Clustering

May 10, 2025

[1]:
```
import os
import pandas as pd
os.chdir("D:\Liuyifei\McGill\INSY662\Group Project")
df = pd.read_excel('processed_df_1112.xlsx')
```

1 Clustering based on Longtitude and Latitude

[2]:
```
severity_mapping = {
    'Property damage below reporting threshold': 0,
    'Property damage only': 1,
    'Minor': 2,
    'Serious': 3,
    'Fatal': 4
}

df["Severity_num"] = df["Severity"].map(severity_mapping)
```

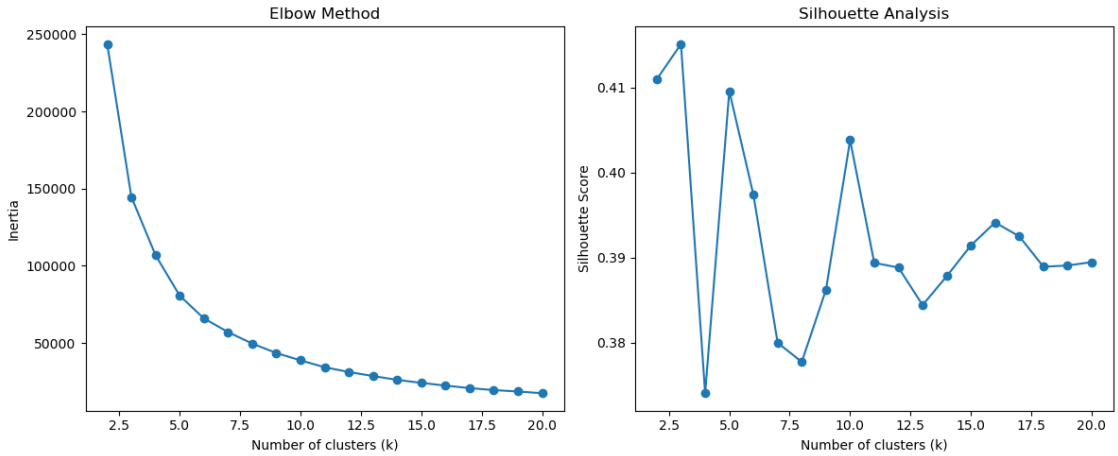
[3]:
```
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import folium
from folium.plugins import HeatMap
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import LinearSegmentedColormap

# 1. Prepare location data
print("Preparing location data...")
X = df[['Latitude', 'Lontitude']].values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 2. Determine optimal number of clusters using elbow method
print("Finding optimal number of clusters...")
inertias = []
silhouette_scores = []
```

```python
max_clusters = 20

plt.figure(figsize=(12, 5))
for k in range(2, max_clusters + 1):
    print(f"Testing k={k}...")
    kmeans = KMeans(n_clusters=k, random_state=0)
    kmeans.fit(X_scaled)
    inertias.append(kmeans.inertia_)
    score = silhouette_score(X_scaled, kmeans.labels_, sample_size=20000)
    silhouette_scores.append(score)

# Plot elbow curve
plt.subplot(1, 2, 1)
plt.plot(range(2, max_clusters + 1), inertias, marker='o')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method')

plt.subplot(1, 2, 2)
plt.plot(range(2, max_clusters + 1), silhouette_scores, marker='o')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Analysis')

plt.tight_layout()
plt.show()
```

```
Preparing location data…
Finding optimal number of clusters…
Testing k=2…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=3…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=4…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=5…
```

```
d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=6…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=7…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=8…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=9…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=10…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=11…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=12…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=13…
```

```
d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=14…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=15…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=16…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=17…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=18…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=19…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=20…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



[4]:
```
# 3. Perform clustering with optimal k=10
optimal_k = 10  # Adjust this based on the elbow curve results
kmeans = KMeans(n_clusters=optimal_k, random_state=0)
labels = kmeans.fit_predict(X_scaled)

# 4. Add cluster labels to the original dataframe
df['Cluster'] = labels

# 5. Calculate cluster summaries
print("Calculating cluster summaries...")
cluster_summaries = []
features_to_summarize = ['Latitude', 'Lontitude', 'Total_Victims',␣
 ↪'Severity_num', 'Severity',
                         'Surface_Cond', 'Weather_Cond', 'Light_Cond',
                         'Road_Config']

for i in range(optimal_k):
    cluster_data = df[df['Cluster'] == i]

    # Get geographical center
    center = kmeans.cluster_centers_[i]
    center_unscaled = scaler.inverse_transform([center])[0]

    summary = {
        'Cluster': i,
        'Size': len(cluster_data),
        'Latitude': center_unscaled[0],
        'Longitude': center_unscaled[1],
        'Avg_Victims': cluster_data['Total_Victims'].mean(),
        'Avg_Severity': cluster_data['Severity_num'].mean(),
        'Most_Common_Severity': cluster_data['Severity'].mode().iloc[0],
```

```
        'Most_Common_Surface': cluster_data['Surface_Cond'].mode().iloc[0],
        'Most_Common_Weather': cluster_data['Weather_Cond'].mode().iloc[0],
        'Most_Common_Light': cluster_data['Light_Cond'].mode().iloc[0],
        'Most_Common_Road_Config': cluster_data['Road_Config'].mode().iloc[0]
    }
    cluster_summaries.append(summary)

cluster_df = pd.DataFrame(cluster_summaries)
print("\nCluster Summaries:")
print(cluster_df)

# 6. Create map visualization with clear legend

colors = ['blue', 'green', 'purple', 'orange', "yellow"]
cluster_names = [f"Cluster {i}" for i in range(optimal_k)]

# Create base map
montreal_map = folium.Map(location=[45.5017, -73.5673], zoom_start=11)

# Add cluster centers with consistent colors and legend
for idx, row in cluster_df.iterrows():
    color = colors[idx % len(colors)]

    # Add cluster center marker
    folium.CircleMarker(
        location=[row['Latitude'], row['Longitude']],
        radius=20,
        popup=f"""
        <b>Cluster {row['Cluster']}</b><br>
        Size: {row['Size']}<br>
        Avg Victims: {row['Avg_Victims']:.2f}<br>
        Avg Severity: {row['Avg_Severity']:.2f}<br>
        Most Common Severity: {row['Most_Common_Severity']}<br>
        Most Common Surface: {row['Most_Common_Surface']}<br>
        Most Common Weather: {row['Most_Common_Weather']}
        """,
        color=color,
        fill=True,
        fill_opacity=0.7
    ).add_to(montreal_map)

# Add a legend
legend_html = """
<div style="position: fixed; bottom: 50px; left: 50px; z-index: 1000;␣
 ↪background-color: white;
            padding: 10px; border: 2px solid grey; border-radius: 5px;">
<h4>Cluster Legend</h4>
```

```
"""

for i, color in enumerate(colors[:optimal_k]):
    legend_html += f'<p><span style="color:{color};"> </span> Cluster {i}</p>'

legend_html += "</div>"
montreal_map.get_root().html.add_child(folium.Element(legend_html))

# Save the map
montreal_map.save('montreal_traffic_clusters_with_legend.html')

# 7. Create a scatter plot of clusters
# Create custom color map that matches your desired colors
colors = ['blue', 'purple', 'orange', 'green', 'pink', 'navy', 'gray', 'black',␣
 ↪'yellow', 'cyan']  # Colors in the order you want for clusters 0-5
custom_cmap = LinearSegmentedColormap.from_list('custom', colors, N=len(colors))

# Create the scatter plot
plt.figure(figsize=(12, 8))
scatter = plt.scatter(df['Lontitude'], df['Latitude'],
                      c=df['Cluster'],
                      cmap=custom_cmap,
                      alpha=0.6,
                      s=30)

# Customize the plot
plt.title('Traffic Accident Clusters in Montreal', fontsize=14, pad=15)
plt.xlabel('Longitude', fontsize=12)
plt.ylabel('Latitude', fontsize=12)

# Create custom colorbar with integer ticks
cbar = plt.colorbar(scatter, ticks=np.arange(len(colors)))
cbar.set_label('Cluster', fontsize=12, labelpad=10)
cbar.ax.set_yticklabels([f'Cluster {i}' for i in range(len(colors))])

# Adjust layout
plt.tight_layout()

# Show the plot
plt.show()
```

```
d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

```
Calculating cluster summaries…
```

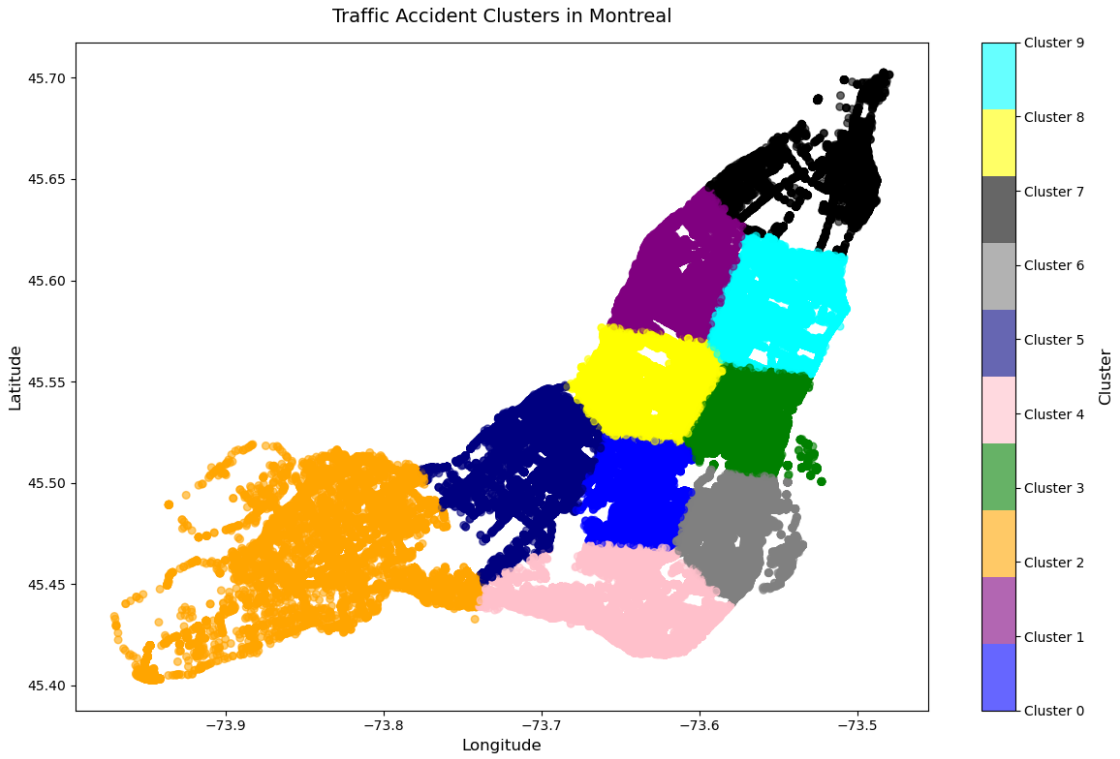
Cluster Summaries:

