Internship at Qualcomm India Private Limited

"Automated Scenario Extraction using Machine Learning"

Submitted by:

Kalyani Kulkarni [redacted]

Under the guidance of

[redacted]

Six Months

[redacted]

ABSTRACT

I. Introduction

Qualcomm is enabling a world where everyone and everything can be intelligently connected. Qualcomm's one technology roadmap allows them to efficiently scale the technologies that launched the mobile revolution – including advanced connectivity, high-performance, low-power compute, on-device intelligence and more – to the next generation of connected smart devices across industries. Innovations from Qualcomm and the family of Snapdragon platforms will help enable cloud-edge convergence, transform industries, accelerate the digital economy, and revolutionize how we experience the world, for the greater good.

Qualcomm Incorporated includes their licensing business, QTL, and the vast majority of their patent portfolio. Qualcomm Technologies, Inc., a subsidiary of Qualcomm Incorporated, operates, along with its subsidiaries, substantially all of their engineering, research, and development functions, and substantially all of their products and services businesses, including the QCT semiconductor business. Snapdragon and Qualcomm branded products are products of Qualcomm Technologies, Inc. and/or its subsidiaries. Qualcomm patented technologies are licensed by Qualcomm Incorporated. Qualcomm is a trademark or registered trademark of Qualcomm Incorporated.

The internship project titled “Automated Scenario Extraction using Machine Learning” was conducted at Qualcomm India Private Limited, aimed at dynamically extracting scenario information through the application of a machine learning model that utilizes data from multiple sensors. The projects spanned a duration of 6 months and aimed to allow users to be able to search for specific scenarios and consequently be able to find said scenarios and the timestamps at which they occur. The aim of this internship was also to improve on a pre-existing tool which did something similar using a heuristic approach.

II. Job Responsibilities and Duties

During my internship at Qualcomm, I was a part of the Data Enrichment Team. This team is tasked with the critical function of processing sensor data from various sources such as Radar, Lidar, GPS, and Camera systems. Our objective is to enrich this data, enhancing its value through detailed analysis, which is pivotal for innovations in the automotive and self-driving sectors.

As an Interim Engineering Intern, my responsibilities were centered around the application of machine learning algorithms to improve the existing data processing tools. The model I developed was particularly focused on identifying specific scenarios within the sensor data. This capability is essential for recognizing patterns and critical events that are fundamental to the development and safety of autonomous driving systems.

My role was situated within the broader organizational structure, contributing to the company's mission of leading the world in next-generation technology solutions. The work done by our team supports Qualcomm's strategic goals by ensuring high-quality data processing, which is a cornerstone of reliable and efficient autonomous vehicle technology.

III. Project Details:

1. **Project Introduction:**

   The project commenced with a comprehensive analysis of the existing tool, which involved a meticulous process of reverse engineering. Due to the tool being dormant at the time, it was imperative to procure the correct versions of the associated libraries. This step was crucial to ensure compatibility and functionality. Additionally, minor code adjustments were required—specifically one or two lines—though the exact nature of these modifications remains deliberately ambiguous to maintain the focus on the broader aspects of the project.

2. **Analysis and Optimization of the Tool**

   Subsequent to the initial setup, I delved into the tool's functionality, which primarily focused on extracting scenarios pertaining to vehicular movements. These scenarios were categorized into two distinct types: 1) host scenarios, which involve only the ego vehicle, and 2) target

scenarios, encompassing interactions between the ego and target vehicles. The decision was made to initially concentrate on the host scenarios exclusively.

In an effort to optimize the tool's performance, I implemented a timing function for each segment of the pipeline. It was observed that the most time-consuming process was the retrieval of target data. Given the project's current focus on host scenarios, I modified the code to extract solely the host data. This strategic exclusion of target scenario extraction resulted in a significant reduction in the tool's runtime. The performance enhancement was quantifiable, with the runtime decreasing from approximately **14.3747 seconds** to **2.4788 seconds**, equating to an **82.75%** reduction in total runtime, as calculated by the formula:

$$Percentage\ Increase = \left(\frac{Original\ Time - New\ time}{Original\ Time}\right) \times 100$$

3. **Scenario Analysis and Machine Learning Model Preparation**

Moving forward, I conducted an in-depth analysis of the various host scenarios and the parameters they entailed. This was a pivotal step to align the scenarios with the inputs required by the machine learning model. The scenarios and their corresponding parameters are as follows: (Please note that although the tool has identified a range of scenarios, I have detailed below only those that I intend to refine and enhance through the application of machine learning techniques.)

- **Deceleration Scenarios:**
  - Longitudinal Velocity (`longVel`)
  - Longitudinal Acceleration (`longAcc`)
  - Time (`time`)
- **Stationary Scenarios:**
  - Longitudinal Velocity (`longVel`)
  - Time (`time`)
- **Lane Change Scenarios:**
  - Time (`Time`)
  - Lateral Velocity (`latVel`)

- Lateral Distance Between Centers (`latDistCenters`)
- Lane Assignment (`laneAssignment`)
- Relative Lanes (`relLanes`)

- **Ego Turning Scenarios:**
  - Ego Heading (`egoHeading`)
  - Time (`time`)
  - Yaw Rate (`yawRate`)

4. **Dataset Creation and Timestamp Association**

With the necessary parameters identified, I embarked on creating the initial dataset. This involved processing approximately 15 files and modifying the code to extract the required inputs such as `longVel`, `time`, `longAcc`, `latDistCenters`, `relLanes`, `egoHeading`, and `yawRate`. Initially, the input signals were extracted in their raw form with the intention of appending the identified scenarios at a later stage.

Initial Dataset:

A	B	C	D	E	F	G	H	I
longVel	time	longAcc	laneAssign	latDistCen	relLanes	egoHeadin	yawRate	
[9.1547787 9.01661024	[1.21284	[-		[[			[	
8.94212151 8.85070183	538e+09	1.191258		nan]			2.244558	
8.749525 8.61574826	1.212845	71e+00 -		[			09e-02	
8.4967869 8.39486587	38e+09	9.656690		nan]			2.669555	
8.30902985 8.20690044	1.212845	78e-01 -		[	[[nan nan		49e-02	
8.06937107 7.93329425	38e+09	8.273693		nan]	nan ...		1.909006	
7.79016963 7.63752693	1.212845	65e-01 -		[	nan nan		18e-02	
7.4746485 7.32225075	38e+09	9.650123		nan]	nan]		2.045008	
7.20539572 7.06868044		91e-01		[	[nan nan		46e-02	
6.92984884 6.80442638	1.212845	-		nan]	nan ...			
6.68766182 6.6240349	38e+09	1.176797		[	nan nan		2.488000	
6.57528021 6.52754616	1.212845	21e+00 -		nan]	nan]		00e-02	

However, during a collaborative sync-up meeting, it was realized that a more granular approach—specifically, a per-timestamp extraction—would be more beneficial. The rationale behind this decision was to enhance the precision of the dataset by ensuring that each data point was accurately associated with its respective scenario.

Updated Dataset:

	A	B	C	D	E	F	G	H	I
1	time	longVel	longAcc	laneAssign	latDistCen	relLanes	egoHeadin	yawRate	
2	1.21E+09	0	0	[]	[nan]	[nan nan n	-93.1083	0	
3	1.21E+09	0	0	[]	[nan]	[nan nan n	-93.116	0	
4	12130900	0	0	[]	[nan]	[nan nan n	-93.1223	0	
5	1.21E+09	0	0	[]	[nan]	[nan nan n	-93.1291	0	
6	1.21E+09	0	0	[]	[nan]	[nan nan n	-93.1372	0	
7	1.21E+09	0	0	[]	[nan]	[nan nan n	-93.1439	0	

To implement this refined approach, I referred to the outputs from the previous tool, which provided the timestamps marking the commencement and conclusion of each scenario within the files. A new column was introduced in the dataset, correlating sensor values with scenario classes based on the timestamps. This was achieved by labeling all sensor values falling within the start and end timestamps with the corresponding scenario class, as identified by the tool. All other values were categorized as 'NoScenario'.

This methodology was initially applied to a single file to validate the logic. Upon confirmation of its semantic correctness, the process was scaled to encompass multiple files, thereby streamlining the dataset preparation for the machine learning model.

