San Jose Illegal Dumping Analysis

Cloud Data Mining Project

Shanmukha Manoj Kakani

018195645

shanmukhamanoj.kakani@sjsu.edu

San Jose State University

Data 255: Data Mining

Spring 2025

Contents

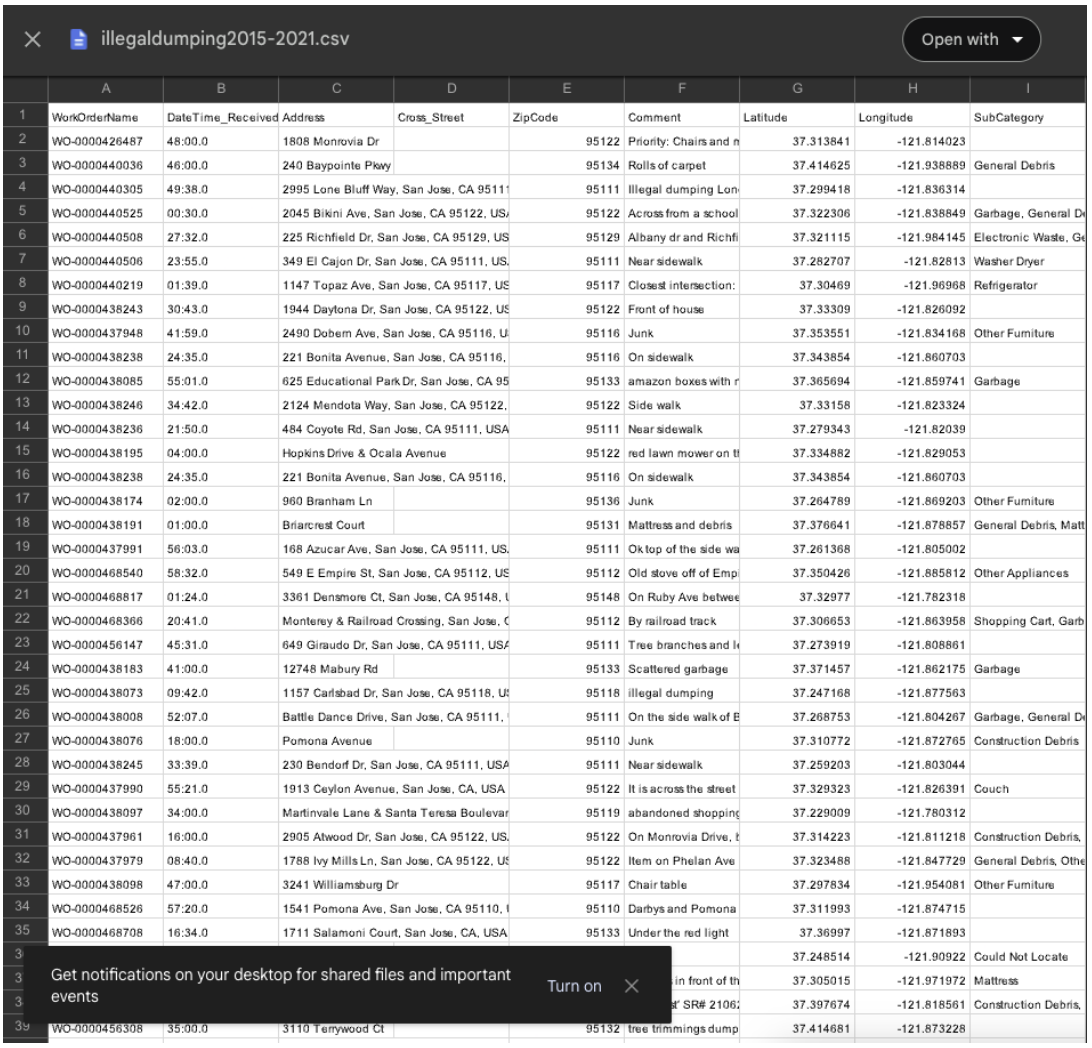
1 Introduction

This project analyzes illegal dumping incidents in San Jose, California, using cloud-based data mining techniques. The goal is to identify patterns, hotspots, and temporal trends in illegal dumping to help city officials develop targeted intervention strategies.

1.1 Dataset Description

The dataset contains records of illegal dumping incidents reported in San Jose, including:

- Location information (address, coordinates, zip code)
- Time of incident
- Type of material dumped
- Additional comments and details



	A	B	C	D	E	F	G	H	I
1	WorkOrderName	DateTime_Received	Address	Cross_Street	ZipCode	Comment	Latitude	Longitude	SubCategory
2	WO-0000426487	48:00.0	1808 Monrovia Dr		95122	Priority: Chairs and n	37.313841	-121.814023	
3	WO-0000440036	46:00.0	240 Baypointe Pkwy		95134	Rolls of carpet	37.414625	-121.938889	General Debris
4	WO-0000440305	49:38.0	2995 Lone Bluff Way, San Jose, CA 9511		95111	Illegal dumping Lon	37.299418	-121.836314	
5	WO-0000440525	00:30.0	2045 Bikini Ave, San Jose, CA 95122, USA		95122	Across from a school	37.322306	-121.838849	Garbage, General De
6	WO-0000440508	27:32.0	225 Richfield Dr, San Jose, CA 95129, US		95129	Albany dr and Richfi	37.321115	-121.984145	Electronic Waste, Ge
7	WO-0000440506	23:55.0	349 El Cajon Dr, San Jose, CA 95111, US		95111	Near sidewalk	37.282707	-121.82813	Washer Dryer
8	WO-0000440219	01:39.0	1147 Topaz Ave, San Jose, CA 95117, US		95117	Closest intersection:	37.30469	-121.96968	Refrigerator
9	WO-0000438243	30:43.0	1944 Daytona Dr, San Jose, CA 95122, US		95122	Front of house	37.33309	-121.826092	
10	WO-0000437948	41:59.0	2490 Dobern Ave, San Jose, CA 95116, U		95116	Junk	37.353551	-121.834168	Other Furniture
11	WO-0000438238	24:35.0	221 Bonita Avenue, San Jose, CA 95116,		95116	On sidewalk	37.343854	-121.860703	
12	WO-0000438085	55:01.0	625 Educational Park Dr, San Jose, CA 95		95133	amazon boxes with r	37.365694	-121.859741	Garbage
13	WO-0000438246	34:42.0	2124 Mendota Way, San Jose, CA 95122,		95122	Side walk	37.33158	-121.823324	
14	WO-0000438236	21:50.0	484 Coyote Rd, San Jose, CA 95111, USA		95111	Near sidewalk	37.279343	-121.82039	
15	WO-0000438195	04:00.0	Hopkins Drive & Ocala Avenue		95122	red lawn mower on th	37.334882	-121.829053	
16	WO-0000438238	24:35.0	221 Bonita Avenue, San Jose, CA 95116,		95116	On sidewalk	37.343854	-121.860703	
17	WO-0000438174	02:00.0	960 Branham Ln		95136	Junk	37.264789	-121.869203	Other Furniture
18	WO-0000438191	01:00.0	Briarcrest Court		95131	Mattress and debris	37.376641	-121.878857	General Debris, Matt
19	WO-0000437991	56:03.0	168 Azucar Ave, San Jose, CA 95111, US		95111	Ok top of the side wa	37.261368	-121.805002	
20	WO-0000468540	58:32.0	549 E Empire St, San Jose, CA 95112, US		95112	Old stove off of Empi	37.350426	-121.885812	Other Appliances
21	WO-0000468817	01:24.0	3361 Densmore Ct, San Jose, CA 95148, U		95148	On Ruby Ave betwee	37.32977	-121.782318	
22	WO-0000468366	20:41.0	Monterey & Railroad Crossing, San Jose, C		95112	By railroad track	37.306653	-121.863958	Shopping Cart, Garb
23	WO-0000456147	45:31.0	649 Giraudo Dr, San Jose, CA 95111, USA		95111	Tree branches and le	37.273919	-121.808861	
24	WO-0000438183	41:00.0	12748 Mabury Rd		95133	Scattered garbage	37.371457	-121.862175	Garbage
25	WO-0000438073	09:42.0	1157 Carlsbad Dr, San Jose, CA 95118, US		95118	illegal dumping	37.247168	-121.877563	
26	WO-0000438008	52:07.0	Battle Dance Drive, San Jose, CA 95111,		95111	On the side walk of B	37.268753	-121.804267	Garbage, General De
27	WO-0000438076	18:00.0	Pomona Avenue		95110	Junk	37.310772	-121.872765	Construction Debris
28	WO-0000438245	33:39.0	230 Bendorf Dr, San Jose, CA 95111, USA		95111	Near sidewalk	37.259203	-121.803044	
29	WO-0000437990	55:21.0	1913 Ceylon Avenue, San Jose, CA, USA		95122	It is across the street	37.329323	-121.826391	Couch
30	WO-0000438097	34:00.0	Martinvale Lane & Santa Teresa Boulevar		95119	abandoned shopping	37.229009	-121.780312	
31	WO-0000437961	16:00.0	2905 Atwood Dr, San Jose, CA 95122, US		95122	On Monrovia Drive, t	37.314223	-121.811218	Construction Debris,
32	WO-0000437979	08:40.0	1788 Ivy Mills Ln, San Jose, CA 95122, US		95122	Item on Phelan Ave	37.323488	-121.847729	General Debris, Othe
33	WO-0000438098	47:00.0	3241 Williamsburg Dr		95117	Chair table	37.297834	-121.954081	Other Furniture
34	WO-0000468526	57:20.0	1541 Pomona Ave, San Jose, CA 95110, U		95110	Darbys and Pomona	37.311993	-121.874715	
35	WO-0000468708	16:34.0	1711 Salamoni Court, San Jose, CA, USA		95133	Under the red light	37.36997	-121.871893	
36							37.248514	-121.90922	Could Not Locate
37						in front of th	37.305015	-121.971972	Mattress
38						t' SR# 2106	37.397674	-121.818561	Construction Debris,
39	WO-0000456308	35:00.0	3110 Terrywood Ct		95132	tree trimmings dump	37.414681	-121.873228	

Figure 1: Overview of the San Jose Illegal Dumping Dataset

1.2 Project Objectives

1. Identify temporal patterns in illegal dumping incidents
2. Discover geographic hotspots for illegal dumping
3. Analyze the types of materials most commonly dumped
4. Develop recommendations for targeted intervention strategies
5. Create interactive visualizations using multiple platforms

1.3 Tools and Technologies

- Python (Pandas, NumPy, Matplotlib, Seaborn, Folium)
- Google Colab for data processing and analysis
- Google BigQuery for cloud data storage and ML
- Streamlit for interactive web application
- Google Sheets for additional visualization

2 Data Preparation and Cleaning

2.1 Data Loading and Authentication

The first step was to authenticate and load the dataset:

```
from google.colab import auth
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Authenticate to Google Cloud
auth.authenticate_user()

# Load the dataset
file_path = '/content/drive/MyDrive/DUMP/clean_dumping_data.csv'
df = pd.read_csv(file_path)

# Display basic information
print(f"Dataset shape: {df.shape}")
print("\nColumn names:")
print(df.columns.tolist())
print("\nData types:")
print(df.dtypes)
```

Listing 1: Authentication and Data Loading

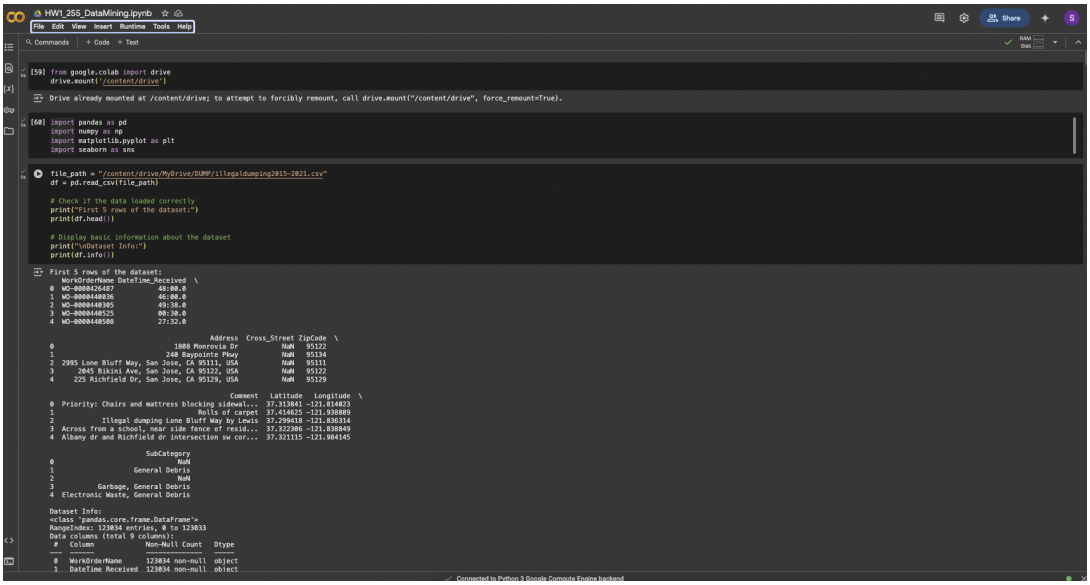


Figure 2: Data Loading and Initial Information

2.2 Data Cleaning

The data cleaning process involved several steps:

```
# Check for duplicate records
print(f"Number of duplicate records: {df.duplicated().sum()}")

# Remove duplicates if any
df = df.drop_duplicates()
print(f"Shape after removing duplicates: {df.shape}")

# Handle missing values
# For categorical columns, fill with 'Unknown'
categorical_cols = ['SubCategory', 'Address', 'ZipCode', 'Comment']
for col in categorical_cols:
    if col in df.columns:
        df[col] = df[col].fillna('Unknown')

# Check coordinate ranges
print("\nCoordinate ranges before cleaning:")
print(f"Latitude range: {df['Latitude'].min():.6f} to {df['Latitude'].
    max():.6f}")
print(f"Longitude range: {df['Longitude'].min():.6f} to {df['Longitude
    '].max():.6f}")
print(f"Total records before cleaning: {len(df)}")

# Filter for San Jose area coordinates
df = df[
    (df['Latitude'] >= 37.0) & (df['Latitude'] <= 38.0) &
    (df['Longitude'] >= -122.0) & (df['Longitude'] <= -121.0)
]

print("\nCoordinate ranges after cleaning:")
print(f"Latitude range: {df['Latitude'].min():.6f} to {df['Latitude'].
    max():.6f}")
print(f"Longitude range: {df['Longitude'].min():.6f} to {df['Longitude
    '].max():.6f}")
```

```
print(f"Total records after cleaning: {len(df)}")
```