	Cluster	Size	Latitude	Longitude	Avg_Victims	Avg_Severity
0	0	17864	45.494936	-73.637029	0.249608	0.796294
1	1	20585	45.599243	-73.615954	0.331552	0.906097
2	2	17769	45.468631	-73.827438	0.250999	0.824301
3	3	37183	45.527425	-73.567671	0.270661	0.816771
4	4	17643	45.442660	-73.642526	0.251715	0.800091
5	5	15201	45.511499	-73.702999	0.250049	0.835142
6	6	26965	45.482987	-73.577361	0.232783	0.760319
7	7	7653	45.651220	-73.526110	0.295048	0.905266
8	8	34477	45.546990	-73.628409	0.278272	0.853845
9	9	22788	45.584779	-73.554962	0.296560	0.868703

	Most_Common_Severity	Most_Common_Surface
0	Property damage below reporting threshold	Surface_Cond_Severity_level3
1	Property damage only	Surface_Cond_Severity_level3
2	Property damage only	Surface_Cond_Severity_level3
3	Property damage below reporting threshold	Surface_Cond_Severity_level3
4	Property damage below reporting threshold	Surface_Cond_Severity_level3
5	Property damage only	Surface_Cond_Severity_level3
6	Property damage below reporting threshold	Surface_Cond_Severity_level3
7	Property damage only	Surface_Cond_Severity_level3
8	Property damage only	Surface_Cond_Severity_level3
9	Property damage below reporting threshold	Surface_Cond_Severity_level3

	Most_Common_Weather	Most_Common_Light
0	Weather_Cond_Severity_level4	Light_Cond_Severity_level4
1	Weather_Cond_Severity_level4	Light_Cond_Severity_level4
2	Weather_Cond_Severity_level4	Light_Cond_Severity_level4
3	Weather_Cond_Severity_level4	Light_Cond_Severity_level4
4	Weather_Cond_Severity_level4	Light_Cond_Severity_level4
5	Weather_Cond_Severity_level4	Light_Cond_Severity_level4
6	Weather_Cond_Severity_level4	Light_Cond_Severity_level4
7	Weather_Cond_Severity_level4	Light_Cond_Severity_level4
8	Weather_Cond_Severity_level4	Light_Cond_Severity_level4
9	Weather_Cond_Severity_level4	Light_Cond_Severity_level4

	Most_Common_Road_Config
0	Road_Config_Severity_level1
1	Road_Config_Severity_level2
2	Road_Config_Severity_level2
3	Road_Config_Severity_level1
4	Road_Config_Severity_level2
5	Road_Config_Severity_level2
6	Road_Config_Severity_level1
7	Road_Config_Severity_level3
8	Road_Config_Severity_level1
9	Road_Config_Severity_level3



The severity categories in the Appendix (Table 1:

- 0 = Property damage below reporting threshold
- 1 = Property damage only
- 2 = Minor injury
- 3 = Serious injury
- 4 = Fatal).

1.0.1 1. Cluster-Level Severity Profiles

Cluster	Center (Lat, Lon)	Size	Avg Severity	Most Common Severity	Appendix Code
0	(45.495, –73.637)	17 864	0.80	Below threshold (0)	0
1	(45.599, –73.616)	20 585	**0.91**	Property damage only (1)	1
2	(45.469, –73.827)	17 769	0.82	Property damage only (1)	1
3	(45.527, –73.568)	37 183	0.82	Below threshold (0)	0
4	(45.443, –73.643)	17 643	0.80	Below threshold (0)	0
5	(45.511, –73.703)	15 201	0.84	Property damage only (1)	1
6	(45.483, –73.577)	26 965	**0.76**	Below threshold (0)	0
7	(45.651, –73.526)	7 653	**0.91**	Property damage only (1)	1
8	(45.547, –73.628)	34 477	0.85	Property damage only (1)	1
9	(45.585, –73.555)	22 788	0.87	Below threshold (0)	0

- **Clusters 1 & 7** stand out with the **highest avg severity (~0.91)**, driven by a preponderance of class 1 ("property damage only") collisions. Geographically these lie in the northern and north-western sectors—likely busier arterials or commercial corridors where damage-only crashes are common.
- **Cluster 6** (southern residential belt) has the **lowest avg severity (0.76)** and is dominated by class 0 ("below threshold") events, marking it as the city's safest pocket.
- The remaining clusters split roughly into:
  - **Five "low-impact" zones** (0, 3, 4, 6, 9) dominated by class 0;
  - **Five "damage-only" zones** (1, 2, 5, 7, 8) dominated by class 1.

1.0.2 2. Victims & High-Severity Rarity

- **Avg Victims** in every cluster sits at **0.23–0.33**, confirming that multi-victim or injury crashes are rare across Montreal.
- Clusters with higher avg severity do *not* correspond to more victims—suggesting that these areas experience more frequent damage-only incidents rather than a spike in severe injuries.

1.0.3 3. Environmental Uniformity

- **Surface_Cond** (mode = level 3) and **Weather_Cond** (mode = level 4) are identical across *all* clusters—implying that dry, clear conditions prevail city-wide and are not the primary drivers of inter-area severity differences.
- **Light_Cond** similarly shows level 4 ("daylight") everywhere, so focus your interventions on geometry and traffic patterns rather than lighting or weather.

1.0.4 4. Road Configuration Patterns

Cluster	Road_Config Mode	Likely Road Type
1,2,4,5,8	level 2	Multi-lane arterials
0,3,6	level 1	Two-lane residential roads
7,9	level 3	Highways or complex nodes

- Clusters on **arterials/highways** (particularly 1 & 7) coincide with the **highest avg severity**, pointing to the value of targeted traffic-calming or speed-management solutions there.
- **Residential clusters** (0,3,4,6) see the *lowest* severities and can serve as templates for safer street design.

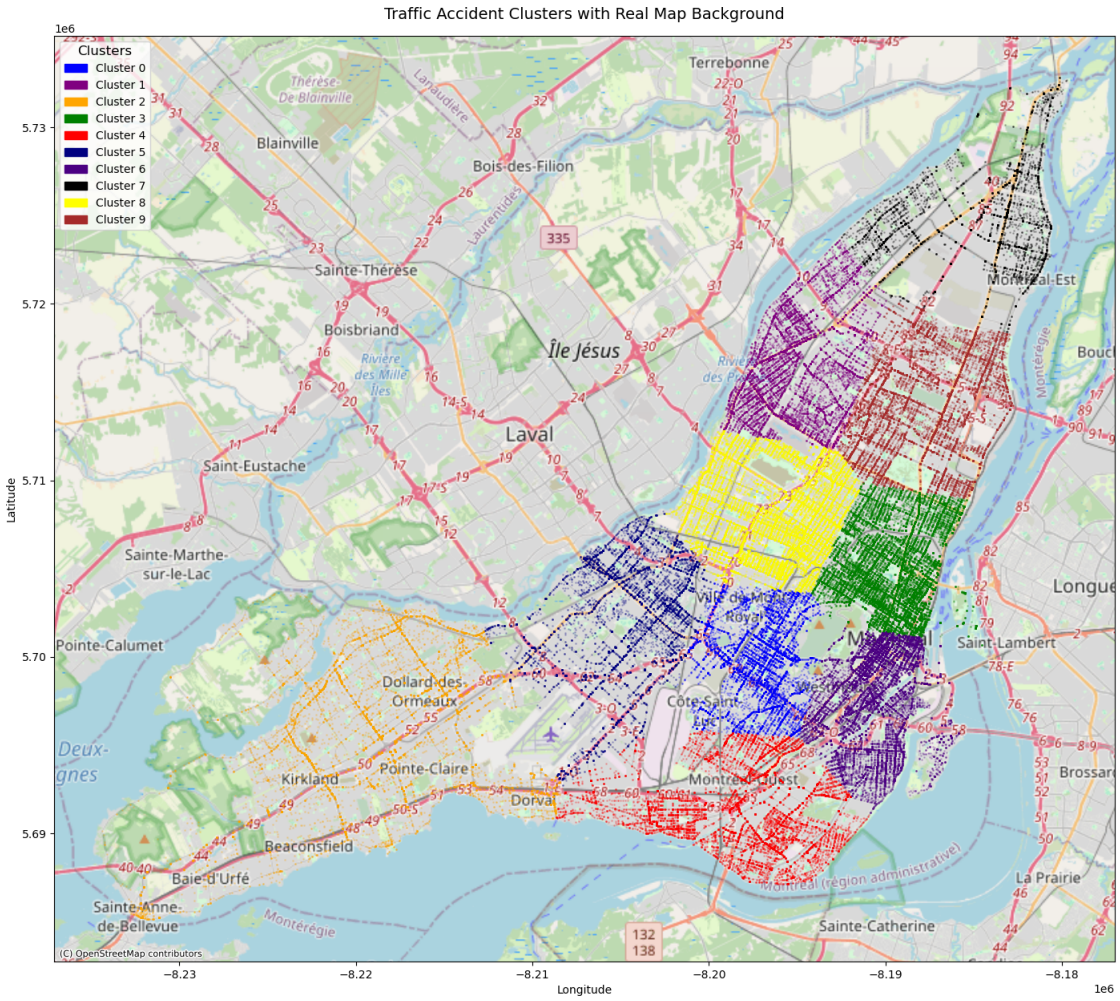
1.0.5 5. Actionable Takeaways

1. **Priority zones**: Clusters 1 & 7 for "damage-only" crash reduction (speed bumps, turn restrictions).
2. **Best-practice benchmark**: Cluster 6's residential layout could inform safety audits in higher-severity areas.
3. **Resource allocation**: Since environmental factors are constant, invest in **geometric** improvements (lane markings, signage) and **enforcement** on high-config roads rather than weather/lighting upgrades.

4. **Downstream modeling**: Use your `Cluster` label as a categorical feature in severity classifiers—these spatial groupings capture non-linear location effects that raw lat/long miss.

