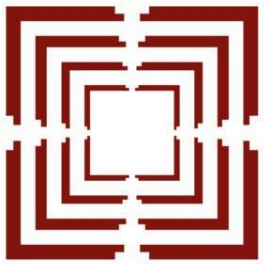


Ahmedabad University

CSE623 - Machine Learning Theory and Practice
Section - 1

Weekly Report 3
Identify Hard stop and momentary stop using the vehicle trajectory dataset

---

Team Name: **The Overfitters**

Name	Enrolment Number
Jinil Savaj	AU2240159
Jay Raval	AU2240151
Meet Suthar	AU2240198
Karan Prajapati	AU2240161
Vishesh Bhatia	AU2240027

Identifying Optimal Thresholds for Vehicle Movement Classification

Overview

This week, we changed the Feature extraction code and focused on refining the thresholds for classifying vehicle movements, specifically identifying hard stops, momentary stops, and moving vehicles. The goal was to determine the best threshold values that accurately distinguish these states, and started working on our machine-learning model.

Changes made in Feature extraction code

```
df_corrected.loc[:, 'distance'] = np.sqrt(df_corrected['x']*2 +
df_corrected['y']*2)

df_corrected.loc[:, 'velocity'] =
df_corrected.groupby('trk')['distance'].diff()

df_corrected.loc[:, 'acceleration'] =
df_corrected.groupby('trk')['velocity'].diff()

df_corrected.loc[:, 'speed'] =
(df_corrected['Vx_Instant_Not_Smoothed(kmph)']*2 +
df_corrected['Vy_Instant_Not_Smoothed(kmph)']2)*0.5

df_corrected.loc[:, 'avg_velocity_50'] =
df_corrected.groupby('trk')['velocity'].rolling(window=50).mean().reset_in
dex(level=0, drop=True)
df_corrected.loc[:, 'avg_velocity_100'] =
df_corrected.groupby('trk')['velocity'].rolling(window=100).mean().reset_i
ndex(level=0, drop=True)

df_corrected.loc[:, 'stops_5000'] =
df_corrected.groupby('trk')['velocity'].rolling(window=5000).apply(lambda
x: (x == 0).sum()).reset_index(level=0, drop=True)
```

```
df_corrected['consecutive_stop_frames'] =
df_corrected.groupby('trk')['velocity'].transform(
    lambda x: x.rolling(window=5, min_periods=1).apply(lambda y: (y.abs()
< 0.1).sum())
)

df_corrected['jerk'] = df_corrected.groupby('trk')['acceleration'].diff()

df_corrected['smoothed_velocity'] =
df_corrected.groupby('trk')['velocity'].transform(
    lambda x: x.ewm(alpha=0.3).mean()
)
df_corrected.head()
```

We have made some changes from the previous version of this code as per our accordance to find the threshold values.

Code explanation that determines the best threshold values

The code iterates over different threshold values for three key parameters:

1. **Hard Stop Threshold**
2. **Momentary Stop Threshold**
3. **Moving Vehicle Threshold**

```
hard_stop_thresholds = [0.0005, 0.0007, 0.0009, 0.001, 0.0011, 0.0013,
0.0015]

momentary_stop_thresholds = [1.5, 1.7, 1.9, 2.0, 2.1, 2.3, 2.5]

moving_vehicle_thresholds = [8.0, 9.0, 10.0, 11.0, 12.0]
```

- Defines different threshold values to test a range of conditions for classifying stops and moving vehicles.

```
for hard_stop_threshold in hard_stop_thresholds:

    for momentary_stop_threshold in momentary_stop_thresholds:

        for moving_vehicle_threshold in moving_vehicle_thresholds:
```

- Iterates through all possible combinations of hard stop, momentary stop, and moving vehicle thresholds.

```
df_corrected['hard_stop'] = (df_corrected['speed'].diff().abs() >
hard_stop_threshold) & (df_corrected['speed'] == 0)
```

- Identifies hard stops where speed changes sharply within a short time and becomes zero.

```
df_corrected['moving_vehicle'] = (df_corrected['velocity'] >
moving_vehicle_threshold) & (df_corrected['acceleration'] != 0)
```

- Classifies moving vehicles where velocity is above the defined threshold and acceleration is nonzero.

```
df_corrected['momentary_stop'] = (

                (df_corrected['avg_velocity_50'].abs() <
momentary_stop_threshold) &  # Looser threshold

                (df_corrected['velocity'] > 0) &  # Ensure it was moving
before stopping

                ~df_corrected['hard_stop']  # Ensure it's not classified
as a hard stop

            )
```

- Identifies momentary stops that do not qualify as hard stops but indicate temporary halts.

```
x = df_corrected['x']

y = df_corrected['y']

hard_stop = df_corrected['hard_stop']

momentary_stop = df_corrected['momentary_stop']

moving_vehicle = df_corrected['moving_vehicle']
```