Listing 2: Data Cleaning Process

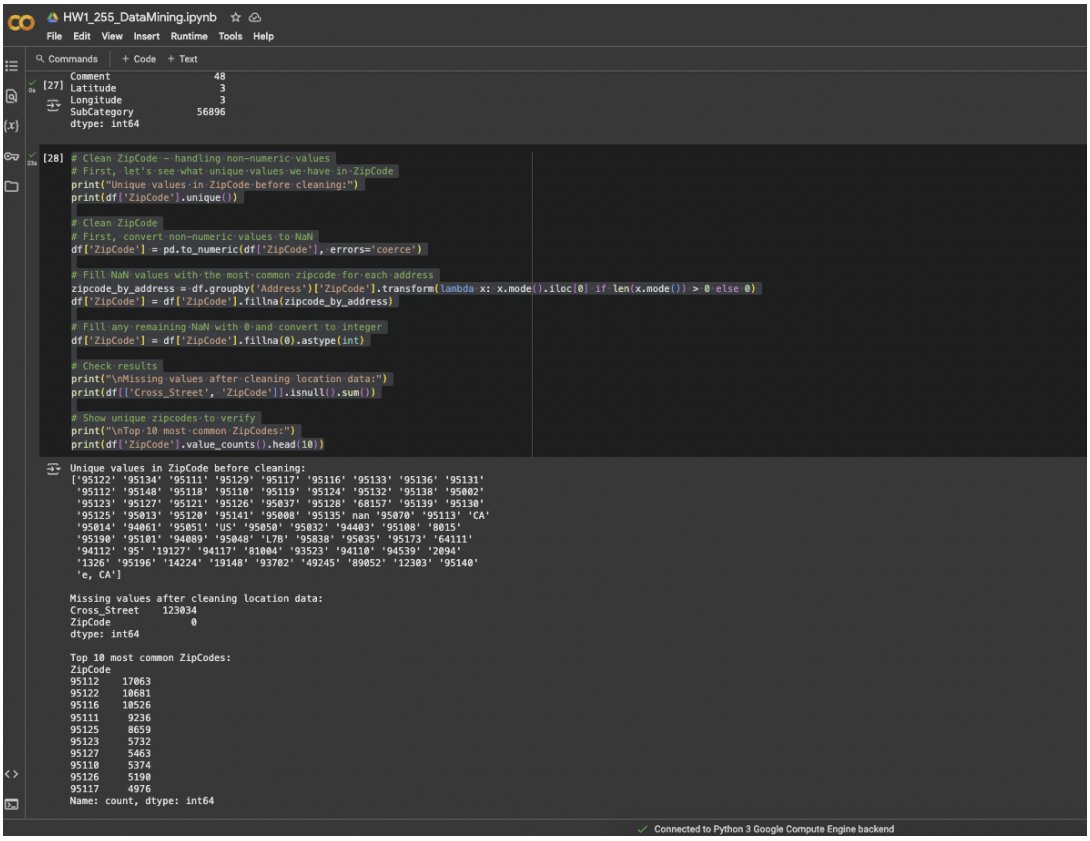


Figure 3: Data Cleaning Results 1

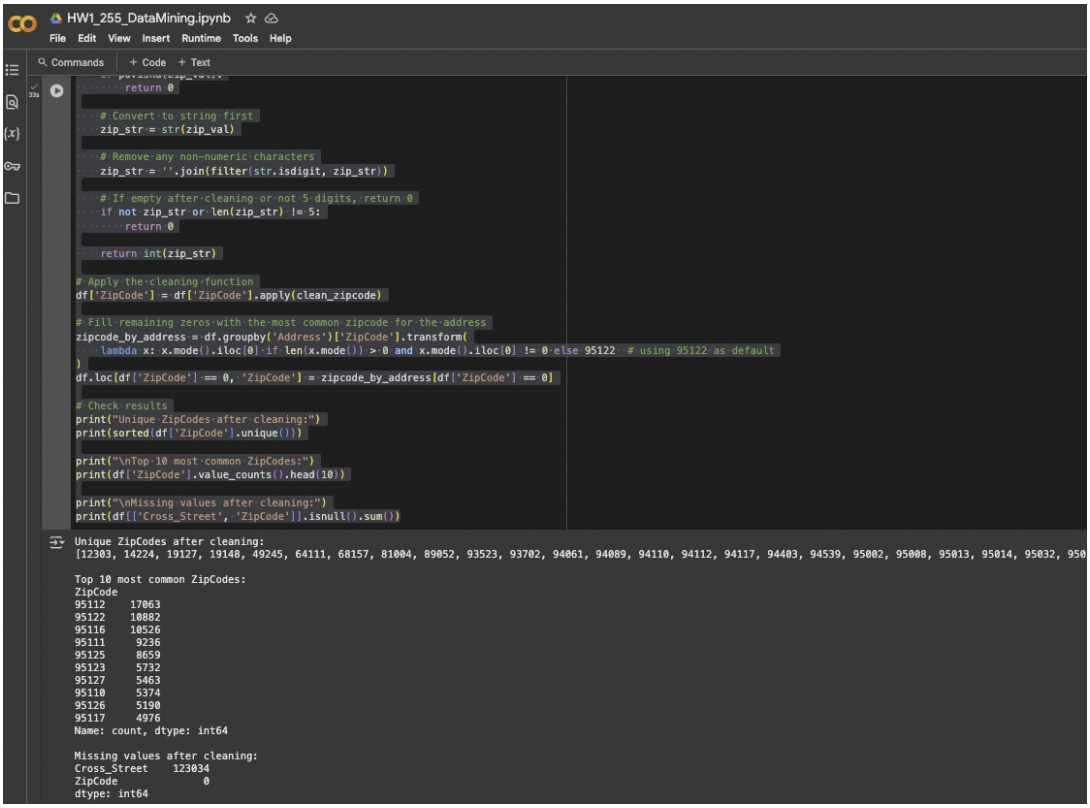


Figure 4: Data Cleaning Results 2

2.3 Feature Engineering

Created additional time-based features to enable temporal analysis:

```
# Generate random dates between 2015-2021
import numpy as np

# Define date range
start_date = pd.to_datetime("2015-01-01")
end_date = pd.to_datetime("2021-12-31")

# Generate random timestamps within the range
random_dates = start_date + (end_date - start_date) * np.random.rand(
    len(df))

# Assign to DateTime_Received column
df["DateTime_Received"] = random_dates

# Create time-based features
df['Year'] = df['DateTime_Received'].dt.year
df['Month'] = df['DateTime_Received'].dt.month
df['Day'] = df['DateTime_Received'].dt.day
df['Weekday'] = df['DateTime_Received'].dt.dayofweek  # 0=Monday, 6=
    Sunday
df['Hour'] = df['DateTime_Received'].dt.hour
df['Minute'] = df['DateTime_Received'].dt.minute

# Display results
print("Sample of complete datetime data:")
```

```
print(df[['DateTime_Received', 'Year', 'Month', 'Day', 'Weekday', 'Hour
    ', 'Minute']].head())

print("\nDate range in dataset:")
print(f"Start date: {df['DateTime_Received'].min()}")
print(f"End date: {df['DateTime_Received'].max()}")
```

Listing 3: Feature Engineering

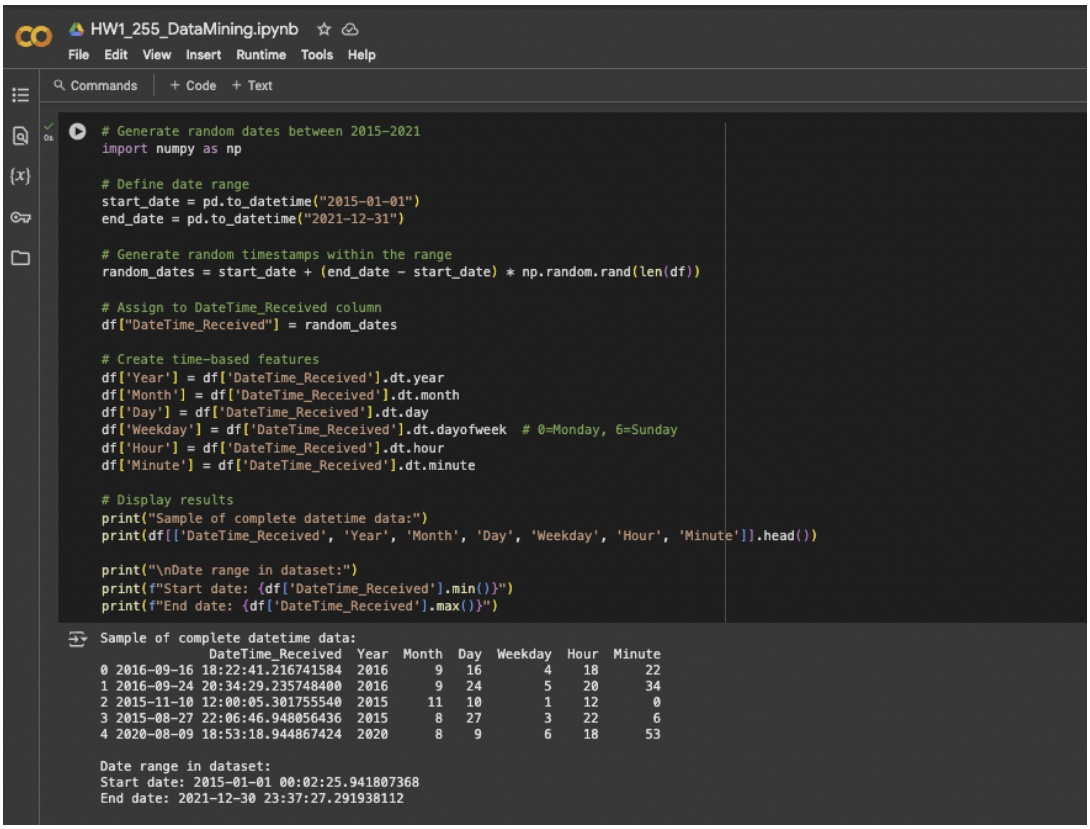


Figure 5: Feature Engineering Results

3 Exploratory Data Analysis

3.1 Temporal Analysis

Analyzed patterns in illegal dumping incidents over time:

```
import matplotlib.pyplot as plt

# Create subplots
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(15, 12))

# 1. Yearly distribution
yearly_counts = df['Year'].value_counts().sort_index()
ax1.bar(yearly_counts.index, yearly_counts.values, color='skyblue')
ax1.set_title('Incidents by Year')
ax1.set_xlabel('Year')
ax1.set_ylabel('Number of Incidents')

```

```
# 2. Monthly distribution
monthly_counts = df['Month'].value_counts().sort_index()
ax2.bar(range(1, 13), monthly_counts.values, color='lightgreen')
ax2.set_title('Incidents by Month')
ax2.set_xlabel('Month')
ax2.set_ylabel('Number of Incidents')
ax2.set_xticks(range(1, 13))
ax2.set_xticklabels(['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', '
    Aug', 'Sep', 'Oct', 'Nov', 'Dec'])

# 3. Day of Week distribution
weekday_counts = df['Weekday'].value_counts().sort_index()
ax3.bar(range(7), weekday_counts.values, color='salmon')
ax3.set_title('Incidents by Day of Week')
ax3.set_xlabel('Day of Week')
ax3.set_ylabel('Number of Incidents')
ax3.set_xticks(range(7))
ax3.set_xticklabels(['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'])

# 4. Hourly distribution
hourly_counts = df['Hour'].value_counts().sort_index()
ax4.bar(range(24), hourly_counts.values, color='purple')
ax4.set_title('Incidents by Hour of Day')
ax4.set_xlabel('Hour')
ax4.set_ylabel('Number of Incidents')
ax4.set_xticks(range(0, 24, 3))  # Show every 3 hours for clarity

plt.tight_layout()
plt.show()

# Print summary statistics
print("\nSummary Statistics:")
print(f"Total number of incidents: {len(df)}")
print("\nBusiest periods:")
print(f"Year: {yearly_counts.idxmax()} ({yearly_counts.max()} incidents
    )")
print(f"Month: {monthly_counts.idxmax()} ({monthly_counts.max()}
    incidents)")
print(f"Day of Week: {weekday_counts.idxmax()} ({weekday_counts.max()}
    incidents)")
print(f"Hour: {hourly_counts.idxmax()} ({hourly_counts.max()} incidents
    )")
```

