    values += values[:1]
    plt.polar(angles, values, label=radar['Line'][i]) values = [np.float64(94.87), np.float64(92.26), np.float64(1.72), np.float64(
```


Figure 1
Average Yield % by Line
100
98
96
94
92
90
88
86
Yield %
Alpha Line
Beta Line
Gamma Line
Line
(x, y) = (Gamma Line, 96.37)


Yield % Distribution per Line
100
98
96
94
92
Yield %
Alpha Line
Beta Line
Gamma Line
Line

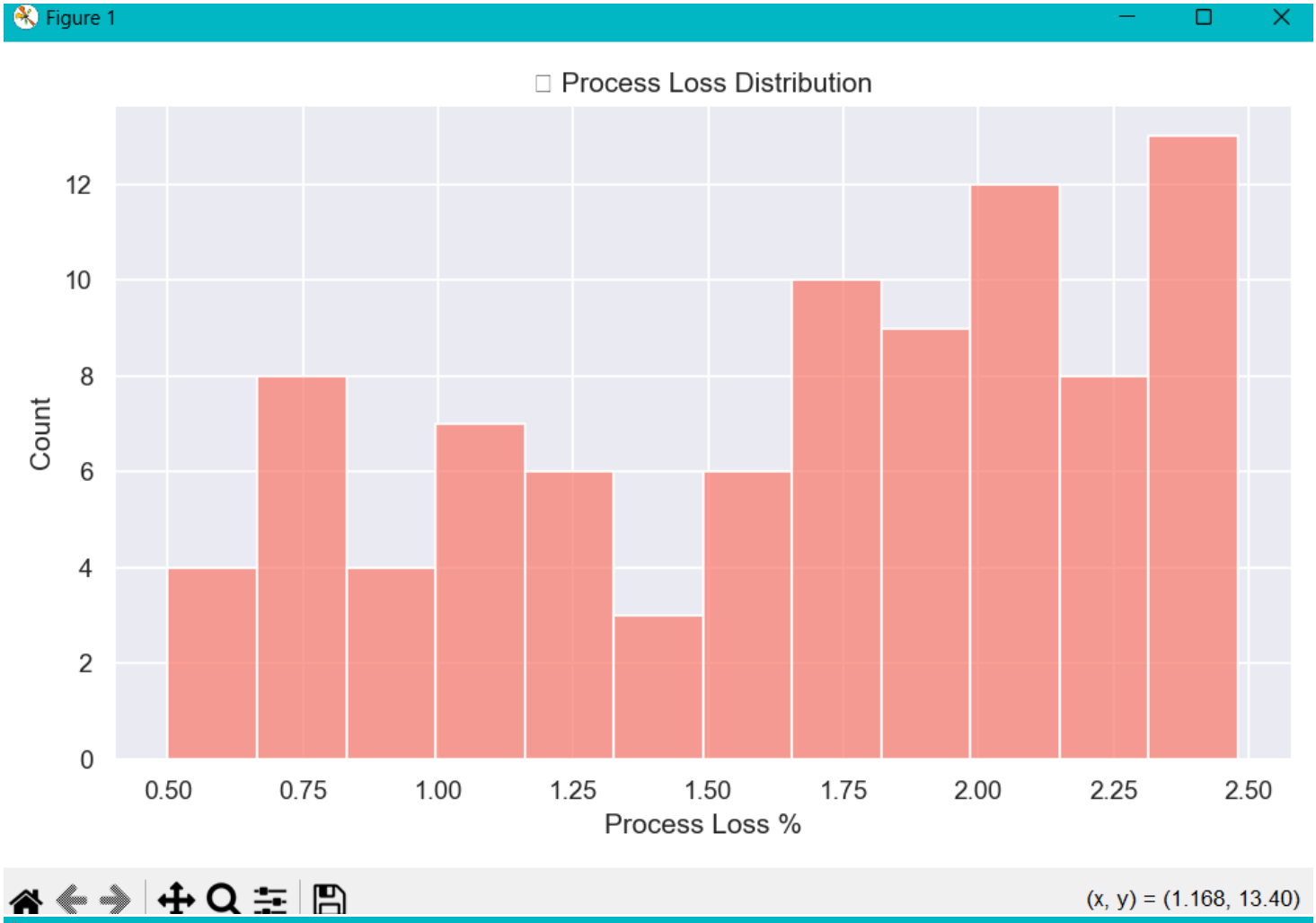


Figure 1
Process Loss Distribution
Count
0
2
4
6
8
10
12
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25
2.50
Process Loss %
(x, y) = (1.168, 13.40)

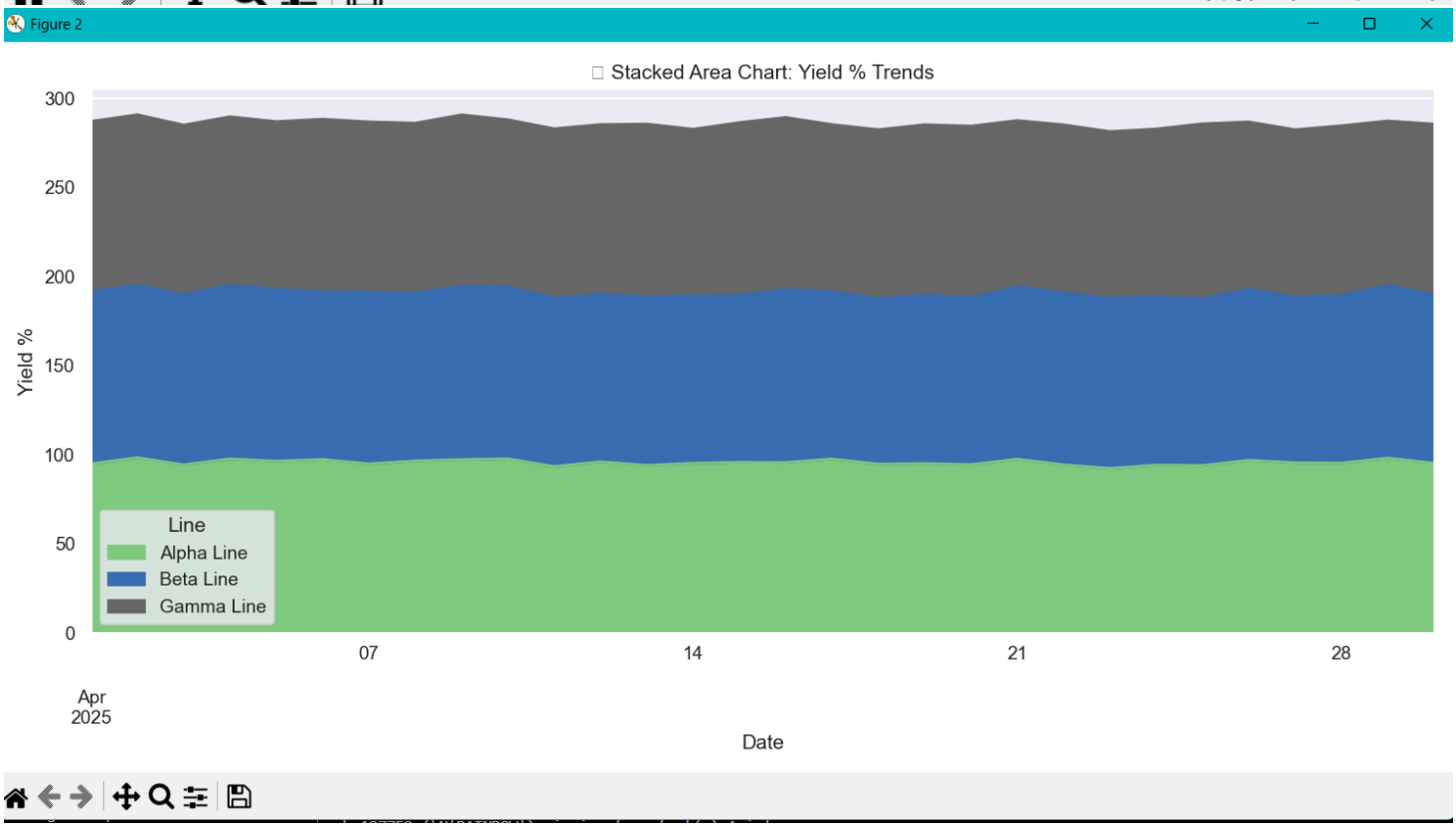


Figure 2
Stacked Area Chart: Yield % Trends
Yield %
0
50
100
150
200
250
300
Line
Alpha Line
Beta Line
Gamma Line
07
14
21
28
Apr
2025
Date