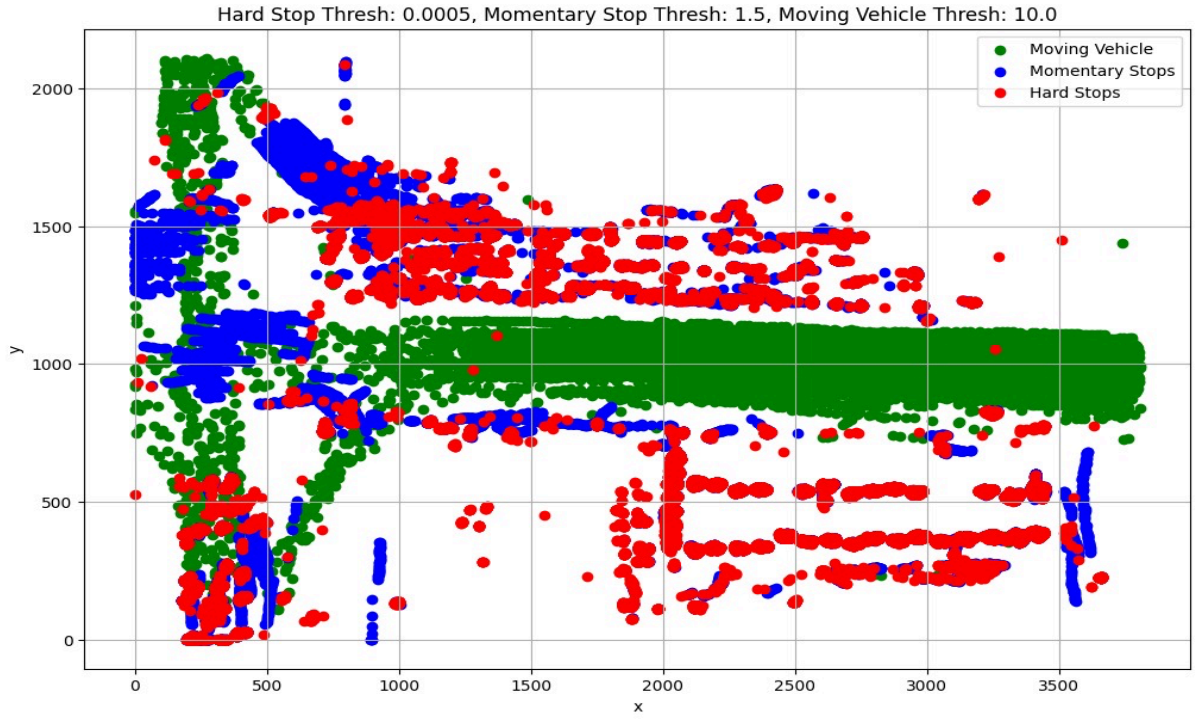
- Extracts x and y coordinates along with classified stop and movement labels for visualization.

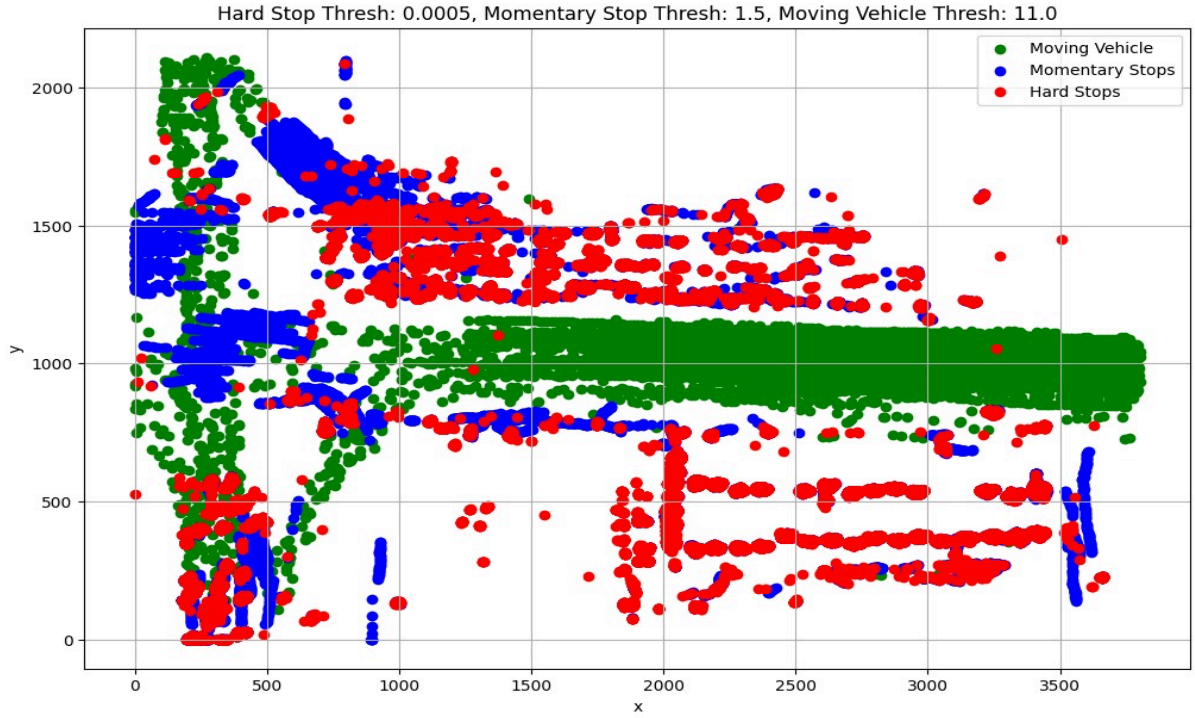
```
plt.figure(figsize=(12, 8))

plt.scatter(x[moving_vehicle], y[moving_vehicle], c='green', label='Moving
Vehicle')

plt.scatter(x[momentary_stop], y[momentary_stop], c='blue',
label='Momentary Stops')

plt.scatter(x[hard_stop], y[hard_stop], c='red', label='Hard Stops')

plt.xlabel('x')

plt.ylabel('y')

plt.title(f'Hard Stop Thresh: {hard_stop_threshold}, Momentary Stop
Thresh: {momentary_stop_threshold}, Moving Vehicle Thresh:
{moving_vehicle_threshold}')

plt.legend()

plt.grid(True)

plt.show()
```