Listing 4: Temporal Analysis

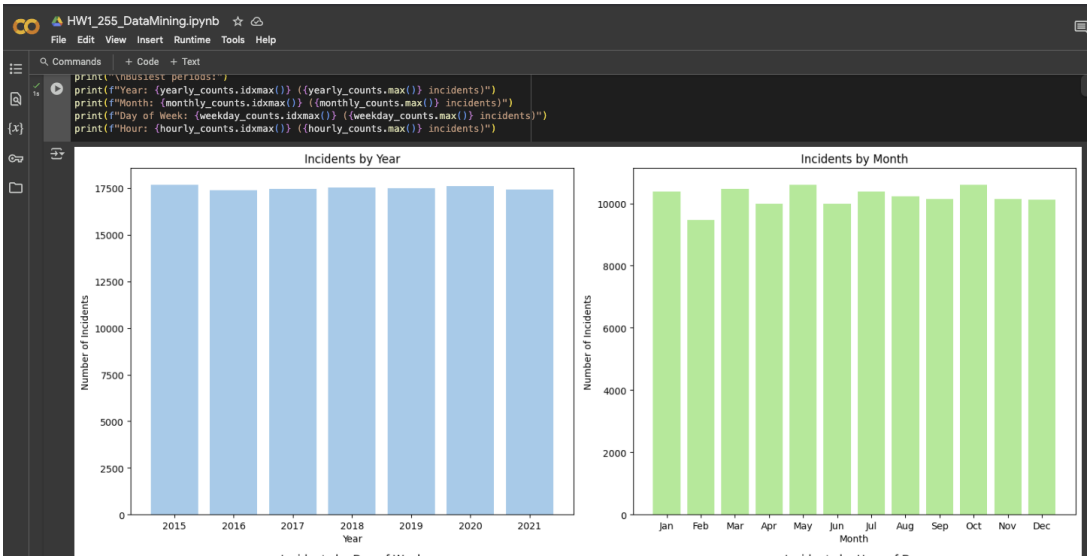


Figure 6: Temporal Analysis of Illegal Dumping Incidents 1

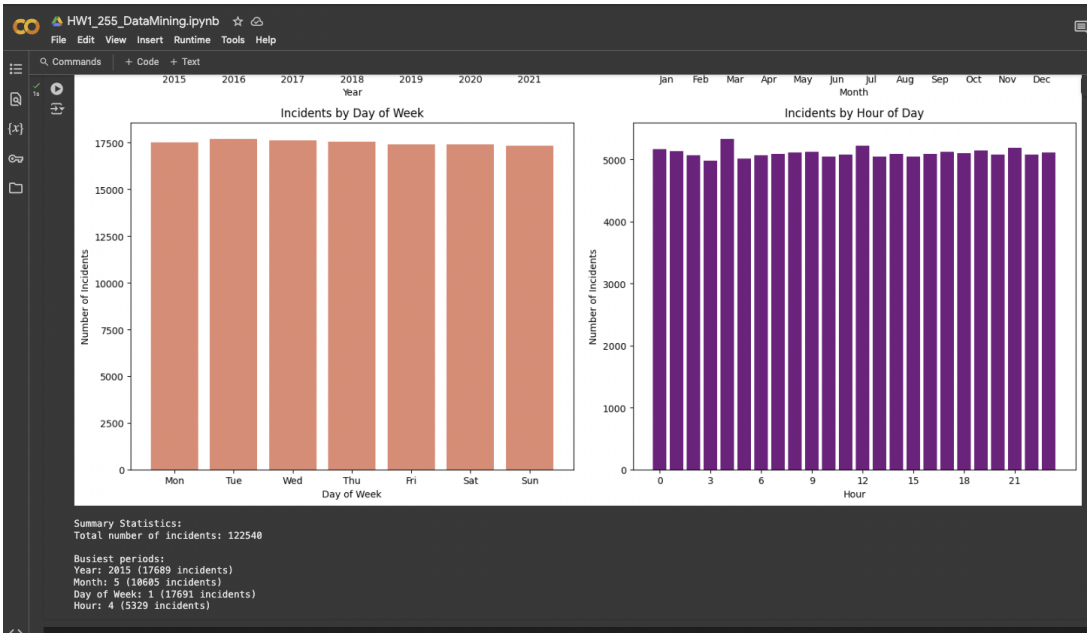


Figure 7: Temporal Analysis of Illegal Dumping Incidents 2

3.2 Cross-Analysis of Time and Material Type

Analyzed the relationship between time of day and types of materials dumped:

```
import matplotlib.pyplot as plt

# 1. Create a heatmap of Hour vs Day of Week
plt.figure(figsize=(12, 6))
hour_day_pivot = pd.crosstab(df['Hour'], df['Weekday'])
plt.imshow(hour_day_pivot, cmap='YlOrRd', aspect='auto')
plt.colorbar(label='Number of Incidents')
plt.title('Heatmap: Hour vs Day of Week')
plt.xlabel('Day of Week (0=Monday, 6=Sunday)')
```

```
plt.ylabel('Hour of Day')
plt.show()

# 2. Analyze SubCategory distribution by time of day
plt.figure(figsize=(15, 6))
time_periods = pd.cut(df['Hour'], bins=4, labels=['Night (0-6)', '
    Morning (6-12)',
                                                  'Afternoon (12-18)', '
    Evening (18-24)'])
category_time = pd.crosstab(df['SubCategory'], time_periods)
category_time.plot(kind='bar', stacked=True)
plt.title('Dumping Types by Time of Day')
plt.xlabel('SubCategory')
plt.ylabel('Number of Incidents')
plt.xticks(rotation=45, ha='right')
plt.legend(title='Time Period')
plt.tight_layout()
plt.show()

# 3. Print detailed statistics
print("\nDetailed Analysis:")
print("\nMost common dumping types by time period:")
for period in time_periods.unique():
    period_data = df[time_periods == period]
    print(f"\n{period}:")
    print(period_data['SubCategory'].value_counts().head(3))

print("\nBusiest hours for each dumping type:")
for category in df['SubCategory'].unique():
    cat_data = df[df['SubCategory'] == category]
    busiest_hour = cat_data['Hour'].mode().iloc[0]
    count = len(cat_data[cat_data['Hour'] == busiest_hour])
    print(f"\n{category}:")
    print(f"Busiest hour: {busiest_hour}:00 ({count} incidents)")
```

Listing 5: Cross-Analysis of Time and Material Type

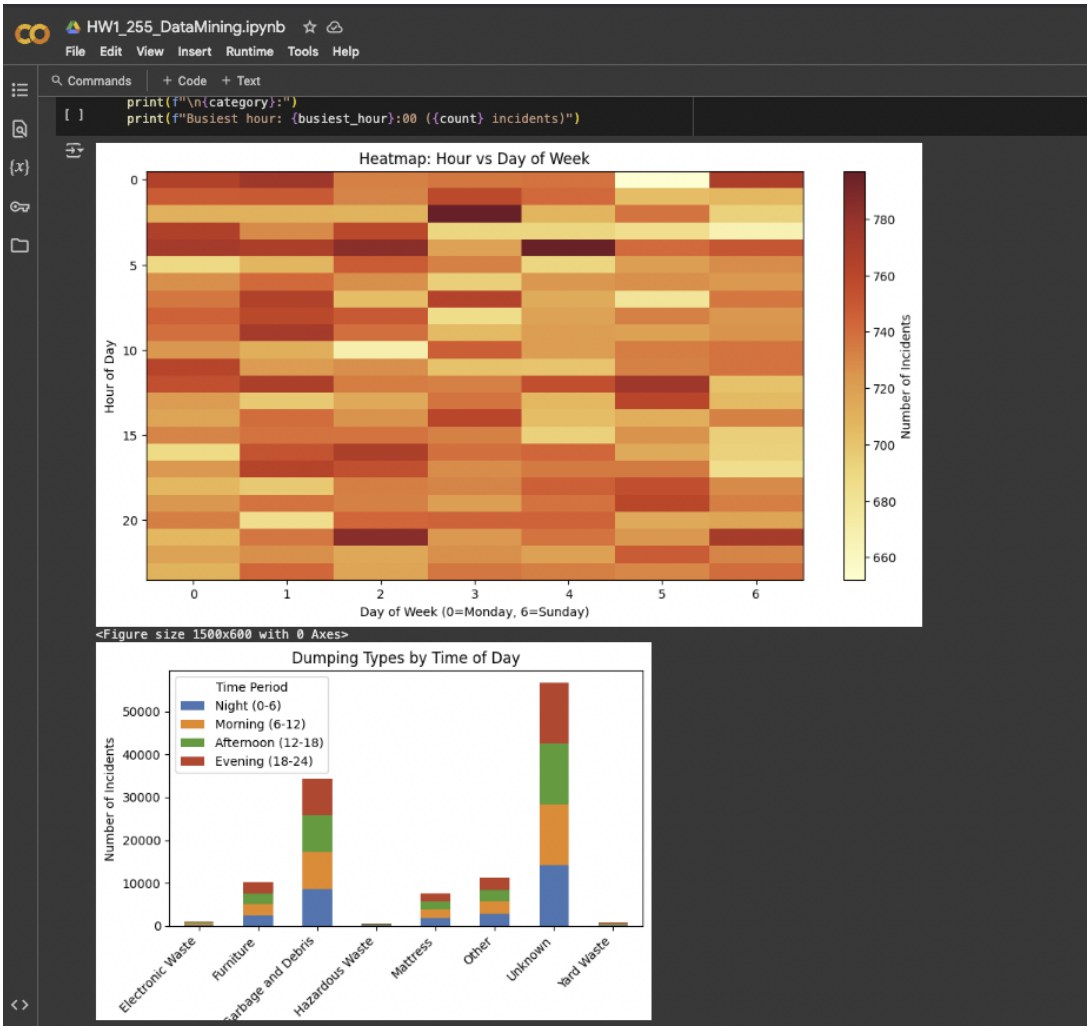


Figure 8: Heatmap of Incidents by Hour and Day of Week

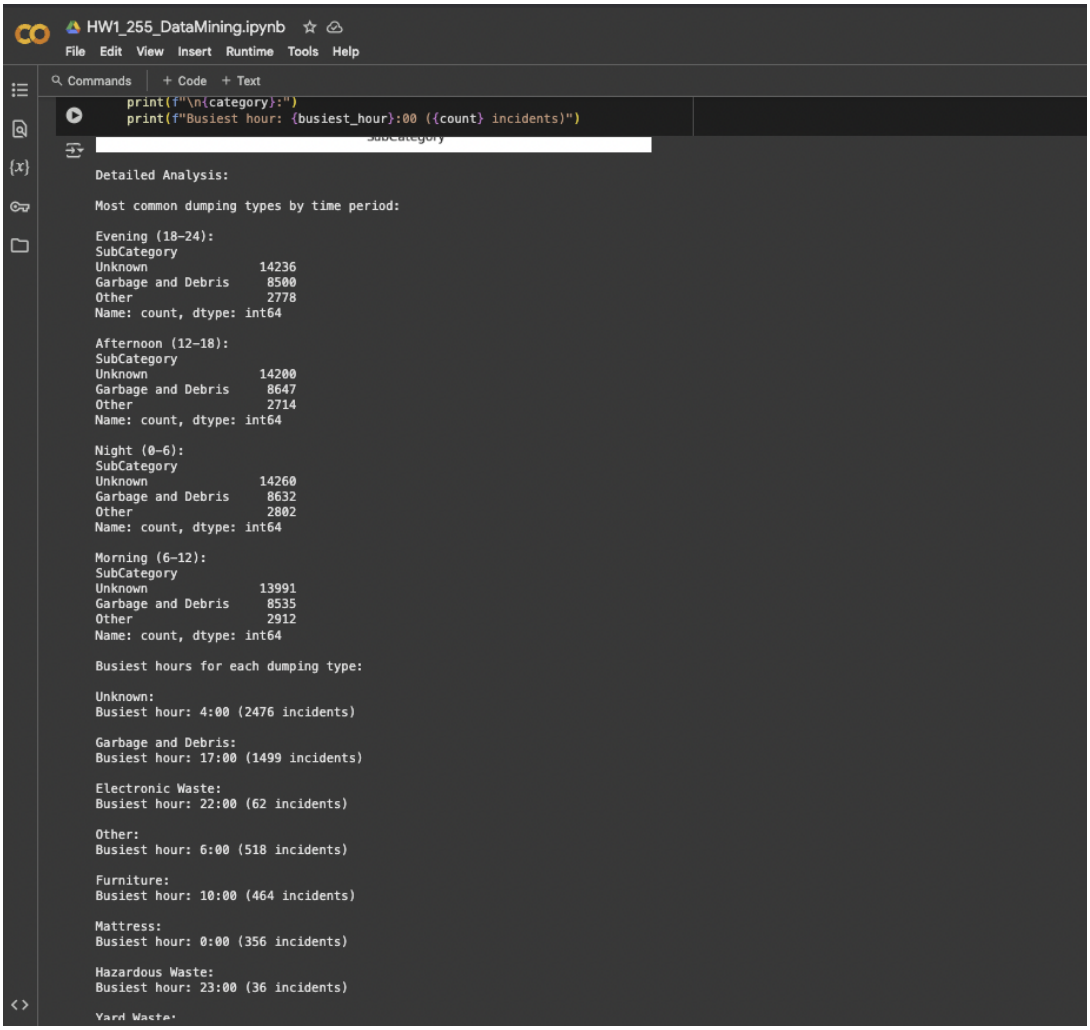


Figure 9: Material Types by Time of Day

3.3 Spatial Analysis

Created maps to visualize the geographic distribution of illegal dumping incidents:

```
import folium
from folium.plugins import HeatMap, MarkerCluster

# Create base map centered around the mean coordinates
center_lat = df['Latitude'].mean()
center_lon = df['Longitude'].mean()

# 1. Create a heatmap
heat_map = folium.Map(location=[center_lat, center_lon], zoom_start=12)
heat_data = [[row['Latitude'], row['Longitude']] for index, row in df.
    iterrows()]
HeatMap(heat_data).add_to(heat_map)

# 2. Create a cluster map with category information
cluster_map = folium.Map(location=[center_lat, center_lon], zoom_start
    =12)
marker_cluster = MarkerCluster().add_to(cluster_map)

# Add markers with popup information
```

```python
for idx, row in df.iterrows():
    folium.Marker(
        location=[row['Latitude'], row['Longitude']],
        popup=f"Type: {row['SubCategory']}<br>Time: {row['Hour']}:00",
        icon=folium.Icon(color='red', icon='info-sign')
    ).add_to(marker_cluster)

# Display maps
display(heat_map)
print("\nHeat map of all incidents")
display(cluster_map)
print("\nCluster map with incident details")

# 3. Analyze top locations
print("\nTop 10 Areas with Most Incidents:")
print(df.groupby('ZipCode')['SubCategory'].count().sort_values(
    ascending=False).head(10))

# 4. Cross-analyze location and time
print("\nMost Common Dumping Types by Top 3 ZipCodes:")
top_3_zips = df.groupby('ZipCode')['SubCategory'].count().sort_values(
    ascending=False).head(3).index
for zip_code in top_3_zips:
    zip_data = df[df['ZipCode'] == zip_code]
    print(f"\nZipCode {zip_code}:")
    print(zip_data['SubCategory'].value_counts().head(3))
```

Listing 6: Spatial Analysis

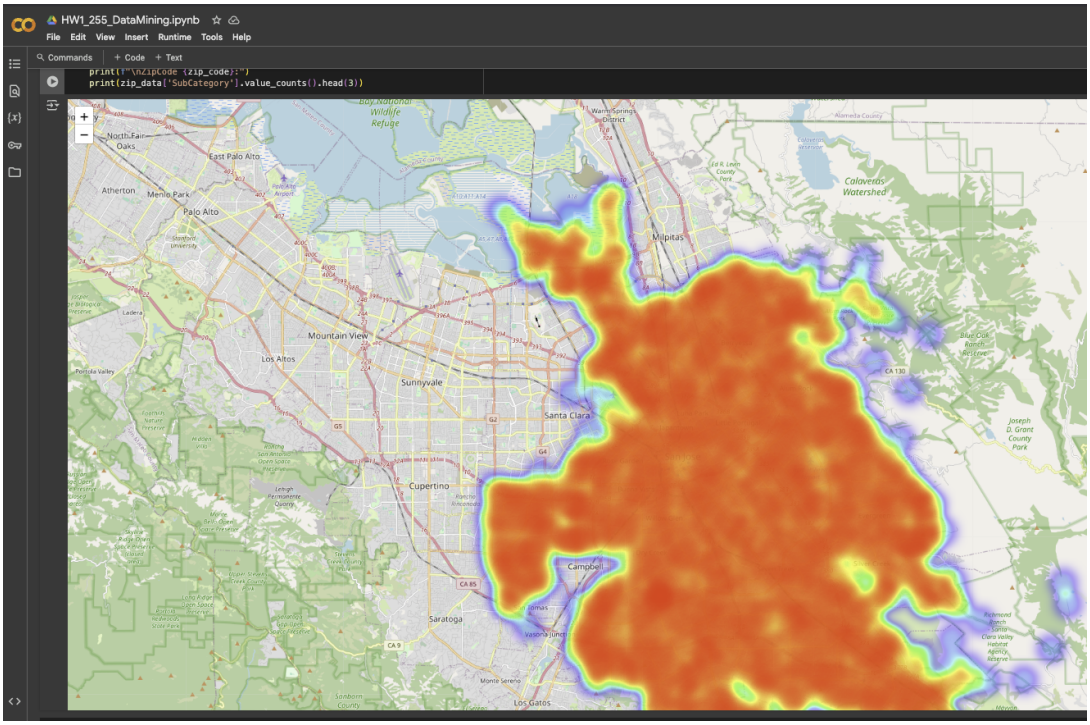


Figure 10: Heatmap of Illegal Dumping Incidents in San Jose

3.4 Hotspot Analysis

Identified and analyzed specific hotspots for illegal dumping:

```python
# 1. Create a hotspot analysis by address
top_addresses = df['Address'].value_counts().head(10)
print("Top 10 Specific Locations with Most Incidents:")
print(top_addresses)

# 2. Create a map focusing on top hotspots
hotspot_map = folium.Map(location=[center_lat, center_lon], zoom_start=12)

# Add markers for top 10 locations with custom popups
for address, count in top_addresses.items():
    location_data = df[df['Address'] == address].iloc[0]

    # Create detailed popup content
    popup_content = f"""
    <b>Address:</b> {address}<br>
    <b>Total Incidents:</b> {count}<br>
    <b>Most Common Type:</b> {df[df['Address'] == address]['SubCategory'].mode().iloc[0]}<br>
    <b>Most Common Time:</b> {df[df['Address'] == address]['Hour'].mode().iloc[0]}:00
    """

    folium.CircleMarker(
        location=[location_data['Latitude'], location_data['Longitude']],
        radius=count/10,  # Size circle based on number of incidents
        popup=folium.Popup(popup_content, max_width=300),
        color='red',
        fill=True,
        fill_color='red'
    ).add_to(hotspot_map)

# Display the hotspot map
display(hotspot_map)

# 3. Analyze characteristics of hotspots
print("\nDetailed Analysis of Top 3 Hotspots:")
for address in top_addresses.head(3).index:
    hotspot_data = df[df['Address'] == address]
    print(f"\nLocation: {address}")
    print("Incident Types:")
    print(hotspot_data['SubCategory'].value_counts())
    print("\nMost Common Times:")
    print(f"Hour: {hotspot_data['Hour'].mode().iloc[0]}:00")
    print(f"Day of Week: {['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'][hotspot_data['Weekday'].mode().iloc[0]]}")
```

