Structured Data Assignment

Thadimudupula Sahith

GUVI

001

Problem Statement

Problem Statement: "Analyzing Prescription Patterns for 'Target Drug'"

The goal is to identify and analyze the dominant prescription patterns for the 'Target Drug' administered to patients. This involves discovering recurring intervals or sequences in which the drug is prescribed.

Problem

The problem is to analyze the prescription patterns of the 'Target Drug' to understand the intervals or sequences at which it is administered to patients.

Objective

To identify and analyze dominant prescription patterns for the 'Target Drug', providing insights for effective drug administration strategies.

Specific Goals:

Identify recurring prescription patterns.

Visualize prescription patterns over time.

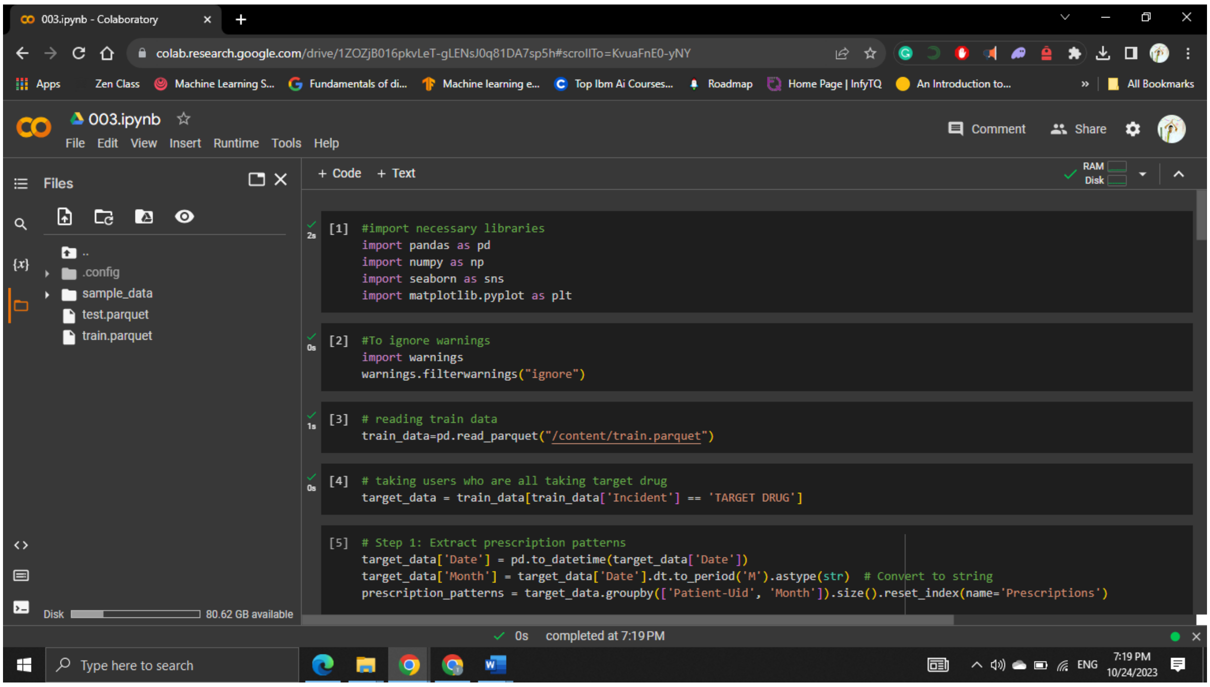
Extract insights for optimizing drug administration.

Potential Applications

Healthcare Optimization: Tailoring drug administration schedules for better patient outcomes.

Clinical Protocol Optimization: Enhancing treatment protocols based on identified patterns.

Patient Adherence Improvement: Designing strategies to improve patient compliance.