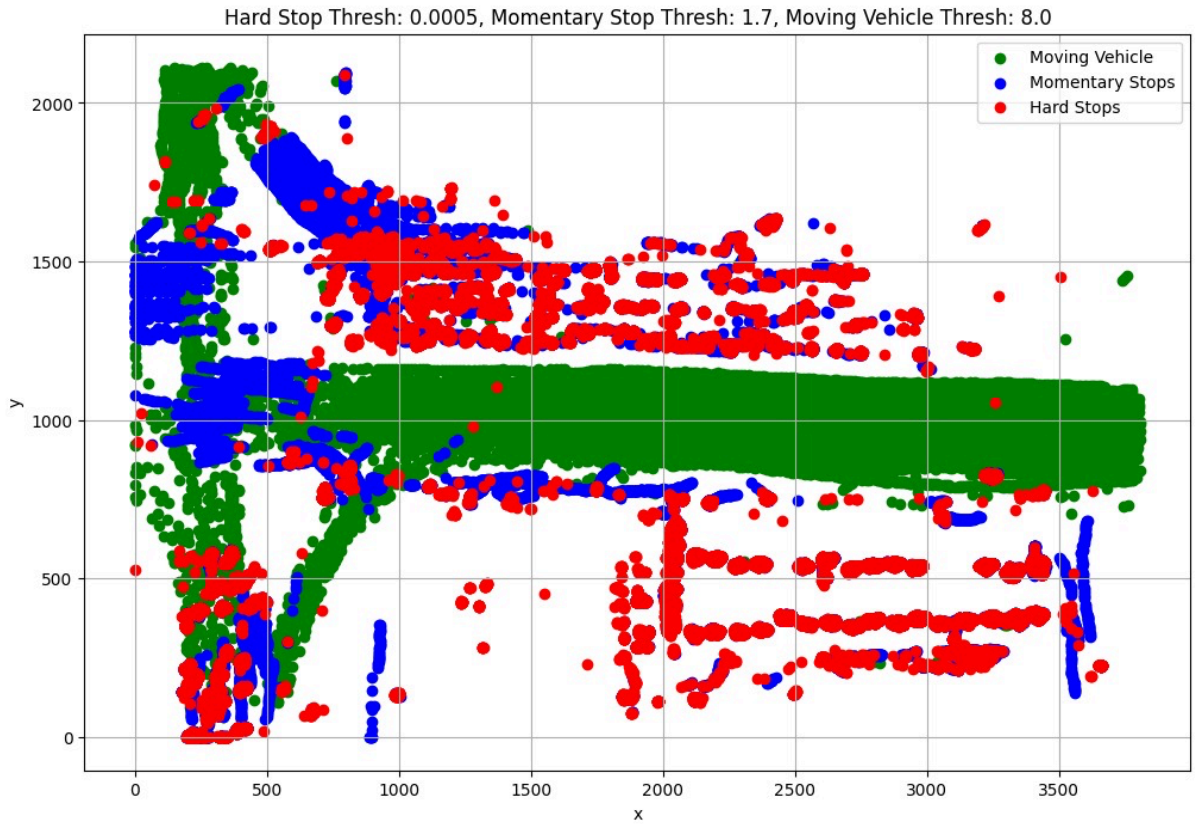
- Plotted the graphs for the Different Values of **hard_stop_thresholds, momentary_stop_thresholds and moving_vehicle_thresholds**.