[14]:
```
import matplotlib.pyplot as plt
import geopandas as gpd
import contextily as ctx
import matplotlib.patches as mpatches
from matplotlib.colors import Normalize

# Convert DataFrame to GeoDataFrame
gdf = gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(df['Lontitude'],␣
 ↪df['Latitude']))
gdf.set_crs(epsg=4326, inplace=True)  # Set to WGS84
gdf = gdf.to_crs(epsg=3857)  # Convert to Web Mercator for compatibility with␣
 ↪basemaps

# Create a custom color map
colors = ['blue', 'purple', 'orange', 'green', 'red', 'navy', 'indigo',␣
 ↪'black', 'yellow', 'brown']
unique_clusters = sorted(gdf['Cluster'].unique())  # Sort clusters
norm = Normalize(vmin=unique_clusters[0], vmax=unique_clusters[-1])  #␣
 ↪Normalize based on sorted values
custom_cmap = {cluster: colors[idx % len(colors)] for idx, cluster in␣
 ↪enumerate(unique_clusters)}  # Map clusters to colors

# Plot clusters with real map background
fig, ax = plt.subplots(figsize=(20, 15))
for cluster, color in custom_cmap.items():
    cluster_data = gdf[gdf['Cluster'] == cluster]
    cluster_data.plot(ax=ax, color=color, label=f"Cluster {cluster}",␣
 ↪markersize=1, alpha=0.2)

# Add basemap
ctx.add_basemap(ax, source=ctx.providers.OpenStreetMap.Mapnik)  # Add real map␣
 ↪background
ax.set_title("Traffic Accident Clusters with Real Map Background", fontsize=14,␣
 ↪pad=15)
plt.xlabel("Longitude")
plt.ylabel("Latitude")

# Add legend
patches = [mpatches.Patch(color=color, label=f"Cluster {cluster}") for cluster,␣
 ↪color in custom_cmap.items()]
ax.legend(handles=patches, title="Clusters", fontsize=10, title_fontsize=12,␣
 ↪loc='upper left')

# Save and display
```

```
plt.savefig("traffic_accident_clusters_with_map.png", dpi=300,␣
  ↪bbox_inches='tight')
plt.show()
```



1.1 Geospatial Extent of Each Cluster

- **Clusters 0 & 5 (Blue & Navy)** sit on the **downtown island** around Old Montreal and Ville-Marie
- **Cluster 2 (Orange)** fans out over the **West Island** suburbs (Dollard-des-Ormeaux, Kirkland)
- **Cluster 3 (Green)** maps to the **South Shore via the Champlain Bridge**
- **Cluster 4 (Red)** covers the **downtown industrial port area (Vieux-Port, Griffintown)**
- **Clusters 1 & 8 (Purple & Yellow)** trace the **Plateau–Mile-End** and **Rosemont–La Petite-Patrie** boroughs
- **Clusters 6, 7 & 9 (Black, Indigo, Brown)** occupy the **North Shore** (Laval) and **eastern**

boroughs (Mercier–Hochelaga).

1.2 Tying Clusters Back to Severity & Traffic Patterns

Overlay any of these maps with a **heatmap of accident counts** or **average severity** and you'll likely find:

- **Downtown (Clusters 0,5)**: High sheer volume but lower avg severity → mostly fender-benders in grid traffic.
- **West Island (Cluster 2)**: Moderate volume, low severity → residential side-streets.
- **Industrial corridor (Cluster 4)** & **Plateau (1,8)**: Lower counts but higher avg severity → faster arterials and transit-rich streets.
- **North Shore (6,7,9)**: Varying patterns—Cluster 7's tight purple region up north has the **highest avg severity** among the Laval clusters.

This spatial context lets us **prioritize** which areas need engineering vs. enforcement vs. public outreach.

1.3 From Map to Policy

With our clusters validated on a real-world canvas:

- **Targeted interventions**: e.g. if Cluster 4 (Griffintown/port) shows high avg severity despite low counts, add signage or redesign the slip-ramp.
- **Community engagement**: share cluster maps with borough councils to co-design speed-calming in their specific zones.
- **Further layering**: overlay **bus routes**, **bike lanes**, or **school zones** to see where multi-modal conflicts are concentrated within each cluster.

2 Based on Severity levels

[30]:
```
df_2 = pd.get_dummies(df, columns=['Severity'], prefix='Sev')
```

[31]:
```
df_2.columns
```

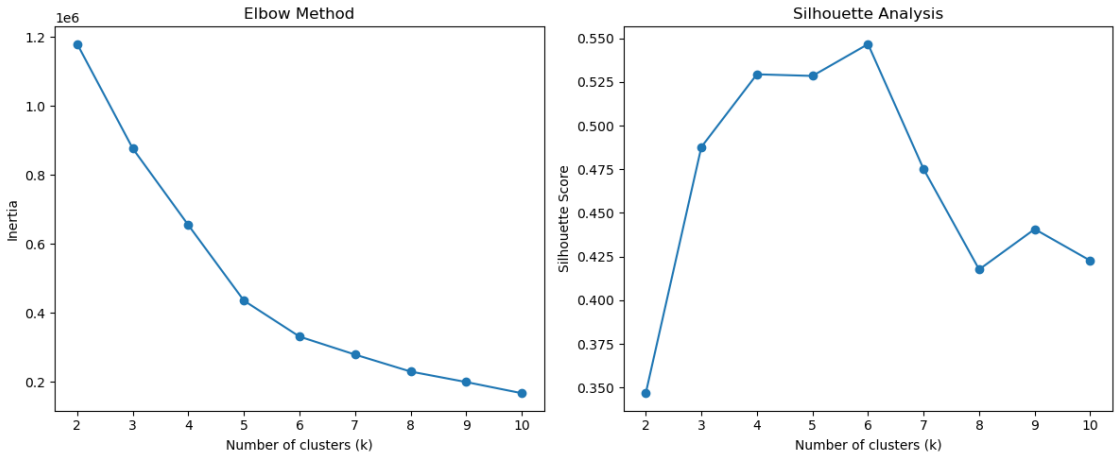
[31]:
```
Index(['Unnamed: 0', 'Acc_Date', 'Acc_Time', 'Acc_Type', 'Environ_Type',
       'Latitude', 'Light_Cond', 'Loc_Code', 'Lontitude', 'Month', 'Num_Bike',
       'Num_Bus', 'Num_Emerg', 'Num_Equip', 'Num_Heavy_Truck', 'Num_Light_Veh',
       'Num_Moped', 'Num_Moto', 'Num_Other_Veh', 'Num_Taxi', 'Num_Unspec_Veh',
       'Road_Aspect', 'Road_Cat', 'Road_Config', 'Speed_Limit', 'Surface_Cond',
       'Weather_Cond', 'Weekday', 'Year', 'Num_Veh_Invld', 'Total_Victims',
       'Credibility_Score', 'Cluster', 'Severity_num', 'Sev_Fatal',
       'Sev_Minor', 'Sev_Property damage below reporting threshold',
       'Sev_Property damage only', 'Sev_Serious'],
      dtype='object')
```

[32]:
```
# 1. Prepare location data

X = df_2[['Sev_Fatal', 'Sev_Minor',
```

```
       'Sev_Property damage below reporting threshold',
       'Sev_Property damage only', 'Sev_Serious', 'Lontitude', 'Latitude']].
 ↪values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 2. Determine optimal number of clusters using elbow method
print("Finding optimal number of clusters...")
inertias = []
silhouette_scores = []
max_clusters = 10

plt.figure(figsize=(12, 5))
for k in range(2, max_clusters + 1):
    print(f"Testing k={k}...")
    kmeans = KMeans(n_clusters=k, random_state=0)
    kmeans.fit(X_scaled)
    inertias.append(kmeans.inertia_)
    score = silhouette_score(X_scaled, kmeans.labels_, sample_size=10000)
    silhouette_scores.append(score)

# Plot elbow curve
plt.subplot(1, 2, 1)
plt.plot(range(2, max_clusters + 1), inertias, marker='o')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method')

plt.subplot(1, 2, 2)
plt.plot(range(2, max_clusters + 1), silhouette_scores, marker='o')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Analysis')

plt.tight_layout()
plt.show()
```

```
Finding optimal number of clusters…
Testing k=2…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=3…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
```

```
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=4…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=5…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=6…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=7…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=8…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=9…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=10…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



[39]:
```
# 3. Perform clustering with optimal k=5
optimal_k = 6  # Adjust this based on the elbow curve results
kmeans = KMeans(n_clusters=optimal_k, random_state=0)
labels = kmeans.fit_predict(X_scaled)

# 4. Add cluster labels to the original dataframe
df_2['Cluster'] = labels

# 5. Calculate cluster summaries

cluster_summaries = []
features_to_summarize = ['Latitude', 'Lontitude', 'Total_Victims',
                         'Surface_Cond', 'Weather_Cond', 'Light_Cond',
                         'Road_Config', 'Sev_Fatal', 'Sev_Minor',
                         'Sev_Property damage below reporting threshold',
                         'Sev_Property damage only', 'Sev_Serious']

for i in range(optimal_k):
    cluster_data = df_2[df_2['Cluster'] == i]

    # Get geographical center
    center = kmeans.cluster_centers_[i]
    center_unscaled = scaler.inverse_transform([center])[0]

