Minhaz Uddin, S M Rifaiya Abrar, MD Tarek Shikdar, S M Asif Azad

GNSS Disruption Detection and Early Warning System

Applied Machine Learning

School of Technology and Innovations
Master of Science

Vaasa 2024

UNIVERSITY OF VAASA
School of Technology and Innovations
Author: Minhaz Uddin, S M Rifaiya Abrar, MD Tarek Shikdar, S M Asif Azad
Title of the Project: GNSS Disruption Detection and Early Warning System
Group Name: Jammer 3
Subject Applied Machine Learning
Subject Code ICAT3210-3007
Degree: Master of Science
Supervisor: Petri Välisuo
Year: 2024 **Pages:** 23

ABSTRACT : Global Navigation Satellite System (GNSS) disruptions generated by intentional or unintentional jamming represents significant challenges to the navigation and location based services. This project integrates two core objectives: GNSS jamming detection and classification, and the development of an early warning system. Then we used datasets (odometer readings, GNSS logs) to implement preprocessing (handle missing values, standardize, engineer features) on data for analysis. To detect and classify anomalies, we employed machine learning algorithms like Isolation Forest and DBSCAN, then employed an LSTM model to predict future disruptions. Results are visualized using high quality graphics, providing evidence that the system is good at identifying GNSS anomalies, and forecasting potential disruptions. Although directed on regions, this work provides a few critical insights to advance GNSS reliability and safety in real world applications.

KEYWORDS: (Jamming Detection, Early Warning System, LSTM prediction, Anomaly detections, GNSS Disruption)

Contents

1 Introductions

GNSS are required by many modern technologies such as self-driving cars, aviation, and maritime activities. Unfortunately, though, these systems tend to struggle with signal disruptions, particularly from jamming. Because such problems can have an impact on navigation, communication and safety, it is desirable to be able to detect and predict disruptions.

This project aims to solve these problems by combining two systems: Two of these are designed to detect and classify GNSS disruptions due to jamming, and another to predict possible future ones using past data. Together, these systems offer solutions to address preexisting problems, as well as to mitigate future ones.

The data sets that are used are location, time, speed as well as signal quality. The data was prepared by taking steps like fixing missing data, removal of errors done and features such as speed, weighted speed and change in direction etc. were calculated. This was achieved using Machine learning models such as Isolation Forest, DBSCAN, LSTM to detect disruptions and predict future issues.

This report explains the methods and details of project, data preparation, feature calculations, models and results. Our findings indicate that through detection and prediction, building safety and reliability with GNSS can be enhanced. The system, though, is today most effective in certain regions, and may require refinement for use in cities or in a hostile environment.

2 Project Objective

We combine the two projects to detect, classify, and predict GNSS signal disruptions to enhance the reliability and safety of navigation and location-based services. It detects, classifies, and predicts GNSS anomalies, offering reactive and proactive solutions for improving GNSS reliability in real-world applications. But we used only specific area data. So maybe it will not be fully functional for the city area or those areas with high rise constrictions.

2.1 Problem Task of the Project

1) **Jamming Detection and Classification:** This part of the project aims to detect and classify GNSS signal anomalies, mainly those caused by intentional or unintentional jamming, which can severely impact navigation and communication systems.

2) **Early Warning System:** The early warning system aims to predict potential future signal disruptions based on past patterns and current data, giving users a heads-up about likely issues.

2.2 Importance

1. **Jamming Detection:** This helps to protect critical systems like autonomous vehicles, aviation and maritime from GNSS disruptions

2. **Early Warning System:** This system provides mainly predictive insights and allows preventive measures to minimize impacts of GNSS signal loss or degradation.

3 Data

3.1 Data Documentation

Describe the Datasets: For the jamming detection part used odometer data and nmea.csv data file we used filed such as latitude, longitude, timestamp and signal to noise ratio etc.

Data Source: GNSS logs in the form of .csv and .log files, such as nmea.csv, gnss_log_2024_09_10_14_21_50.pos, and
gnss_log_2024_09_10_14_21_50.24o.
Data includes key fields such as timestamp, latitude, longitude, speed, and signal-to-noise ratio (SNR).

The nmea.csv file provides latitude (lat), longitude (lon), and timestamps for plotting paths and analyzing GNSS signal variations.

Other files contain additional metadata for signal quality, speed, and potential anomalies. For the Early warning system, we are using nmea data file to train our data set and based on that data set forecasting future disruptions.