Final Dataset:

	A	B	C	D	E	F	G	H	I	J
1	time	longVel	longAcc	laneAssign	latDistCen	relLanes	egoHeadin	yawRate	scenarioClass	
2	1.21E+09	0	0	[]	[nan]	[nan nan n	-93.1083	0	stationary	
3	1.21E+09	0	0	[]	[nan]	[nan nan n	-93.116	0	stationary	
4	12130900	0	0	[]	[nan]	[nan nan n	-93.1223	0	stationary	
5	1.21E+09	0	0	[]	[nan]	[nan nan n	-93.1291	0	stationary	
6	1.21E+09	0	0	[]	[nan]	[nan nan n	-93.1372	0	stationary	
7	1.21E+09	0	0	[]	[nan]	[nan nan n	-93.1439	0	stationary	

5. **Dataset Preprocessing and Model Selection:**

In the initial phase of my project, I embarked on the crucial task of data preprocessing. Given that the final results were categorical, it was imperative to transform them into numerical data.

This conversion is essential because most machine learning models are designed to work with numerical inputs. They require numbers to perform calculations and derive patterns. To facilitate this transformation, I employed Label Encoders. Label Encoders are an excellent choice for this task as they convert categorical labels into a numerical sequence, preserving the order which can be vital for certain models that assume ordinality.

During the nan value removal process, I observed a significant loss of data when eliminating rows with nan values. This was particularly true for lane-related columns, which predominantly contained nan values. To mitigate this issue, I decided to bifurcate the models into lane and non-lane related changes.

For the model selection, I initially opted for Logistic Regression for several reasons:

a. **Establishing a Training and Evaluation Framework**: It was essential to set up an end-to-end training and evaluation pipeline to establish baseline metrics. Jumping straight into complex neural networks without a proper understanding of the data could lead to suboptimal results.
b. **Benchmarking**: Logistic Regression provides a baseline performance metric. This metric is invaluable as it serves as a reference point for future model comparisons and justifies the need for more sophisticated algorithms.
c. **Simplicity and Interpretability**: Logistic Regression is a straightforward model that offers interpretability. This simplicity is beneficial in comprehending the influence of individual features on the predictions, thus elucidating the input-output relationship.

6. **Model Results:**

An important realization during this process was the significance of fixing the random seed. Ensuring reproducibility by fixing the seed guarantees consistent outputs across multiple runs, eliminating variations that could confound result analysis. This practice was instrumental in maintaining sanity while interpreting the outcomes.

Upon finalizing the model, I achieved an accuracy of **86.87%** on the multi-class Logistic Regression model. The classification report generated revealed the following metrics:

Class	Precision	Recall	F1-Score	Support
0	0.89	0.93	0.91	1152
1	0.00	0.00	0.00	41
2	0.82	0.70	0.76	20
3	0.88	0.27	0.41	78
4	1.00	0.20	0.33	41
5	0.82	1.00	0.90	344

The classes correspond to:

0 » No scenarios

1 » Lane Change

2 » Deceleration

3 » Ego Turning

4 » Between Lanes

5 » Stationary

However, I inadvertently included classifications for Lane Change and Deceleration. Despite this, the precision for classes 1 and 4 raised suspicions, prompting me to revise the code. Subsequent to the modifications, the classification report altered significantly, underscoring the importance of a fixed random seed for reproducibility as I had mentioned previously.

7. **Refinement of Model and Addressing Class Imbalance**

   After careful consideration, I decided to exclude scenarios 1 (Lane Change) and 4 (Between Lanes) from the initial model, with the intention of developing a separate model for these cases. This strategic exclusion led to the creation of a new model, which impressively achieved an accuracy of **91.01%**. The classification report for this model is as follows:

Class	Precision	Recall	F1-Score	Support
0	0.94	0.94	0.94	1159
1	0.67	0.15	0.25	13
2	0.90	0.26	0.40	73
3	0.84	0.96	0.90	335

   The labels for this model represent:

   0 » No scenarios

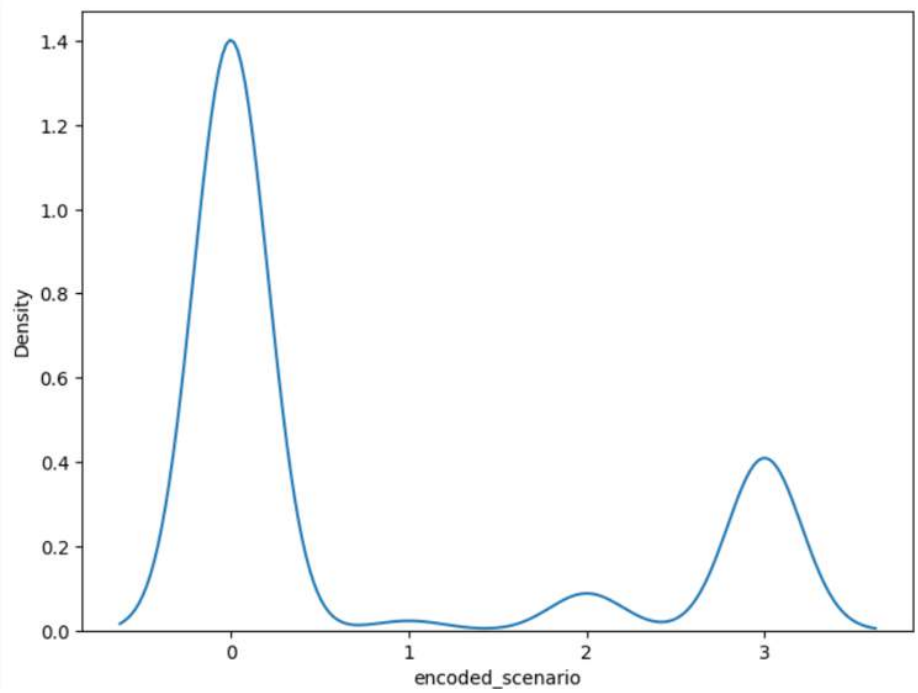
1 » Deceleration

2 » Ego Turning

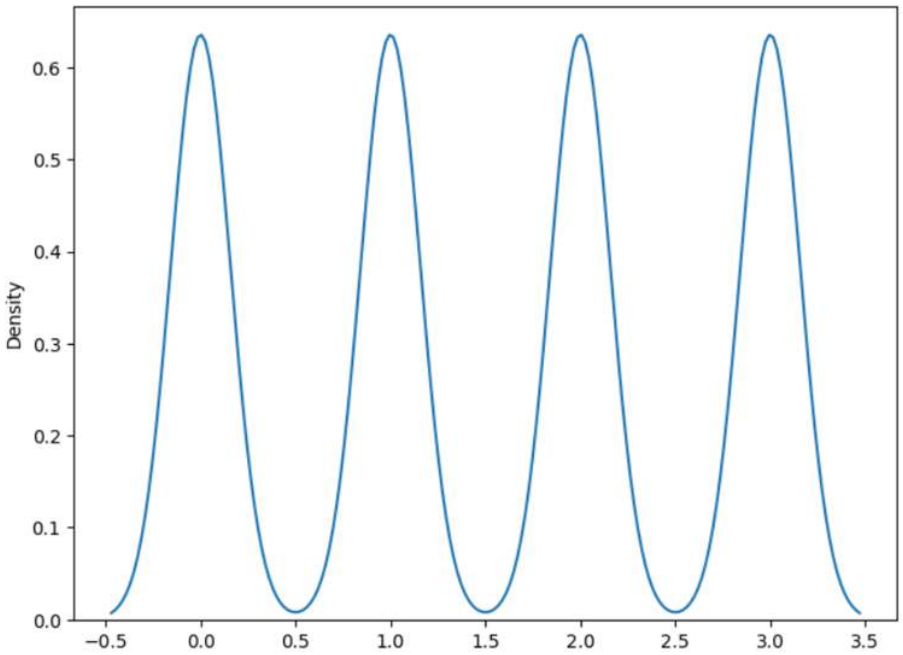
3 » Stationary

It is important to note that the high accuracy initially observed was partly due to the highly imbalanced nature of the classes. Class imbalance can lead to misleadingly high accuracy scores, as the model may simply predict the majority class for most instances. To address this, I embarked on class balancing, which is crucial for training robust models. Balanced classes ensure that the model does not develop a bias toward the majority class and can generalize better to unseen data.