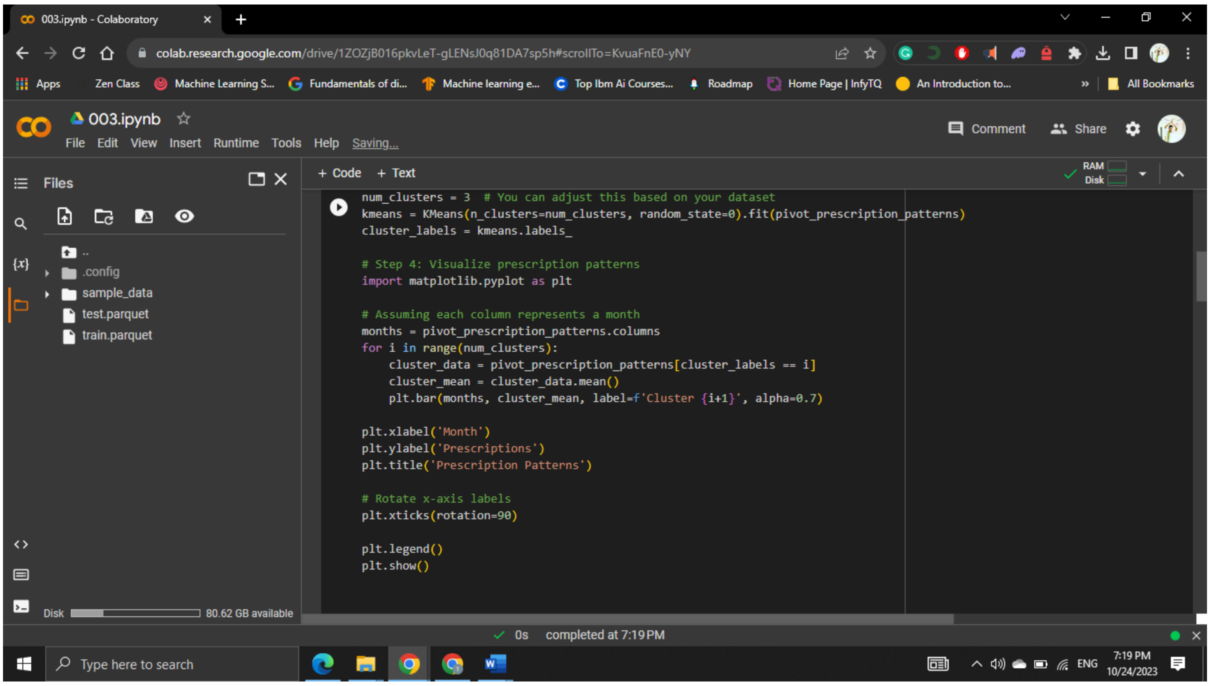
```
[1] #import necessary libraries
    import pandas as pd
    import numpy as np
    import seaborn as sns
    import matplotlib.pyplot as plt
```

```
[2] #To ignore warnings
    import warnings
    warnings.filterwarnings("ignore")
```

```
[3] # reading train data
    train_data=pd.read_parquet("/content/train.parquet")
```

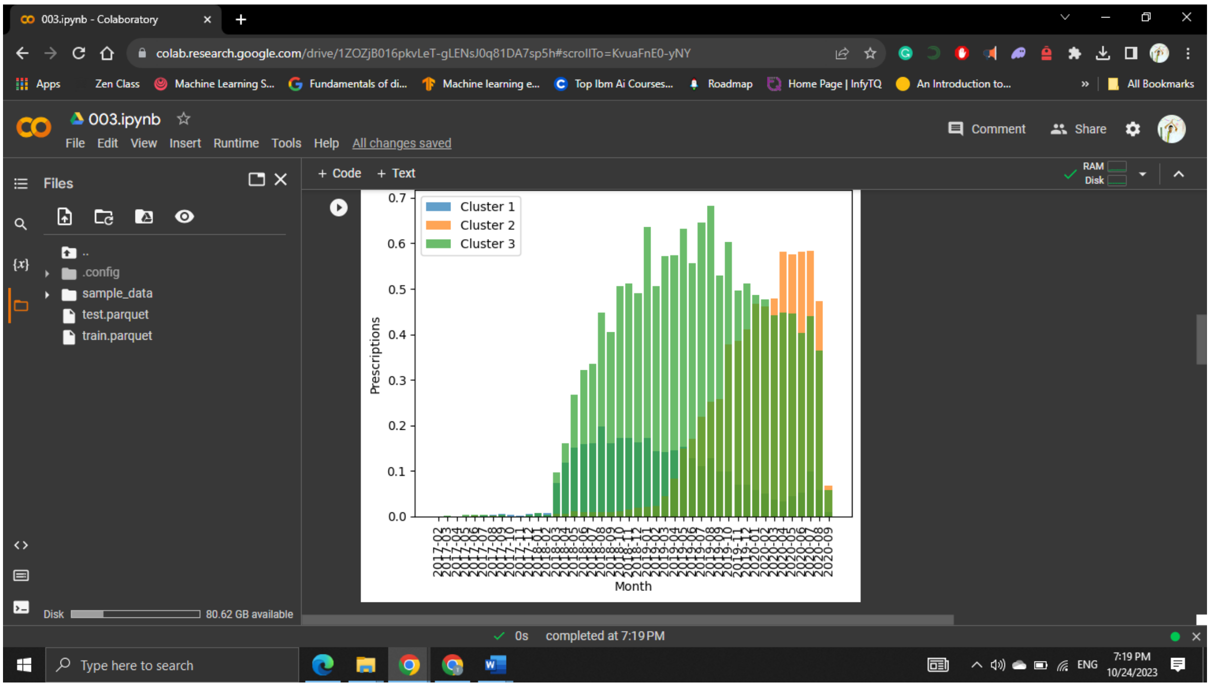
```
[4] # taking users who are all taking target drug
    target_data = train_data[train_data['Incident'] == 'TARGET DRUG']
```