3.2 Data Structure

1. **Lat And Lon:** Geographic coordinates.
2. **Timestamp:** Time of GNSS measurements.
3. **Speed:** Derived from latitude and longitude differences over time.
4. **Time_Diff:** Time differences between consecutive measurements.
5. **Direction_Change:** Change in movement direction calculated from latitude and longitude.

4 The Data Preprocessing and Feature Calculation

4.1 The Data Preprocessing

1. **Handling Missing Values:** Missing values in critical columns (lat, lon, speed) were filled to ensure continuity.
2. **Handling Outliers:** Outliers in features like speed and SNR were capped to avoid skewing machine learning models.
3. **Standardization:** Features were standardized using StandardScaler for compatibility with machine learning algorithms like DBSCAN and Isolation Forest.

4.2 Feature Calculation

1. Speed: Calculated using the Haversine formula based on differences in latitude and longitude over time.
2. Time Difference (time_diff): Derived from consecutive timestamps to understand the temporal aspect of anomalies.
3. Direction Change: Calculated as the arctangent of latitude and longitude changes to identify sudden directional shifts, often linked to GNSS interference.

5 Machine Learning Algorithms

5.1 Algorithms

Isolation Forest:

1) Used for detecting GNSS anomalies as it isolates outliers based on feature distributions.
2) Suitable for high-dimensional datasets and unsupervised anomaly detection.
3) The contamination parameter was optimized for this dataset.

DBSCAN:

1) Used for clustering anomalies, identifying groups of related disruptions (e.g., clusters of jamming events).
2) Effective for handling spatial data and clustering anomalies with arbitrary shapes.

LSTM (Early Warning System):

1) Applied for the Early Warning System to predict potential GNSS anomalies based on historical patterns in speed, time difference, and direction changes.

6 Results

6.1 Visualization

1. **Speed Over Time:** Line plots showing speed variations with anomalies marked in red, providing a clear view of disruption patterns.
2. **Path Visualization:** Interactive map using Folium to plot the journey, marking the start, end, and detected anomalies. Anomalies are color-coded for better interpretation.
3. **Predicted vs. Detected Anomalies:** Time-series plots comparing predicted anomalies (from the Early Warning System) with detected anomalies.
4. **Clustering Results:** Scatter plots with clusters identified by DBSCAN, showing geographical zones of GNSS disruptions.

6.2 GNSS Data Collection Map and Distance

Here folium.Marker is used to mark specific locations on the map using latitude and longitude coordinates.