Before Balancing:



After Balancing:



After balancing the classes, the model's accuracy shifted to **77.15%**. The updated classification report is:

Class	Precision	Recall	F1-Score	Support
0	0.97	0.71	0.82	1159
1	0.20	1.00	0.33	13
2	0.21	0.67	0.32	73
3	0.77	1.00	0.87	335

While the accuracy appears to have decreased, this is a more realistic representation of the model's performance across all classes. Balancing the classes is essential because it allows the model to learn from an equal representation of each class, preventing the overfitting to the majority class and improving the model's ability to predict minority class instances accurately. Additionally, I made a pragmatic decision to manually assign numerical values to categorical variables. This approach was taken as the LabelEncoder was not providing consistent labeling across different files. Although this method required additional coding, it simplified the process and ensured consistent labeling, which is vital for the model's performance.

Feature Importance Analysis and Model Refinement

In the subsequent phase of my analysis, I directed my efforts toward discerning the feature importance as assigned by the model. Understanding feature importance is pivotal as it sheds

light on which variables most significantly influence the model's predictions. This insight not only informs feature engineering efforts but also aids in interpreting the model's decision-making process.

The coefficients derived from the model provide the following insights:

1. **NoScenario:** The coefficients are [0.49971586, 2.7176363, 0.5609786, -0.18426916, -0.07564402]. This suggests that longVel has the highest positive impact on predicting this scenario, followed by time and longAcc. egoHeading and yawRate have a negative impact.
2. **Deceleration**: The coefficients are [0.10276207, 0.7788238, -1.58772, 0.3786329, 0.17540923]. Here, longVel has the highest positive impact, and longAcc has a strong negative impact.
3. **EgoTurning**: The coefficients are [-0.16166583, 0.9943247, 0.92568564, 0.13201937, -0.31550178]. LongVel and longAcc have a positive impact, while time and yawRate have a negative impact.
4. **Stationary**: The coefficients are [0.31691685, -3.9088707, 0.655433, 0.3304591, 0.11929816]. Time and longAcc have a positive impact, but longVel has a strong negative impact.

Upon reflection, it became apparent that the time feature, despite its use in preprocessing, should have been excluded from the model as it does not contribute meaningfully to the model's understanding. The removal of the time column, represented in Unix time format, resulted in an accuracy decrease to **72.15%**. This decrease is intriguing and could be attributed to the model's reliance on the temporal aspect as a distinguishing feature, despite its apparent lack of direct relevance to the scenarios.

The classification report post-removal is as follows:

Class	Precision	Recall	F1-Score	Support
0	0.97	0.64	0.77	1159
1	0.18	1.00	0.31	13
2	0.16	0.66	0.26	73
3	0.76	1.00	0.86	335

To address class imbalance, initially oversampling was employed using SMOTE (Synthetic Minority Over-sampling Technique), which synthetically generates new examples in the minority class, thereby balancing the dataset. However, to explore alternative methods, experiments were also conducted with undersampling using RandomUnderSampler, which reduces the size of the majority class by random selection. This method led to an accuracy of **60.89%**, potentially due to the loss of valuable information as majority class instances are discarded, which can be detrimental to the model's learning capacity.

Results after Undersampling:

	precision	recall	f1-score	support
0	0.97	0.65	0.78	1159
1	0.03	1.00	0.07	13
2	0.49	0.52	0.50	73
3	0.46	0.47	0.46	335
accuracy			0.61	1580
macro avg	0.49	0.66	0.45	1580
weighted avg	0.83	0.61	0.69	1580

An observation of note is that recall of 1 indicates that the model has perfectly identified all actual positives for a given class, which is an ideal scenario but more often than not it suggests overfitting to the training data, especially in the context of imbalanced datasets.

Throughout this project, I have utilized PyTorch for implementing the logistic regression model. PyTorch is a robust choice due to its flexibility, ease of use, and compatibility with complex models, which aligns with my intention to implement more advanced models in the future. To validate my approach, I also replicated the model in scikit-learn and achieved similar results, bolstering my confidence in the correctness of my implementation.

Results from sklearn:

	precision	recall	f1-score	support
0	0.97	0.62	0.76	1159
1	0.43	1.00	0.60	13
2	0.12	0.71	0.21	73
3	0.86	1.00	0.92	335
accuracy			0.71	1580
macro avg	0.60	0.83	0.62	1580
weighted avg	0.90	0.71	0.77	1580

Transitioning from Jupyter Notebook to a Python script was a strategic move to enhance reproducibility and scalability. Running code in parallel and avoiding the pitfalls of saved variables in Jupyter cells are compelling reasons for this switch, especially for the final model evaluation.

In a focused analysis, I provided only acceleration as input for the deceleration scenario and achieved an accuracy of **100%** with the logistic regression model. This result underscores the straightforward relationship within the data, likely due to its rule-based origins.

Training for Deceleration:

```
Number of NaN values: 0
{'NoScenario': 0, 'deceleration': 1}
5854
(array([0, 1]), array([5762,   92]))
(array([0, 1]), array([92, 92]))
1
2
Epoch 10/100, Loss: 0.4113001227378845, Val_accuracy = 100.00%
Epoch 20/100, Loss: 0.24941472709178925, Val_accuracy = 100.00%
Early stopping at epoch 24. Best validation accuracy: 100.00%

And the result is   100.00%
```

Training for EgoTurning:

```
Number of NaN values: 0
{'NoScenario': 0, 'egoTurning': 1}
6122
(array([0, 1]), array([5762,  360]))
(array([0, 1]), array([360, 360]))
1
2
Epoch 10/100, Loss: 0.8823246955871582, Val_accuracy = 66.67%
Epoch 20/100, Loss: 0.6887373924255371, Val_accuracy = 68.75%
Epoch 30/100, Loss: 0.91523677110672, Val_accuracy = 73.61%
Epoch 40/100, Loss: 0.6646091938018799, Val_accuracy = 77.78%
Epoch 50/100, Loss: 0.6305821537971497, Val_accuracy = 78.47%
Epoch 60/100, Loss: 0.7597455382347107, Val_accuracy = 79.86%
Epoch 70/100, Loss: 0.5839229226112366, Val_accuracy = 83.33%
Epoch 80/100, Loss: 0.6026303768157959, Val_accuracy = 84.72%
Epoch 90/100, Loss: 0.6314123868942261, Val_accuracy = 85.42%
Epoch 100/100, Loss: 0.6790584325790405, Val_accuracy = 85.42%
Early stopping at epoch 107. Best validation accuracy: 85.42%

And the result is   84.03%
              precision    recall  f1-score   support

           0       0.81      0.94      0.87        82
           1       0.90      0.71      0.79        62

    accuracy                           0.84       144
   macro avg       0.85      0.82      0.83       144
weighted avg       0.85      0.84      0.84       144
```

Similarly, for ego turning, by isolating yaw rate as the sole parameter, the model attained an accuracy of **84.03%**. The classification report for this model is:

Class	Precision	Recall	F1-Score	Support
0	0.81	0.94	0.87	82
1	0.90	0.71	0.79	62

8. **Reassessing Model Training on Rule-Based Systems**

The outcomes of the logistic regression model have reinforced an intuition that training on a dataset derived solely from a rule-based system inherently limits the model to, at best, replicating the rule-based tool's performance. This was exemplified by the model's **100% accuracy** in identifying deceleration scenarios, which adhered strictly to the predefined rule: a scenario is classified as deceleration if the acceleration value falls below a certain negative threshold. Similarly, the ego turning scenario was determined by the yaw rate exceeding a specific range, coupled with a duration criterion. While the model adeptly recognized these relationships, this approach does not align with our objective to overcome the limitations of the rule-based tool.

To enhance the dataset's quality and the model's predictive power, I embarked on manually labeling at least 50 files. The labeling process presented two options: analyzing video data to identify turns and correlating them with Unix Timestamps, or plotting latitude and longitude data to ascertain vehicle turns. I opted for the latter, considering its efficiency and ease of labelling in addition to availability.