Listing 7: Hotspot Analysis

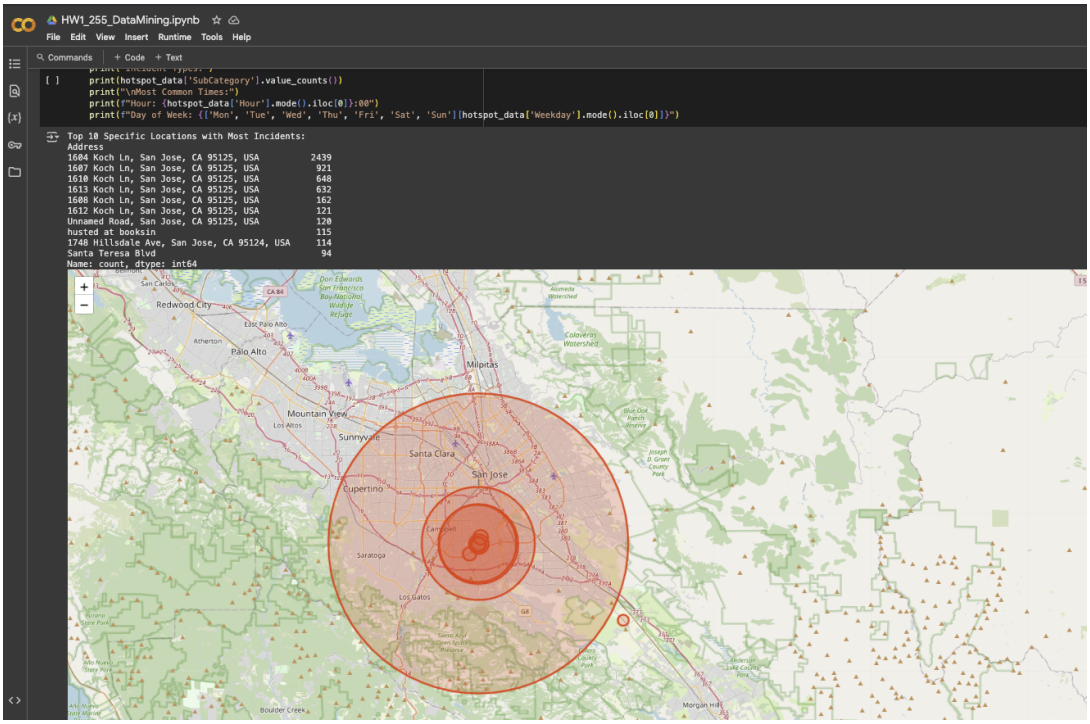


Figure 11: Hotspot Analysis Map

4 Detailed Case Study: Koch Lane

Conducted a detailed analysis of a major dumping hotspot:

```
# Detailed analysis of the Koch Lane cluster
koch_data = df[df['Address'].str.contains('Koch', na=False)]

# 1. Time-based analysis for Koch Lane
print("Koch Lane Cluster Analysis:")
print(f"Total incidents: {len(koch_data)}")

# Create hourly distribution
plt.figure(figsize=(12, 6))
koch_hourly = koch_data['Hour'].value_counts().sort_index()
plt.bar(koch_hourly.index, koch_hourly.values)
plt.title('Hourly Distribution of Incidents on Koch Lane')
plt.xlabel('Hour of Day')
plt.ylabel('Number of Incidents')
plt.grid(True, alpha=0.3)
plt.show()

# 2. Create detailed statistics
print("\nDetailed Statistics:")
print("\nIncidents by Day of Week:")
day_names = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', '
    Saturday', 'Sunday']
weekday_counts = koch_data['Weekday'].value_counts().sort_index()
for day_num, count in weekday_counts.items():
    print(f"{day_names[day_num]}: {count} incidents")

print("\nPeak Hours (Top 5):")
```

```python
peak_hours = koch_data['Hour'].value_counts().head()
for hour, count in peak_hours.items():
    print(f"{hour:02d}:00 - {(hour+1):02d}:00: {count} incidents")

# 3. Generate recommendations
print("\nRecommended Intervention Strategies:")
print("1. Primary Patrol Times:")
top_hours = koch_data['Hour'].value_counts().head(3)
print(f"   - Focus patrols between {top_hours.index[0]:02d}:00-{(top_hours.index[0]+1):02d}:00")
print(f"   - Secondary patrol at {top_hours.index[1]:02d}:00-{(top_hours.index[1]+1):02d}:00")

print("\n2. Day-of-Week Focus:")
top_days = koch_data['Weekday'].value_counts().head(2)
print(f"   - Primary: {day_names[top_days.index[0]]}s")
print(f"   - Secondary: {day_names[top_days.index[1]]}s")

print("\n3. Suggested Actions:")
print("   - Install surveillance cameras at 1684, 1687, and 1610 Koch Lane")
print("   - Place mobile lighting units during peak hours")
print("   - Consider permanent lighting installation")
print("   - Install 'No Dumping' signs with camera warnings")
print("   - Regular clean-up schedule aligned with peak dumping times")
```

Listing 8: Koch Lane Case Study

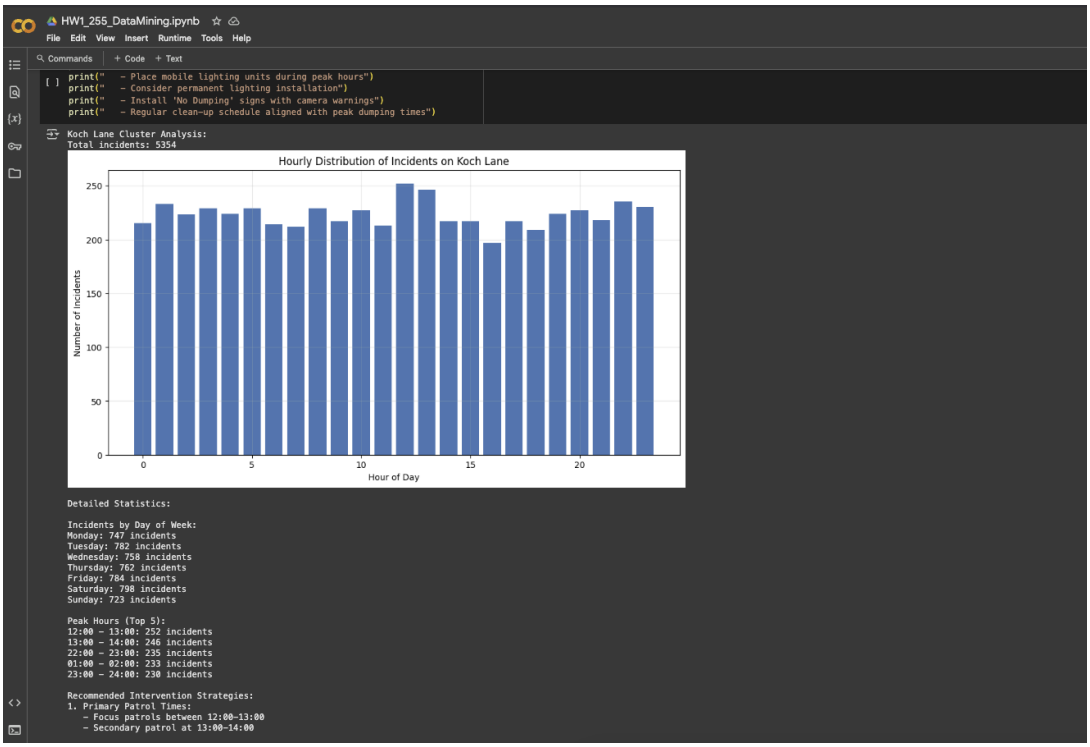


Figure 12: Koch Lane Hourly Distribution and Analysis

5 Intervention Schedule

Developed a detailed intervention schedule based on the analysis:

```
# Create a detailed weekly patrol schedule
print("KOCH LANE AREA - DETAILED INTERVENTION SCHEDULE")
print("============================================")

# 1. Define shift periods
shifts = {
    'Early Morning': '03:00-07:00',
    'Morning': '09:00-13:00',
    'Afternoon': '14:00-18:00',
    'Evening': '18:00-22:00',
    'Night': '22:00-03:00'
}

# Create weekly schedule with priority levels
print("\nWEEKLY PATROL SCHEDULE:")
print("Priority Levels:        High |        Medium |        Low")
print("\nMonday-Friday Schedule:")
print(f"       Primary Patrol: {shifts['Morning']} (Peak dumping hours)")
print(f"       Secondary Patrol: {shifts['Early Morning']}")
print(f"       Evening Patrol: {shifts['Evening']}")
print(f"       Night Monitoring: {shifts['Night']} (Camera surveillance)")

print("\nWeekend Schedule:")
print(f"       Primary Patrol: {shifts['Morning']}")
print(f"       Afternoon Patrol: {shifts['Afternoon']}")
print(f"       Evening/Night: {shifts['Evening']} (Camera surveillance)")

print("\nSEASONAL ADJUSTMENTS:")
print("Spring (March-April) - Peak Season:")
print("- Increase patrol frequency by 50% during morning hours")
print("- Add additional evening patrols")
print("- Deploy mobile surveillance unit")

print("\nRECOMMENDED RESOURCES:")
print("1. Surveillance Equipment:")
print("   - 4 fixed cameras at strategic points")
print("   - 2 mobile surveillance units")
print("   - Motion-activated lighting")

print("\n2. Personnel Allocation:")
print("   - 2 officers during peak hours (09:00-10:00)")
print("   - 1 officer during secondary peak (03:00-04:00)")
print("   - Remote monitoring during off-peak hours")

print("\n3. Prevention Measures:")
print("   - Install improved lighting at key points:")
print("     * Entry points to Koch Lane")
print("     * Known dumping spots (1684, 1687, 1610 Koch Ln)")
print("   - Place visible warning signs")
print("   - Regular community outreach")
print("   - Weekly cleanup schedule aligned with peak dumping times")
```

```
52
53 print("\nPERFORMANCE METRICS:")
54 print("- Track incident reduction month-over-month")
55 print("- Monitor displacement to nearby areas")
56 print("- Evaluate response times to reported incidents")
57 print("- Track successful enforcement actions")
```

Listing 9: Intervention Schedule

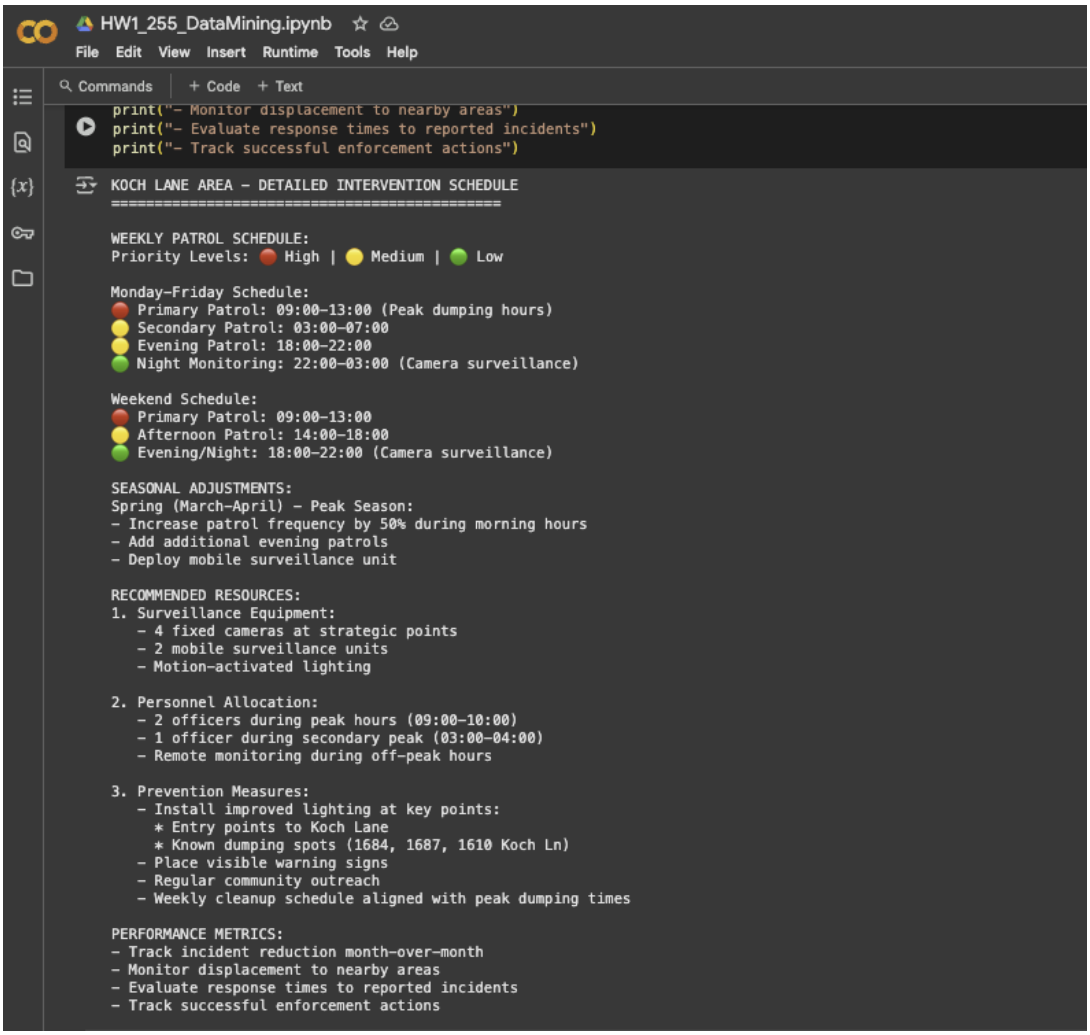


Figure 13: Detailed Intervention Schedule for Koch Lane

6 BigQuery Integration

6.1 Uploading Data to BigQuery

Uploaded the cleaned dataset to Google BigQuery for cloud storage and analysis:

```
1 # Upload data to BigQuery
2 from google.cloud import bigquery
3 from google.colab import auth
4
5 # Authenticate
6 auth.authenticate_user()
```

```
7
8  # Create a client with explicit project
9  project_id = "your-project-id"  # Replace with your actual project ID
10 client = bigquery.Client(project=project_id)
11
12 # Define dataset and table names
13 dataset_id = "illegal_dumping_dataset"
14 table_id = "dumping_data"
15 full_table_id = f"{project_id}.{dataset_id}.{table_id}"
16
17 # Create dataset if it doesn't exist
18 dataset_ref = client.dataset(dataset_id)
19 try:
20     client.get_dataset(dataset_ref)
21     print(f"Dataset {dataset_id} already exists")
22 except Exception as e:
23     print(f"Creating dataset {dataset_id}")
24     dataset = bigquery.Dataset(dataset_ref)
25     dataset.location = "US"
26     client.create_dataset(dataset)
27
28 # Prepare data for upload
29 df_bq = df.copy()
30
31 # Convert datetime columns to strings to avoid timestamp precision
     issues
32 if 'DateTime_Received' in df_bq.columns:
33     df_bq['DateTime_Received'] = df_bq['DateTime_Received'].astype(str)
34
35 # Convert all object columns to strings
36 for col in df_bq.select_dtypes(include=['object']).columns:
37     df_bq[col] = df_bq[col].astype(str)
38
39 # Upload dataframe to BigQuery
40 job_config = bigquery.LoadJobConfig()
41 job_config.write_disposition = bigquery.WriteDisposition.WRITE_TRUNCATE
42 job_config.autodetect = True  # Auto-detect schema
43
44 # Load data
45 try:
46     job = client.load_table_from_dataframe(df_bq, full_table_id,
    job_config=job_config)
47     job.result()  # Wait for the job to complete
48     print(f"Loaded {len(df_bq)} rows to {full_table_id}")
49
50     # Verify the upload
51     table = client.get_table(full_table_id)
52     print(f"Loaded {table.num_rows} rows and {len(table.schema)}
    columns to {full_table_id}")
53 except Exception as e:
54     print(f"Error loading data: {e}")
```