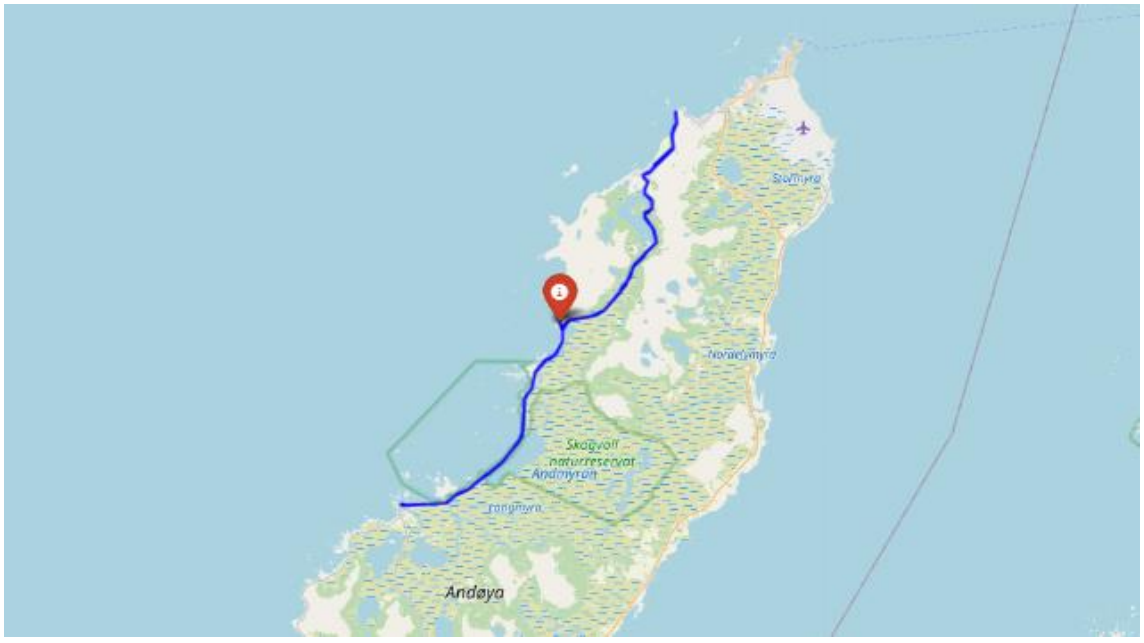

Figure 1: shows the GNSS data collection location

Our analysis indicates that the subject traveled a cumulative distance of 58.86 kilometers. According to the provided dataset, module function is used for identifing the the distance

```
total_distance = 0.0

for i in range(1, len(data)):

    prev_coord = (data[lat_column].iloc[i-1], data[lon_col-
umn].iloc[i-1])
    curr_coord = (data[lat_column].iloc[i], data[lon_col-
umn].iloc[i])
    distance = geodesic(prev_coord, curr_coord).meters
    total_distance += distance

total_distance_km = total_distance / 1000

print(f"Total distance traveled: {total_distance_km:.2f} km")
```

6.3 Description for the Analysis

In this analysis for GNSS Jamming Detection and Classification, we processed with GPS data from given nmea.csv data file. Which contains latitudinal and longitudinal coordinates, as well as timestamp data. Here a timestamp converted to datetime format, enabling the calculation of **time differences** between consecutive points, allowing us to account for the Earth's curvature in determining the actual distances traveled.

The timestamp was converted to datetime format, enabling the calculation of **time differences** between consecutive points. We employed the **Haversine formula** to calculate distances between these points, allowing us to account for the Earth's curvature in determining the actual distances traveled.

Then we computed the **speed** by dividing the distance by the time difference between consecutive GPS readings. Using these calculated metrics, we identified **anomalies** by defining thresholds for both speed and time differences. anomalies were marked if the speed or time difference exceeded the **95th percentile** of the respective distribution.

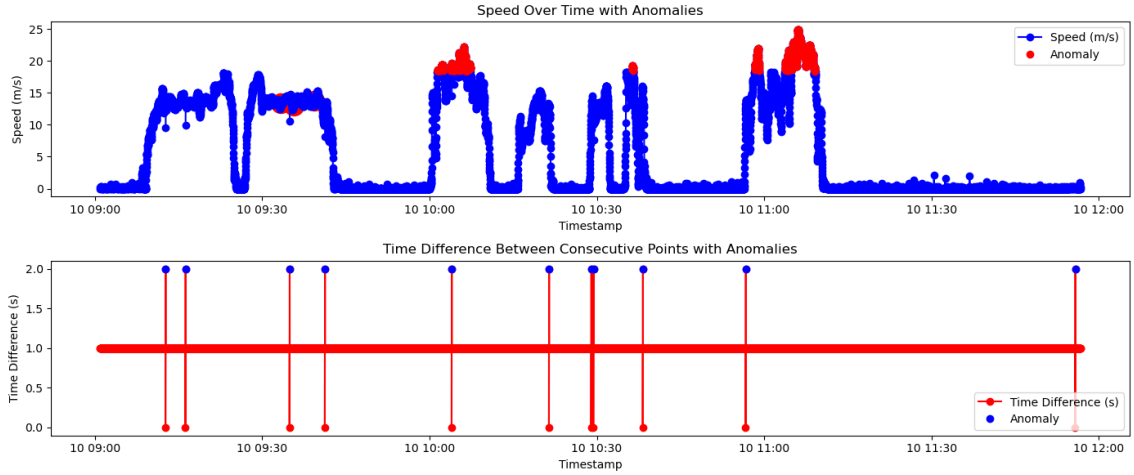

Figure 2: Speed and Time Differences with Detected Anomalies

6.4 Algorithmns for Detection and Classification

6.4.1.1 DBSCAN

We used unsupervised learning **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise) for clustering data based on density. It groups data points that are close to each other into clusters and identifies anomalies. our goals is to identify anomalies, such as irregular speed changes or unexpected time differences.

```
# Apply DBSCAN with adjusted parameters
dbscan = DBSCAN(eps=eps_value, min_samples=min_samples_value)
clusters = dbscan.fit_predict(features_scaled)
data_sample['cluster'] = clusters

# Mark anomalies with more lenient criteria
data_sample['cluster_anomaly'] = data_sample['cluster'] == -1

# Simplify cross-referenced anomalies for testing
data_sample['cross_referenced_anomaly'] = (
    data_sample['speed_anomaly'] | data_sample['time_diff_anomaly']
)
# Save anomalies again
anomalies = data_sample[data_sample['cross_referenced_anomaly']]
anomalies.to_csv("cross_referenced_anomalies.csv", index=False)
```