Focusing on Ego Turning Scenarios

My attention was particularly drawn to ego turning scenarios due to several shortcomings observed in the heuristic tool, which I aimed to address:

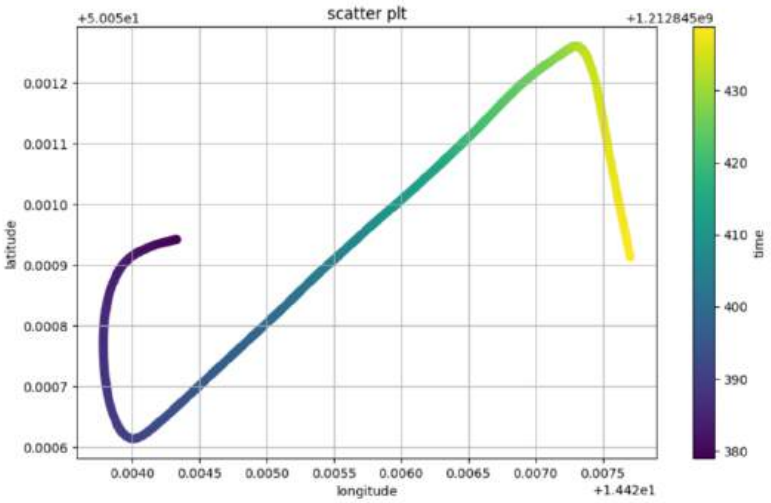
- **Curvy Roads**: The existing tool at times misclassifies curvy roads as turning scenarios, leading to erroneous predictions and behavioral modeling. A machine learning model, in contrast, could leverage data to better adapt to these complex road geometries.
- **Roundabouts and Straight Roads**: The tool's misclassification of roundabouts—where the vehicle navigates around to continue straight—as turning scenarios poses significant challenges. Roundabouts entail distinct rules and behaviors that differ from conventional turns. A machine learning model could more accurately distinguish between roundabouts and genuine turning scenarios.
- **Breaking Up Turns into Parts**: The tool's propensity to fragment a single turning scenario into multiple segments is both confusing and inefficient. Ideally, a continuous turn should be recognized as a singular event. A machine learning model could be trained to identify and process complete turns with greater efficacy.

These improvements are crucial for developing a model that not only matches but also transcends the rule-based tool, providing more accurate and holistic interpretations of driving scenarios.

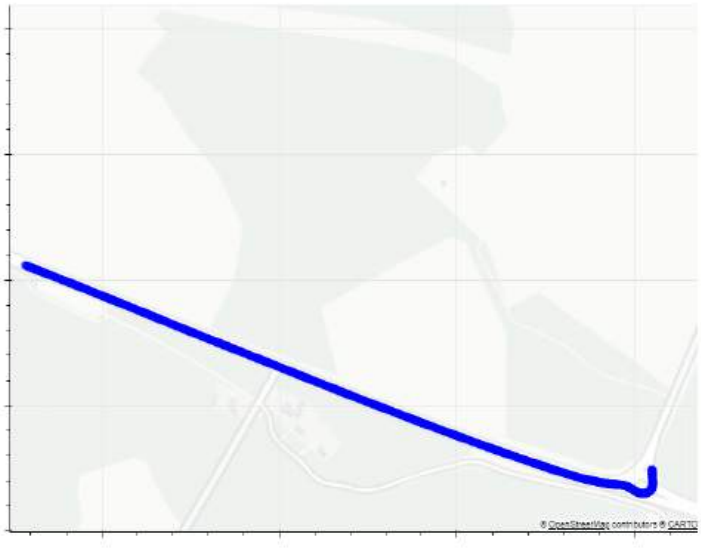
9. Enhancing Data Visualization and Labeling Efficiency

The journey of refining our data labeling process began with the extraction of latitude and longitude information from our dataset. Initially, I utilized **Matplotlib** to visualize the trajectory of the vehicle, which provided a basic understanding of the route taken. However, during my research for a more effective visualization tool, I discovered the **Bokeh** library, a powerful tool that not only plots the geographical data but also overlays it onto an interactive map. This advanced visualization significantly improved my comprehension of the vehicle's trajectory.

Initial Visualization with Matplotlib:

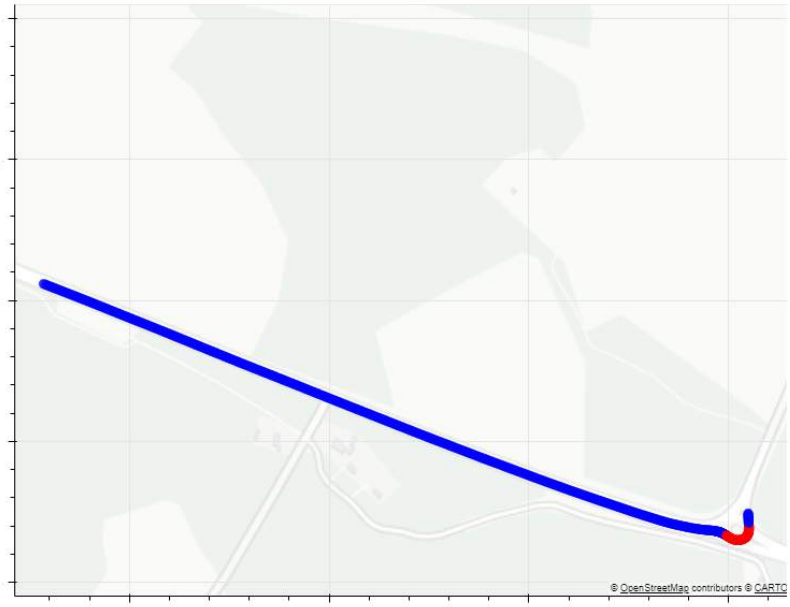


Updated Visualization with Bokeh using CartoDB as vendor:



My initial plots were simple representations of the vehicle's path. To add depth to the analysis, I incorporated a feature to highlight the points where the previous tool identified turns, using a distinct color to differentiate them on the map. Once I achieved this for a single file, I expanded the code to automate the process for an entire directory.

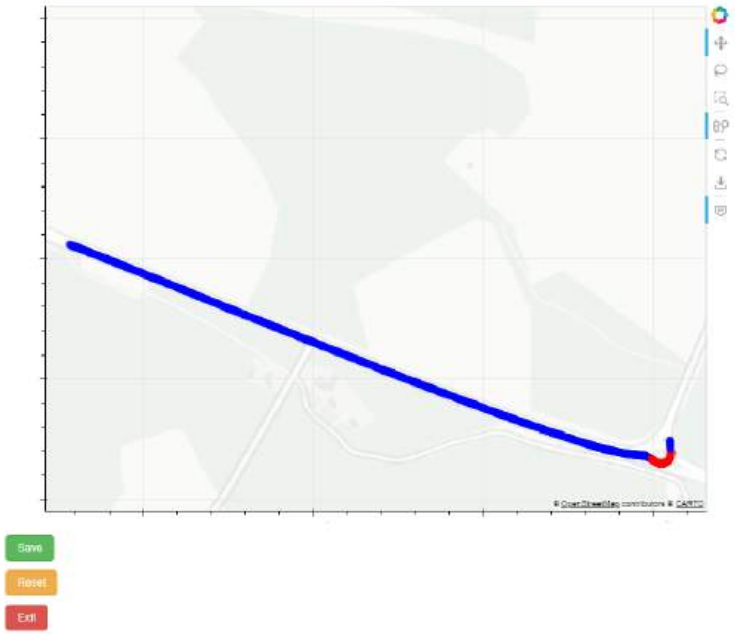
Final Plots with tool mapping included:



While the initial Bokeh visualization served its purpose, I sought to streamline the labeling process further. I began by integrating the **Tap Tool** functionality into my Bokeh application, but it soon became clear that selecting individual points was insufficient for our needs. After exploring various interactive tools available within Bokeh, I settled on the **Lasso Select Tool**. This tool allowed me to select multiple points of interest on the map, enhancing the precision of our labeling efforts.

To complement the Lasso Select Tool, I implemented a **Save Button** to record the selected points directly into the corresponding .csv file under the 'GT' (Ground Truth) column. Additionally, a **Reset Button** was introduced to discard selections if necessary. Upon verifying the tool's functionality with a single file, I further refined the code to handle batch processing for directories. The final touch was the integration of an **Exit Button** that prompts the user in the terminal to proceed to the next file. With a simple 'n' response, the Bokeh visualization for the subsequent file is automatically generated and displayed. This bespoke tool has dramatically accelerated our labeling process, reducing what would have typically taken a week to a mere couple of hours.

Created Tool:



Map Vendors for Enhanced Data Visualization

In the process of enhancing our data visualization capabilities, I evaluated several map vendors provided by Bokeh, each with its unique features and benefits. Here's a brief overview of the three primary options considered:

- **OpenStreetMap (OSM)**: Known for its comprehensive and open-source mapping data, OSM offers an invaluable layer of traffic direction indicators, such as arrows, which significantly aids in interpreting the flow of traffic within our labeled data. Its extensive geographical coverage ensures that we have a broad representation of diverse regions, making it a versatile choice for global datasets. Additionally, OSM provides detailed information on buildings and surrounding infrastructure, offering a rich context for our data analysis and interpretation.
- **CartoDB**: While CartoDB is a robust mapping service known for its user-friendly design and visualization tools, it was observed that OSM offers a more extensive range of geographical coverage. This broader coverage by OSM is crucial for our project as it encompasses a wider array of regions and scenarios.
- **Stamen**: Stamen's maps are aesthetically pleasing and offer a variety of creative cartographic designs. However, during our testing phase, we encountered issues with map loading reliability.