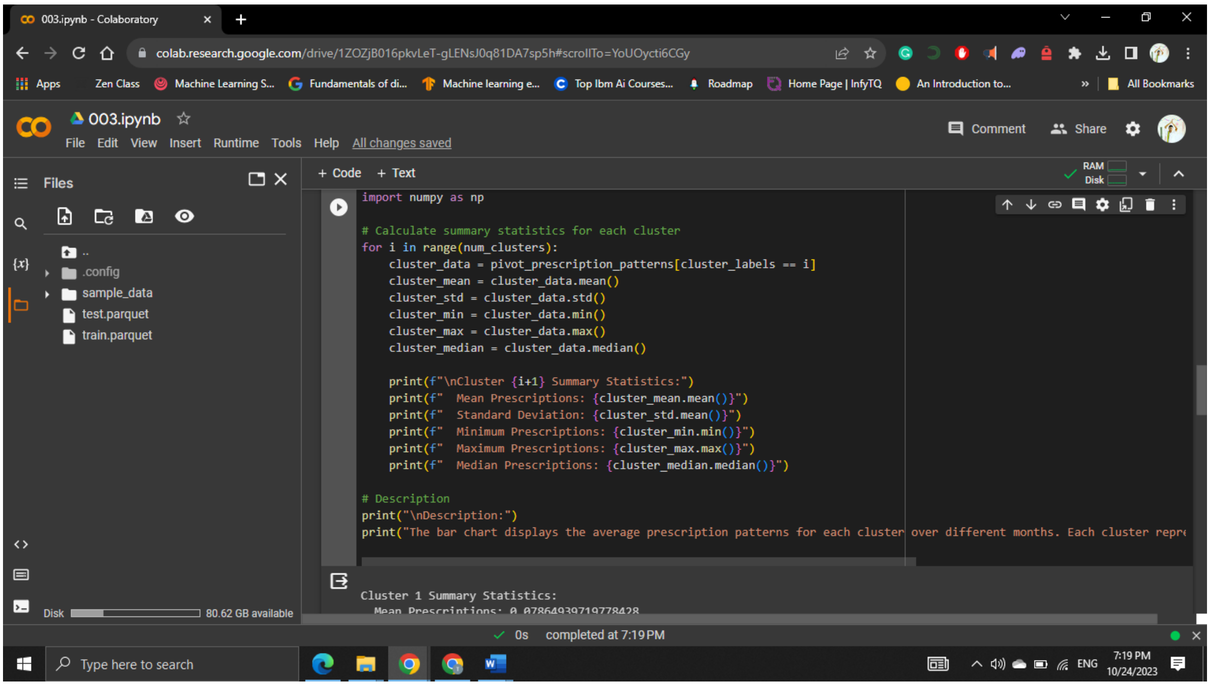
```
[5] # Step 1: Extract prescription patterns
    target_data['Date'] = pd.to_datetime(target_data['Date'])
    target_data['Month'] = target_data['Date'].dt.to_period('M').astype(str)  # Convert to string
    prescription_patterns = target_data.groupby(['Patient-Uid', 'Month']).size().reset_index(name='Prescriptions')
```



```
num_clusters = 3  # You can adjust this based on your dataset
kmeans = KMeans(n_clusters=num_clusters, random_state=0).fit(pivot_prescription_patterns)
cluster_labels = kmeans.labels_

# Step 4: Visualize prescription patterns
import matplotlib.pyplot as plt

# Assuming each column represents a month
months = pivot_prescription_patterns.columns
for i in range(num_clusters):
    cluster_data = pivot_prescription_patterns[cluster_labels == i]
    cluster_mean = cluster_data.mean()
    plt.bar(months, cluster_mean, label=f'Cluster {i+1}', alpha=0.7)

plt.xlabel('Month')
plt.ylabel('Prescriptions')
plt.title('Prescription Patterns')

# Rotate x-axis labels
plt.xticks(rotation=90)

plt.legend()
plt.show()
```

