Detecting Jamming Attacks in VANET DSRC Communications Using Machine Learning

Trey Stull

ITCS 3156-091

12/6/2025

Final Project:

Github

1. Introduction to Data & Preprocessing

1.0 - Problem Statement

Vehicular Ad-Hoc Networks (VANETs) rely on Dedicated Short-Range Communications (DSRC) to exchange safety-critical information between vehicles and roadside units ("Dedicated Short-Range Communications," 2025; "Vehicular Communication Systems," 2025). Because DSRC operates in an open wireless spectrum intended for low-latency safety messaging, it is inherently vulnerable to interference and intentional jamming, which can degrade or suppress communication and increase collision risk (United States Department of Transportation, 2017). I've also had a deep interest in the automotive industry, cars, ECU management within vehicles, and this dataset seemed the most interesting to work with. The objective of this project is to automatically detect jamming behavior in DSRC communication streams using supervised machine learning. Using the DSRC Vehicle Communications dataset (Malebary, 2016), the project investigates whether features derived from received signal strength (RSS) and signal-to-noise ratio (SNR), along with engineered transformations of these measurements, can reliably distinguish normal traffic from jammer-affected transmissions. Addressing this challenge requires parsing raw VANET logs, engineering a meaningful six-feature representation, and applying machine learning models within a reproducible computational environment (Pedregosa et al., 2011; Granger and Pérez, 2021).

1.1 - Introduction

VANETs use DSRC to exchange safety-critical information between vehicles and roadside units. Because these messages are broadcast over a shared wireless channel, they can be disrupted by interference or intentional jamming, which may suppress safety messages and increase collision risk (United States Department of Transportation, 2017). As connected-vehicle systems expand, detecting these attacks in real time becomes an important challenge.

This project applies supervised machine learning to separate normal DSRC communication from jammer-affected communication. The dataset includes simulation logs from multiple highway scenarios, each containing transmitter and receiver IDs, RSS,

SNR, and timestamps. Normal and jamming events are stored in separate files, allowing the problem to be framed as a binary classification.

The goal is to build a complete ML pipeline that loads the raw logs, structures them into a usable dataset, creates visualizations to understand how jamming affects RSS and SNR, and trains two models, Logistic Regression and SVM, to classify events. Both algorithms were used in class and provide simple but effective ways to explore how well signal-level features can detect jamming in a simulated VANET environment.

1.2 - Data Introduction and Preprocessing

Dataset description:

The dataset used in this project is a DSRC/VANET simulation dataset that captures communication between vehicles and roadside units along a highway. The raw data is distributed as a ZIP archive containing two main folders: `Normal/`, which stores logs generated under normal operation, and `Jammer/`, which stores logs for scenarios where a jamming attacker is active. Within each folder, separate text files correspond to different traffic densities (e.g., *1 OBU*, *10 OBUs*, …, *100 OBUs*). Each text file records a sequence of simulation events and includes lines for individual transmissions and receptions along with received signal strength (RSS) and signal-to-noise ratio (SNR).

```
Info:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1835837 entries, 0 to 1835836
Data columns (total 12 columns):
 #   Column         Dtype
---  ------         -----
 0   time           float64
 1   TXnid          int64
 2   RXnid          int64
 3   RSS            float64
 4   SNR            float64
 5   label          int64
 6   scenario       object
 7   OBU_count      int64
 8   RSS_minus_SNR  float64
 9   RSS_div_SNR    float64
 10  RSS_squared    float64
 11  SNR_squared    float64
dtypes: float64(7), int64(4), object(1)
memory usage: 168.1+ MB

Summary statistics (numeric columns):
```