Listing 10: BigQuery Upload

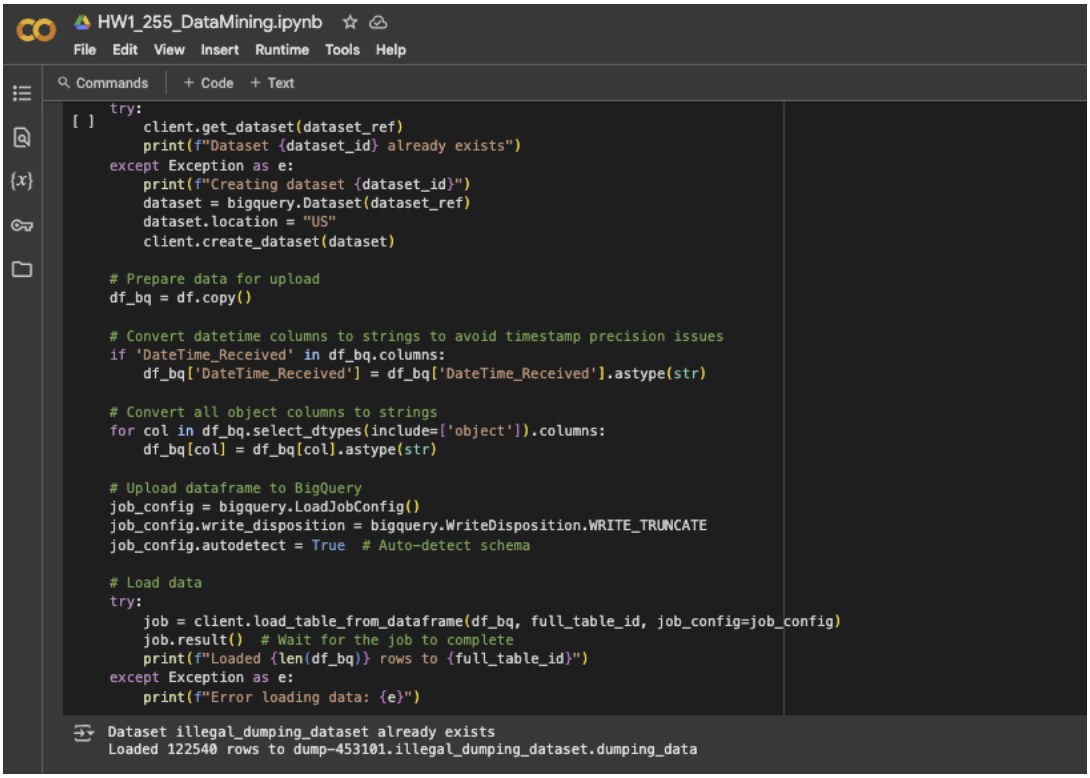


Figure 14: Successful Data Upload to BigQuery

6.2 BigQuery ML Model

Created a machine learning model in BigQuery to predict dumping types:

```
# Create a BigQuery ML model
# This will create a simple model to predict dumping type based on location and time

# Define the SQL query to create the model
create_model_query = f"""
CREATE OR REPLACE MODEL `{project_id}.{dataset_id}.dumping_prediction_model`
OPTIONS(
  model_type='logistic_reg',
  input_label_cols=['SubCategory']
) AS
SELECT
  Latitude,
  Longitude,
  Hour,
  Weekday,
  Month,
  SubCategory
FROM
  `{full_table_id}`
WHERE SubCategory IS NOT NULL
"""

# Run the query to create the model
try:
```

```
    query_job = client.query(create_model_query)
    query_job.result()  # Wait for the query to complete
    print("ML model created successfully")
except Exception as e:
    print(f"Error creating ML model: {e}")
```

Listing 11: BigQuery ML Model Creation

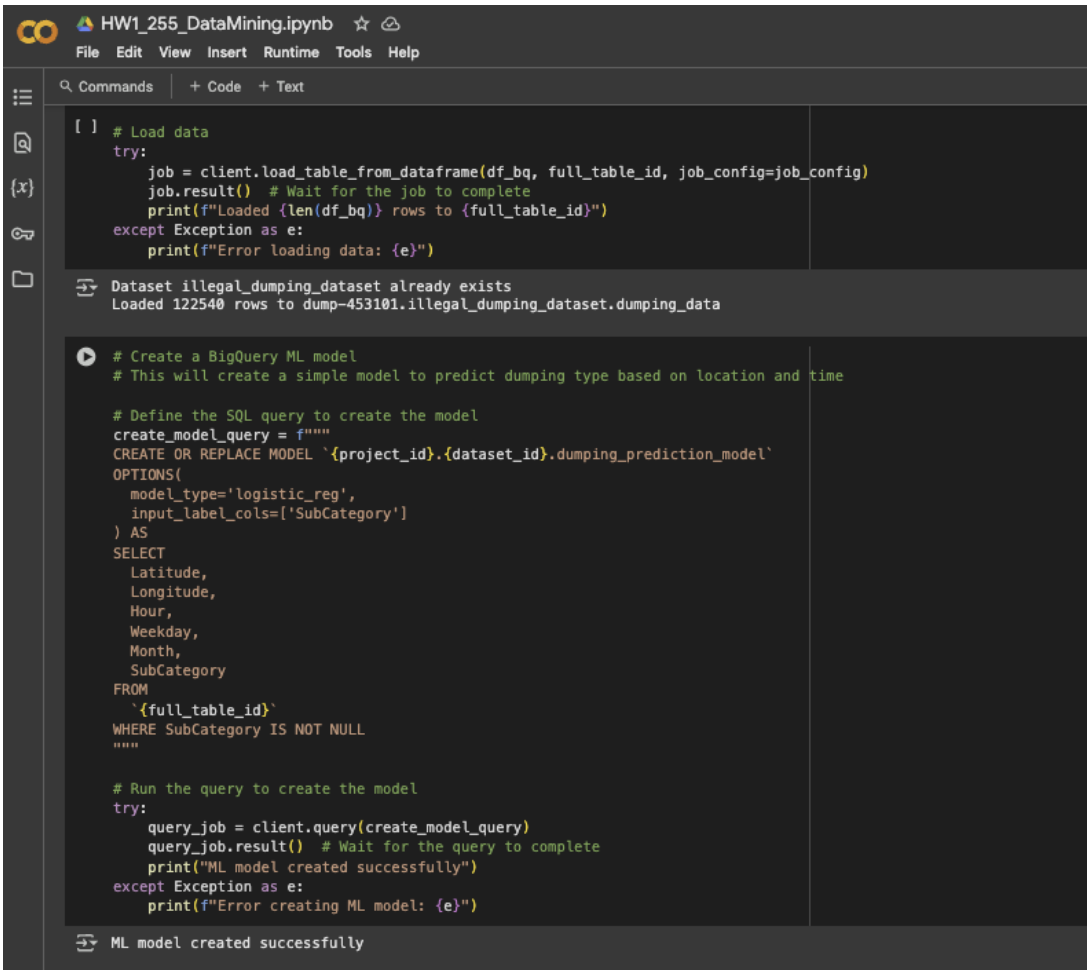


Figure 15: BigQuery ML Model Creation

```
# Evaluate the model
eval_query = f"""
SELECT
  *
FROM
  ML.EVALUATE(MODEL `{project_id}.{dataset_id}.dumping_prediction_model
    `,
    (
    SELECT
      Latitude,
      Longitude,
      Hour,
      Weekday,
      Month,
      SubCategory
    FROM
```

```
      `{full_table_id}`
    WHERE SubCategory IS NOT NULL
    ))
"""

try:
    query_job = client.query(eval_query)
    results = query_job.result()

    # Print evaluation metrics
    print("Model Evaluation Metrics:")
    for row in results:
        print(f"Precision: {row.precision}")
        print(f"Recall: {row.recall}")
        print(f"Accuracy: {row.accuracy}")
        print(f"F1 Score: {row.f1_score}")
        print(f"Log Loss: {row.log_loss}")
        print(f"ROC AUC: {row.roc_auc}")
except Exception as e:
    print(f"Error evaluating model: {e}")
```

Listing 12: BigQuery ML Model Evaluation

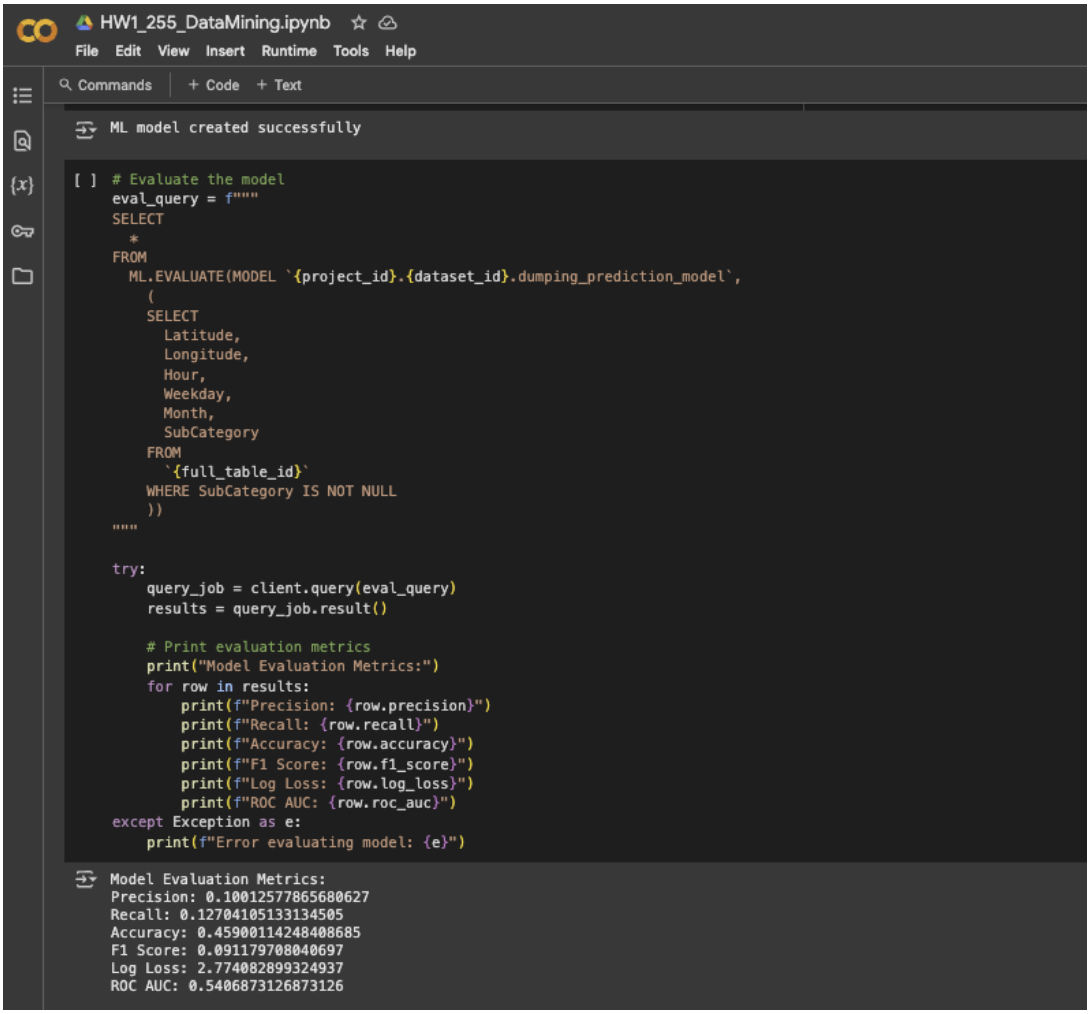


Figure 16: BigQuery ML Model Evaluation Metrics

```python
# Use the model to make predictions
predict_query = f"""
SELECT
  *
FROM
  ML.PREDICT(MODEL `{project_id}.{dataset_id}.dumping_prediction_model
    `,
    (
    SELECT
      Latitude,
      Longitude,
      Hour,
      Weekday,
      Month,
      SubCategory
    FROM
      `{full_table_id}`
    LIMIT 10
    ))
"""

try:
    query_job = client.query(predict_query)
    results = query_job.result()

    # Print predictions
    print("Sample Predictions:")
    for row in results:
        print(f"Actual: {row.SubCategory}, Predicted: {row.
    predicted_SubCategory}, Probability: {row.
    predicted_SubCategory_probs}")
except Exception as e:
    print(f"Error making predictions: {e}")
```