Using this lerning we saved the **cross-referenced anomalies** into a CSV file. By saving anomalies to a CSV file, we can **access the results later** without needing to rerun the

entire analysis thats why we save the data file . we save this to generate **charts and visualizations** of the anomalous behavior.

	timestamp	lat	lon	height	speed	time_diff	direction_	cluster	cluster_anomaly	speed_anomaly	time_diff_anomaly	cross_referenced_anomaly
2	9/10/2024 9:12	69.19486	15.84092	5.5	0.000178	2	-2.78611	0	FALSE	FALSE	TRUE	TRUE
3	9/10/2024 9:16	69.17088	15.81819	9.4	8.95E-05	2	-1.55404	0	FALSE	FALSE	TRUE	TRUE
4	9/10/2024 9:23	69.1371	15.71793	12.5	0.000458	1	-3.11758	0	FALSE	TRUE	FALSE	TRUE
5	9/10/2024 9:23	69.13708	15.71702	13.6	0.000457	1	-3.11971	0	FALSE	TRUE	FALSE	TRUE
6	9/10/2024 9:23	69.13707	15.71656	14	0.000458	1	-3.11976	0	FALSE	TRUE	FALSE	TRUE
7	9/10/2024 9:34	69.16425	15.81511	14.2	0.000135	2	0.707025	0	FALSE	FALSE	TRUE	TRUE
8	9/10/2024 9:41	69.20495	15.86258	14.7	8.12E-05	2	2.064738	0	FALSE	FALSE	TRUE	TRUE
9	9/10/2024 10:01	69.21219	15.87206	14.2	0.000459	1	0.0741	0	FALSE	TRUE	FALSE	TRUE
10	9/10/2024 10:01	69.21222	15.87253	13.9	0.000464	1	0.062553	0	FALSE	TRUE	FALSE	TRUE

Figure 3: Cross- referenced anomalies values

By using this data we filter out and remove any remaining NaN or infinite values in speed and time_diff.Then we plot thos data to find **Geographical Distribution of Anomalies**. Based on Longitude, Latitude, Speed and Time Difference. Our main goal was here to find out Statistical Summary of Anomalies.

```
Statistical Summary of Anomalies:
              speed   time_diff
count   117.000000  117.000000
mean      0.000462    1.094017
std       0.000120    0.293108
min       0.000002    1.000000
25%       0.000460    1.000000
50%       0.000482    1.000000
75%       0.000524    1.000000
max       0.000561    2.000000
```

Our goal of the plot is to visually identify patterns in the geographical distribution of GNSS anomalies that were detected in the dataset. It shows the **locations of anomalies** with additional context provided by **speed** and **time difference** - making it easier to understand the conditions under which these anomalies occurred.