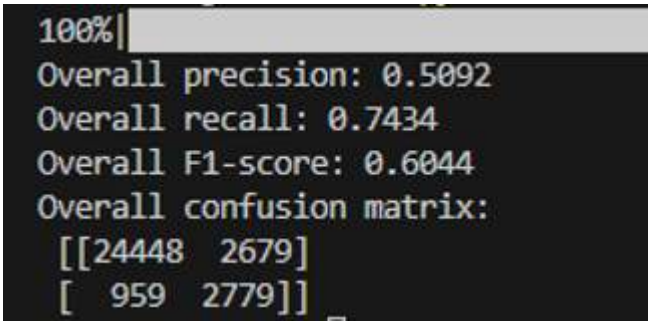
In comparison, OSM proved to be more dependable, with consistent performance and no significant loading problems.

Based on these assessments, OSM emerged as the preferred choice for our project, aligning with our needs for reliable, detailed, and comprehensive map visualizations. However, please note that the map plots shown with this report are with CARTODB to comply with guidelines.

10. **Evaluating Tool Accuracy with Ground Truth Data**

The subsequent phase involved assessing the accuracy of the heuristic tool against the meticulously compiled ground truth data. Upon executing the tool with this data, the outcomes were as follows:

Results:



Significance of the matrix:

	A	B	C
		Predicted Negative (0)	Predicted Positive (1)
	Actual Negative (0)	24448 (True Negative)	2679 (False Positive)
	Actual Positive (1)	959 (False Negative)	2779 (True Positive)

Logistic Regression Model Training on GT

Post-evaluation, I proceeded to train a logistic regression model leveraging the new ground truth data. The model incorporated features such as longitudinal velocity (longVel), longitudinal acceleration (longAcc), ego vehicle heading (egoHeading), and yaw rate. The preliminary results, derived from oversampling, are delineated below. Subsequently, the results obtained from

undersampling are also presented. Additionally, the significance of each feature was determined, the details of which are outlined below.

Model training with oversampling:

```
Epoch 10/100, Loss: 0.600227952003479, Val_Accuracy = 65.52%
Epoch 20/100, Loss: 0.5960045456886292, Val_Accuracy = 65.57%
Epoch 30/100, Loss: 0.6883357167243958, Val_Accuracy = 65.58%
Early stopping at epoch 30. Best validation accuracy: 66.04%
```

Final Accuracy on test set with oversampling:

```
And the result is     62.75%
```

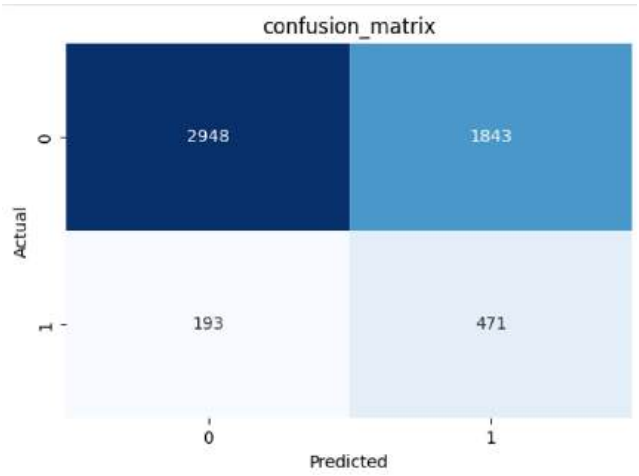
```
print(classification_report(y1_test, y_pred))
✓ 0.0s

              precision    recall  f1-score   support

           0       0.93      0.62      0.75      4763
           1       0.20      0.66      0.31       692

    accuracy                           0.63      5455
   macro avg       0.56      0.64      0.53      5455
weighted avg       0.83      0.63      0.69      5455
```

Training using Under sampling:

```
Epoch 10/100, Loss: 0.791435182094574, Val_Accuracy = 67.07%
Early stopping at epoch 15. Best validation accuracy: 67.29%
```

Results of Under sampling on test set:

```
And the result is     62.07%
```



Significance of each feature:

```
Class 0
longVel : 0.5171114206314087
longAcc : 0.13668979704380035
egoHeading : -0.07170239090919495
yawRate : 0.3530904948711395
Class 1
longVel : -0.2443939596414566
longAcc : 0.37920984625816345
egoHeading : -0.2888610363006592
yawRate : 0.3998565375804901
```

Feature Combination Analysis

To further our understanding, I explored the impact of various feature combinations on the model's performance. The outcomes of employing combinations like yaw rate and longitudinal acceleration, yaw rate and longitudinal velocity, as well as longitudinal velocity and ego vehicle heading, are provided below. Moreover, the results derived from using solely the yaw rate are also included. This analytical approach is instrumental in discerning the contribution of each feature to the model's predictive capabilities, thereby enabling the optimization of feature selection for enhanced model performance.

Results with yawRate and longAcc:

```
[[0.07937197 0.5727222 ]
 [0.2955824  0.6637378 ]]
Class 0
longAcc : 0.07937196642160416
yawRate : 0.5727221965789795
Class 1
longAcc : 0.2955824136734009
yawRate : 0.6637377738952637
```

```
And the result is     60.33%
```

Results with yawRate and longVel:

```
[[ 0.5385646   0.58757526]
 [-0.16360822  0.64888525]]
Class 0
longVel : 0.5385646224021912
yawRate : 0.5875752568244934
Class 1
longVel : -0.16360822319984436
yawRate : 0.6488852500915527
```

```
And the result is     58.96%
```

Results with longVel and EgoHeading:

```
[[ 0.5654615   0.71699965]
 [-0.19050758  0.51946044]]
Class 0
longVel : 0.5654615163803101
egoHeading : 0.7169996500015259
Class 1
longVel : -0.19050757586956024
egoHeading : 0.5194604396820068
```

```
And the result is     60.82%
```

Results with only yawRate:

```
And the result is     41.54%
```

Comparison with the Heuristic Model

Finally, I re-evaluated the pre-existing model, which was initially trained using the tool, against the new ground truth data. The performance was markedly subpar, underscoring the necessity of our current endeavor—manually labeling new data as ground truth to refine the model's accuracy.

```
print(classification_report(y3, y_pred))
✓ 0.0s
```

	precision	recall	f1-score	support
0	0.88	0.99	0.93	23928
1	0.09	0.01	0.01	3346
accuracy			0.87	27274
macro avg	0.49	0.50	0.47	27274
weighted avg	0.78	0.87	0.82	27274

11. **Transition to LSTM Model for Contextual Understanding**

After a comprehensive analysis of the dataset and scrutinizing the outcomes from the logistic regression model, it became increasingly clear that a model with the capability to understand sequences of data would be significantly more beneficial. The logistic regression model was limited to evaluating individual timestamps, which is a substantial constraint when attempting to identify patterns that unfold over time. To address this limitation, we decided to shift our focus to an LSTM (Long Short-Term Memory) model.

LSTMs are particularly well-suited for this task as they are designed to remember information for long periods, which is key in understanding the context within sequences. This characteristic makes them ideal for tasks like extracting ego vehicle turning events from sensor data, where the temporal sequence and the relationship between consecutive data points are crucial for accurate predictions.

The results from the logistic regression model were instrumental in this decision. They highlighted the model's shortcomings in capturing temporal dependencies, thereby reinforcing the need for a model that can process data in sequences to provide context and improve prediction accuracy.

DataLoader and LSTM Implementation

Before we could implement the LSTM model, it was necessary to modify our PyTorch dataset class. This modification was aimed at enabling the DataLoader to supply windows of data rather than isolated data points. After implementing these changes and training the LSTM model, we observed the following results:

Results of the initial LSTM model:

	precision	recall	f1-score	support
0	1.00	0.79	0.88	5396
1	0.04	0.91	0.08	53
accuracy			0.79	5449
macro avg	0.52	0.85	0.48	5449
weighted avg	0.99	0.79	0.87	5449

These results indicate that while the model is exceptionally precise in predicting class 0, it shows a tendency to overpredict class 1. This could imply that the model is very conservative in predicting class 1, potentially due to class imbalance or other issues.