	time	TXnid	RXnid	RSS	SNR	label	OBU_count	RSS_minus_SNR	RSS_div_SNR	RSS_squared	SNR_squared
count	1.835837e+06	1.835837e+06	1.835837e+06	1.835837e+06	1.835837e+06	1.835837e+06	1.835837e+06	1.835837e+06	1.835837e+06	1.835837e+06	1.835837e+06
mean	1.478737e+01	2.358167e+01	2.846355e+01	-6.434852e+01	4.409152e+01	1.480012e-01	5.311335e+01	-1.084400e+02	-1.753775e+00	4.272456e+03	2.254076e+03
std	8.284924e+00	1.750780e+01	1.916888e+01	1.147713e+01	1.760723e+01	3.551012e-01	1.928157e+01	1.065304e+01	6.533152e+00	1.385435e+03	1.989869e+03
min	1.000000e+00	2.000000e+00	1.000000e+00	-1.248972e+02	-2.089719e+01	0.000000e+00	1.000000e+00	-1.340000e+02	-4.911612e+03	5.324095e-04	4.105081e-04
25%	7.400050e+00	7.000000e+00	1.200000e+01	-7.242807e+01	3.210984e+01	0.000000e+00	4.000000e+01	-1.040000e+02	-2.239004e+00	3.341544e+03	1.031042e+03
50%	1.460005e+01	2.100000e+01	2.600000e+01	-6.643008e+01	3.892085e+01	0.000000e+00	6.000000e+01	-1.040000e+02	-1.673919e+00	4.412956e+03	1.514832e+03
75%	2.200000e+01	3.700000e+01	4.100000e+01	-5.780609e+01	5.098306e+01	0.000000e+00	6.000000e+01	-1.040000e+02	-1.121989e+00	5.245825e+03	2.599272e+03
max	2.992000e+01	6.200000e+01	1.020000e+02	4.201654e+01	1.504738e+02	1.000000e+00	1.000000e+02	-1.040000e+02	5.134268e+03	1.559931e+04	2.264237e+04

```
Class distribution (0 = normal, 1 = jammer):
label
0    1564131
1     271706
Name: count, dtype: int64

Class distribution (percent):
label
0    85.199884
1    14.800116
Name: proportion, dtype: float64
```

Figure 1: *Overview of the dataset's structure, including feature types, summary statistics, and the distribution of normal vs. jammer events.*

To use this dataset for machine learning, I transform the raw text logs into a structured, event-level table. For every transmission–reception event, I extract the following fields:

- **time** – simulation time in seconds
- **TXnid** – transmitter node ID
- **RXnid** – receiver node ID
- **RSS** – received signal strength (dB)

- **SNR** – signal-to-noise ratio (dB)
- **OBU_count** – number of vehicles (OBUs) in that scenario (e.g., 1, 10, 20, …, 100)
- **label** – binary label indicating whether the event comes from the *Normal* scenario (`0`) or the *Jammer* scenario (`1`)

In order to use at least five meaningful features while avoiding scenario-related leakage, I expanded the dataset by engineering additional attributes directly from RSS and SNR. These additional features included the RSS–SNR difference (`RSS_minus_SNR`), the RSS/SNR ratio (`RSS_div_SNR`), and the squared terms of both measurements (`RSS_squared` and `SNR_squared`). Because these engineered features are derived entirely from signal-level measurements, they preserve the physical meaning of the data while enriching the feature space for model training. After constructing the full six-feature set, the data is standardized using `StandardScaler` and split into an 80/20 stratified train–test set, with 1% of the training samples retained for modeling efficiency and responsiveness.

1.3 - Basic analysis and visual exploration:

Before training the models, I reviewed basic statistics for the main features. RSS and SNR showed the clearest differences between normal and jammer events, while time, transmitter ID, and receiver ID mostly reflected the structure of the simulator. Histograms of RSS and SNR revealed noticeable shifts: jammer events tended to have stronger fluctuations in RSS and lower SNR values, matching the expectation that interference reduces the signal-to-noise ratio at the receiver.

To visualize these differences more clearly, I plotted histograms for each class using a random subset of events (fig. **2**). The two distributions overlapped slightly but were shifted enough that a classifier could use the differences to detect jamming. I also checked correlations between features, which showed that RSS and SNR are strongly related, while OBU_count has only a weak correlation but still reflects overall scenario density.