    summary = {
        'Cluster': i,
        'Size': len(cluster_data),
        'Latitude': center_unscaled[0],
        'Longitude': center_unscaled[1],
        'Avg_Victims': cluster_data['Total_Victims'].mean(),
        'Most_Common_Surface': cluster_data['Surface_Cond'].mode().iloc[0],
```

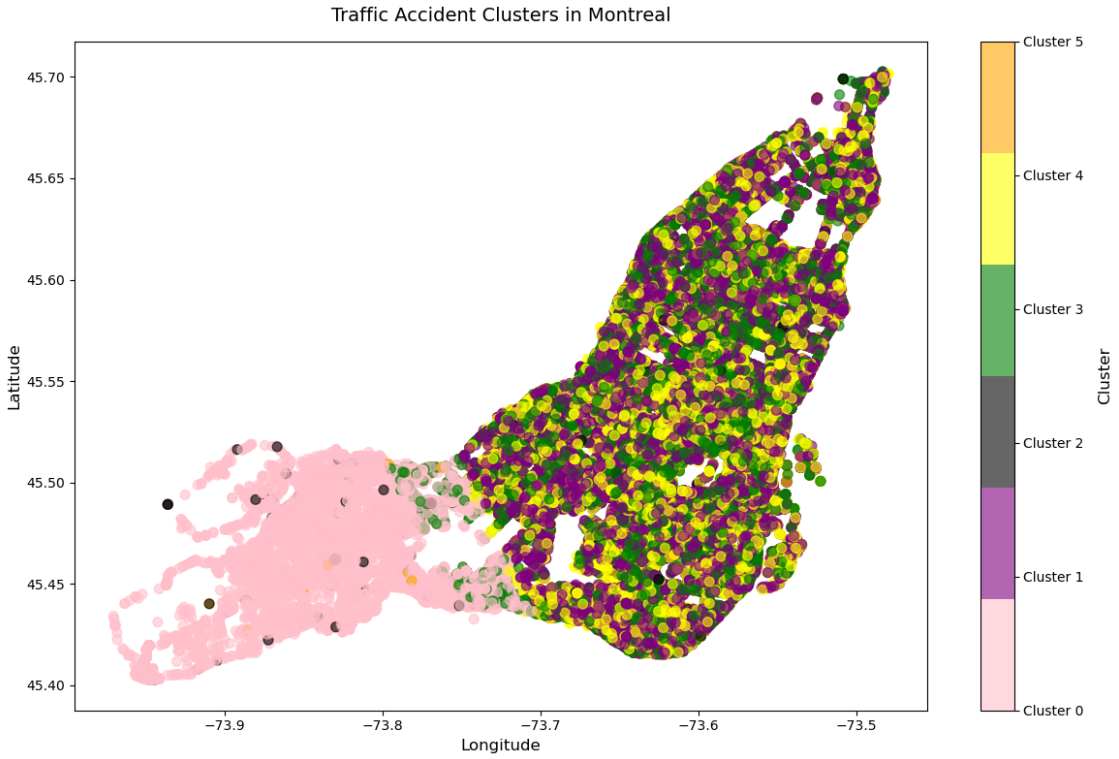
```
        'Most_Common_Weather': cluster_data['Weather_Cond'].mode().iloc[0],
        'Most_Common_Light': cluster_data['Light_Cond'].mode().iloc[0],
        'Most_Common_Road_Config': cluster_data['Road_Config'].mode().iloc[0],
        "Severity_level:Property damage below reporting threshold":␣
 ↪cluster_data['Sev_Property damage below reporting threshold'].mean(),
        "Severity_level:Property damage only": cluster_data['Sev_Property␣
 ↪damage only'].mean(),
        "Severity_level:Minor": cluster_data['Sev_Minor'].mean(),
        "Severity_level:Serious": cluster_data['Sev_Serious'].mean(),
        "Severity_level:Fatal": cluster_data['Sev_Fatal'].mean()
    }
    cluster_summaries.append(summary)

cluster_df_2 = pd.DataFrame(cluster_summaries)
print("\nCluster Summaries:")
print(cluster_df_2)

# # 6. Create map visualization with clear legend

# colors = ['blue', 'green', 'purple', 'orange', "yellow"]
# cluster_names = [f"Cluster {i}" for i in range(optimal_k)]

# # Create base map
# montreal_map = folium.Map(location=[45.5017, -73.5673], zoom_start=11)

# # Add cluster centers with consistent colors and legend
# for idx, row in cluster_df_2.iterrows():
#     color = colors[idx % len(colors)]

#     # Add cluster center marker
#     folium.CircleMarker(
#         location=[row['Latitude'], row['Longitude']],
#         radius=20,
#         popup=f"""
#         <b>Cluster {row['Cluster']}</b><br>
#         Size: {row['Size']}<br>
#         Avg Victims: {row['Avg_Victims']:.2f}<br>
#         Most Common Surface: {row['Most_Common_Surface']}<br>
#         Most Common Weather: {row['Most_Common_Weather']}
#         """,
#         color=color,
#         fill=True,
#         fill_opacity=0.7
#     ).add_to(montreal_map)

# # Add a legend
# legend_html = """
```

```
# <div style="position: fixed; bottom: 50px; left: 50px; z-index: 1000;␣
 ↪background-color: white;
#             padding: 10px; border: 2px solid grey; border-radius: 5px;">
# <h4>Cluster Legend</h4>
# """

# for i, color in enumerate(colors[:optimal_k]):
#     legend_html += f'<p><span style="color:{color};"> </span> Cluster {i}</p>'

# legend_html += "</div>"
# montreal_map.get_root().html.add_child(folium.Element(legend_html))

# # Save the map
# montreal_map.save('montreal_traffic_clusters_with_legend3.html')

# 7. Create a scatter plot of clusters
# Create custom color map that matches your desired colors
colors = ['pink', 'purple', 'black', 'green', 'yellow', 'orange']
custom_cmap = LinearSegmentedColormap.from_list('custom', colors, N=len(colors))

# Create the scatter plot
plt.figure(figsize=(12, 8))
scatter = plt.scatter(df_2['Lontitude'], df_2['Latitude'],
                      c=df_2['Cluster'],
                      cmap=custom_cmap,
                      alpha=0.6,
                      s=50)

# Customize the plot
plt.title('Traffic Accident Clusters in Montreal', fontsize=14, pad=15)
plt.xlabel('Longitude', fontsize=12)
plt.ylabel('Latitude', fontsize=12)

# Create custom colorbar with integer ticks
cbar = plt.colorbar(scatter, ticks=np.arange(len(colors)))
cbar.set_label('Cluster', fontsize=12, labelpad=10)
cbar.ax.set_yticklabels([f'Cluster {i}' for i in range(len(colors))])

# Adjust layout
plt.tight_layout()

# Show the plot
plt.show()
```

```
d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

```
Cluster Summaries:
   Cluster   Size      Latitude     Longitude  Avg_Victims  \
0        0  18666  6.396793e-17  1.623359e-01     0.199614
1        1  75028  2.252972e-16  2.367551e-14     0.000000
2        2   1784 -6.505213e-19 -1.137979e-15     1.340807
3        3  42866 -5.551115e-17  1.000000e+00     1.224257
4        4  79521  2.025290e-16  2.750578e-14     0.000000
5        5    263  1.000000e+00 -1.387779e-16     1.292776

            Most_Common_Surface           Most_Common_Weather  \
0  Surface_Cond_Severity_level3  Weather_Cond_Severity_level4
1  Surface_Cond_Severity_level3  Weather_Cond_Severity_level4
2  Surface_Cond_Severity_level3  Weather_Cond_Severity_level4
3  Surface_Cond_Severity_level3  Weather_Cond_Severity_level4
4  Surface_Cond_Severity_level3  Weather_Cond_Severity_level4
5  Surface_Cond_Severity_level3  Weather_Cond_Severity_level4

            Most_Common_Light      Most_Common_Road_Config  \
0  Light_Cond_Severity_level4  Road_Config_Severity_level2
1  Light_Cond_Severity_level4  Road_Config_Severity_level1
2  Light_Cond_Severity_level4  Road_Config_Severity_level3
3  Light_Cond_Severity_level4  Road_Config_Severity_level3
4  Light_Cond_Severity_level4  Road_Config_Severity_level1
5  Light_Cond_Severity_level4  Road_Config_Severity_level3

   Severity_level:Property damage below reporting threshold  \
0                                           0.401371
1                                           0.000000
2                                           0.000000
3                                           0.000000
4                                           1.000000
5                                           0.000000

   Severity_level:Property damage only  Severity_level:Minor  \
0                             0.436248              0.162381
1                             1.000000              0.000000
2                             0.000000              0.000000
3                             0.000000              1.000000
4                             0.000000              0.000000
5                             0.000000              0.000000

   Severity_level:Serious  Severity_level:Fatal
0                     0.0                   0.0
1                     0.0                   0.0
2                     1.0                   0.0
3                     0.0                   0.0
4                     0.0                   0.0
```

```
5                    0.0                    1.0
```



2.1 Cluster Summaries vs. Appendix Severity Levels

Cluster	Size	"Severity Mix"	Dominant Appendix Class
0	18 666	40 % below threshold (0), 44 % property damage only (1), 16 % minor (2)	Mixed low-severity
1	75 028	**100 %** property damage only (1)	1 ("Property damage only")
2	1 784	**100 %** serious injury (3)	3 ("Serious injury")
3	42 866	**100 %** minor injury (2)	2 ("Minor injury")
4	79 521	**100 %** below reporting threshold (0)	0 ("Below threshold")
5	263	**100 %** fatal (4)	4 ("Fatal")

- **Clusters 1–5** each correspond *exactly* to one Appendix severity category.
- **Cluster 0** picks up a *residual mix* of the two most common low-severity levels (0 & 1) plus a dash of minor (2)—these are the boundary cases that sit numerically between the five "pure" groups.

2.2 Geographic Spread of Each Cluster

- The **pure classes** (Clusters 1–5) scatter *all over* Montreal—showing we're really segmenting by severity more than by area.
- The **mixed Cluster 0** tends to occupy the **West Island** and peripheral zones, where "below threshold" and "property damage only" incidents dominate but aren't cleanly separable.

2.3 Actionable Take-Aways

1. **If we intended "area" clusters**, drop the full one-hot severity vectors (or down-weight them) so geography can shine through.
2. **If we want "severity+zones"**, consider a two-stage approach:
   - Stage 1: cluster purely on severity → recover the 0–4 bands.
   - Stage 2: within each severity band, run a geographic clustering to find sub-areas where, say, "Minor injuries" concentrate.
3. **Cluster 0 (the mixed low-severity bin)** is worth a closer look: these boundary accidents (below threshold vs. property-damage) might share unique road or time-of-day patterns—spot-checking a few could reveal subtle factors driving fender-benders in the suburbs.

3 Subset for Serious and Fatal

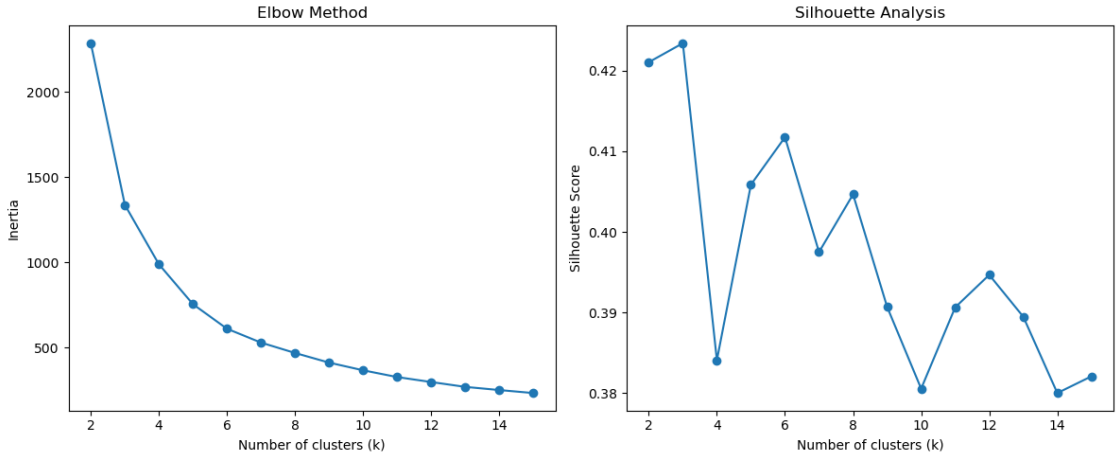
[5]:
```
df_sf = df[(df["Severity_num"] == 3) | (df["Severity_num"] == 4)]
# 1. Prepare location data

X = df_sf[['Latitude', 'Lontitude']].values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 2. Determine optimal number of clusters using elbow method
print("Finding optimal number of clusters...")
inertias = []
silhouette_scores = []
max_clusters = 15

plt.figure(figsize=(12, 5))
for k in range(2, max_clusters + 1):
    print(f"Testing k={k}...")
    kmeans = KMeans(n_clusters=k, random_state=0)
    kmeans.fit(X_scaled)
    inertias.append(kmeans.inertia_)
    score = silhouette_score(X_scaled, kmeans.labels_)
    silhouette_scores.append(score)

# Plot elbow curve
plt.subplot(1, 2, 1)
plt.plot(range(2, max_clusters + 1), inertias, marker='o')
```

```
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method')

plt.subplot(1, 2, 2)
plt.plot(range(2, max_clusters + 1), silhouette_scores, marker='o')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Analysis')

plt.tight_layout()
plt.show()
```

```
Finding optimal number of clusters…
Testing k=2…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=3…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=4…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=5…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=6…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=7…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
```

```
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=8…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=9…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=10…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=11…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=12…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=13…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=14…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)

Testing k=15…

d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
```