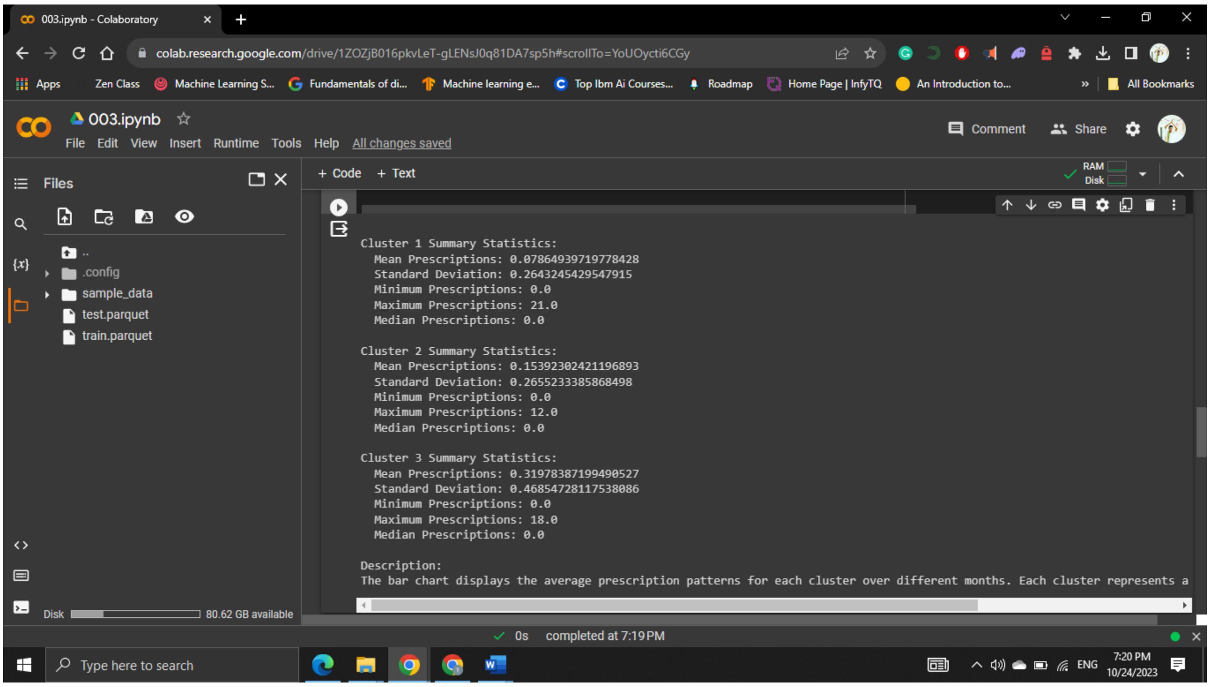
✓ 0s completed at 7:19 PM





```python
import numpy as np

# Calculate summary statistics for each cluster
for i in range(num_clusters):
    cluster_data = pivot_prescription_patterns[cluster_labels == i]
    cluster_mean = cluster_data.mean()
    cluster_std = cluster_data.std()
    cluster_min = cluster_data.min()
    cluster_max = cluster_data.max()
    cluster_median = cluster_data.median()

    print(f"\nCluster {i+1} Summary Statistics:")
    print(f"  Mean Prescriptions: {cluster_mean.mean()}")
    print(f"  Standard Deviation: {cluster_std.mean()}")
    print(f"  Minimum Prescriptions: {cluster_min.min()}")
    print(f"  Maximum Prescriptions: {cluster_max.max()}")
    print(f"  Median Prescriptions: {cluster_median.median()}")

# Description
print("\nDescription:")
print("The bar chart displays the average prescription patterns for each cluster over different months. Each cluster repre
```

```
Cluster 1 Summary Statistics:
  Mean Prescriptions: 0.07864939719778428
```