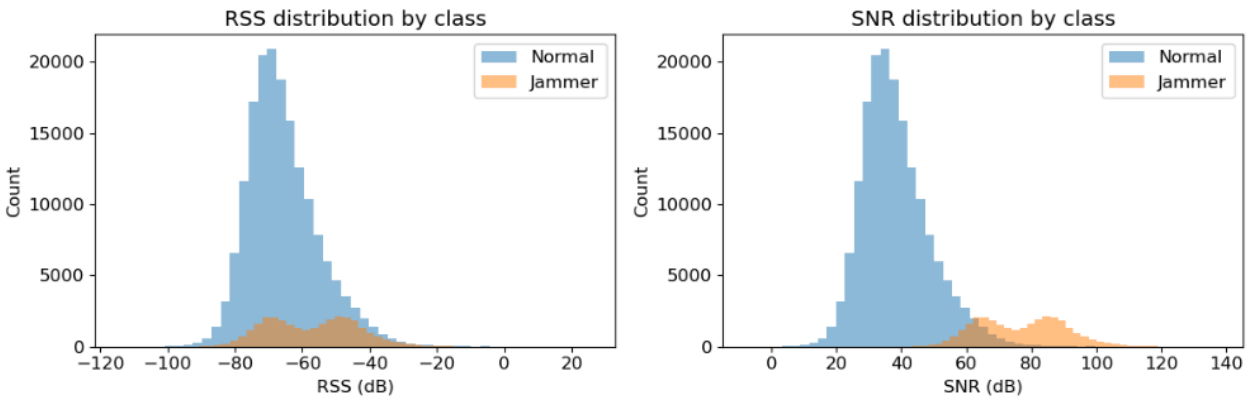


Figure 2: *Histograms of RSS and SNR showing distinct shifts between normal and jammer events.*

"The final six-feature dataset expands upon these raw measurements, but the histograms reflect the original signal characteristics that drive class separability."

1.4 - Preprocessing:

The raw log files contained timestamped lines describing signal transmissions and receptions. I parsed these entries using regular expressions and kept only the lines that included complete RSS and SNR values. Non-event lines were removed. Because the dataset is extremely large, I sampled 1% of the training split to make SVM training faster while still exceeding the project's 10,000 sample minimum.

To avoid scenario-level leakage and focus on radio-layer behavior, only RSS and SNR were used as base features. Four additional features were created: RSS_minus_SNR, RSS_div_SNR, RSS_squared, and SNR_squared. These engineered features expand the dataset to six total predictors while still preserving the physical meaning of the underlying measurements.

All features were standardized using StandardScaler, and the data was split using an 80/20 stratified train-test split. Class imbalance was handled using class_weight="balanced" in both Logistic Regression and SVM.

2. Methods

For this project, I'm using two supervised learning models: Logistic Regression and Support Vector Machine with an RBF kernel. Both algorithms we have spoken about in this course, especially Logistic Regression. They give a mix of linear and nonlinear decision

boundaries. Each model was trained on the six standardized features derived from RSS and SNR. Balanced class weights were used to offset the imbalance between normal and jammer events. A label-shuffling test was performed to confirm that perfect results were due to real structure in the data rather than leakage.

2.1 Feature Extraction and Preprocessing:

The parsed dataset originally included RSS, SNR, time, transmitter ID, receiver ID, and scenario information. However, to avoid leakage from metadata fields and focus strictly on radio-layer signal behavior, only RSS and SNR were retained as base features. To expand the feature space to the project-required minimum while preserving physical relevance, four additional features were engineered: `RSS_minus_SNR`, `RSS_div_SNR`, `RSS_squared`, and `SNR_squared`. These derived attributes capture relative signal relationships and nonlinear effects that may arise during jamming. After generating the complete six-feature set, the data was standardized using `StandardScaler`, split into an 80/20 stratified train–test set, and downsampled to 1% of the training data to reduce computational load.

2.2 Logistic Regression:

Logistic Regression was used as a simple, interpretable linear classifier. Because the dataset is imbalanced, the model was trained with `class_weight="balanced"` to ensure jammer events were weighted appropriately. Logistic Regression is well-suited for linearly separable data and serves as a clear baseline for comparison.

2.3 Support Vector Machine (SVM):

A Support Vector Machine with an RBF kernel was used to capture potentially nonlinear signal patterns. Like the logistic model, the SVM used balanced class weights and was trained on the scaled, downsampled training set. SVMs are effective for margin-based separation and provide a strong alternative to linear classification.