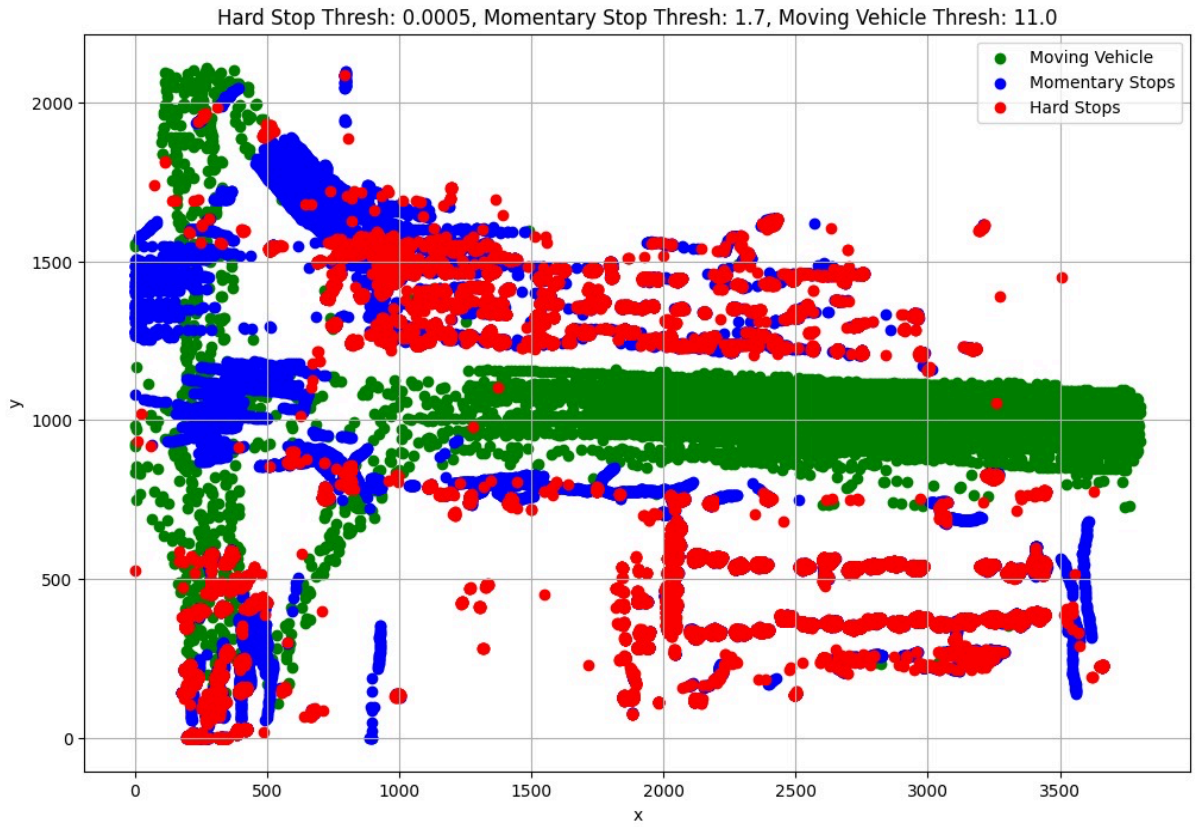
Graphs:

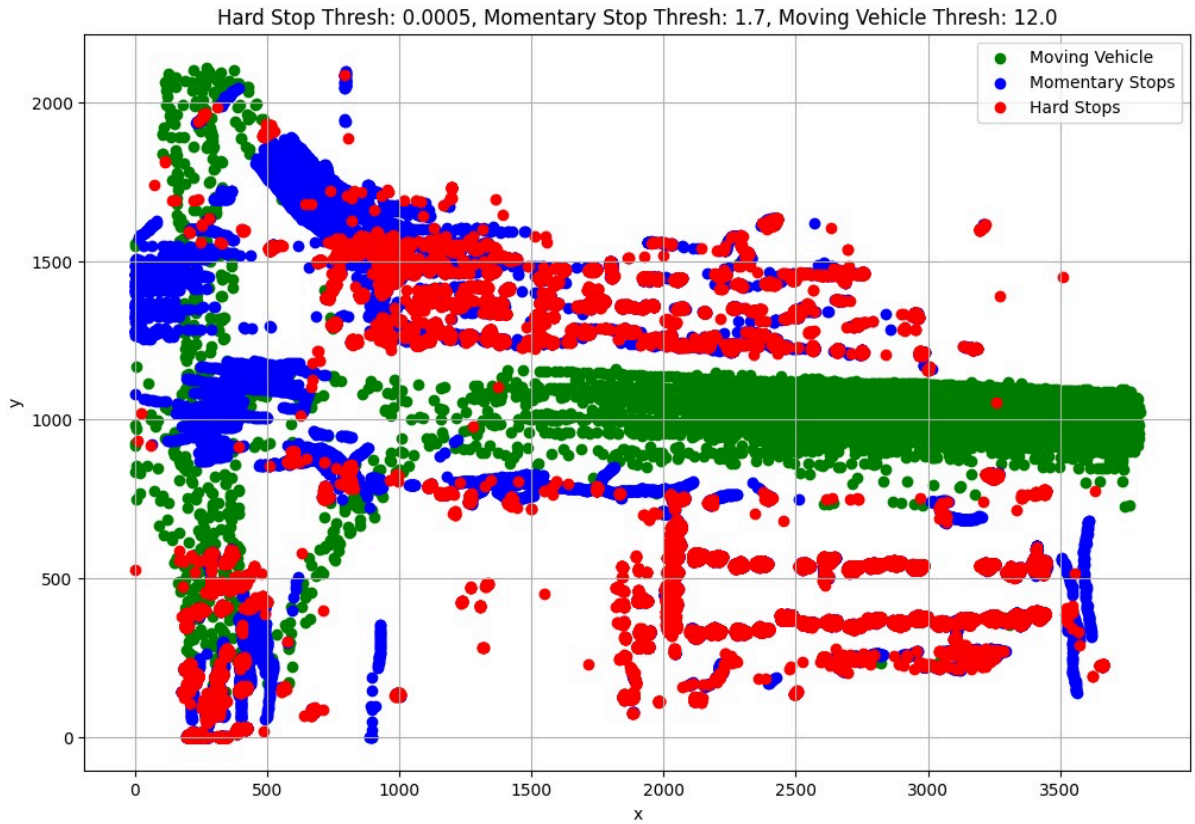
We plotted several graphs to compare distribution of vehicle labels spatially. Here are some of them.Post comparing and evaluating the graphs for each threshold value we found the graph plotted(label distribution hard stop, momentary stop and moving vehicle) was best for only some unique values of the thresholds.