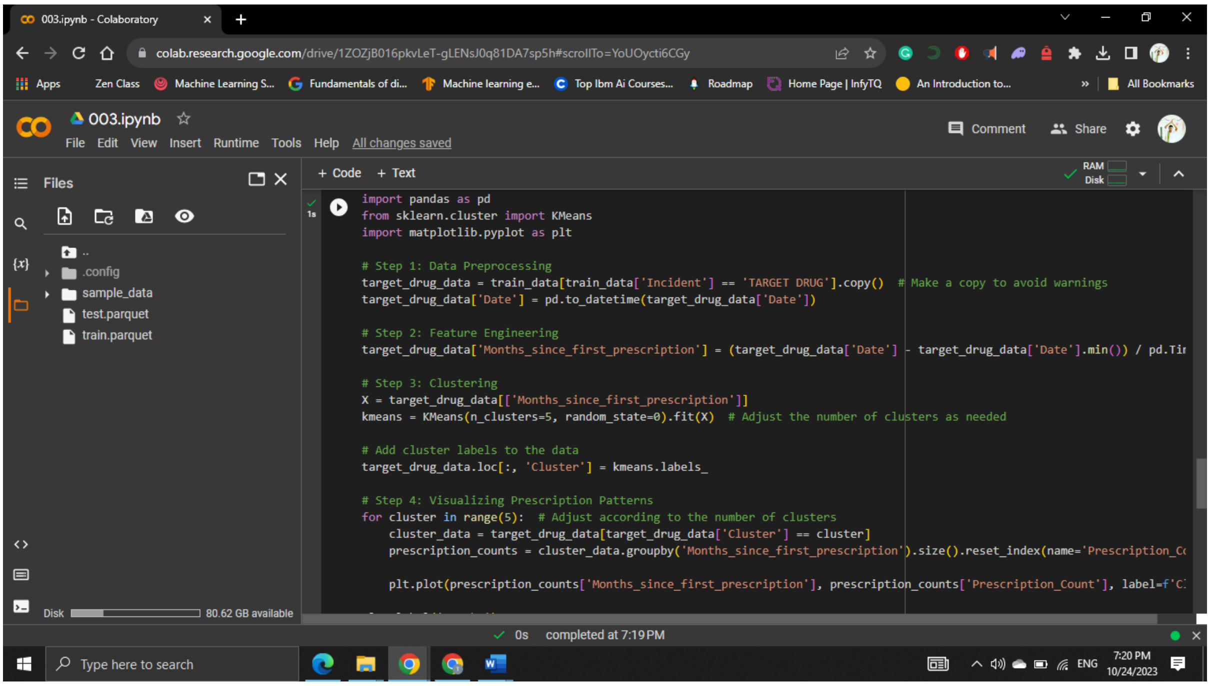


```
Cluster 1 Summary Statistics:
  Mean Prescriptions: 0.07864939719778428
  Standard Deviation: 0.2643245429547915
  Minimum Prescriptions: 0.0
  Maximum Prescriptions: 21.0
  Median Prescriptions: 0.0

Cluster 2 Summary Statistics:
  Mean Prescriptions: 0.15392302421196893
  Standard Deviation: 0.2655233385868498
  Minimum Prescriptions: 0.0
  Maximum Prescriptions: 12.0
  Median Prescriptions: 0.0

Cluster 3 Summary Statistics:
  Mean Prescriptions: 0.31978387199490527
  Standard Deviation: 0.46854728117538086
  Minimum Prescriptions: 0.0
  Maximum Prescriptions: 18.0
  Median Prescriptions: 0.0

Description:
The bar chart displays the average prescription patterns for each cluster over different months. Each cluster represents a
```