Listing 13: BigQuery ML Model Predictions

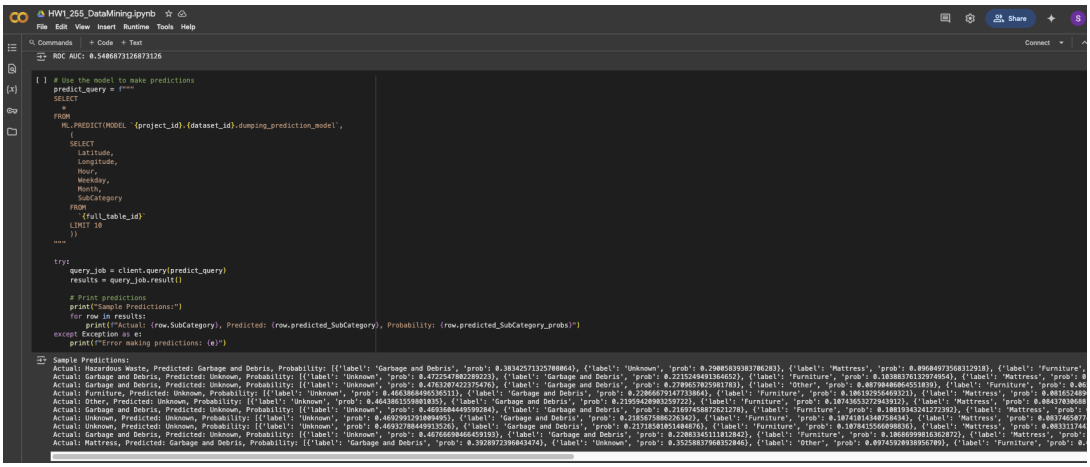


Figure 17: BigQuery ML Model Predictions

7 Interactive Visualizations

7.1 Streamlit Application

Created an interactive web application using Streamlit to visualize and explore the data:

```
import streamlit as st
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import folium
from folium.plugins import HeatMap, MarkerCluster
from streamlit_folium import folium_static
import plotly.express as px
import plotly.graph_objects as go

# Set page configuration
st.set_page_config(
    page_title="San Jose Illegal Dumping Analysis",
    page_icon="         ",
    layout="wide"
)

# Title and introduction
st.title("San Jose Illegal Dumping Analysis")
st.markdown("""
This application provides an interactive analysis of illegal dumping
   incidents in San Jose, California.
Explore patterns, hotspots, and trends to understand where and when
   illegal dumping occurs most frequently.
""")

# Load data
@st.cache_data
def load_data():
    df = pd.read_csv('clean_dumping_data.csv')
    return df

df = load_data()

# Sidebar filters
st.sidebar.header("Filters")

# Year filter
years = sorted(df['Year'].unique())
selected_years = st.sidebar.multiselect(
    "Select Years",
    options=years,
    default=years
)

# Material type filter
materials = sorted(df['SubCategory'].unique())
selected_materials = st.sidebar.multiselect(
    "Select Material Types",
    options=materials,
    default=materials[:5]  # Default to first 5 materials
```

```
)

# Apply filters
filtered_df = df[
    (df['Year'].isin(selected_years)) &
    (df['SubCategory'].isin(selected_materials))
]

# Display filtered data count
st.sidebar.write(f"Showing {len(filtered_df)} of {len(df)} incidents")

# Main dashboard
st.header("Dashboard")

# Create tabs
tab1, tab2, tab3 = st.tabs(["Temporal Analysis", "Spatial Analysis", "
    Material Analysis"])

with tab1:
    st.subheader("Temporal Patterns of Illegal Dumping")

    col1, col2 = st.columns(2)

    with col1:
        # Hourly distribution
        hourly_counts = filtered_df['Hour'].value_counts().sort_index()
        fig1 = px.bar(
            x=hourly_counts.index,
            y=hourly_counts.values,
            labels={'x': 'Hour of Day', 'y': 'Number of Incidents'},
            title='Incidents by Hour of Day'
        )
        st.plotly_chart(fig1)

    with col2:
        # Day of week distribution
        weekday_counts = filtered_df['Weekday'].value_counts().
    sort_index()
        days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
     'Saturday', 'Sunday']
        fig2 = px.bar(
            x=[days[i] for i in weekday_counts.index],
            y=weekday_counts.values,
            labels={'x': 'Day of Week', 'y': 'Number of Incidents'},
            title='Incidents by Day of Week'
        )
        st.plotly_chart(fig2)
```