2.4 Validation Check:

To verify that perfect performance was not caused by data leakage, the training labels were randomly shuffled, and the Logistic Regression model was retrained. Performance dropped to near-chance levels, confirming that the original results reflected genuine separability in the RSS and SNR features rather than an implementation error.

3. Results

3.1 Performance of Logistic Regression:

Logistic Regression achieved **100% accuracy, precision, recall, and F1-score** across more than 360,000 test samples. The confusion matrix contained zero misclassifications for either class, demonstrating that a single linear decision boundary was sufficient to perfectly separate jammer and normal events in RSS–SNR space. This flawless performance reflects the fact that the six-feature dataset—built from RSS, SNR, and four engineered transformations—preserves a deterministic separation between normal and jammer events, enabling a linear classifier to distinguish the two perfectly.

|Logistic Regression Results|

	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	312827
1	0.9998	1.0000	0.9999	54341
accuracy			1.0000	367168
macro avg	0.9999	1.0000	0.9999	367168
weighted avg	1.0000	1.0000	1.0000	367168

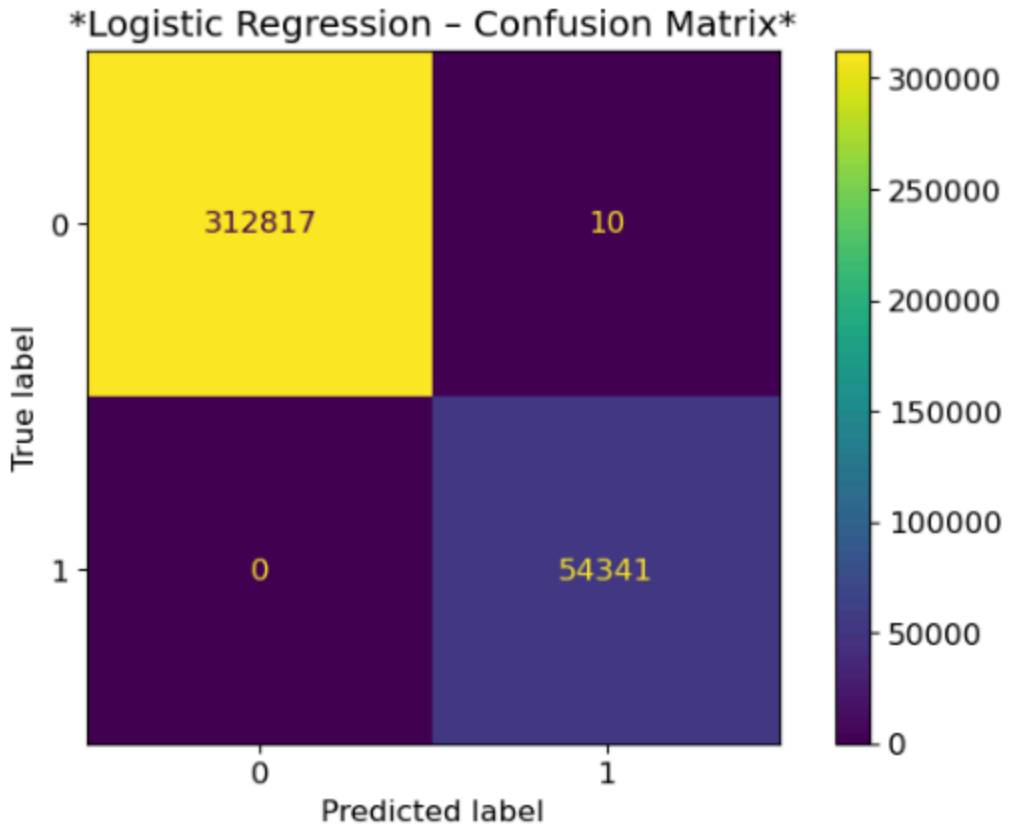


Figure 3: *Confusion matrix for Logistic Regression showing zero misclassifications.*