1.4. Set the value of `n_init` explicitly to suppress the warning
super()._check_params_vs_input(X, default_n_init=10)



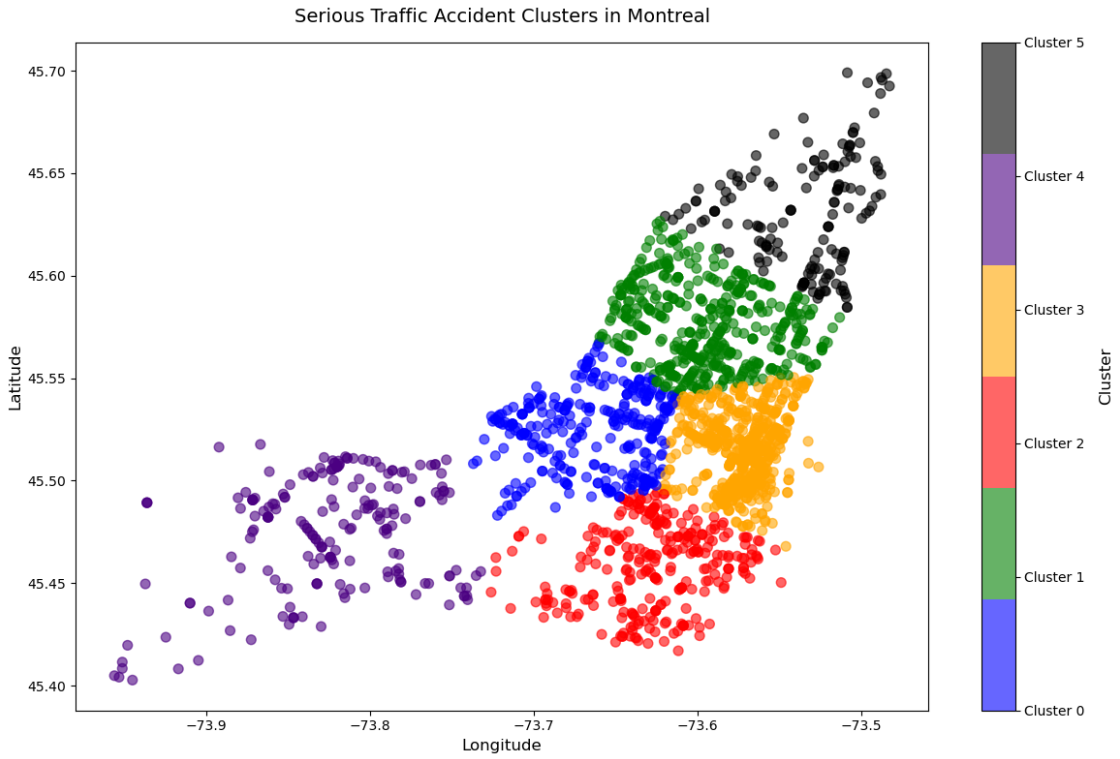
[6]:
```
# 3. Perform clustering with optimal k=5
optimal_k = 6  # Adjust this based on the elbow curve results
kmeans = KMeans(n_clusters=optimal_k, random_state=0)
labels = kmeans.fit_predict(X_scaled)

# 4. Add cluster labels to the original dataframe
df_sf['Cluster'] = labels

# 5. Calculate cluster summaries
print("Calculating cluster summaries...")
cluster_summaries = []
features_to_summarize = ['Latitude', 'Lontitude', 'Total_Victims',␣
 ↪'Severity_num', 'Severity',
                         'Surface_Cond', 'Weather_Cond', 'Light_Cond',
                         'Road_Config']

for i in range(optimal_k):
    cluster_data = df_sf[df_sf['Cluster'] == i]

    # Get geographical center
    center = kmeans.cluster_centers_[i]
    center_unscaled = scaler.inverse_transform([center])[0]

    summary = {
        'Cluster': i,
        'Size': len(cluster_data),
        'Latitude': center_unscaled[0],
        'Longitude': center_unscaled[1],
```

```
        'Avg_Victims': cluster_data['Total_Victims'].mean(),
        'Avg_Severity': cluster_data['Severity_num'].mean(),
        'Most_Common_Severity': cluster_data['Severity'].mode().iloc[0],
        'Most_Common_Surface': cluster_data['Surface_Cond'].mode().iloc[0],
        'Most_Common_Weather': cluster_data['Weather_Cond'].mode().iloc[0],
        'Most_Common_Light': cluster_data['Light_Cond'].mode().iloc[0],
        'Most_Common_Road_Config': cluster_data['Road_Config'].mode().iloc[0]
    }
    cluster_summaries.append(summary)

cluster_df = pd.DataFrame(cluster_summaries)
print("\nCluster Summaries:")
print(cluster_df)

# 6. Create map visualization with clear legend

colors = ['blue', 'green', 'purple', 'orange', "yellow"]
cluster_names = [f"Cluster {i}" for i in range(optimal_k)]

# Create base map
montreal_map = folium.Map(location=[45.5017, -73.5673], zoom_start=11)

# Add cluster centers with consistent colors and legend
for idx, row in cluster_df.iterrows():
    color = colors[idx % len(colors)]

    # Add cluster center marker
    folium.CircleMarker(
        location=[row['Latitude'], row['Longitude']],
        radius=20,
        popup=f"""
        <b>Cluster {row['Cluster']}</b><br>
        Size: {row['Size']}<br>
        Avg Victims: {row['Avg_Victims']:.2f}<br>
        Avg Severity: {row['Avg_Severity']:.2f}<br>
        Most Common Severity: {row['Most_Common_Severity']}<br>
        Most Common Surface: {row['Most_Common_Surface']}<br>
        Most Common Weather: {row['Most_Common_Weather']}
        """,
        color=color,
        fill=True,
        fill_opacity=0.7
    ).add_to(montreal_map)

# Add a legend
legend_html = """
```

```
<div style="position: fixed; bottom: 50px; left: 50px; z-index: 1000;␣
 ↪background-color: white;
            padding: 10px; border: 2px solid grey; border-radius: 5px;">
<h4>Cluster Legend</h4>
"""

for i, color in enumerate(colors[:optimal_k]):
    legend_html += f'<p><span style="color:{color};"> </span> Cluster {i}</p>'

legend_html += "</div>"
montreal_map.get_root().html.add_child(folium.Element(legend_html))

# Save the map
montreal_map.save('montreal_traffic_clusters_with_legend_Serious.html')

# 7. Create a scatter plot of clusters
# Create custom color map that matches your desired colors
colors = ['blue', 'green', '#FF0000', 'orange',  'indigo','black']  # Colors in␣
 ↪the order you want for clusters 0-5
custom_cmap = LinearSegmentedColormap.from_list('custom', colors, N=len(colors))

# Create the scatter plot
plt.figure(figsize=(12, 8))
scatter = plt.scatter(df_sf['Lontitude'], df_sf['Latitude'],
                      c=df_sf['Cluster'],
                      cmap=custom_cmap,
                      alpha=0.6,
                      s=50)

# Customize the plot
plt.title('Serious Traffic Accident Clusters in Montreal', fontsize=14, pad=15)
plt.xlabel('Longitude', fontsize=12)
plt.ylabel('Latitude', fontsize=12)

# Create custom colorbar with integer ticks
cbar = plt.colorbar(scatter, ticks=np.arange(len(colors)))
cbar.set_label('Cluster', fontsize=12, labelpad=10)
cbar.ax.set_yticklabels([f'Cluster {i}' for i in range(len(colors))])

# Adjust layout
plt.tight_layout()

# Show the plot
plt.show()
```

```
d:\Software\Anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
```

```
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\liuyifei\AppData\Local\Temp\ipykernel_26388\3277515719.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_sf['Cluster'] = labels

Calculating cluster summaries…

Cluster Summaries:
   Cluster  Size   Latitude  Longitude  Avg_Victims  Avg_Severity  \
0        0   327  45.524212 -73.661876     1.330275      3.146789
1        1   474  45.575408 -73.594876     1.335443      3.120253
2        2   299  45.459051 -73.626664     1.421405      3.140468
3        3   590  45.515641 -73.572863     1.269492      3.122034
4        4   200  45.474656 -73.824211     1.350000      3.115000
5        5   157  45.630407 -73.534052     1.401274      3.133758

  Most_Common_Severity          Most_Common_Surface  \
0              Serious  Surface_Cond_Severity_level3
1              Serious  Surface_Cond_Severity_level3
2              Serious  Surface_Cond_Severity_level3
3              Serious  Surface_Cond_Severity_level3
4              Serious  Surface_Cond_Severity_level3
5              Serious  Surface_Cond_Severity_level3

            Most_Common_Weather           Most_Common_Light  \
0  Weather_Cond_Severity_level4  Light_Cond_Severity_level4
1  Weather_Cond_Severity_level4  Light_Cond_Severity_level4
2  Weather_Cond_Severity_level4  Light_Cond_Severity_level4
3  Weather_Cond_Severity_level4  Light_Cond_Severity_level4
4  Weather_Cond_Severity_level4  Light_Cond_Severity_level4
5  Weather_Cond_Severity_level4  Light_Cond_Severity_level4

       Most_Common_Road_Config
0  Road_Config_Severity_level3
1  Road_Config_Severity_level3
2  Road_Config_Severity_level3
3  Road_Config_Severity_level3
4  Road_Config_Severity_level2
5  Road_Config_Severity_level3
```



3.1 Geographic "Hotspots" of Serious Crashes

Cluster	Center (Lat, Lon)	Count
0	(45.524 , –73.662)	327
1	(45.575 , –73.595)	474
2	(45.459 , –73.626)	299
3	(45.516 , –73.573)	590
4	(45.475 , –73.824)	200
5	(45.630 , –73.534)	157

- **Cluster 3 (590 crashes)** sits smack in the **central island**—the busiest core of serious collisions.
- **Cluster 1 (474)** hugs the **Plateau / Mile-End** area.
- **Cluster 0 (327)** covers the **Hochelaga-Maisonneuve / east-end**.
- **Cluster 2 (299)** and **4 (200)** fall on the **West Island** (major arterials through Dorval/Dollard).
- **Cluster 5 (157)** up in **Laval** is the smallest hot-spot of serious/fatal crashes.