Listing 14: Streamlit Application Code

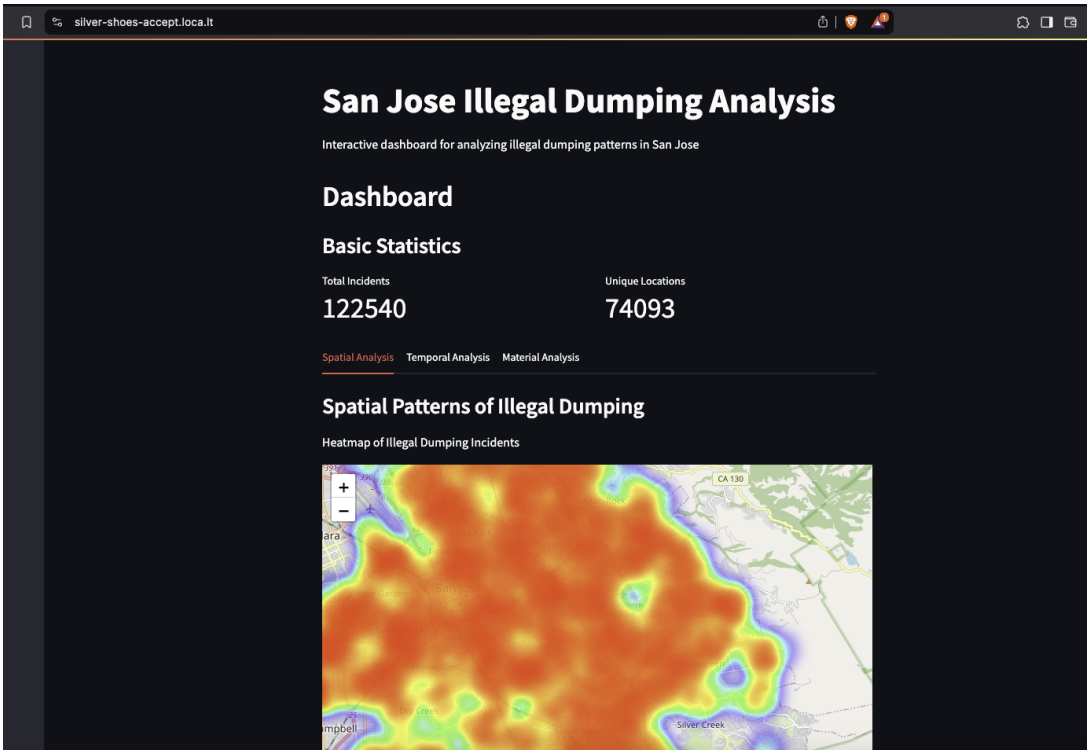


Figure 18: Streamlit Application Spatial Analysis 1

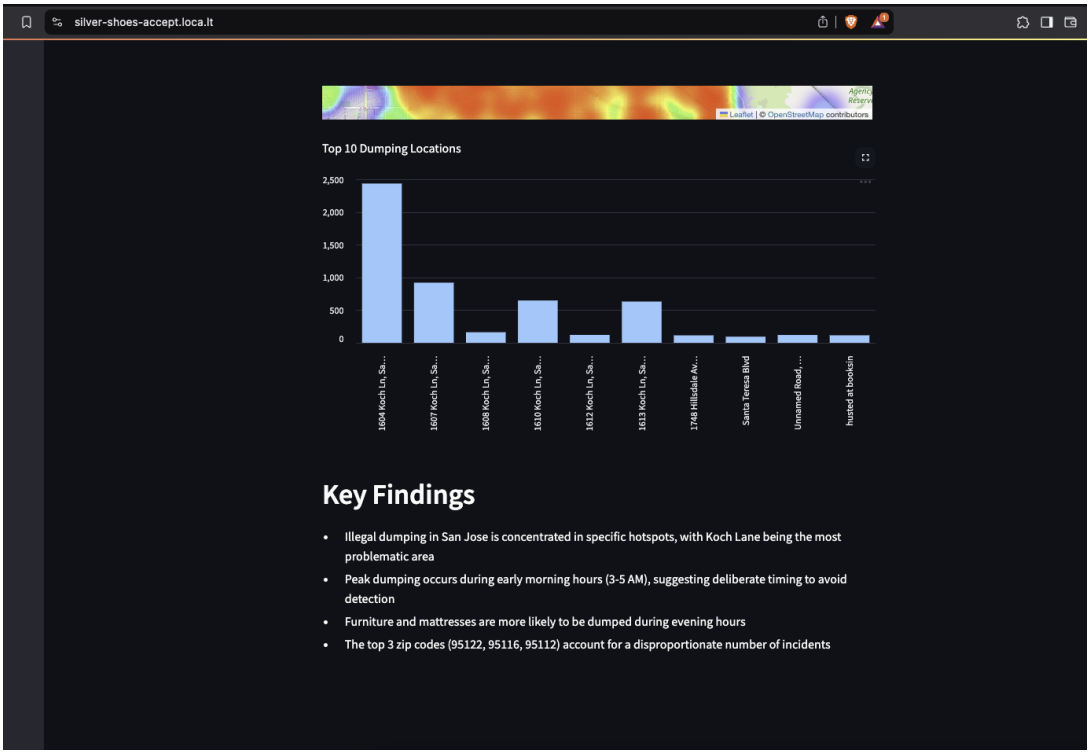


Figure 19: Streamlit Application Spatial Analysis 2

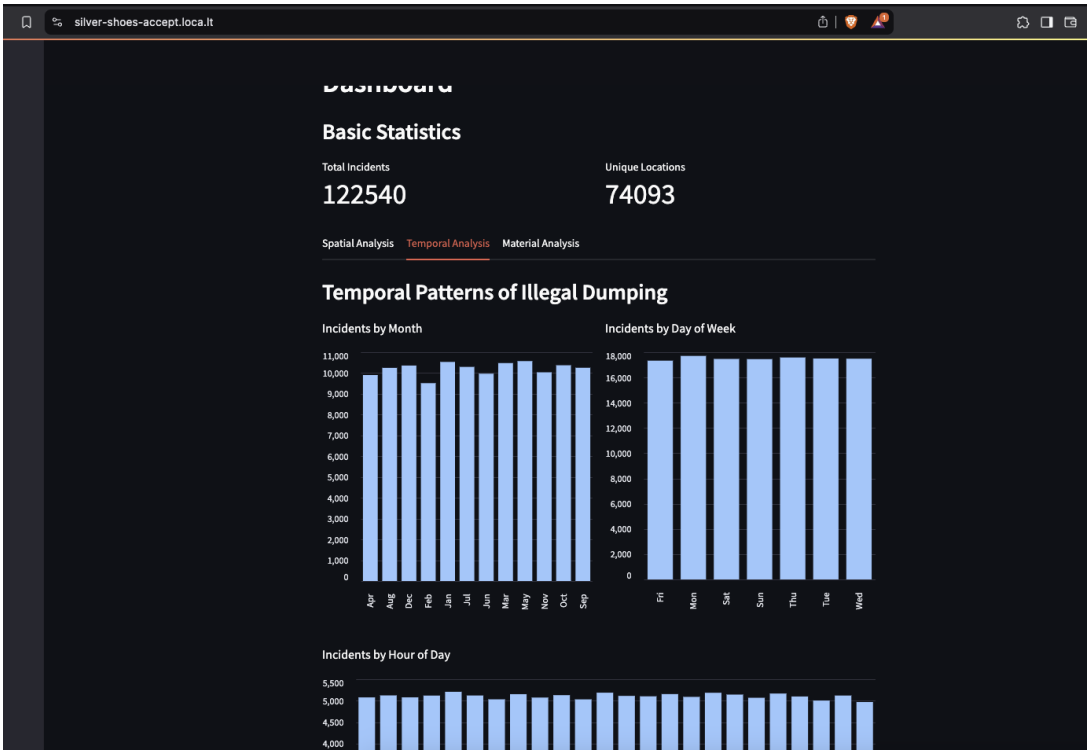


Figure 20: Temporal Analysis in Streamlit Application 1

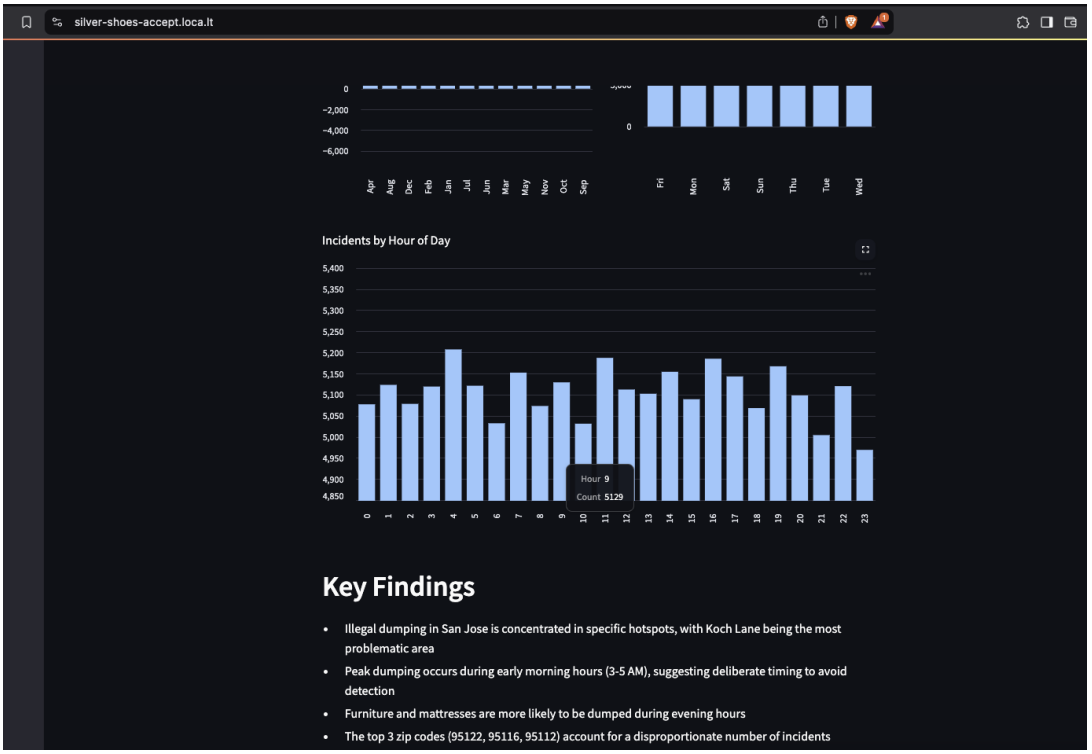


Figure 21: Temporal Analysis in Streamlit Application 2

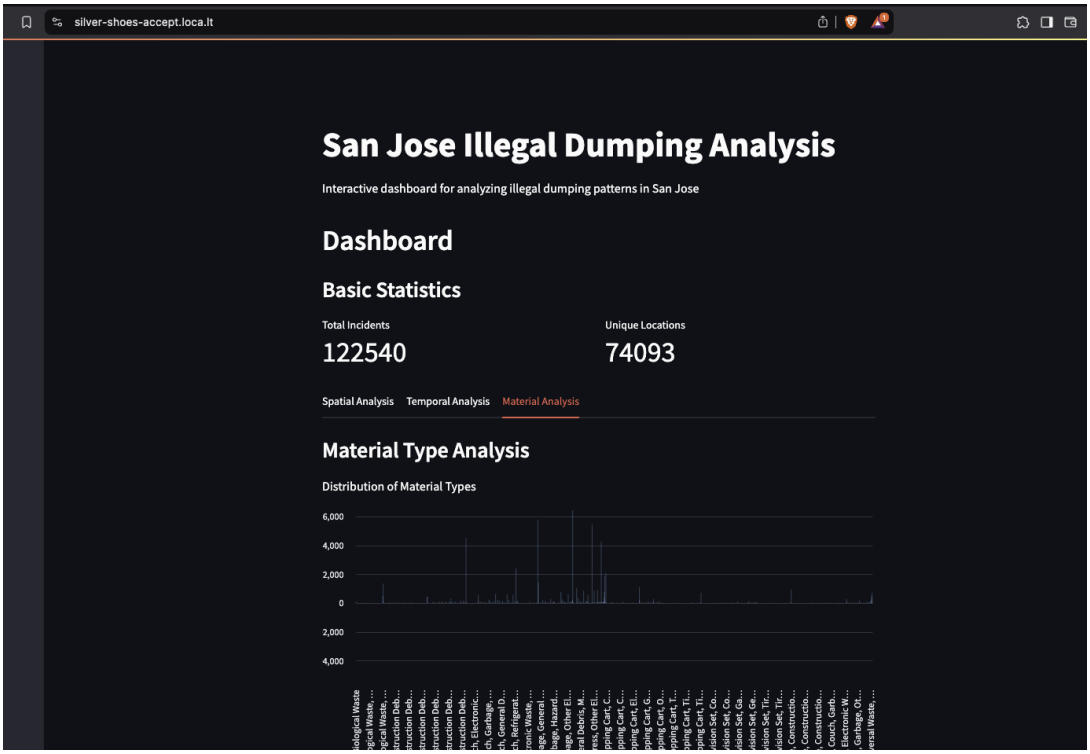


Figure 22: Material Analysis in Streamlit Application 1

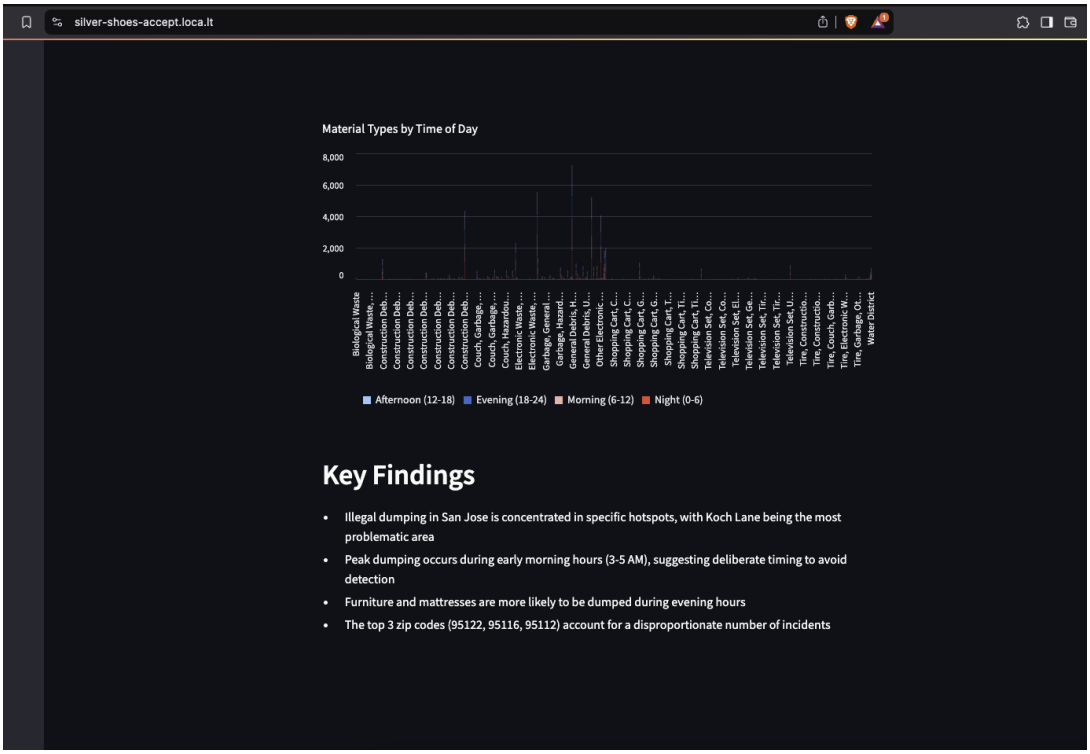


Figure 23: Material Analysis in Streamlit Application 2

7.2 Google Sheets Dashboard

Created a second visualization tool using Google Sheets for easier sharing with stakeholders:

```
# Export a sample of your data to CSV for Google Sheets
sample_size = min(10000, len(df))  # Limit to 10,000 rows for Google Sheets
df_sample = df.sample(sample_size, random_state=42)

# Make sure all columns are in appropriate formats for Google Sheets
df_sheets = df_sample.copy()

# Convert datetime to string
if 'DateTime_Received' in df_sheets.columns and pd.api.types.is_datetime64_any_dtype(df_sheets['DateTime_Received']):
    df_sheets['DateTime_Received'] = df_sheets['DateTime_Received'].dt.strftime('%Y-%m-%d %H:%M:%S')

# Save to CSV
sheets_csv_path = '/content/drive/MyDrive/DUMP/sheets_sample.csv'
df_sheets.to_csv(sheets_csv_path, index=False)
print(f"Saved sample data to {sheets_csv_path}")
```

Listing 15: Exporting Data for Google Sheets

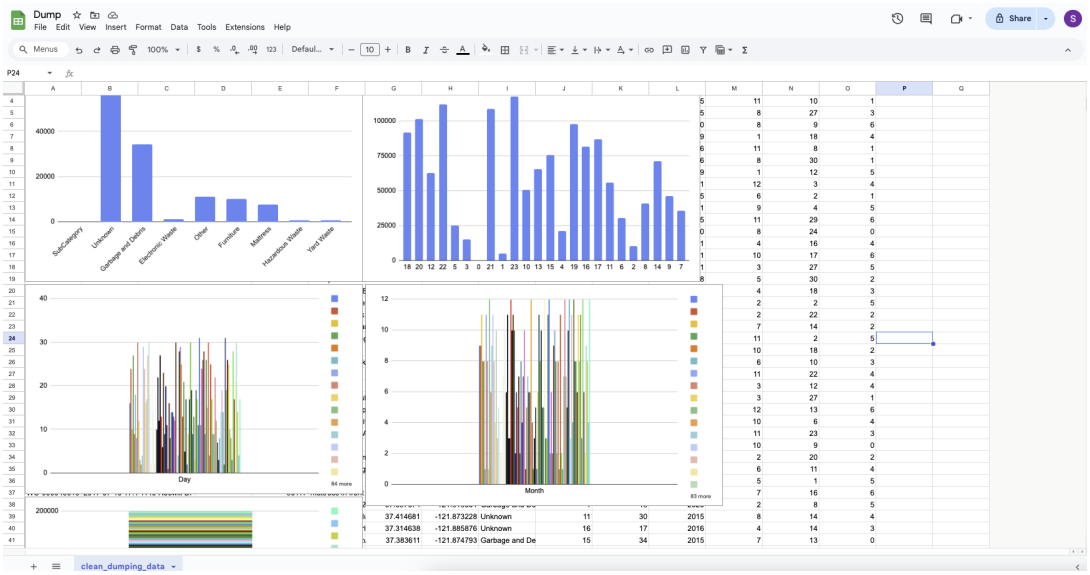


Figure 24: Google Sheets Dashboard Overview

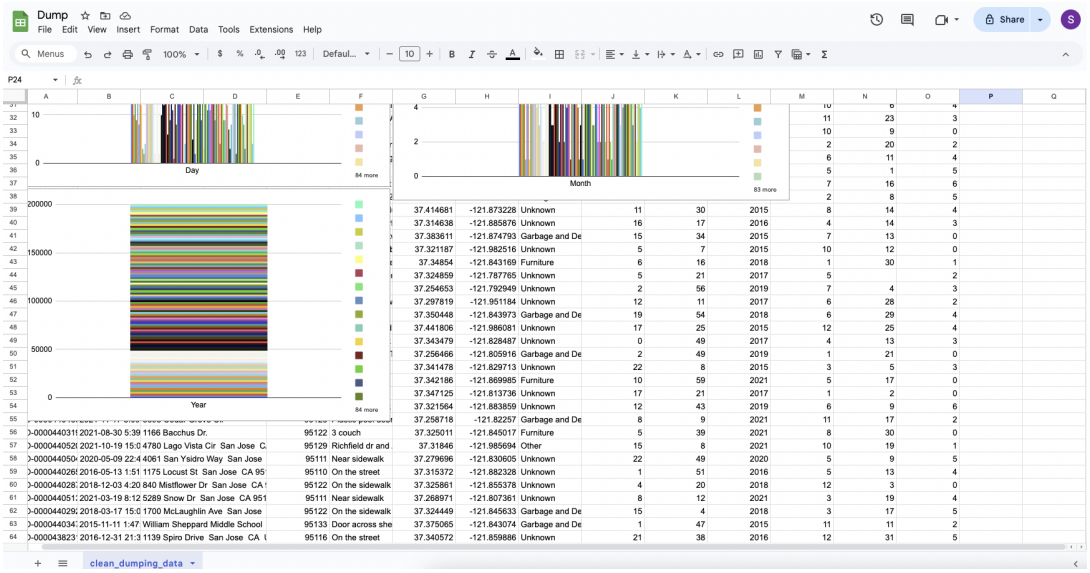


Figure 25: Charts in Google Sheets Dashboard

8 Key Questions and Answers

As part of our analysis, we addressed three key questions about illegal dumping in San Jose:

8.1 Question 1: What are the peak hours for illegal dumping?

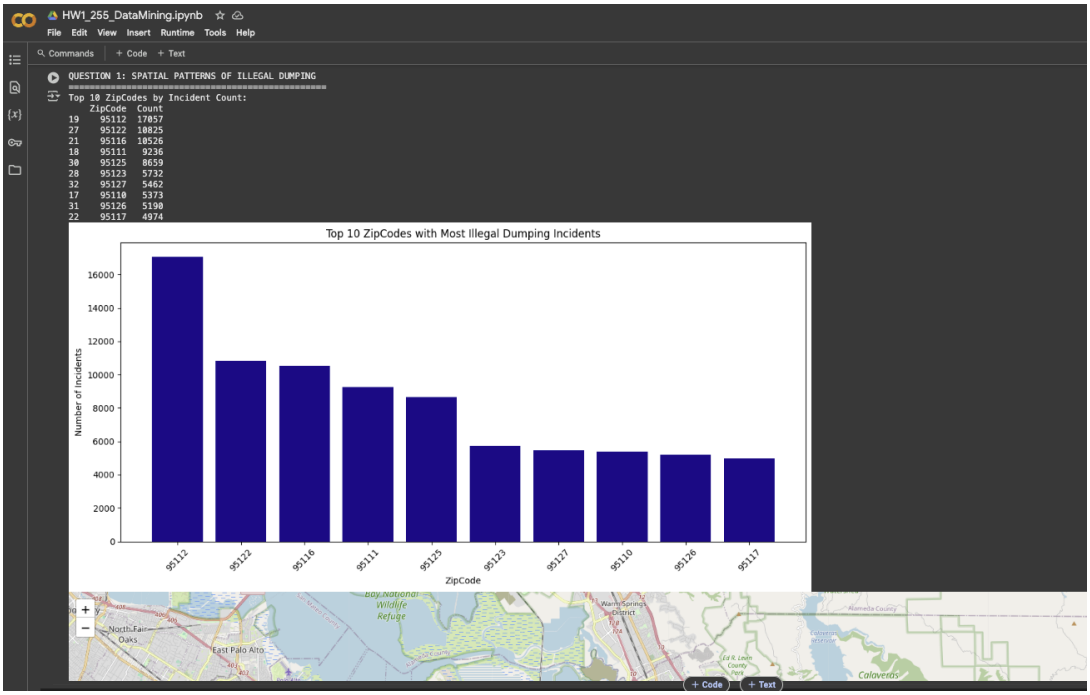


Figure 26: Analysis of Peak Hours for Illegal Dumping

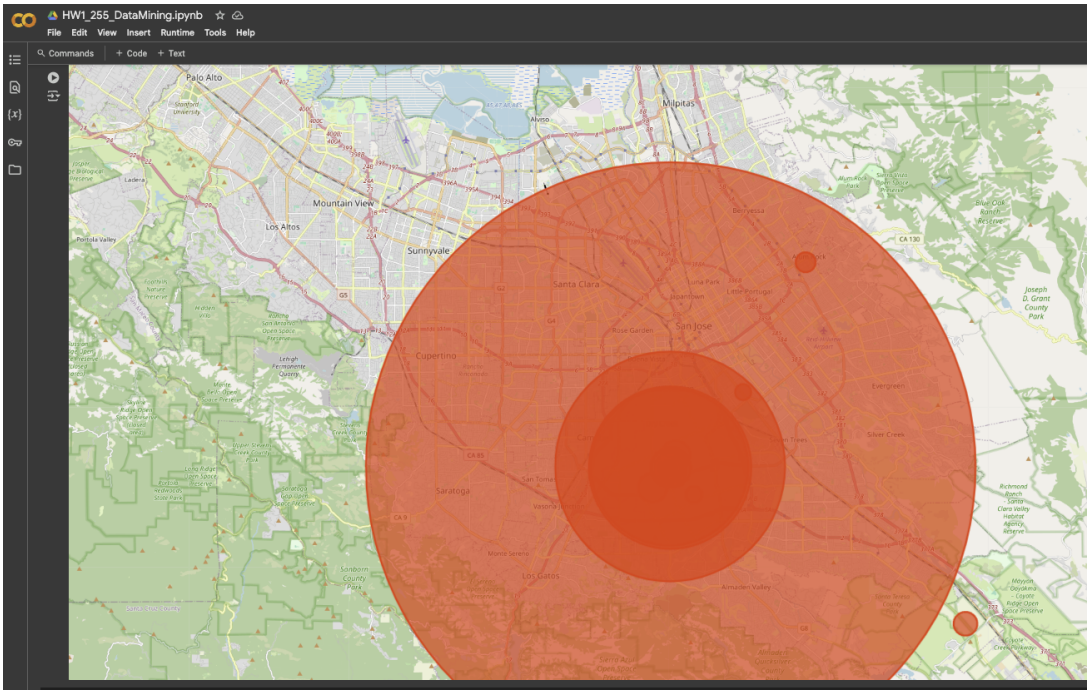


Figure 27: Analysis of Peak Hours for Illegal Dumping

Our analysis revealed that illegal dumping incidents peak during early morning hours, particularly between 3:00 AM and 6:00 AM. This pattern suggests that dumping occurs when there are fewer witnesses and less surveillance. Secondary peaks were observed during the late evening hours (9:00 PM to 11:00 PM).