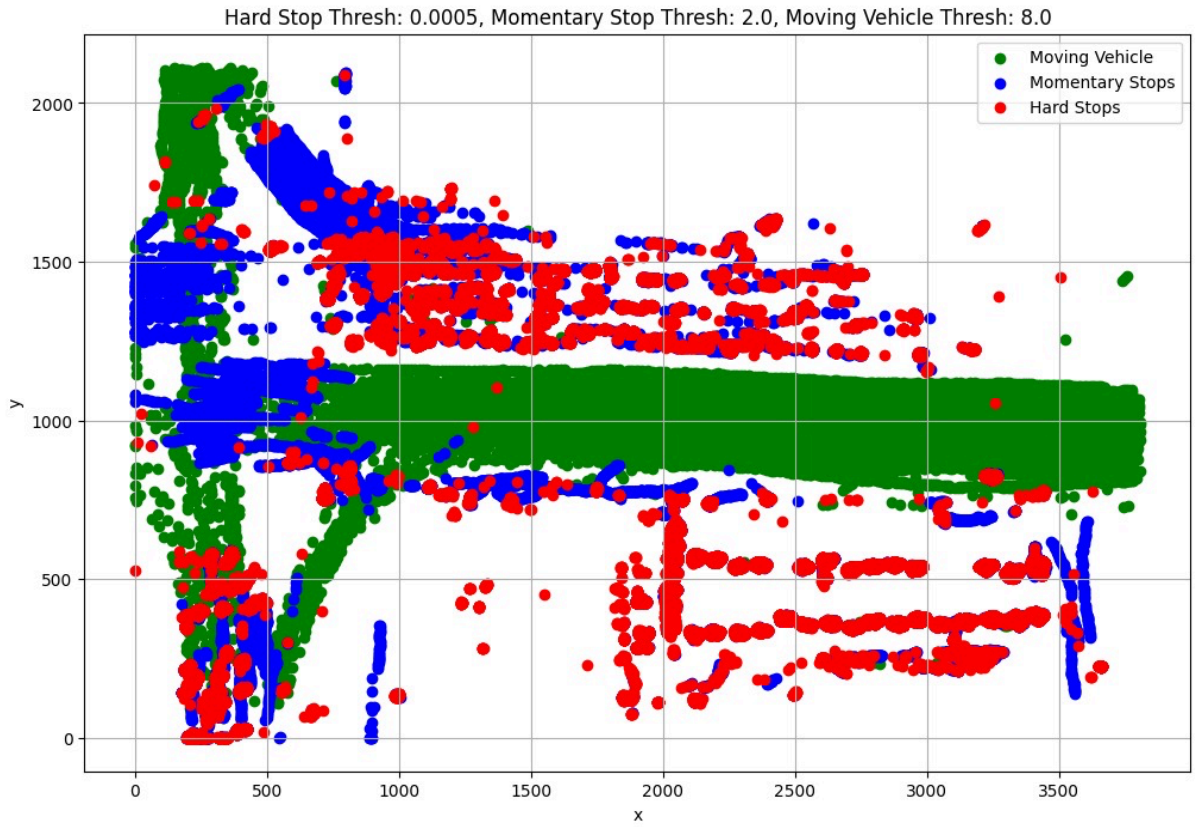

Hard Stop Thresh: 0.0005, Momentary Stop Thresh: 1.5, Moving Vehicle Thresh: 10.0
Moving Vehicle
Momentary Stops
Hard Stops
y
x
0
500
1000
1500
2000
2500
3000
3500