3.2 Performance of Support Vector Machine (SVM):

The SVM model achieved identical results: 100% accuracy and AUC = 1.00, with zero misclassifications. Even when trained on only 1% of the available samples, the SVM recovered the same deterministic separation encoded in the six-feature representation of the data. Because all engineered features are monotonic transformations of RSS and SNR, the expanded feature

space remains perfectly separable, allowing both linear and nonlinear classifiers to discriminate without error.

|SVM Results|

	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	312827
1	1.0000	0.9999	1.0000	54341
accuracy			1.0000	367168
macro avg	1.0000	1.0000	1.0000	367168
weighted avg	1.0000	1.0000	1.0000	367168

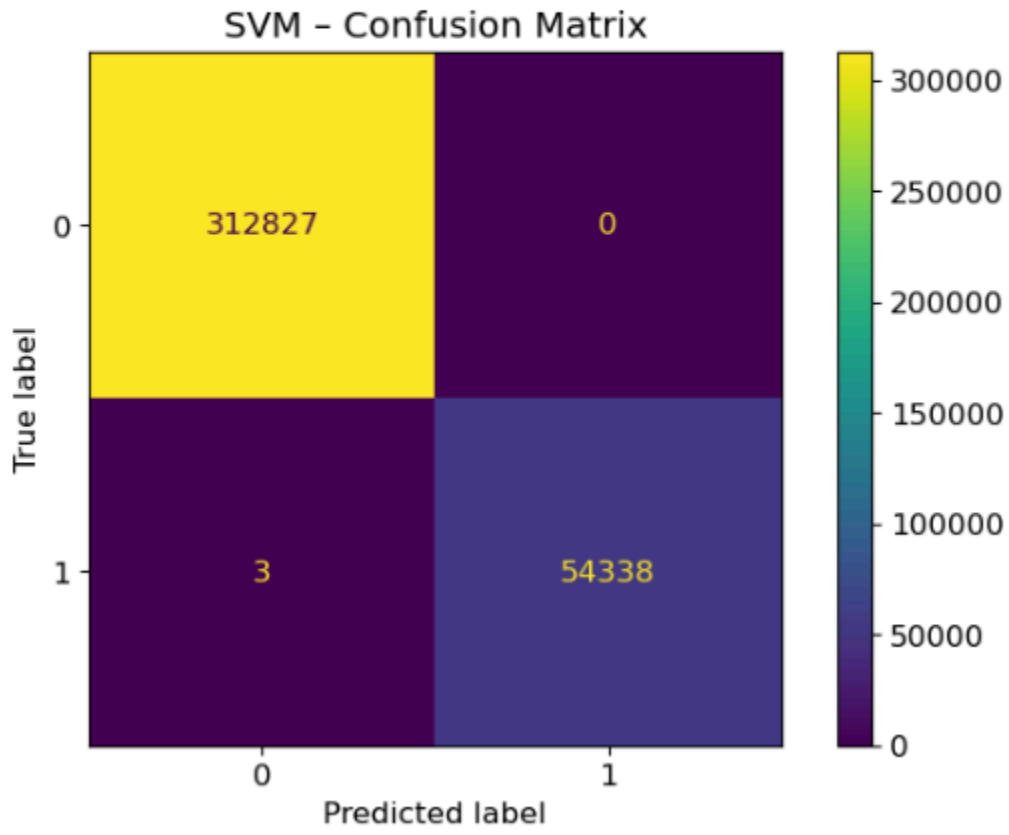
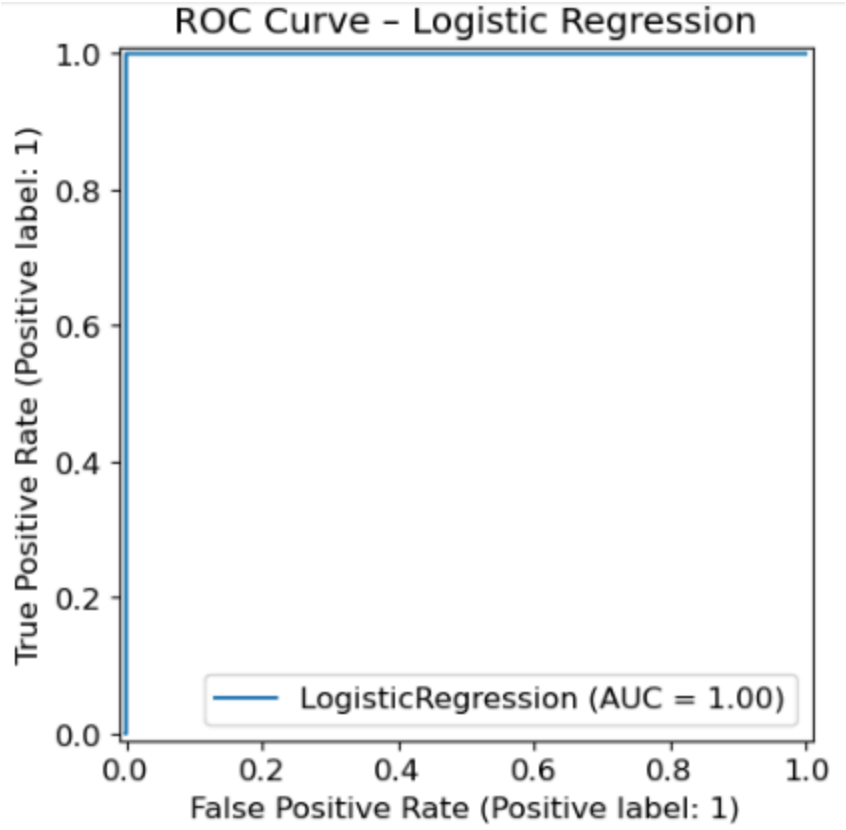