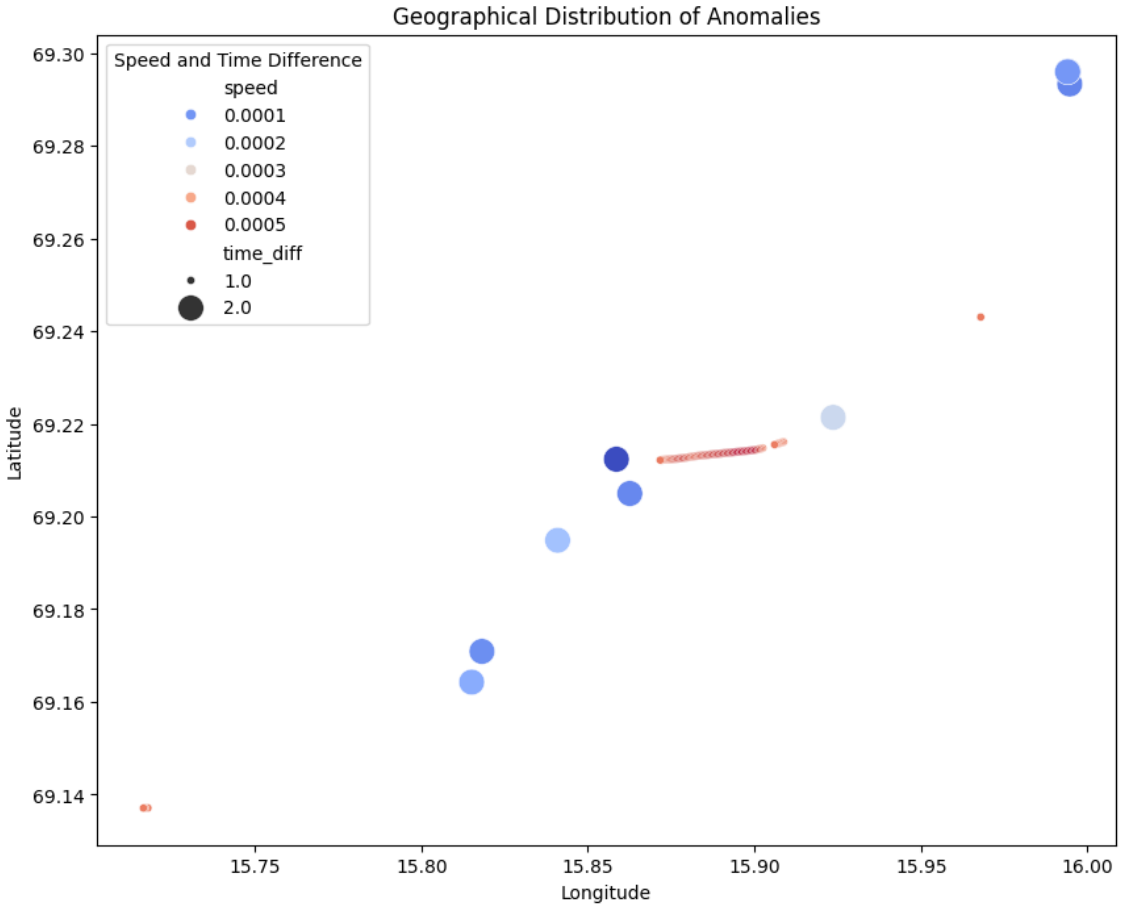

Figure 4: Geographical Clustering of Anomalies

6.4.1.2 KMeans Clustering

We used KMeans clustering to anomalies and used only lat and lon for clustering the anomalies to again identify the and cross verify Geographical Clustering of Anomalies based on location cluster.

```
# Applying KMeans clustering to anomalies
kmeans = KMeans(n_clusters=3, random_state=42).fit(coords)
anomalies['location_cluster'] = kmeans.labels_
# Visualizing clusters
plt.figure(figsize=(10, 8))
sns.scatterplot(data=anomalies, x='lon', y='lat', hue='loca-
tion_cluster', palette='viridis', size='speed', sizes=(20, 200))
plt.title("Geographical Clustering of Anomalies")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.legend(title="Location Clusters")
plt.show()
```