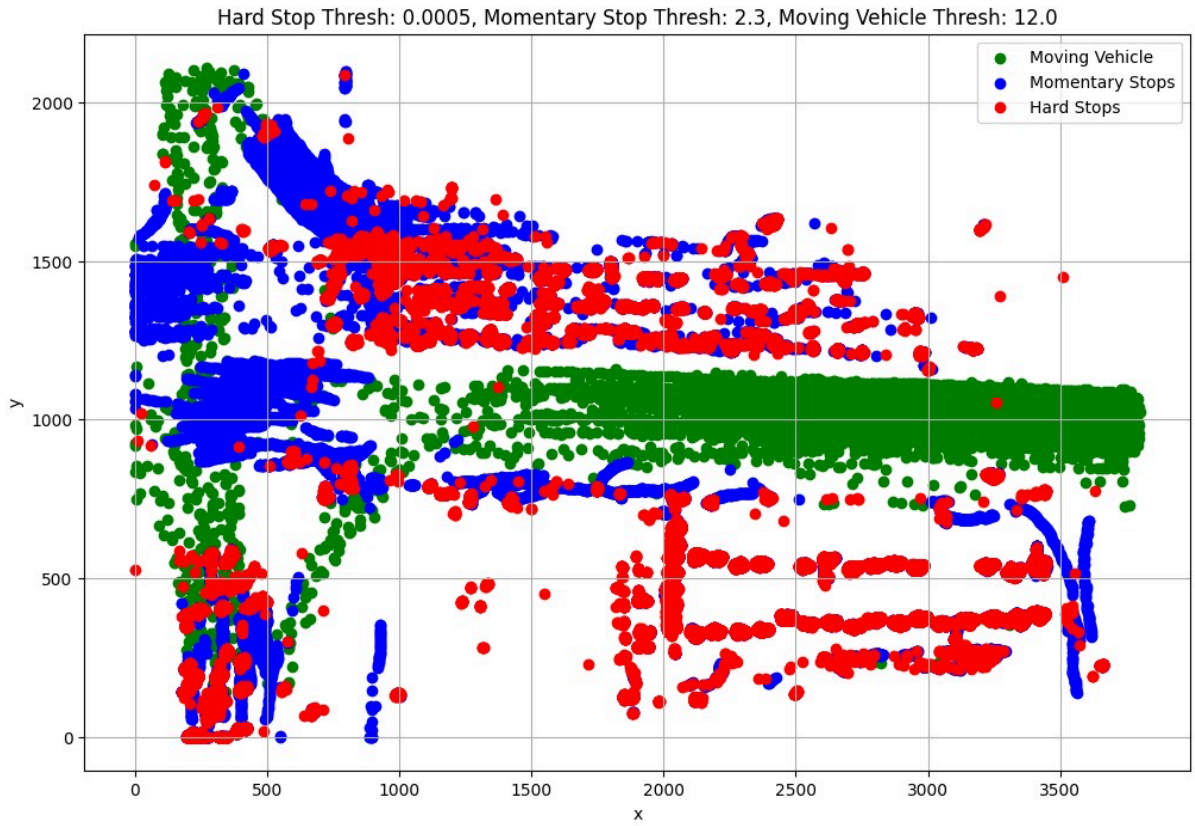

Hard Stop Thresh: 0.0005, Momentary Stop Thresh: 1.5, Moving Vehicle Thresh: 11.0
Moving Vehicle
Momentary Stops
Hard Stops
y
x
0
500
1000
1500
2000
2500
3000
3500