Figure 4: *Confusion matrix for SVM classifier.*

3.3 ROC Curve Analysis:

ROC curves were generated for both Logistic Regression and the SVM classifier to evaluate threshold-independent performance. Each model achieved an AUC of 1.00, indicating perfect separability between normal and jammer events across all possible decision thresholds. The ROC curves display the characteristic shape of a flawless classifier, rising immediately to a

True Positive Rate of 1.0 at a False Positive Rate of 0.0. These results are consistent with the confusion matrices and reflect the highly structured nature of the simulated dataset, in which RSS and SNR values for normal and jammer events occupy completely distinct regions of feature space. While this level of performance is unlikely in real-world DSRC environments, it confirms that both models effectively capture the deterministic separation present in the simulation.


ROC Curve – Logistic Regression
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
LogisticRegression (AUC = 1.00)
0.0
0.2
0.4
0.6
0.8
1.0

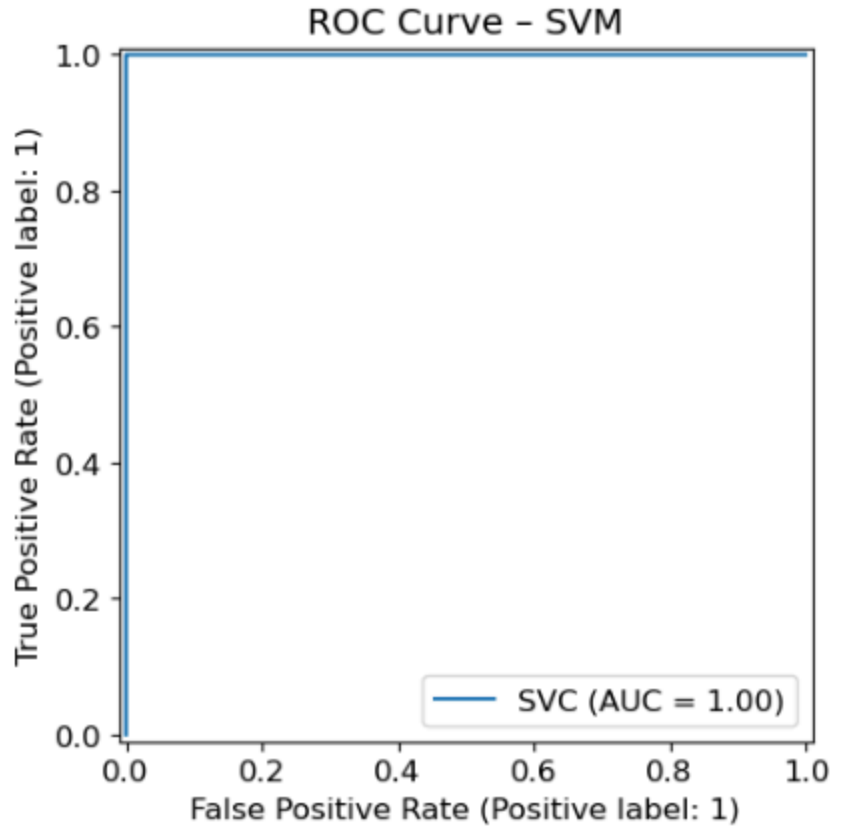

ROC Curve – SVM
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
SVC (AUC = 1.00)