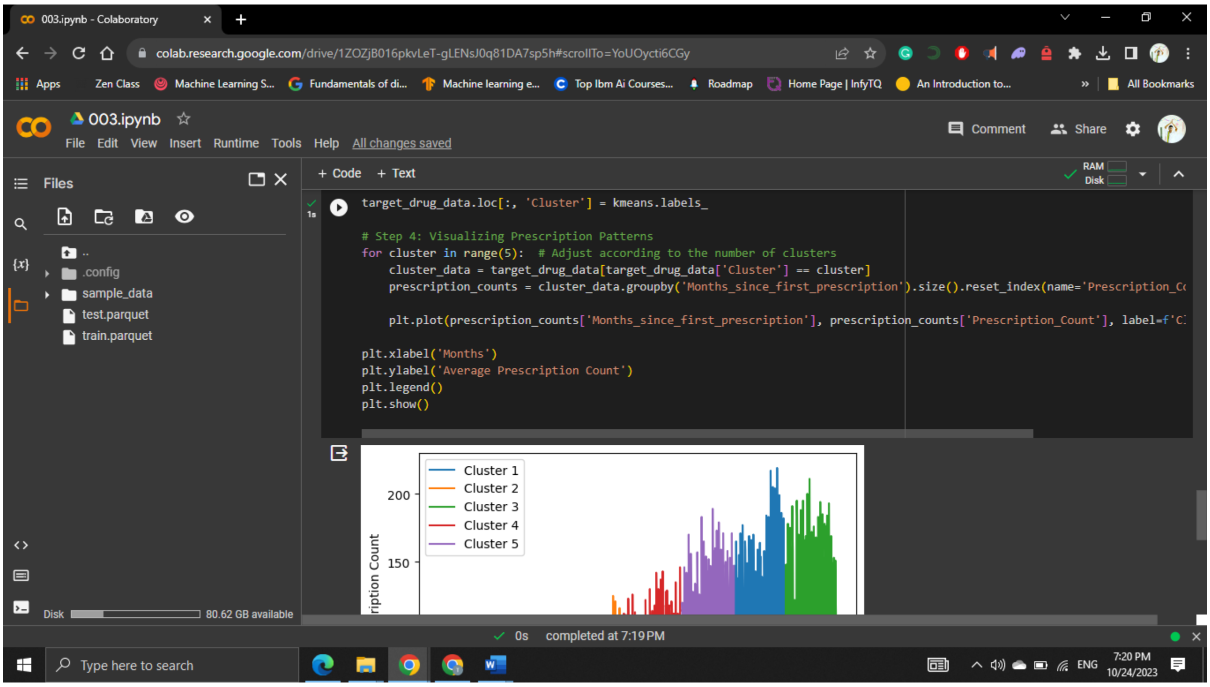
Cluster 1: This cluster has a low average number of prescriptions, with a mean of 0.08. The standard deviation is relatively high, indicating some variability. The maximum number of prescriptions in a month for this cluster is 21, while the minimum is 0.

Cluster 2: This cluster shows a slightly higher average number of prescriptions compared to Cluster 1, with a mean of 0.15. The standard deviation is similar to Cluster 1. The maximum number of prescriptions in a month for this cluster is 12, while the minimum is 0.