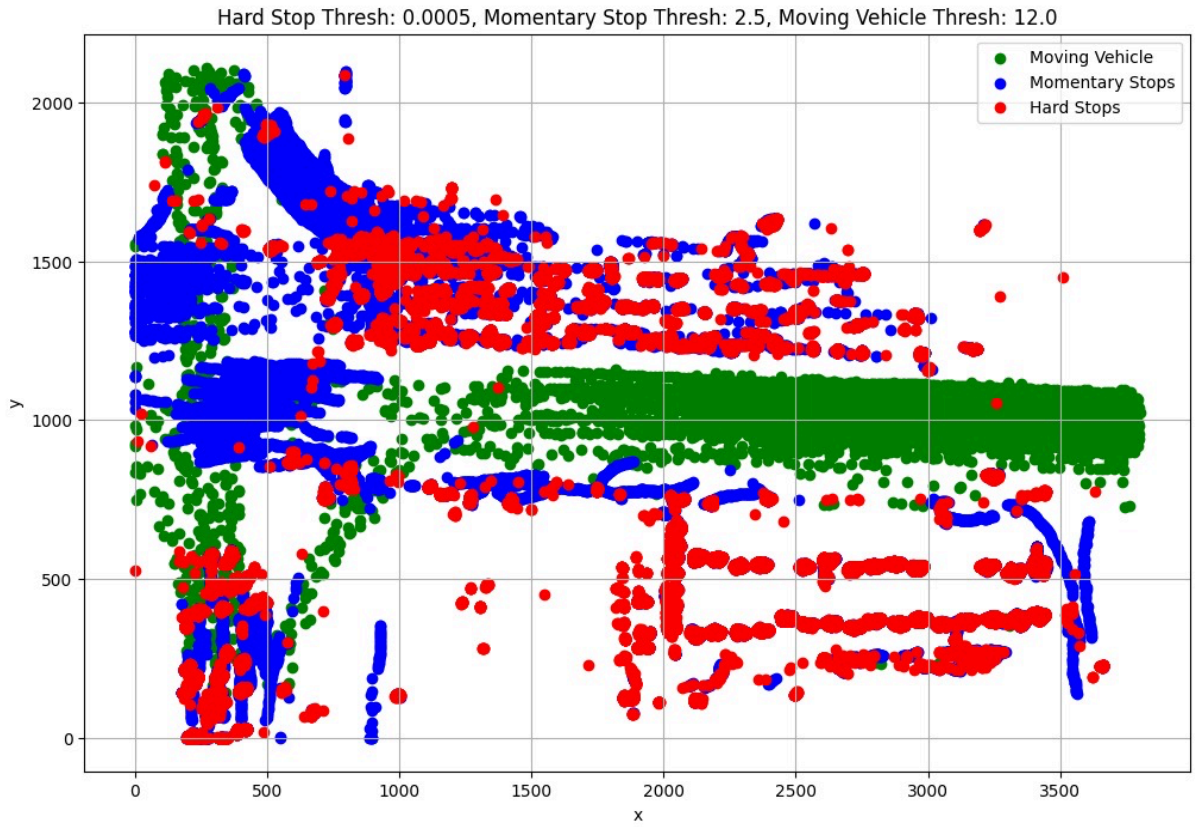

Hard Stop Thresh: 0.0005, Momentary Stop Thresh: 1.7, Moving Vehicle Thresh: 8.0
Moving Vehicle
Momentary Stops
Hard Stops
y
x
0
500
1000
1500
2000
2500
3000
3500

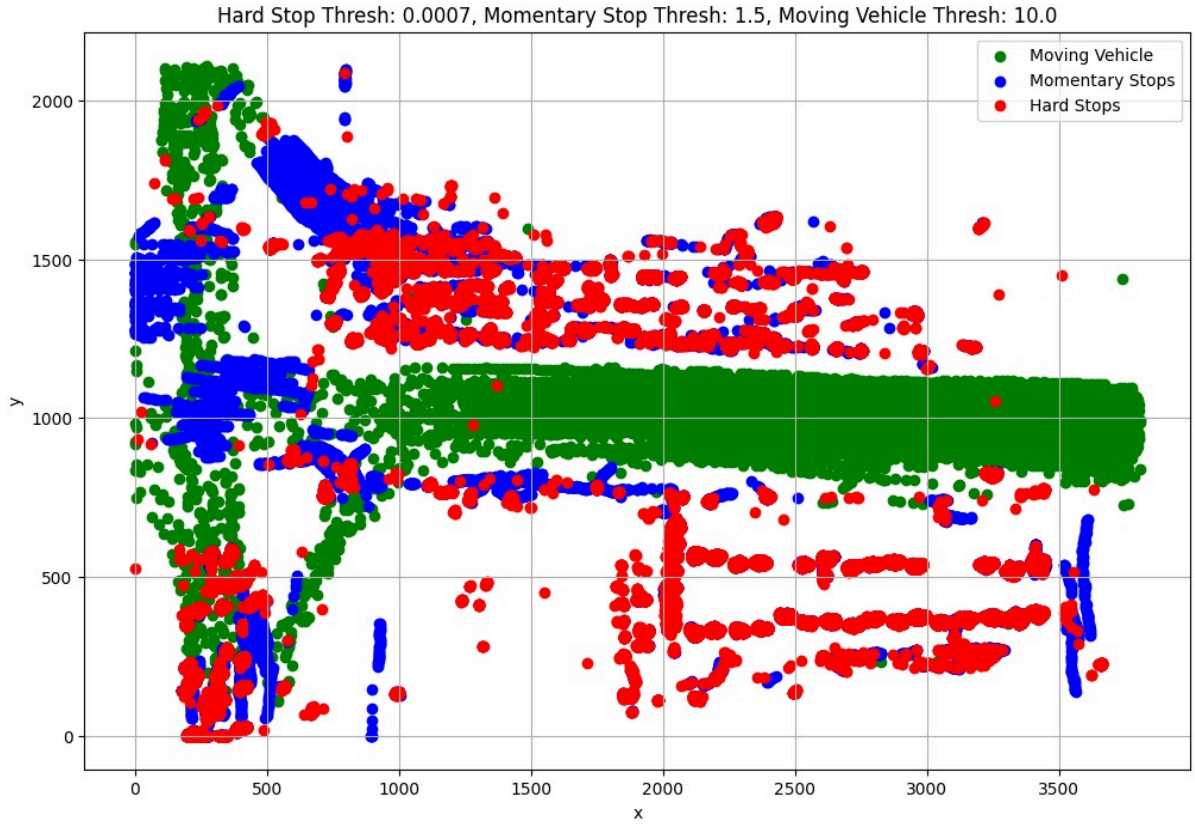

Hard Stop Thresh: 0.0005, Momentary Stop Thresh: 1.7, Moving Vehicle Thresh: 11.0
Moving Vehicle
Momentary Stops
Hard Stops
y
x
0
500
1000
1500
2000
2500
3000
3500


Hard Stop Thresh: 0.0005, Momentary Stop Thresh: 1.7, Moving Vehicle Thresh: 12.0
Moving Vehicle
Momentary Stops
Hard Stops
y
x
0
500
1000
1500
2000
2500
3000
3500