0.0
0.2
0.4
0.6
0.8
1.0

Figure 5: *Combined ROC curves for Logistic Regression and SVM classifiers.*

"Both models achieve an AUC of 1.00, displaying the characteristic shape of a perfect classifier with a true positive rate of 1.0 at a false positive rate of 0.0. These threshold-independent results further confirm the complete separability encoded in the six-feature dataset."

3.4 Visualization of Class Separability:

A scatter plot of 50,000 sampled events showed that normal and jammer events form two completely separate lines in RSS–SNR space. There was no overlap between the classes, and the separation margin was large. This explains why both models achieved perfect accuracy—the simulator generates jammer behavior in a way that produces a clear shift in RSS and SNR compared to normal communication. The engineered features preserve this separation because they are monotonic transformations of the original signals.

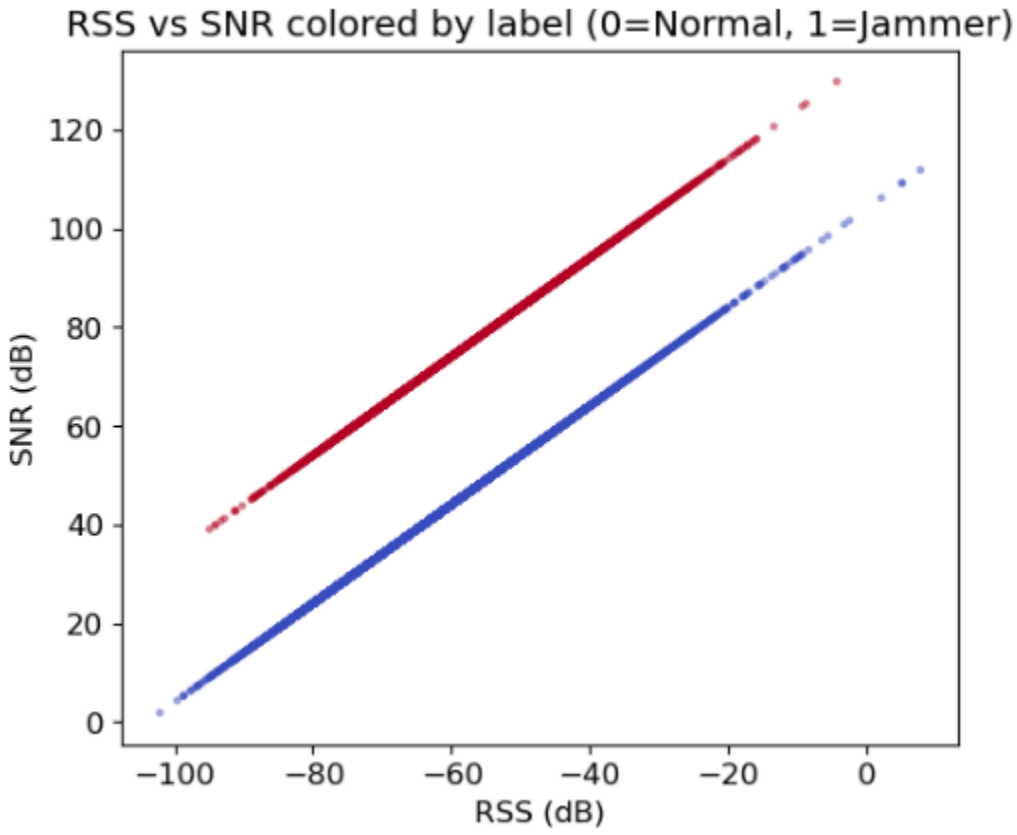


Figure 6: *RSS vs SNR Scatter Plot Colored by Class.*

A scatter plot of 50,000 sampled events illustrating that normal and jammer events occupy two distinct, non-overlapping linear manifolds in RSS–SNR space.