Revisiting the DataLoader and Data Windowing

Upon further examination of the DataLoader, several errors came to light. Notably, when the batch size was larger than the dataset, the DataLoader was inadvertently repeating samples. This was traced back to an incorrect definition of the DataLoader's length function, which was subsequently corrected to ensure that the length is calculated based on the window size and the total data available.

Additionally, the initial approach to data windowing was flawed. The DataLoader was not providing the sliding window effect that is essential for LSTM models. We corrected this by adjusting the windowing technique to ensure continuity in the data sequence:

```
# Incorrect windowing:
      [1, 2, 3]
      [4, 5, 6]
      [7, 8, 9]
```

```
# Corrected windowing:
      [1, 2, 3]
      [2, 3, 4]
      [3, 4, 5]
```

This change ensures that each new window starts with the last data point of the previous window, maintaining the sequence's integrity.

Data Balancing and Sequential Integrity

Another critical observation was related to data balancing. The traditional methods used for balancing, such as RandomUnderSampler or RandomOverSampler, were unsuitable for our windowed data. These methods could disrupt the sequence of information, which is vital for the LSTM model. To resolve this, we introduced a new strategy that balances the data at the window level, ensuring that entire windows are considered for removal or duplication, thus preserving the sequence's integrity.

Furthermore, we discovered that the default behavior of sklearn's train_test_split function, which shuffles the data, was detrimental to the sequential nature required for LSTM training. To maintain the sequential integrity, we opted to manually divide the dataset into training, validation, and test sets. We also implemented assertion tests to verify the accuracy of our data balancing process. The results of the updated model are given below:

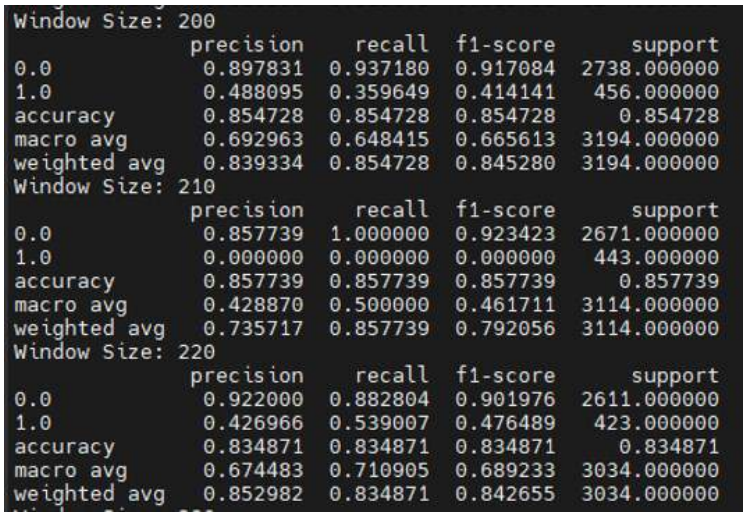
	precision	recall	f1-score	support
0.0	0.95	0.90	0.92	4105
1.0	0.51	0.66	0.58	633
accuracy			0.87	4738
macro avg	0.73	0.78	0.75	4738
weighted avg	0.89	0.87	0.88	4738

12. **Window Size Variation and Model Optimization**

The results mentioned earlier were obtained with a window size of 7. To further refine the model's performance, we embarked on a series of experiments with varying window sizes. We initiated a loop that tested the model with window sizes starting from 7, increasing in increments of 10, up to

a maximum of 300. This allows us to evaluate the model's performance across a range of temporal contexts and identify the optimal window size for our specific task.

A short snippet of the results from running over various window sizes:

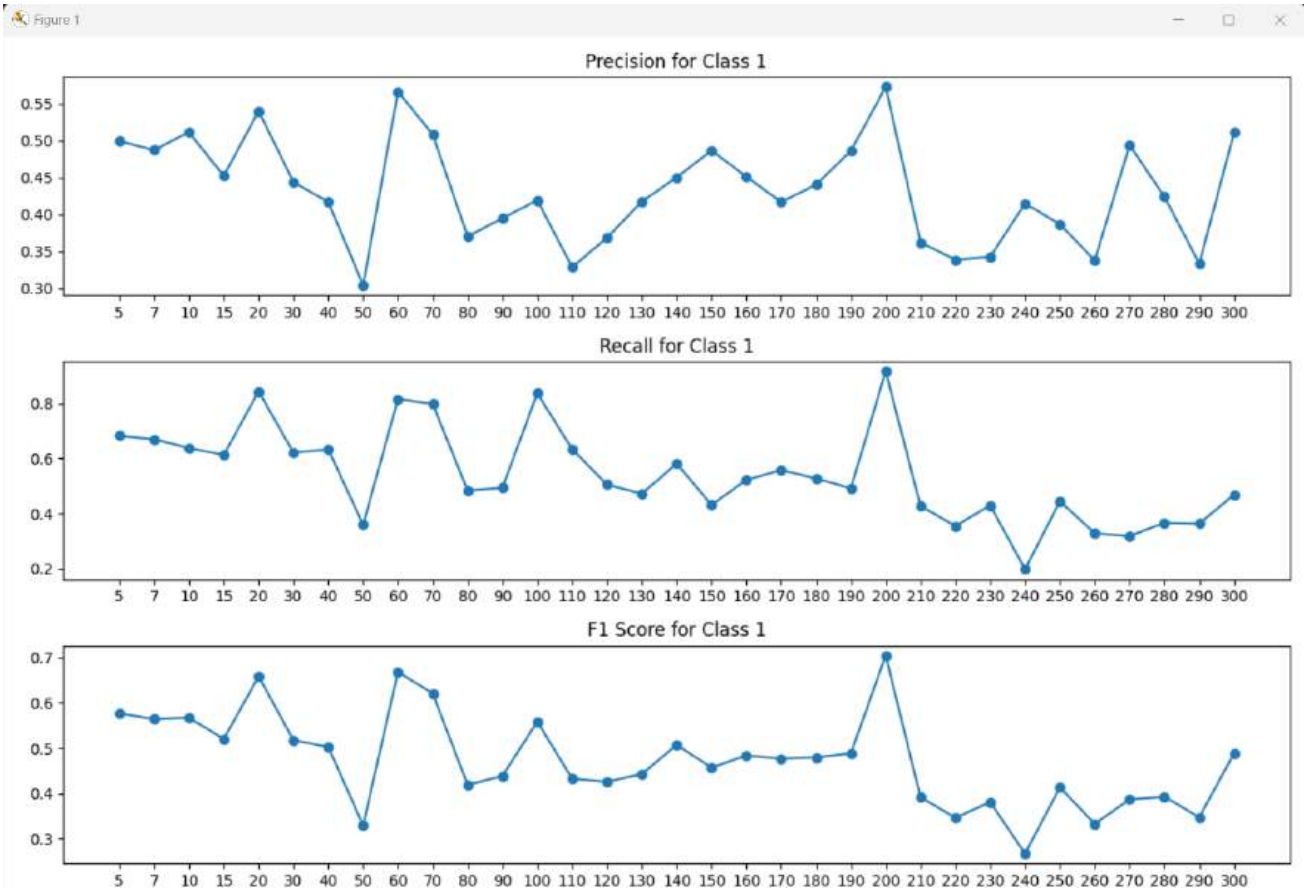


```
Window Size: 200
              precision    recall  f1-score      support
0.0            0.897831  0.937180  0.917084  2738.000000
1.0            0.488095  0.359649  0.414141   456.000000
accuracy       0.854728  0.854728  0.854728     0.854728
macro avg      0.692963  0.648415  0.665613  3194.000000
weighted avg   0.839334  0.854728  0.845280  3194.000000
Window Size: 210
              precision    recall  f1-score      support
0.0            0.857739  1.000000  0.923423  2671.000000
1.0            0.000000  0.000000  0.000000   443.000000
accuracy       0.857739  0.857739  0.857739     0.857739
macro avg      0.428870  0.500000  0.461711  3114.000000
weighted avg   0.735717  0.857739  0.792056  3114.000000
Window Size: 220
              precision    recall  f1-score      support
0.0            0.922000  0.882804  0.901976  2611.000000
1.0            0.426966  0.539007  0.476489   423.000000
accuracy       0.834871  0.834871  0.834871     0.834871
macro avg      0.674483  0.710905  0.689233  3034.000000
weighted avg   0.852982  0.834871  0.842655  3034.000000
```