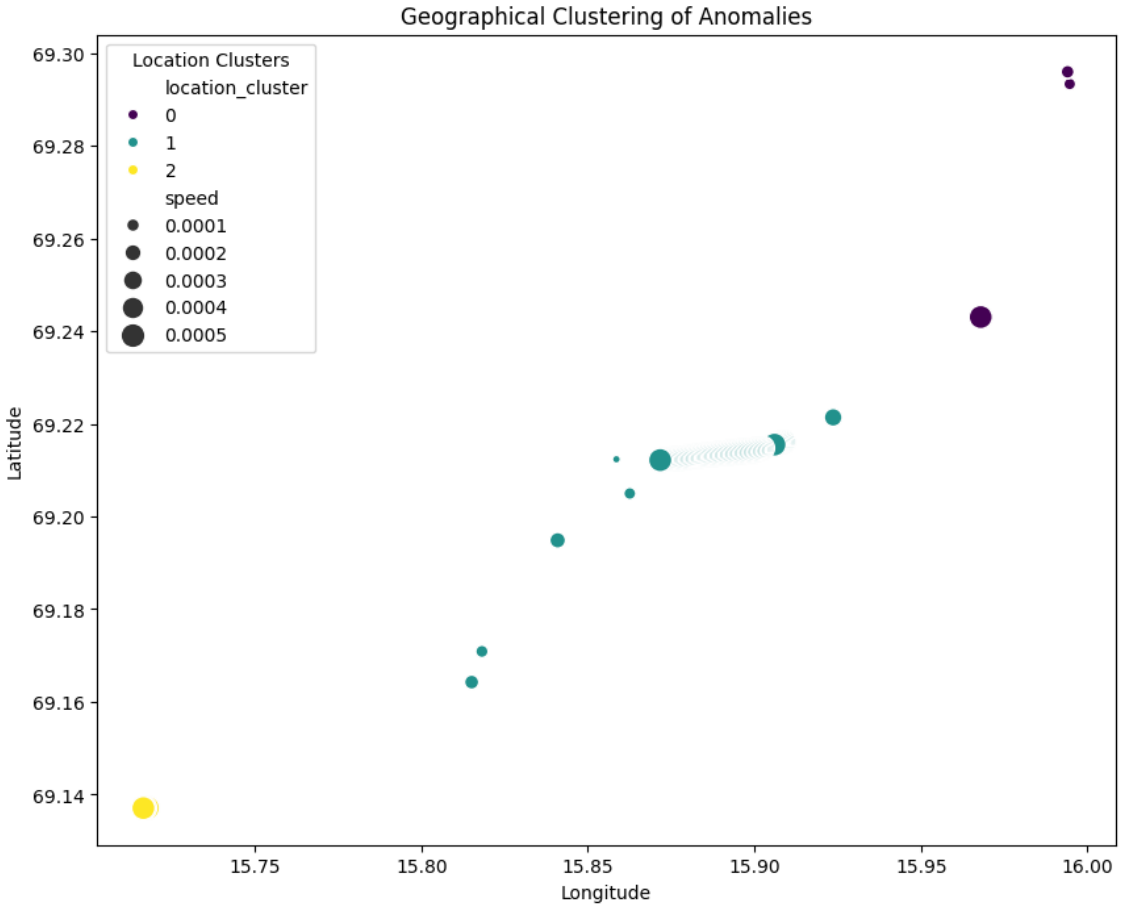

Figure 5: Geographical Clustering of Anomalies (Based on location Cluster)

6.4.1.3 Understanding Cluster Characteristics

Here our plan was to calculate average speed and time_diff for each cluster. We wanted to compare the clusters whether certain clusters are linked with specific speeds or timing behaviors or not.

```
cluster_stats = anomalies.groupby('location_cluster')[['speed',
'time_diff']].mean()
print("Average Speed and Time Difference by Location Cluster:")
print(cluster_stats)
```

Result:

```
Average Speed and Time Difference by Location Cluster:
                      speed  time_diff
location_cluster
0                  0.000158   1.833333
1                  0.000479   1.055556
2                  0.000458   1.000000
```

6.4.1.4 Anomalies Map

Figure 6: Anomalies Map (Red Points)

6.5 GNSS Early Warning System

6.5.1.1 Objective

Our plan to make early warning system that capable of detecting GNSS (Global Navigation Satellite System) anomalies that may indicate signal degradation, interference or jamming. The method we used to detect anomalies is machine learning-based approach. We applied an **Isolation Forest** model to identify anomalous behavior in speed and other derived features from GNSS data in **nmea.csv.**

6.5.1.2 Model Applying

```
features = data[['speed', 'speed_diff', 'time_diff', 'direc-
tion_change']]

# standardize the feature
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)
```

```
#isolation forest apply
isolation_forest = IsolationForest(contamination=0.05, ran-
dom_state=42)
data['anomaly'] = isolation_forest.fit_predict(features_scaled)
data['anomaly'] = data['anomaly'] == -1
```