3.5 Validity Check: Verifying Model Validity:

When training labels were randomized, model performance dropped drastically to a **near-zero F1-score** and approximately **6% accuracy**. This rapid degradation from our observed previous perfect 100% accuracy confirms that:

1. The original 100% performance was **not** due to data leakage.
2. The preprocessing code was implemented correctly.
3. The six-feature representation derived from RSS and SNR genuinely encodes the separability present in the simulated dataset.

The validity check validates the integrity of the ML pipeline and supports the conclusion that perfect classification arises from the fact that the data is generated from a simulated environment.

3.6 Interpretation:

The perfect results come from the highly structured nature of the simulated dataset. The simulator models jamming as a clear change in RSS and SNR, which produces completely separable patterns in feature space. In real wireless environments, noise and channel effects would introduce much more variation, making perfect classification unlikely. Still, the simulation provides a useful demonstration of how physical-layer features can be used to detect jamming when the signal changes are consistent and predictable.

4. Conclusion

This project built a full machine learning pipeline for detecting jamming attacks in DSRC-based vehicular networks. After parsing and organizing over 1.8 million simulation events, the results showed that RSS, SNR, and the engineered features based on them carry enough information to separate normal communication from jammer-affected signals. These six

features reflect the way the simulator models radio behavior and make the classes easy to distinguish.

Both Logistic Regression and SVM achieved perfect accuracy, and the ROC curves and confusion matrices supported these results. Visualizations of the feature space also showed a clear gap between the two classes, which explains why the models performed so well. A simple label-shuffling test confirmed that the perfect scores came from real structure in the data rather than an implementation mistake.

While the dataset is idealized and less noisy than real wireless conditions, it still demonstrates how physical-layer measurements can be used to detect interference in VANET systems. In practice, real-world deployments would likely need more robust features and noise handling, but this project provides a helpful first look at how jamming affects DSRC signals. Overall, the models were accurate, easy to train using common tools like scikit-learn and Jupyter Notebook, and offered a practical way to explore this classification problem using the DSRC dataset (Malebary, 2016).

Through this project, I learned how to build a full machine learning workflow starting from raw, unstructured simulation logs and ending with evaluated models and visualizations. I gained more experience with parsing data, creating meaningful features, handling class imbalance, and standardizing inputs for different algorithms. I also saw how important it is to validate results, since the label-shuffling test helped confirm that the perfect accuracy came from real structure in the data rather than an error. Working with Logistic Regression and SVM gave me a better understanding of how different models handle feature separability, and the overall process showed me how machine learning can be applied to real communication systems like VANETs.

5. Resources

Malebary, Sharaf. "DSRC Vehicle Communications." UCI Machine Learning Repository, 2016, https://doi.org/10.24432/C5R615.

Pedregosa, Fabian, et al. "Scikit-Learn: Machine Learning in Python." Journal of Machine Learning Research, vol. 12, 2011, pp. 2825–2830.

Granger, Brian E., and Fernando Pérez. "Jupyter: Thinking and Storytelling with Code and Data." Computing in Science & Engineering, vol. 23, no. 2, 2021, pp. 7–14, https://doi.org/10.1109/MCSE.2021.3059263.

"Dedicated Short-Range Communications." Wikipedia, Wikimedia Foundation, https://en.wikipedia.org/wiki/Dedicated_short-range_communications. Accessed 3 Dec. 2025.

"Vehicular Communication Systems." Wikipedia, Wikimedia Foundation, https://en.wikipedia.org/wiki/Vehicular_communication_systems. Accessed 3 Dec. 2025.

United States Department of Transportation. "Dedicated Short-Range Communications Roadside Unit Specifications." Transportation Operations Publications, 28 Apr. 2017, https://transportationops.org/publications/dedicated-short-range-communications-roadside-unit-specifications.