Window Size Performance Analysis

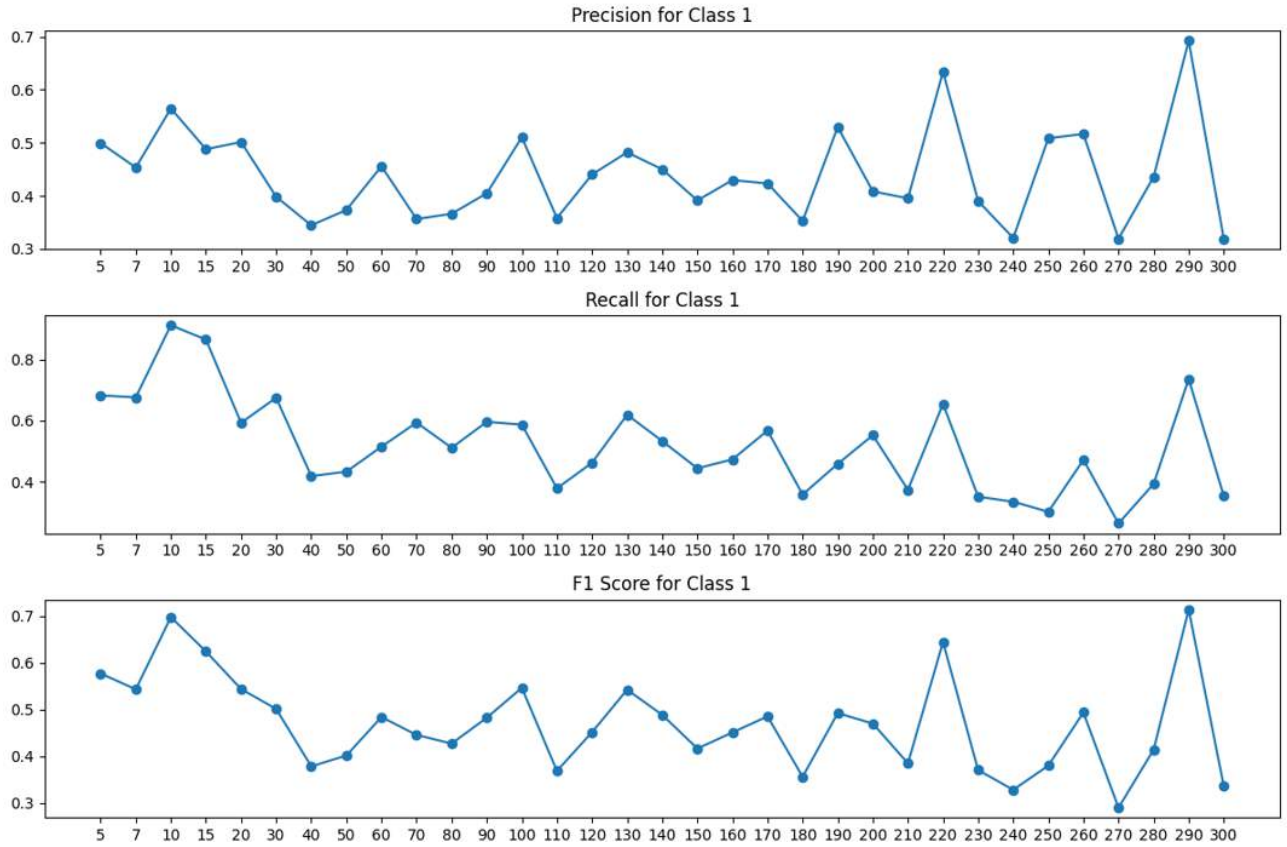
After plotting the performance metrics against various window sizes, the graph exhibited an unexpected pattern. Contrary to the anticipated gradual increase or decrease in performance, the model's accuracy fluctuated erratically. One might expect a decline in performance with larger window sizes due to the model's potential difficulty in retaining earlier information—a phenomenon known as the vanishing gradient problem. Alternatively, performance could improve as the window size increases, providing the model with more context for better predictions.

Plot displaying various window sizes:



However, the observed fluctuations suggest that neither of these scenarios consistently occurred. An initial hypothesis attributed this irregularity to the placement of torch.manual_seed outside the iterative loop, which could have led to the carryover of state from one iteration to the next, thereby affecting the model's initialization and subsequent performance. Even after rectifying the seed initialization to occur within the loop, the performance did not stabilize, indicating that other factors might be influencing the results. Ongoing investigations aim to uncover the underlying causes of these fluctuations.

Updated plot after adding seed initialization in loop:



Incorporating Geospatial Features

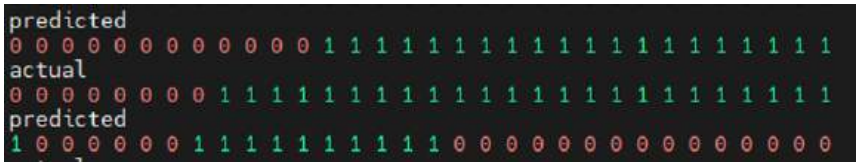
In an effort to enhance the model's predictive power, latitude and longitude features were introduced. The rationale behind this was the assumption that geospatial context could provide additional cues for the model to discern patterns related to vehicle turning events. However, the inclusion of these features did not significantly impact the model's performance. This outcome suggests that the model may already be capturing sufficient spatial information from the existing features, or that the added geospatial data does not correlate strongly with the target variable.

	precision	recall	f1-score	support
0.0	0.94	0.90	0.92	4075
1.0	0.50	0.63	0.55	631
accuracy			0.86	4706
macro avg	0.72	0.76	0.74	4706
weighted avg	0.88	0.86	0.87	4706

Visualization of Model Predictions

To gain a more intuitive understanding of the model's behavior, we attempted to visualize its predictions on a map. While numerical metrics like precision and recall offer quantitative insights, a visual representation can sometimes reveal patterns and anomalies that numbers alone cannot. During the process of converting predictions into CSV files for visualization, we encountered several issues. To troubleshoot these, we printed the predictions and employed colored tensors to better interpret the outputs. This approach eventually led to the successful integration of model predictions into the CSV files.

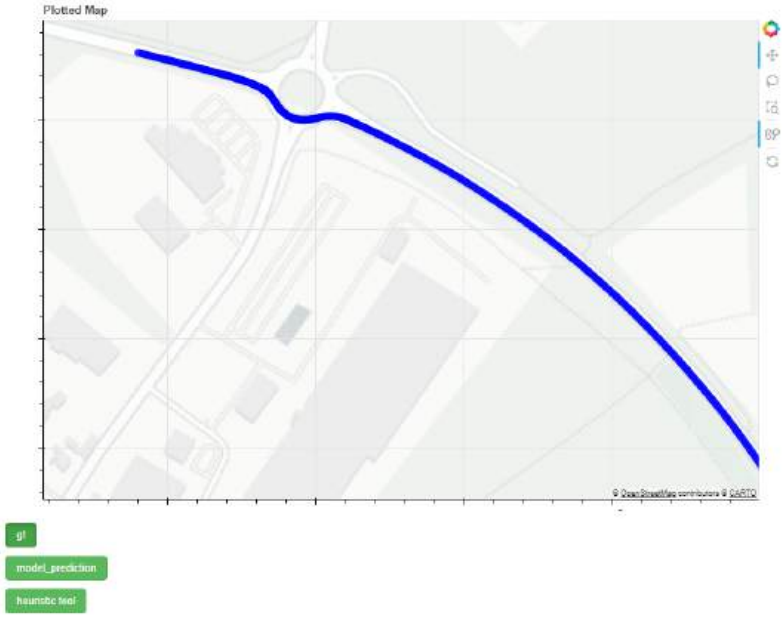
Printing of colored tensors:



We utilized Bokeh for visualization, making modifications to the code to enhance interactivity. Three new buttons were added:

1. **GT**: Displays the ground truth points on the map.
2. **Model Prediction**: Shows the model's predictions.
3. **Tool**: Reveals the predictions made by the previous tool.

Picture of visualization with GT button selected:



Picture of visualization with Model Prediction button selected:

Picture of visualization with heuristic tool button selected:

These enhancements allowed us to observe that our model, like the previous tool, incorrectly predicted that a vehicle traveling straight from a roundabout was turning. To improve upon this, we decided to gather more examples of such scenarios and feed them to our model, aiming to refine its ability to accurately classify these instances.