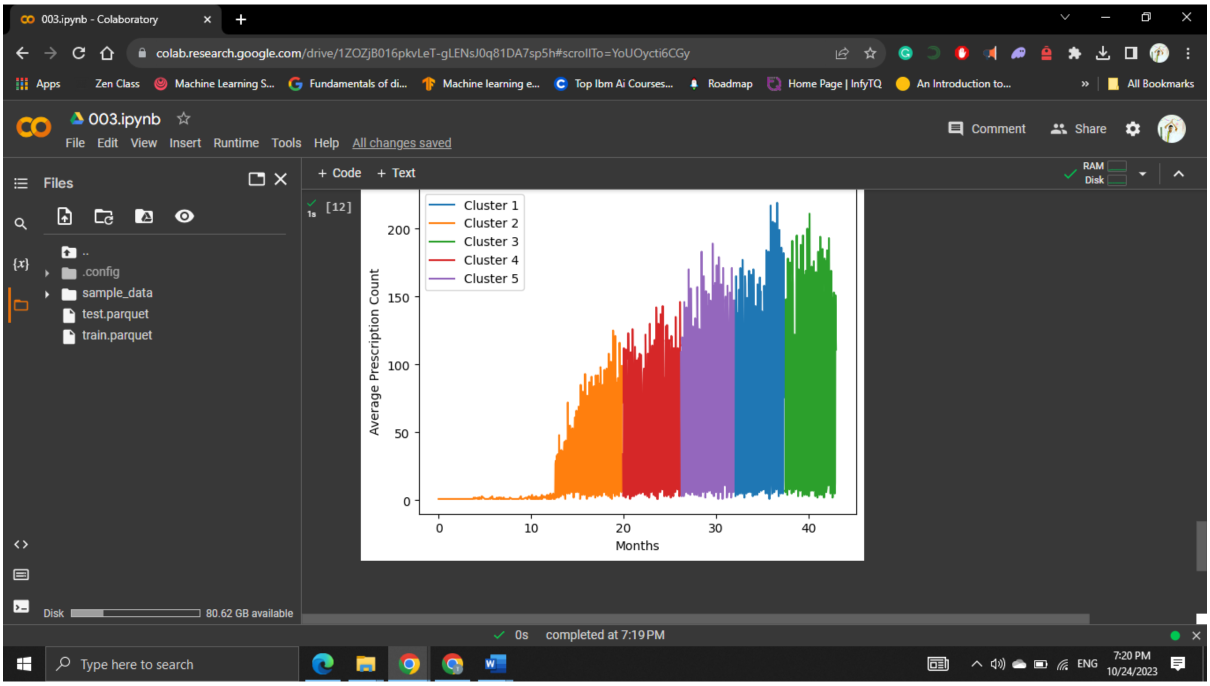
Cluster 3: This cluster has the highest average number of prescriptions, with a mean of 0.32. It also has the highest standard deviation, indicating greater variability. The maximum number of prescriptions in a month for this cluster is 18, while the minimum is 0.



```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Step 1: Data Preprocessing
target_drug_data = train_data[train_data['Incident'] == 'TARGET DRUG'].copy()  # Make a copy to avoid warnings
target_drug_data['Date'] = pd.to_datetime(target_drug_data['Date'])

# Step 2: Feature Engineering
target_drug_data['Months_since_first_prescription'] = (target_drug_data['Date'] - target_drug_data['Date'].min()) / pd.Ti

# Step 3: Clustering
X = target_drug_data[['Months_since_first_prescription']]
kmeans = KMeans(n_clusters=5, random_state=0).fit(X)  # Adjust the number of clusters as needed

# Add cluster labels to the data
target_drug_data.loc[:, 'Cluster'] = kmeans.labels_

# Step 4: Visualizing Prescription Patterns
for cluster in range(5):  # Adjust according to the number of clusters
    cluster_data = target_drug_data[target_drug_data['Cluster'] == cluster]
    prescription_counts = cluster_data.groupby('Months_since_first_prescription').size().reset_index(name='Prescription_C

    plt.plot(prescription_counts['Months_since_first_prescription'], prescription_counts['Prescription_Count'], label=f'C
```



```
target_drug_data.loc[:, 'Cluster'] = kmeans.labels_

# Step 4: Visualizing Prescription Patterns
for cluster in range(5):  # Adjust according to the number of clusters
    cluster_data = target_drug_data[target_drug_data['Cluster'] == cluster]
    prescription_counts = cluster_data.groupby('Months_since_first_prescription').size().reset_index(name='Prescription_C

    plt.plot(prescription_counts['Months_since_first_prescription'], prescription_counts['Prescription_Count'], label=f'C

plt.xlabel('Months')
plt.ylabel('Average Prescription Count')
plt.legend()
plt.show()
```



0s completed at 7:19 PM