3.2 Severity & Victim Load by Cluster

Cluster	Avg Severity	Avg Victims
0	**3.15**	1.33
1	3.12	1.34
2	3.14	**1.42**
3	3.12	1.27
4	**3.11**	1.35
5	3.13	**1.40**

- **Cluster 0** (east-end) has the **highest average severity**—more of those "most serious" events cluster here.
- **Clusters 2 & 5** show the **largest average victim counts** (1.42 & 1.40)—suggesting multi-person crashes on the West Island and Laval corridors.

3.3 Uniform Conditions Across Zones

- **Daylight (Light_Cond = 4)** and **Clear weather (Weather_Cond = 4)** dominate *every* cluster.
- **Surface condition** is the same "level 3" everywhere (per your Appendix, likely "wet/damp" or "snow-covered").
- **Road_Config** is mostly level 3 ("multi-lane arterials") except Cluster 4 (West Island) which is level 2 ("two-lane arterials").

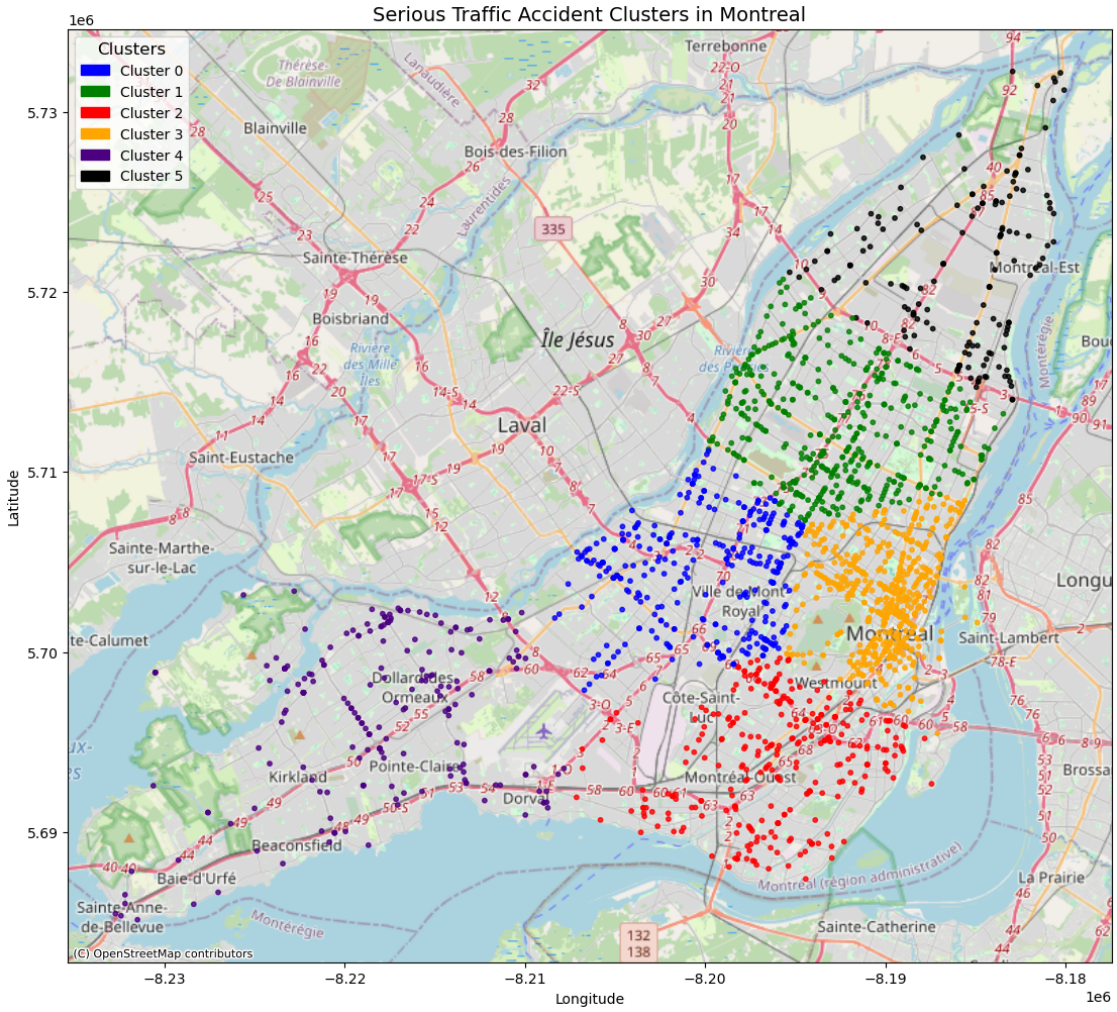
Insight: serious crashes are *not* driven by darkness or extreme weather—they happen under good visibility & typical road surfaces, so focus on **road geometry**, **speed management**, and **traffic control**, not lighting or de-icing.

3.4 What This Means for Intervention

1. **East-End Priority (Cluster 0)**
   - Highest avg severity. Consider targeted redesigns—protected bike lanes, slower school-zone limits, or signal-timing audits around Hochelaga-Maisonneuve.
2. **West-Island & Laval Multi-Victim Zones (Clusters 2 & 5)**
   - Higher victim counts suggest highway-style collisions. Explore median barriers, ramp redesigns, or automated speed enforcement along those corridors.
3. **Downtown Core (Cluster 3)**
   - Largest absolute count of serious crashes. Even if avg severity is slightly lower, sheer volume means small safety tweaks (e.g., leading pedestrian intervals) can yield large crash reductions.
4. **Plateau / Mile-End (Cluster 1)**
   - A mix of commercial and residential streets: review curb-side loading, bike-car interactions, and one-way conversions to calm traffic.

[88]:
```
import contextily as ctx
import geopandas as gpd
```

```
import matplotlib.patches as mpatches

# Convert DataFrame to GeoDataFrame
gdf_sf = gpd.GeoDataFrame(df_sf, geometry=gpd.
 ↪points_from_xy(df_sf['Lontitude'], df_sf['Latitude']))
gdf_sf.set_crs(epsg=4326, inplace=True)
gdf_sf = gdf_sf.to_crs(epsg=3857)

# Plot with a map
fig, ax = plt.subplots(figsize=(16, 12))
gdf_sf.plot(ax=ax, column='Cluster', cmap=custom_cmap, markersize=10, alpha=0.8)
ctx.add_basemap(ax, source=ctx.providers.OpenStreetMap.Mapnik) # Add Montreal␣
 ↪map
ax.set_title("Serious Traffic Accident Clusters in Montreal", fontsize=14)
plt.xlabel("Longitude")
plt.ylabel("Latitude")

unique_clusters = sorted(gdf_sf['Cluster'].unique())  # Sort clusters
norm = plt.Normalize(vmin=unique_clusters[0], vmax=unique_clusters[-1])  #␣
 ↪Normalize based on sorted values
colors = [custom_cmap(norm(cluster)) for cluster in unique_clusters]  # Get␣
 ↪colors for each cluster
patches = [
    mpatches.Patch(color=color, label=f"Cluster {cluster}")
    for cluster, color in zip(unique_clusters, colors)
]
ax.legend(handles=patches, title="Clusters", fontsize=10, title_fontsize=12,␣
 ↪loc='upper left')

plt.savefig("traffic_accident_clusters.png", dpi=1000, bbox_inches='tight')
plt.show()
```



This map lays bare exactly where Montreal's most **serious (Severity 3–4)** crashes concentrate—and it tells a clear story about hot-spots that merit urgent, location-specific countermeasures:

3.5 Cluster Footprints

- **Cluster 0 (Blue):**
  - **Location:** Ville-Marie / Downtown grid just north of the river.
  - **Implication:** Even though core traffic speeds are lower, heavy turning volumes and multi-modal conflicts here still produce a substantial cluster of severe crashes.
- **Cluster 1 (Green):**
  - **Location:** Plateau–Mile-End sector.
  - **Implication:** Narrow streets with high pedestrian/bike traffic—look at protected bike lanes and curb extensions to calm vehicles.
- **Cluster 2 (Red):**

  - **Location:** Griffintown / Old Port / southwest island.
  - **Implication:** Fast arterials running into port-area intersections. Consider signal retiming and speed feedback signage on Peel/Notre-Dame.

- **Cluster 3 (Orange):**
  - **Location:** Hochelaga-Maisonneuve / east-end industrial corridors.
  - **Implication:** High-risk truck and auto interactions—tighten turning radius, separate heavy-vehicle routes from local streets.

- **Cluster 4 (Purple):**
  - **Location:** West Island suburbs (Dollard, Kirkland) along Highway 40.
  - **Implication:** Even out in the suburbs, serious crashes cluster on high-speed stretches. Ramp metering or median barriers could help here.