Hard Stop Thresh: 0.0005, Momentary Stop Thresh: 2.0, Moving Vehicle Thresh: 8.0
Moving Vehicle
Momentary Stops
Hard Stops
y
x
0
500
1000
1500
2000
2500
3000
3500


Hard Stop Thresh: 0.0005, Momentary Stop Thresh: 2.3, Moving Vehicle Thresh: 12.0
Moving Vehicle
Momentary Stops
Hard Stops
y
x
0
500
1000
1500
2000
2500
3000
3500

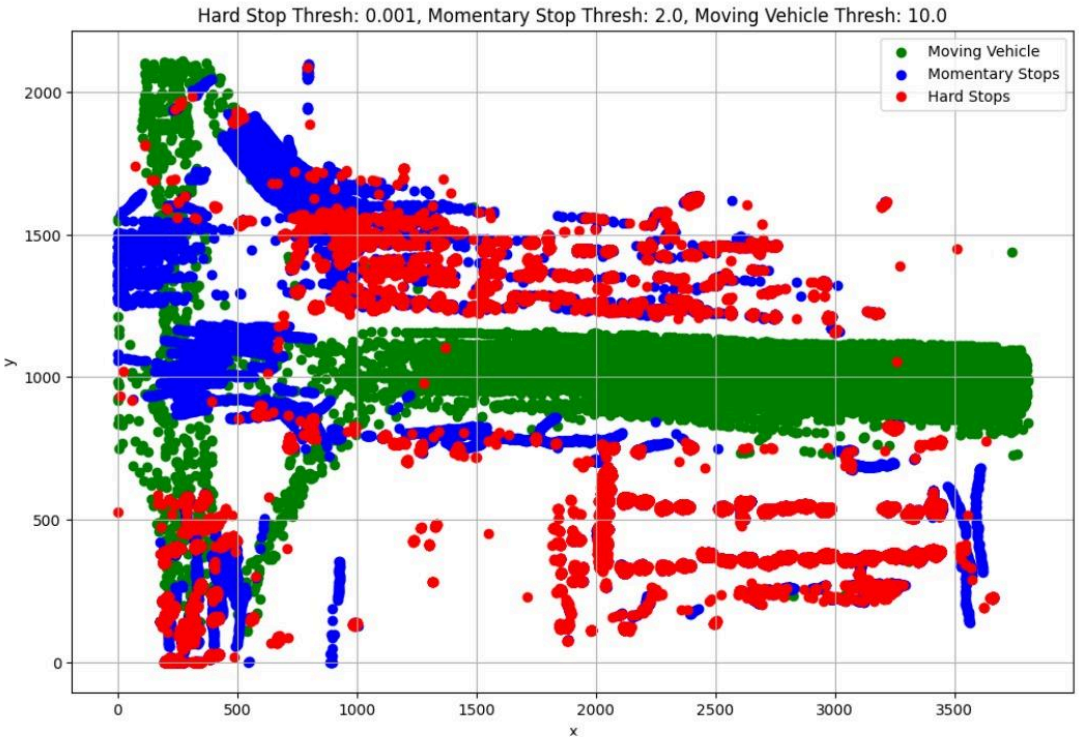

Hard Stop Thresh: 0.0005, Momentary Stop Thresh: 2.5, Moving Vehicle Thresh: 12.0
Moving Vehicle
Momentary Stops
Hard Stops
y
x
0
500
1000
1500
2000
2500
3000
3500



Observations & Next Steps

The generated scatter plots helped us understand how different threshold values affect the classification. Currently we got the best results for:

```
hard_stop_threshold = 0.001

momentary_stop_threshold = 2.0

moving_vehicle_threshold = 10.0
```



Next Steps

- **Refinement Needed:** Based on the visual data, further adjustments may be necessary to optimize the thresholds.
- **Next Steps:** We will try to implement the Random forest model for these threshold values

Initial Model

```
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

imputer = SimpleImputer(strategy='mean')
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)
# Train the model
model = RandomForestClassifier()
model.fit(X_train_imputed, y_train)

# Evaluate the model
```

```
y_pred = model.predict(X_test_imputed)
print("Accuracy Score:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print(y_test,y_pred)
```

This model is designed for classification tasks using the **Random Forest Classifier** from the `sklearn` library. First, it handles missing values in the dataset by replacing them with the **mean** of the respective feature using `SimpleImputer`. It applies this transformation to both the training (`X_train`) and testing (`X_test`) datasets to ensure consistency. Then, it trains a **Random Forest Classifier**, which is an ensemble learning method that builds multiple decision trees and combines their predictions for better accuracy and robustness. Once the model is trained on the imputed training data (`X_train_imputed`), it predicts the labels (`y_pred`) for the test dataset (`X_test_imputed`). Finally, the script evaluates the model's performance by calculating the **accuracy score** and displaying a **classification report**, which includes precision, recall, and F1-score. The actual and predicted labels (`y_test` and `y_pred`) are also printed for further analysis.