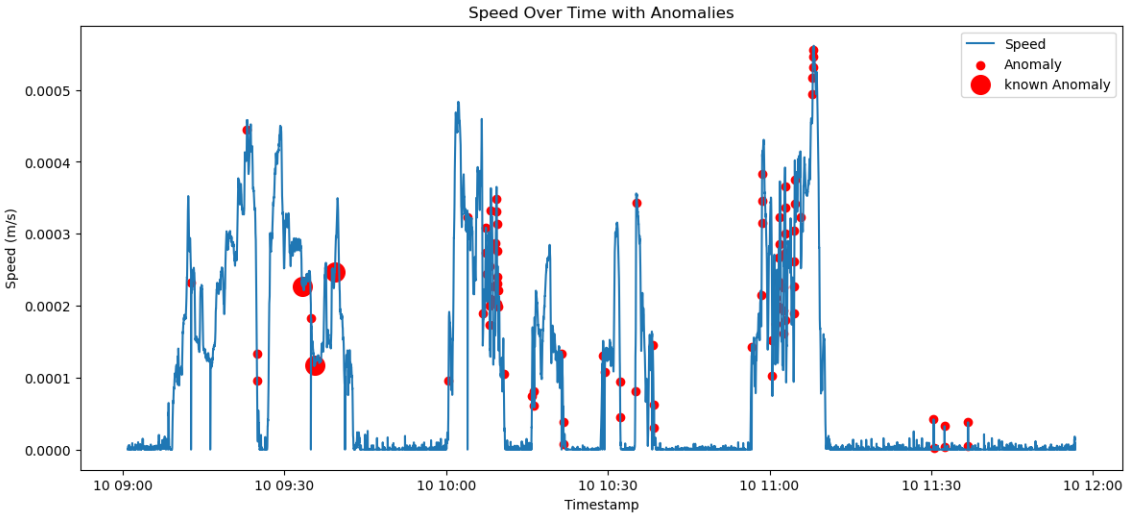

Figure 7: Speed Over Time Anomalies (Isolation Forest Model)

6.5.1.3 LSTM

For capturing complex temporal relationships in GNSS data, we increase our model to include a Long Short-Term Memory (LSTM) network, which is a specialized type of recurrent neural network capable of learning time-dependent patterns effectively. We used LSTM modeling by creating input sequences of length 10 using a sliding window approach and reshaping them into a 3D tensor to be used as LSTM input

```
window_size = 10  #lstm input sequences
X, y = [], []# sequences ,label LSTM
for i in range(len(features_scaled) - window_size):
    X.append(features_scaled[i:i + window_size])
    y.append(data['anomaly'].iloc[i + window_size])
X, y = np.array(X), np.array(y)

# Reshape X to(samples,time steps,features)
X = X.reshape((X.shape[0], X.shape[1], X.shape[2]))
```

6.5.1.4 Train Models For Prediction

We use Sequential, LSTM models to train **input sequences** and using **validation data**, we ensured that the model was able to generalize effectively, identifying anomalies not only during training but also in new, unseen GNSS condition.

```
model = Sequential()
model.add(Input(shape=(window_size, X.shape[2])))
model.add(LSTM(50, activation='relu'))
model.add(Dense(1, activation='sigmoid'))


model.compile(optimizer='adam', loss='binary_crossentropy', met-
rics=['accuracy'])

# train the model
history = model.fit(X, y, epochs=10, batch_size=16, valida-
tion_split=0.3)
```

6.5.1.5 Predict Future Anomalies

By predicting future anomalies, the system can act as an **early warning mechanism** to alert users of potential GNSS signal issues before they significantly impact navigation. So we train our model and using that training we predit a graph .

```
# Predict future anomalies
y_pred = model.predict(X)
predicted_anomalies = y_pred > 0.5
data['predicted_anomaly'] = np.nan
data.loc[window_size:, 'predicted_anomaly'] = predicted_anoma-
lies.flatten().astype(float)
```