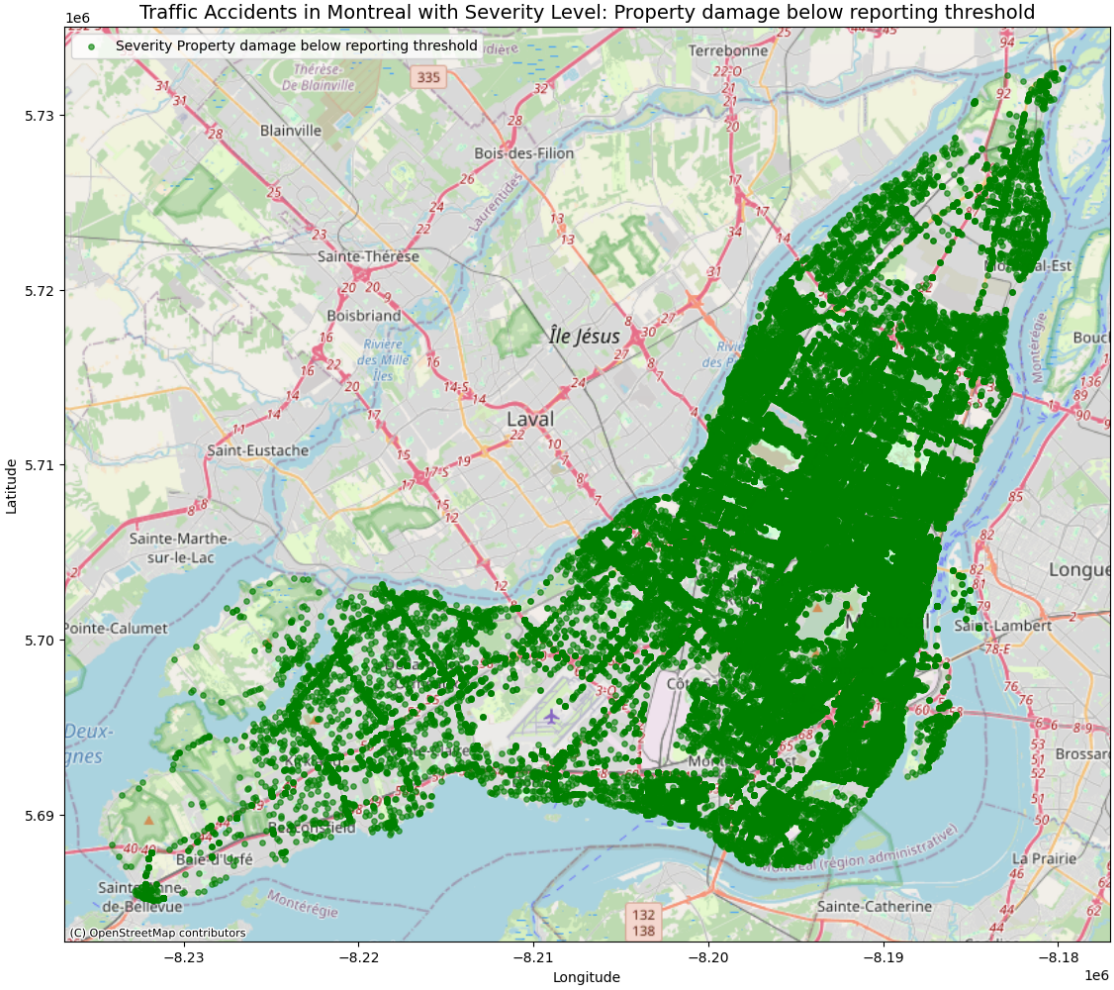
- **Cluster 5 (Black):**
  - **Location:** Laval's southern spine.
  - **Implication:** Commuter corridors into the city generate their own serious-crash cluster—automated speed enforcement on Papineau and Pie-IX bridges may be warranted.

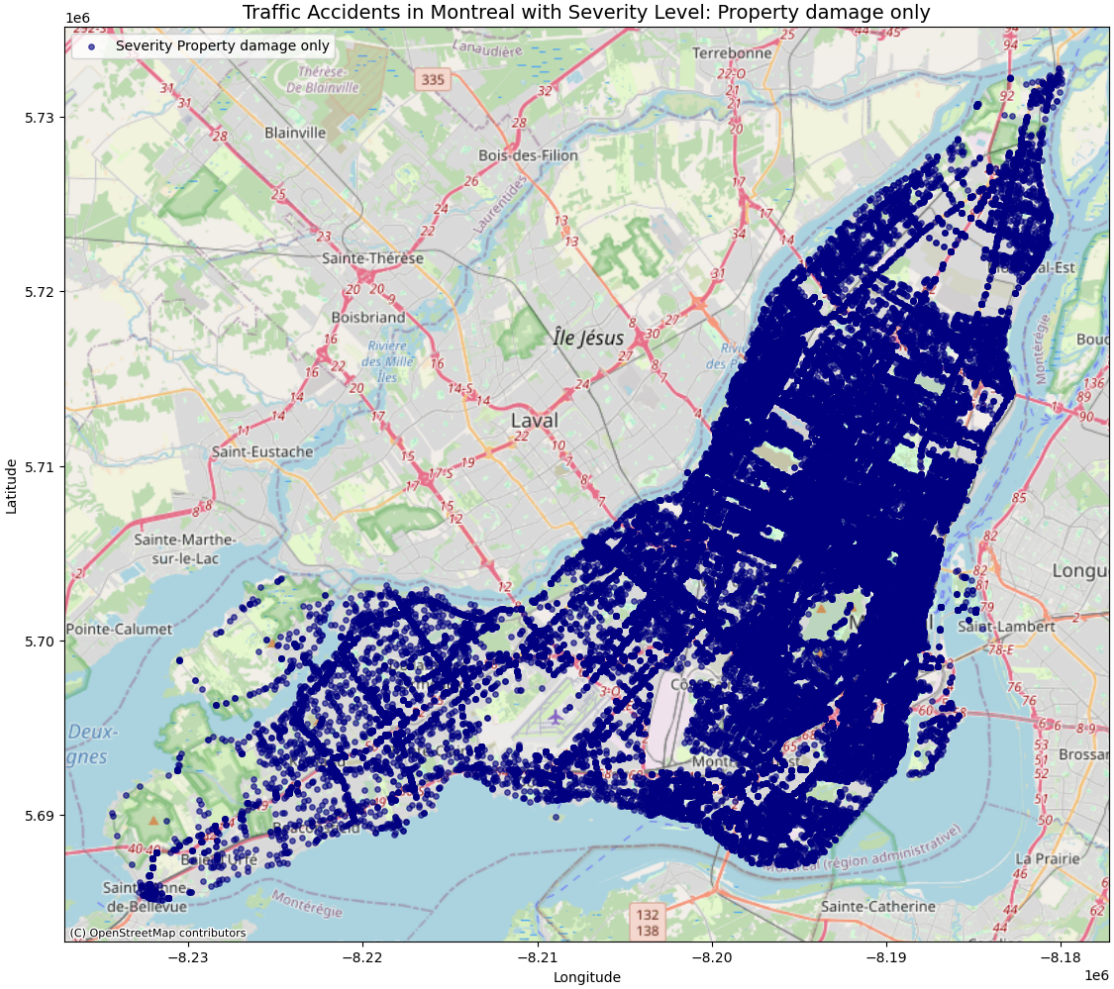
3.6 Key Takeaways & Next Steps

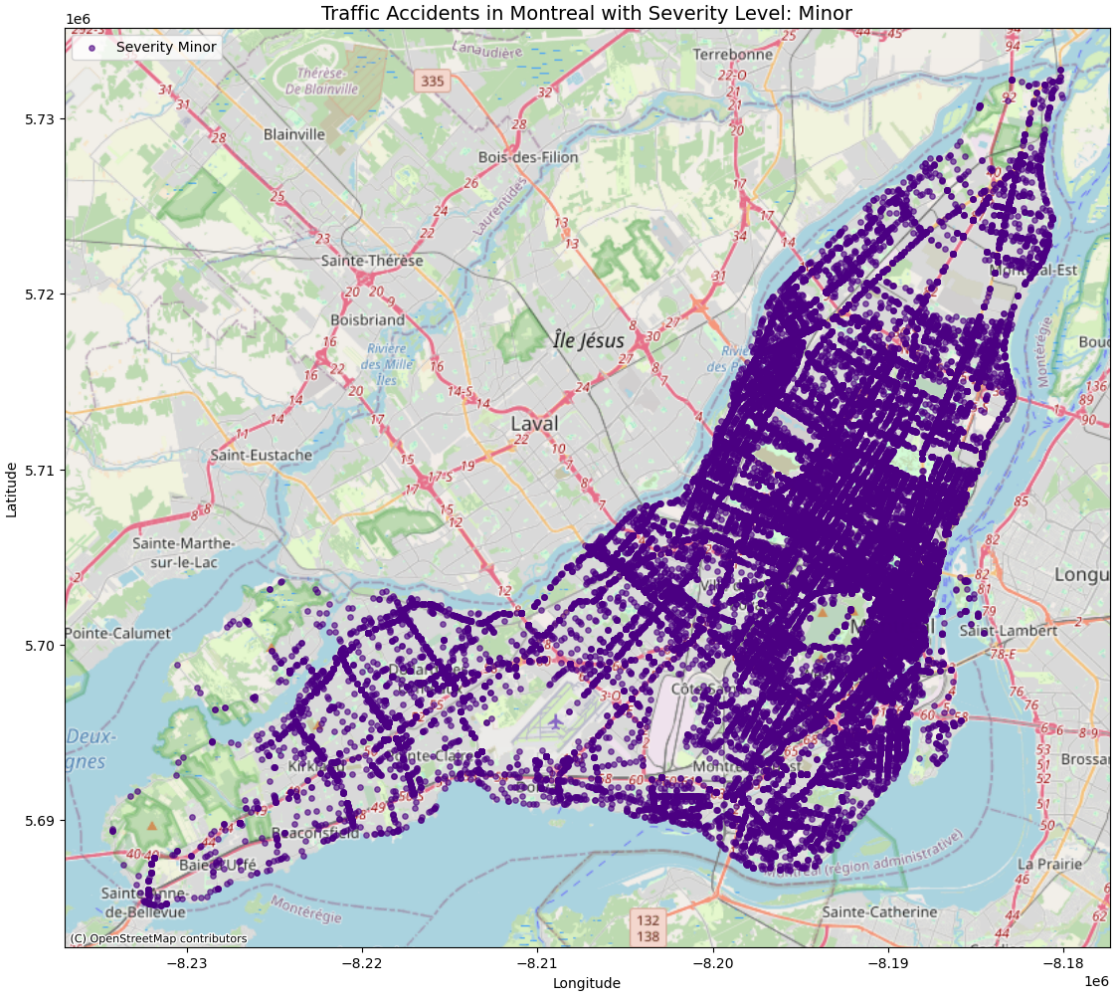
1. **Geo-Targeted Engineering:**
   - Downtown (Cluster 0): Add leading pedestrian intervals, high-visibility crosswalks, and raised intersections.
   - East-end (Cluster 3): Reconfigure truck routes, enforce turn bans at known truck-car conflict points.
2. **Speed Management:**
   - Suburban arterials (4 & 5): Deploy dynamic speed displays and photo-enforcement on highway on-ramps.
3. **Infrastructure Upgrades:**
   - Plateaus (Cluster 1): Install protected bike lanes and curb extensions around high-pedestrian corridors.
   - Griffintown (Cluster 2): Use "daylighting" (clearing vision triangles) at industrial intersections.
4. **Data-Driven Policing & Outreach:**
   - Coordinate with SPVM to increase patrols in peak crash hours for each cluster.
   - Roll out community workshops in high-severity zones to raise awareness of local risk factors.