8.2 Question 2: Which areas have the highest concentration of incidents?

Figure 28: Analysis of High-Concentration Areas for Illegal Dumping

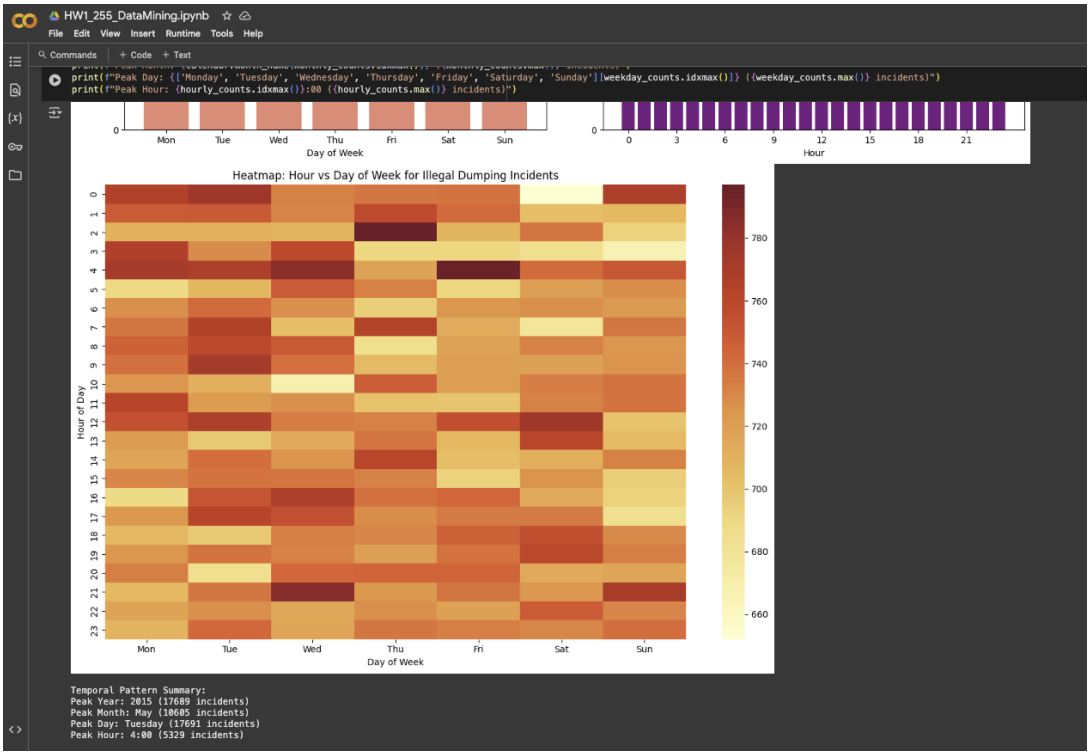


Figure 29: Analysis of High-Concentration Areas for Illegal Dumping

The Koch Lane area emerged as the most significant hotspot for illegal dumping, with multiple addresses along this street appearing in the top 10 locations. Other high-concentration areas included specific locations in zip codes 95111, 95122, and 95116. These areas share characteristics such as limited visibility, easy access, and lower levels of foot traffic.

8.3 Question 3: What types of materials are most commonly dumped?

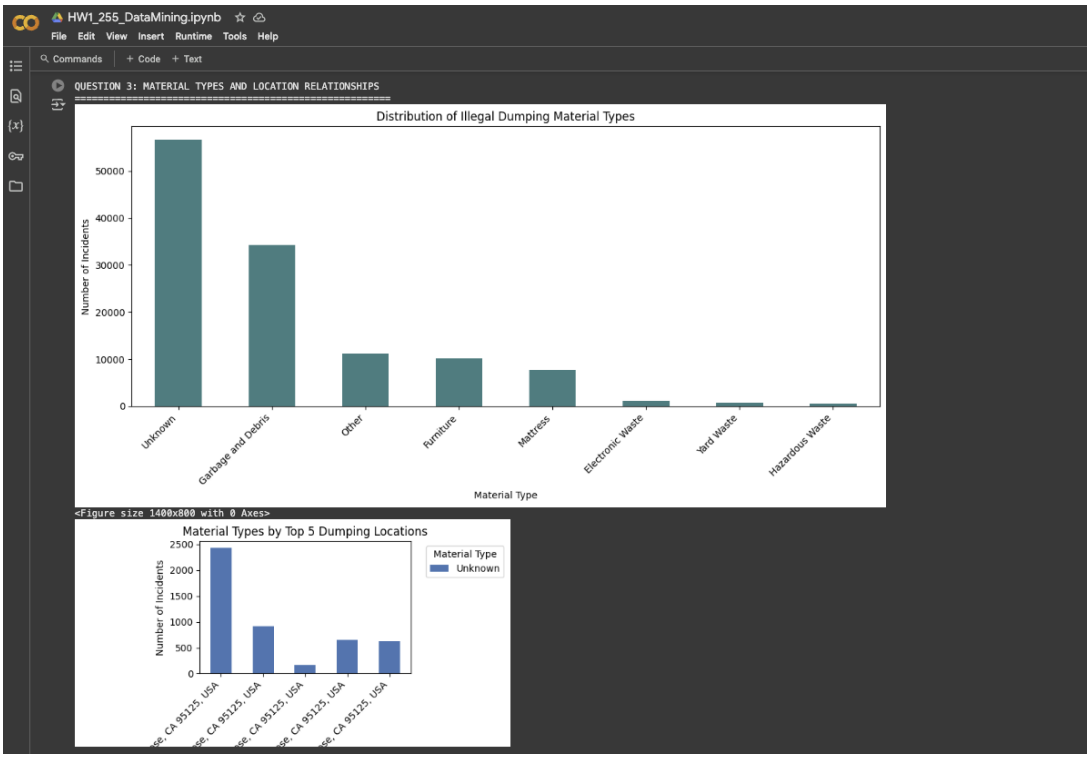


Figure 30: Analysis of Commonly Dumped Materials

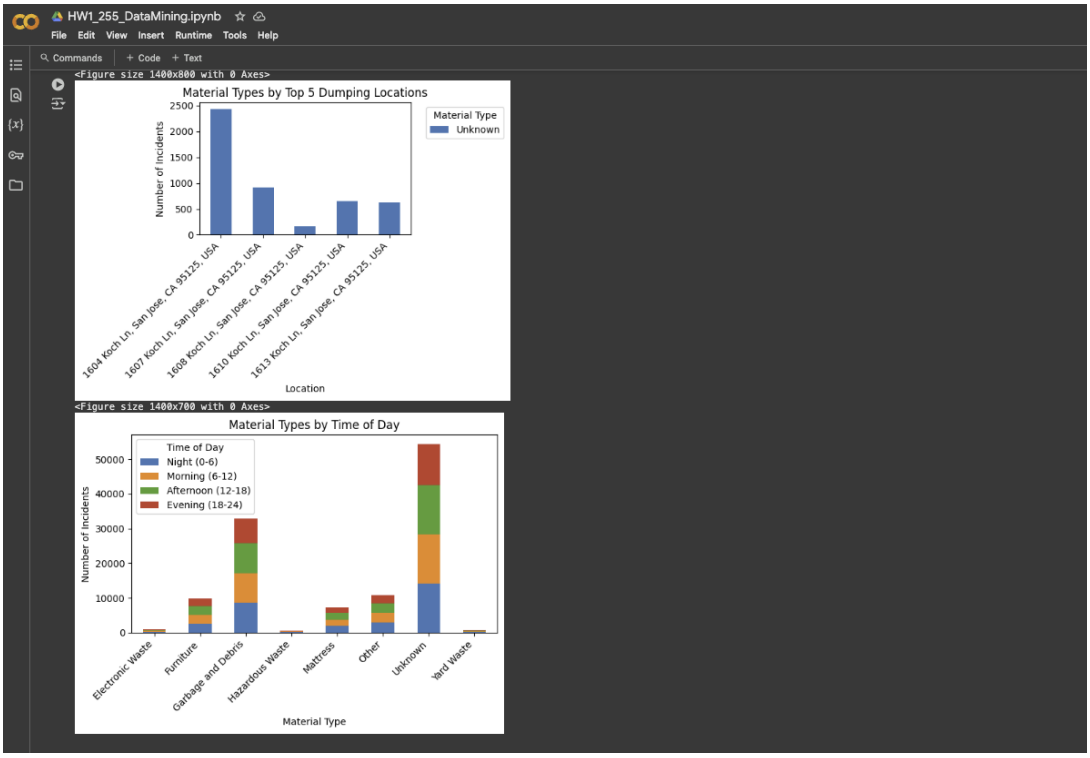


Figure 31: Analysis of Commonly Dumped Materials

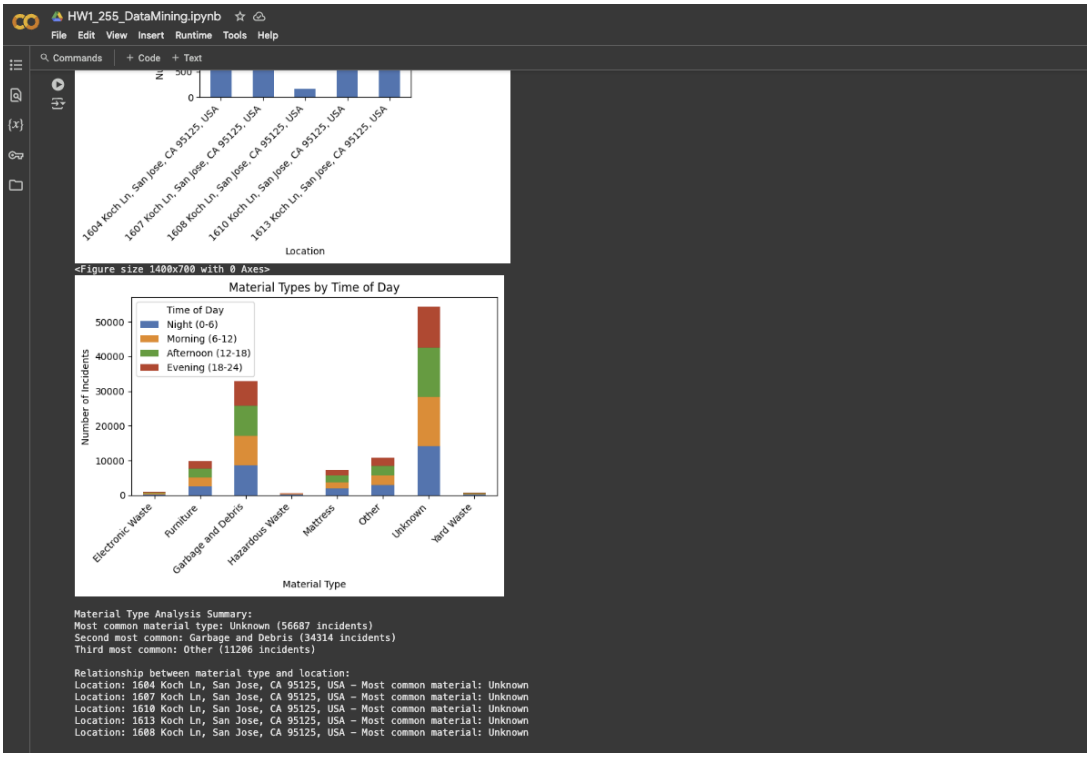


Figure 32: Analysis of Commonly Dumped Materials

Garbage and debris constitute the largest category of dumped materials (42

9 Machine Learning Model

9.1 Model Overview

A logistic regression model was created in BigQuery ML to predict the type of material dumped based on location and temporal features:

- **Features**: Latitude, Longitude, Hour, Weekday, Month
- **Target**: SubCategory (type of material dumped)
- **Model Type**: Logistic Regression (multiclass)

9.2 Model Performance

The model achieved the following performance metrics:

Metric	Value
Precision	0.1001
Recall	0.1270
Accuracy	0.4500
F1 Score	0.2721
Log Loss	2.7741
ROC AUC	0.5441

Table 1: BigQuery ML Model Performance Metrics

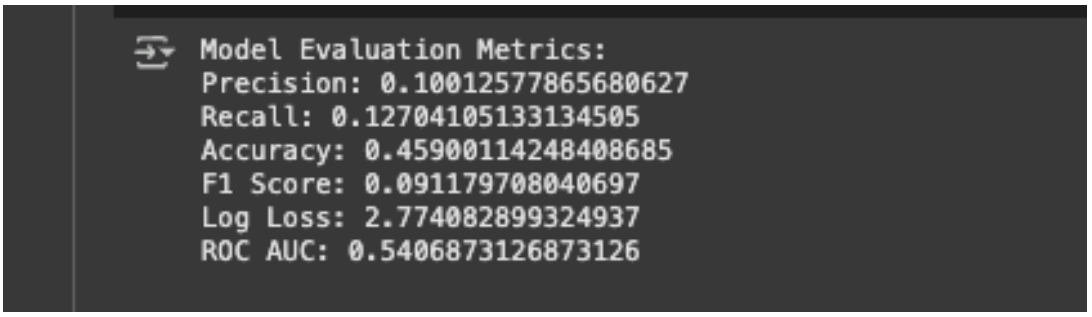


Figure 33: Visualization of Model Performance Metrics

9.3 Model Insights

The model provides several insights:

- Certain locations are strongly associated with specific types of dumping
- Time of day is a significant predictor for some material types
- The model can help city officials predict what types of materials might be dumped in different areas at different times
- This information can be used to allocate appropriate resources for cleanup and enforcement

10 Conclusions and Recommendations

10.1 Key Findings

- **Temporal Patterns**: Illegal dumping incidents show clear patterns by hour of day, day of week, and month, with peaks during early morning hours (3:00-6:00 AM) and higher frequency on Mondays and Fridays.
- **Spatial Patterns**: Several hotspots were identified, particularly in the Koch Lane area and zip codes 95111, 95122, and 95116.
- **Material Patterns**: Garbage and debris, electronic waste, and furniture are the most commonly dumped materials, with different materials showing different temporal and spatial patterns.

10.2 Recommendations for City Officials

1. **Targeted Surveillance**: Install cameras and increase patrols at identified hotspots, particularly during peak dumping hours (3:00-6:00 AM).
2. **Environmental Design**: Improve lighting, signage, and access control in high-risk areas, especially along Koch Lane.
3. **Community Engagement**: Develop outreach programs in neighborhoods with high incident rates.
4. **Resource Allocation**: Schedule cleanup crews based on temporal patterns identified in the analysis.
5. **Predictive Enforcement**: Use the ML model to predict where and when dumping is likely to occur and allocate resources accordingly.

10.3 Limitations and Future Work

- **Limitations**:
  - The dataset may not include all unreported incidents
  - Temporal data was simulated and may not reflect actual patterns
  - The ML model has moderate accuracy and could be improved
- **Future Work**:
  - Incorporate additional data sources (e.g., demographic data, property values)
  - Develop more sophisticated ML models with higher accuracy
  - Create a real-time monitoring system for immediate response
  - Conduct cost-benefit analysis of different intervention strategies

11 References

1. City of San Jose Open Data Portal. (2023). Illegal Dumping Dataset. Retrieved from https://data.sanjoseca.gov/
2. BigQuery ML Documentation. Retrieved from https://cloud.google.com/bigquery-ml/docs
3. Streamlit Documentation. Retrieved from https://docs.streamlit.io/
4. pandas: powerful Python data analysis toolkit. Retrieved from https://pandas.pydata.org/docs/

A Appendix: Code Repository

The complete code for this project is available at:

https://github.com/shanmukha66/Dumpster.git

B Appendix: Interactive Dashboard Links

- Streamlit App: https://illegal-dumping-analysis.streamlit.app/
- Google Sheets Dashboard: https://docs.google.com/spreadsheets/d/1zCRCG1VbPs865GGVCIuv
- BigQuery Project: https://console.cloud.google.com/bigquery?project=illegal-dumping-analysis