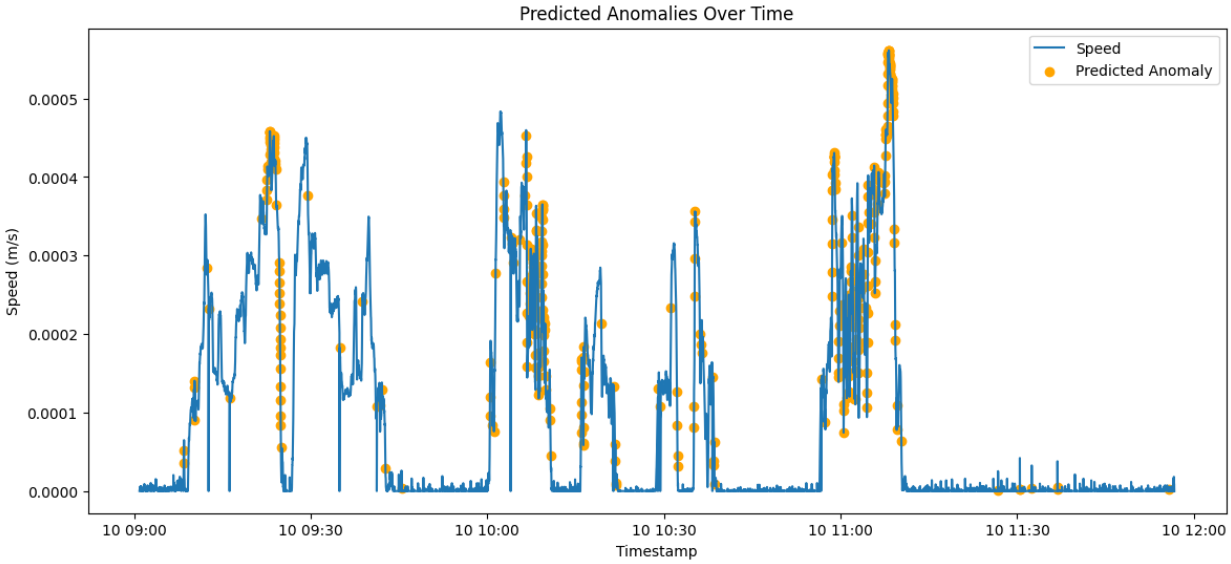

Figure 8: Speed Over Time Anomalies (Isolation Forest Model)

7 Limitations

A limitation of this analysis is that it has been conducted using data from a specific rural countryside area, where GNSS signal patterns are relatively stable and less affected by urban interference. Consequently, we are uncertain about the system's performance in urban or city environments, where factors like high-rise buildings, dense infrastructure, and increased signal obstructions could introduce additional complexities. Currently, no GNSS data is available for such urban areas to validate or refine the system for these conditions. However, we are optimistic that with proper tuning and additional data, the analysis and models can adapt to handle urban GNSS challenges effectively.

8 Ground Truth

The ground truth for this project involves a labeled dataset containing GNSS readings with fields such as timestamp, latitude, longitude, speed, signal-to-noise ratio (SNR), and anomaly indicators. Each data point is annotated as normal or anomalous, with anomalies further classified into types like intentional jamming or unintentional interference. Key timestamps such as 2024-09-10 09:33:20, 09:35:40, and 09:39:20 are included as validation points. The ground truth enables anomaly detection and classification, as well as early warning predictions. This ensures accurate and reliable detection of disruptions and forecasting of potential signal issues.

9 Conclusion

The goal of the project is the constructing of a GNSS Disruption Detection and Early Warning System (DEWS) based on latest machine learning techniques. Addressing the critical challenges posed by intentional or unintentional GNSS signal jamming, the system integrates two essential components: Detection and classification of GNSS disruptions in real time, and prediction of future anomalies... The project was able to perform robust anomaly detection and forecasting using algorithms such as Isolation Forest, DBSCAN and LSTM.

Data preprocessing was premised on handling missing values, standardizing data as well as engineering features such as speed, direction change and time differences. This made the data fit for applied models. Moreover it was proved effective by the machine learning methods used: Isolation Forest and DBSCAN for detecting anomalies and clustering-related disruptions, and LSTM with a predictive mechanism to anticipate disruptions in the future.

Results visualization, including path mapping and clustering anomalies, provided useful means to identify GNSS disruptions and their geographical distribution. In areas of stable GNSS signal patterns, the system's capability in proactively detecting and classifying anomalies has great potential to improve navigation systems' reliability and safety.

One major limitation of the project was the fact that the dataset given to work on was fairly limited, generally focused on rural regions with little consideration for any other similarities. This limited the validity of the system to more complex urban environments in which interference to GNSS signals from infrastructure and densification of the building layout is present. Nevertheless, due to the lack of urban data, the future scalability of the model to diverse settings is questionable, although the system offers the promise of adapting with further tuning as data collection increases.

While these challenges can be expected to persist for some time, the project nevertheless makes a valuable contribution to the applied usage of machine learning and GNSS safety. The combination of reactive detection and predictive capabilities provides a solid platform from which to build more robust GNSS reliability systems. This work can be expanded in future by including varied datasets as well as refine existing models for application in urban and high risk areas to improve the significance of the system in terms of global navigation safety and performance.

References

Välisuo P., Data Collection: https://github.com/pevalisuo/AML/blob/master/Exercises/Jammer_detection