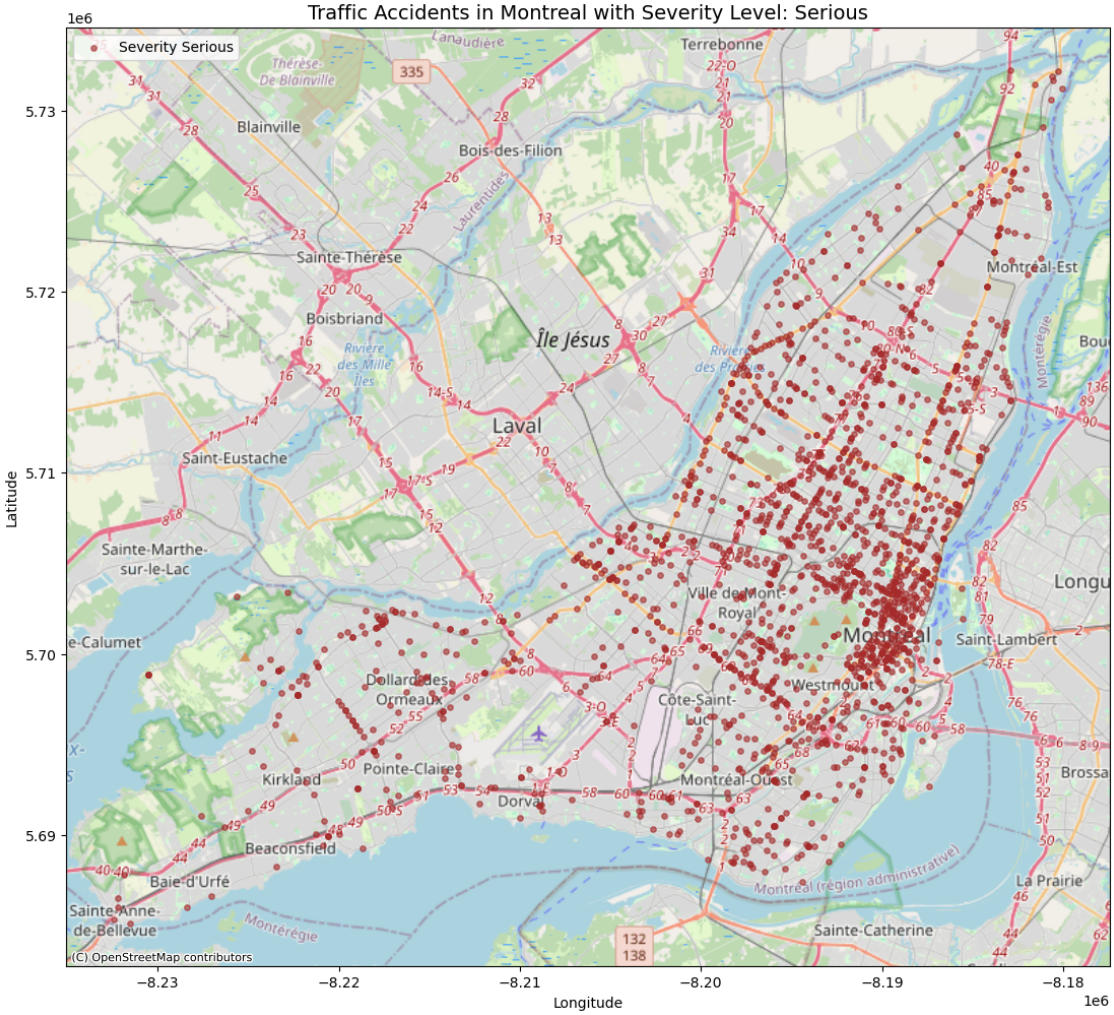
[13]:
```
import geopandas as gpd
import matplotlib.pyplot as plt
import contextily as ctx
```

```
# Convert DataFrame to GeoDataFrame
gdf = gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(df['Lontitude'],␣
 ↪df['Latitude']))
gdf.set_crs(epsg=4326, inplace=True)
gdf = gdf.to_crs(epsg=3857)

# Define a color map for the severity categories
severity_colors = {
    'Property damage below reporting threshold': 'green',
    'Property damage only': 'navy',
    'Minor': 'indigo',
    'Serious': 'brown',
    'Fatal':  'black'
}

# Create separate plots for each severity category
for severity, color in severity_colors.items():
    # Filter the GeoDataFrame by severity
    subset = gdf[gdf['Severity'] == severity]

    # Create the plot
    fig, ax = plt.subplots(figsize=(16, 12))
    subset.plot(ax=ax, color=color, markersize=15, alpha=0.6, label=f"Severity␣
 ↪{severity}")

    # Add basemap
    ctx.add_basemap(ax, source=ctx.providers.OpenStreetMap.Mapnik)

    # Set plot titles and labels
    ax.set_title(f"Traffic Accidents in Montreal with Severity Level:␣
 ↪{severity}", fontsize=14)
    ax.set_xlabel("Longitude")
    ax.set_ylabel("Latitude")

    # Add legend
    ax.legend(fontsize=10, loc='upper left')

    # Save and display each plot
    plt.savefig(f"traffic_accident_severity_{severity}.png", dpi=1000,␣
 ↪bbox_inches='tight')
    plt.show()
```


Traffic Accidents in Montreal with Severity Level: Property damage below reporting threshold
Severity Property damage below reporting threshold
1e6
5.73
5.72
5.71
5.70
5.69
Latitude
−8.23
−8.22
−8.21
−8.20
−8.19
−8.18
1e6
Longitude
Terrebonne
Blainville
Bois-des-Filion
Sainte-Thérèse
Boisbriand
Île Jésus
Laval
Saint-Eustache
Sainte-Marthe-sur-le-Lac
Pointe-Calumet
Saint-Lambert
La Prairie
Sainte-Catherine
Sainte-Anne-de-Bellevue
Baie-d'Urfé
Montérégie
335
132
138
(C) OpenStreetMap contributors


Traffic Accidents in Montreal with Severity Level: Property damage only
Severity Property damage only
1e6
5.73
5.72
5.71
5.70
5.69
Latitude
−8.23
−8.22
−8.21
−8.20
−8.19
−8.18
1e6
Longitude
Terrebonne
Blainville
Bois-des-Filion
Sainte-Thérèse
Boisbriand
Île Jésus
Laval
Saint-Eustache
Sainte-Marthe-sur-le-Lac
Pointe-Calumet
Saint-Lambert
La Prairie
Sainte-Catherine
Montérégie
335
132
138
(C) OpenStreetMap contributors


Traffic Accidents in Montreal with Severity Level: Minor
Severity Minor
1e6
5.73
5.72
5.71
5.70
5.69
Latitude
−8.23
−8.22
−8.21
−8.20
−8.19
−8.18
1e6
Longitude
Terrebonne
Blainville
Bois-des-Filion
Sainte-Thérèse
Boisbriand
Île Jésus
Laval
Saint-Eustache
Sainte-Marthe-sur-le-Lac
Pointe-Calumet
Saint-Lambert
La Prairie
Sainte-Catherine
Sainte-Anne-de-Bellevue
(C) OpenStreetMap contributors


Traffic Accidents in Montreal with Severity Level: Serious
Severity Serious
1e6
5.73
5.72
5.71
5.70
5.69
Latitude
−8.23
−8.22
−8.21
−8.20
−8.19
−8.18
1e6
Longitude
Terrebonne
Blainville
Bois-des-Filion
Sainte-Thérèse
Boisbriand
Île Jésus
Laval
Saint-Eustache
Sainte-Marthe-sur-le-Lac
Montréal-Est
Ville de Mont-Royal
Saint-Lambert
Dollard-des-Ormeaux
Côte-Saint-Luc
Westmount
Kirkland
Pointe-Claire
Dorval
Montréal-Ouest
Beaconsfield
Baie-d'Urfé
Sainte-Anne-de-Bellevue
La Prairie
Sainte-Catherine
Montréal (région administrative)
Rivière des Mille Îles
Rivière des Prairies
Montérégie
Laurentides
335
132
138
(C) OpenStreetMap contributors

Here's what the five "single-severity" maps tell us—and how we can turn each into targeted, severity-appropriate actions:

3.7 1. Below-Threshold Damage (Green)

- **Ubiquitous but Low-Impact:** These tiny fender-benders trace virtually every street grid—downtown, suburbs, and even quiet side streets.
- **Action:** Automated reporting or "dashcam" initiatives could help capture near-miss data here. As the lowest-impact class, this layer is your baseline "exposure" map of where vehicles mingle most.

3.8 2. Property-Damage Only (Navy)

- **Corridor Focused:** You still see the full grid, but darker along major arterials (Décarie, Papineau, Notre-Dame).
- **Action:** These are ripe for **traffic-calming** (speed cushions, narrower lanes) and **intersection redesign** (clearer turn lanes) to reduce the most common "no-injury" crashes.

3.9 3. Minor Injuries (Indigo)

- **Concentrated on High-Volume Routes:** Strong presence on plate-glass grids like Sherbrooke, St-Laurent, and the Highway 20 corridor.
- **Action:** Retrofit **protected bike lanes** and **pedestrian refuges** here—these corridors yield the most "ouch, but okay" crashes.

3.10 4. Serious Injuries (Brown)

- **Hot-Spots in Core & Industrial Zones:** You see clusters on Canal-side boulevards, the east-end industrial arteries, and the Peel/Notre-Dame slip-ramps.
- **Action:** Prioritize **signal-timing audits**, **leading pedestrian intervals**, and **truck-car conflict mitigation** (e.g. turn prohibitions) in these areas.

3.11 5. Fatal Collisions (Black)

- **Sparse but Crucial:** Fatal crashes scatter along expressway on/offs (Highway 40, Champlain Bridge) and a handful on busy downtown stretches.
- **Action: Automated speed enforcement**, **median barrier installation**, and **ramp-metering** should focus on these high-lethality corridors.

3.11.1 Cross-Severity Takeaways

1. **Layered Approach:**
   - Use the "below threshold" map for **network exposure**.
   - Overlay "minor" and "property damage only" for **volume-weighted traffic-calming**.
   - Overlay "serious" and "fatal" for **high-priority engineering & enforcement**.
2. **Resource Allocation:**
   - **High-frequency, low-severity**: low-cost, citywide interventions (signage, awareness).
   - **Medium-frequency, medium-severity**: targeted retrofit (protected lanes, curb extensions).
   - **Low-frequency, high-severity**: expensive countermeasures (barriers, speed cameras) at pinpointed locations.
3. **Next Steps:**
   - **Intersection heatmaps** within each layer to pinpoint the worst crosses.
   - **Before–after evaluation**: pilot a countermeasure in one serious-crash cluster and track changes across all five layers.