13. **Advancing LSTM Architecture for Enhanced Temporal Analysis**

In our pursuit of a more sophisticated model, we ventured to adapt the LSTM architecture, drawing inspiration from methodologies prevalent in natural language processing (NLP). This adaptation involves not only considering the final output from the sequence but also incorporating the intermediate outputs. The rationale behind this innovative approach is threefold:

1. **Temporal Dynamics**: The quartet of features—yaw rate, ego heading, velocity, and acceleration—constitutes a time-series dataset. By evaluating the output at each temporal juncture, we capture the dynamic nature of the data, akin to the sequential interplay of words within a sentence in NLP tasks.

2. **Augmented Information**: The inclusion of intermediate outputs effectively amplifies the informational breadth available to the model. This paradigm shift from a singular focus on the final output to a holistic view of the entire data window could potentially culminate in a model with enhanced robustness and precision.
3. **Sequential Data Comprehension**: This strategy affords a granular understanding of the sequential evolution of input features and their collective influence on the outcome. It enables us to dissect the progression of yaw rate, ego heading, velocity, and acceleration and their cumulative effect on determining vehicular movement patterns.

Refinement of Window Balancing Technique

Subsequent to this architectural modification, it became imperative to revise the window balancing methodology. The original code's practice of assigning the target based on the final row's value within a window was supplanted by a more nuanced approach. In this revised method, we consider the prevalence of each class within the window, assigning the target based on the majority. This approach is adaptable to varying window sizes, ensuring consistency and accuracy in target assignment across the dataset.

Interpreting the Classification Report's Anomalies

The classification report post-implementation revealed an intriguing anomaly: the number of classes equaled the window size. My current hypothesis for this phenomenon is rooted in the model's output handling. Originally, the model selected the last timestep's output for classification (self.fc(out[:,-1,:])). In contrast, the modified model applies the fully connected layer across all timesteps (self.fc(out)), resulting in a multiplicity of outputs per sequence.

Current Output of the Model:

```
  _warn_prf(average, modifier, msg_start, len(result))
              precision    recall  f1-score   support

           0       0.46      0.58      0.51       623
           1       0.47      0.61      0.53       624
           2       0.43      0.59      0.50       625
           3       0.45      0.60      0.51       626
           4       0.47      0.60      0.53       627
           5       0.48      0.59      0.53       628
           6       0.47      0.59      0.52       629
           7       0.46      0.59      0.51       630
           8       0.46      0.59      0.52       631
           9       0.46      0.58      0.51       631
          10       0.45      0.58      0.50       631

   micro avg       0.46      0.59      0.52      6905
   macro avg       0.46      0.59      0.52      6905
weighted avg       0.46      0.59      0.52      6905
 samples avg       0.09      0.08      0.08      6905
```

Synthesizing Multiple Outputs for Cohesive Prediction

To synthesize these multiple outputs into a singular prediction, we can draw parallels from ensemble model techniques. The following methods are under consideration:

- **Mean**: This method posits that each timestep contributes equally to the final prediction, suitable for sequences exhibiting consistent behavior.
- **Maximum**: This approach prioritizes the most confident prediction across timesteps, ideal for scenarios where extreme values are indicative of the correct prediction.
- **Voting**: Here, each timestep's prediction is considered a vote, with the majority class determining the final prediction. This method is apt for categorical data where the most frequent category is presumed correct.
- **Weighted Sum or Average**: This technique involves assigning varying weights to different timesteps, potentially prioritizing later timesteps if deemed more informative. The determination of weights could be empirical or data-driven.

14. In summary, there are approximately 2 and a half months remaining for the internship to conclude. The current progress of the project involves addressing prediction issues in the NLP-like LSTM model and investigating the peculiar behaviour of different window sizes. Additionally, we plan to explore the use of a transformer model for this scenario and extract lane change-based scenarios.

III. Accomplishments and Achievements:

During my internship, I have achieved several significant milestones that have contributed to both my personal growth and the advancement of the projects I was involved in.

Achievements:

- **Delivered Two Key Presentations:** Successfully conducted two major presentations; the first introduced the project scope and objectives to the team, while the second detailed the current progress, demonstrating effective communication and project management skills.
- **Official Recognition:** Received official acknowledgment for my contributions on Qualcomm's recognition page, which included the award of spendable points, reflecting the value and impact of my work.
- **Creation of Multiple Datasets:** Developed various datasets essential for training machine learning models, showcasing my technical expertise and attention to detail.
- **Development of Labeling Tools:** Engineered tools that significantly increased the efficiency of data labeling processes, thereby enhancing the productivity of the team.
- **Training of Various Models :** Trained various models in order to find which one would best suit the data, and concurrent analysis of it's results.

Challenges and Strategies:

- **Navigating New Technologies:** Faced the challenge of mastering new software and tools necessary for project development. Overcame this by dedicating extra hours to self-study and seeking mentorship from experienced colleagues.
- **Presentation Skills:** Initially, public speaking was a hurdle. I tackled this by engaging in thorough preparation and practice, which led to successful presentations and boosted my confidence.

Evidence of Success:

- **Positive Feedback:** Received commendations from supervisors and peers for my proactive approach and the quality of work delivered.
- **Metrics of Improvement:** The tools I developed reduced the data labeling time by **80%**, directly impacting the project's timeline positively.

Additional Points:

- **Collaborative Efforts:** Actively participated in team meetings and brainstorming sessions, contributing innovative ideas that were incorporated into the project.

- **Continuous Learning:** Demonstrated a commitment to continuous learning by completing online courses relevant to the internship.

IV. Lessons Learned

The journey through this internship was both challenging and enriching, endowing me with insights and experiences that will significantly influence my trajectory as a software developer, applied scientist, and researcher.

Technical Skills Enhancement: The internship was a gateway to a plethora of technical skills and tools such as **Bokeh** and **Folium** for interactive map visualization. Utilizing **Matplotlib**, I could create compelling data visualizations. I also delved into various machine learning models, mastering their training, analysis, and troubleshooting. This arsenal of technologies not only broadened my machine learning acumen but also refined my capacity to weave disparate libraries into a unified project.

Software Development Best Practices: My tenure at the internship bolstered my grasp of software development's best practices. I learned the importance of code readability and the necessity to adhere to coding standards. These methodologies not only streamlined the project but also facilitated teamwork. The clarity and documentation of the code were pivotal during problem-solving and laid the groundwork for subsequent phases of the project.

Collaboration and Effective Communication: The internship afforded me numerous collaborative opportunities, allowing me to seek mentorship and exchange expertise with peers. Effective communication was key to navigating project milestones, grasping requirements, and untangling technical quandaries.

Problem-Solving and Debugging: The development phase often presented technical challenges, which became a canvas for problem-solving and debugging. I honed a systematic approach to issues, leveraged debugging tools proficiently, and harnessed research and online resources for solutions. Surmounting these challenges not only sharpened my technical prowess but also fortified my confidence in addressing intricate issues.

Adaptability and Flexibility: The dynamic nature of the project meant that requirements and priorities could shift. I learned to swiftly adjust to new circumstances, embrace alterations, and maintain a versatile stance. This flexibility was crucial in achieving project goals and staying in step with the organization's broader objectives.

V. Conclusion

During my internship, I gained valuable insights and developed essential skills that will undoubtedly shape my future career. Here are the key takeaways from my experience:

1. Working with Various Software Tools

The internship exposed me to a diverse range of software tools. I learned how to navigate different environments, collaborate with team members, and efficiently use tools specific to our project. This experience broadened my technical toolkit and enhanced my adaptability.

2. Time Series Data Challenges

While I had prior experience with machine learning models, working with time series data was a new adventure. I encountered several unique considerations, such as creating time windows, handling class imbalances, and adjusting validation techniques. These challenges pushed me to think creatively and adapt my existing knowledge to this specialized domain.

3. Software Deployment Practices

Understanding the deployment process within a company was eye-opening. I observed how code is meticulously reviewed, merged, and tested before deployment. Learning about version control, merge conflicts, and peer reviews provided me with practical insights that I can apply in future projects.

4. Collaboration and Documentation

The internship emphasized the importance of collaboration. I actively participated in team discussions, learned how to communicate effectively, and contributed to shared goals. Additionally, documenting my code became second nature, ensuring that knowledge was accessible to others and facilitating smoother collaboration.

5. Debugging Mastery

Debugging became a significant part of my daily routine. I honed my skills using the debugger in Visual Studio Code, set breakpoints, and dissected complex issues. This hands-on experience significantly improved my problem-solving abilities and boosted my confidence in tackling intricate code problems.

6. Adaptability and Flexibility

As the project evolved, so did its requirements. I learned to be flexible, adapt to changing circumstances, and pivot when necessary. This adaptability will serve me well in future endeavors, where agility and responsiveness are critical.