Middle East Technical University

Informatics Institute

JAMMER DETECTION IN AUTONOMOUS VEHICLES WITH MACHINE LEARNING

Advisor Name: Prof. Dr. Tuğba TAŞKAYA TEMİZEL

(METU)

Student Name: Erdoğan Mert DEMİRYÜREK

(IS)

January 2025

Orta Doğu Teknik Üniversitesi

Enformatik Enstitüsü

MAKİNE ÖĞRENİMİ İLE OTONOM ARAÇLARDA JAMMER TESPİTİ

Danışman Adı: Prof. Dr. Tuğba TAŞKAYA TEMİZEL

(ODTÜ)

Öğrenci Adı: Erdoğan Mert DEMİRYÜREK

(BS)

Ocak 2025

REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY (Internal Use)	2. REPORT DATE 10.01.2025

3. TITLE AND SUBTITLE

JAMMER DETECTION IN AUTONOMOUS VEHICLES WITH MACHINE LEARNING

4. AUTHOR (S) Erdoğan Mert DEMİRYÜREK	5. REPORT NUMBER (Internal Use)

6. SPONSORING/ MONITORING AGENCY NAME(S) AND SIGNATURE(S)

7. SUPPLEMENTARY NOTES

8. ABSTRACT (MAXIMUM 200 WORDS)

Wireless communication systems of autonomous vehicles may be vulnerable to threats such as jammer attacks. As a result of these attacks, operational failures and security problems may occur. This study aims to classify which jammer attacking scenario occurs, using machine learning in order to select the necessary precaution to be taken against jammer attacks. A simulated dataset consisting of parameters such as RSSI, SNR, PDR and estimated relative speed was used in the study. KNN, Random Forest and XGBoost models were used for jammer detection and their performances were compared. The results showed that all of the models, KNN, Random Forest, and XGBoost models has similar accuracy results which is above 90%. The results show that accurate detection of jammer attacks can be achieved with the machine learning algorithms

9. SUBJECT TERMS	10. NUMBER OF PAGES 74

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

Autonomous vehicles are a technology that meets today's needs in road transportation and ushers in a new era in transportation. These vehicles share data with other autonomous vehicles in real time, process this data and make navigation decisions and ensure passenger safety with this method. Continuing uninterrupted communication is very important for autonomous vehicles to ensure operational success.

As with every wireless communication system, there are also weaknesses in the communication analysis found in autonomous vehicles. They aim to improve the success of jammers by attacking autonomous operating systems. By intervening in this capability environment, they ensure that the communication signal belonging to the autonomous vehicle is disrupted. As a result of the disrupted signal, the autonomous vehicle cannot receive data and cannot receive information from other vehicles. Such attacks, through failures or delays, demonstrate the integrity and durability of an AV system or cause a worse accident. That shows the urgent need to develop some efficient mechanisms for detecting such threats and acting accordingly.

Under these conditions, the knowledge of whether the situation is normal or jammed is important for communication systems because the countermeasure mechanisms change according to this decision. Misclassification can lead to overreaction due to non-malicious conditions or underreaction due to the presence of real threats. The work presented here aims to predict the attack scenarios of jammers using machine learning algorithms.

These jammer attacks severely disrupt AV systems by disrupting key communication metrics such as signal reliability, latency, and packet delivery. Traditional methods for detecting anomalies in wireless networks rely on fixed thresholds or rule-based systems that cannot adapt to dynamic and diverse operational conditions. Since AVs operate in an ever-changing environment, solutions must be more flexible and intelligent.

Jammer attacks can occur in different ways and different precautions should be taken against each scenario. The most popular of these are the Reactive Attack and Constant Attack scenarios. In the Reactive Attack scenario, the jammer does not attack until it receives the broadcast sent by the autonomous vehicle or gets close enough to the autonomous vehicle and starts the attack when it reaches the autonomous vehicle. This type of attack is difficult to detect because there is no jamming broadcast in the environment until the target autonomous vehicle is attacked, and the attack starts when the autonomous vehicle becomes vulnerable. In the Constant Attack scenario, there is a jammer in the environment, and it regularly pollutes the communication channel. While it can be detected more easily than in the Reactive Attack scenario, it seriously reduces the operational success of autonomous vehicles. In addition to jammer scenarios, communication systems may experience packet loss due to signal reflections and fading that may occur even if there is no jammer in the environment. It is also very important to distinguish this situation from jammer scenarios in terms of taking countermeasures.

The countermeasure to be applied is selected as a result of the classification of the jammer attack scenario. If there is no jammer in the environment and packet loss or noise is encountered, the waveform's durability can be increased by switching to modulations with lower coding rates. In the Reactive Attack scenario, frequency hopping mode can be switched to and regular communication can be provided from the nearby jammer at a different frequency.

In the Constant Jammer scenario, since there is a fixed broadcast in the environment, communication can be continued at a fixed frequency other than the currently used frequency and the output power can be increased.

Machine learning has emerged as one of the approaches that can be used to overcome the classification of jammer scenarios by recognizing patterns. Machine learning models can analyze key features such as received signal level, signal-to-noise ratio, and packet throughput to classify scenarios and predict potential jammer attacks. Unlike static methods, ML models learn from the data itself, and their performance improves over time and in various contexts. Since different jammer scenarios such as No Attack, Reactive Attack, or Constant Attack require different countermeasures, machine learning can be used to reliably classify these scenarios to secure an AV communication system.

This study explores how the machine learning algorithms, including K-Nearest Neighbors (KNN), Random Forest (RF), and XGBoost, can be used for jammer detection in AV systems.

Developing different machine learning models to classify the jammer attacking scenarios, comparing the performances of each model, identifying key input features like RSSI, SNR, PDR and Variation in Relative Speed are the objectives of this study.

CHAPTER 2 LITERATURE REVIEW

There have been studies on machine learning in many different areas in RF communication systems. In this Literature Review section, a brief summary of machine learning or deep learning studies used in RF communication systems is given.

The study of Jaganattah et al. (2019b) used machine learning algorithms for optimizing the wireless communication system for Internet of Things. The aim of the study is that optimizing spectrum sharing, medium access control, and routing protocols with machine learning models to improve overall communication efficiency in wireless networks. Using machine learning models is a good choice because traditional methods may be insufficient under variable environmental conditions and may consume more energy in calculations compared to machine learning models. The dataset used in the work includes signal characteristics and network parameters. The inputs of the model are network traffic data and signal characteristics. The proposed model predicts improvements in spectrum utilization and energy efficiency. Thus, it is aimed to increase the efficiency in both physical, network and application layers in terms of the use of machine learning in wireless communication systems. As a result of this study, it has been shown that the problem can be approached with different methods from supervised learning models to deep learning models in areas such as improving spectrum efficiency, optimizing routing and increasing energy efficiency under different environmental conditions of wireless communication systems. (Jagannath et al., 2019b)

The study of Erpek et al. (2019) is about deep learning techniques applied to wireless communication systems in various study areas. The main purpose of the study is detecting signals, sensing spectrum, and modulation classification to improve the performance, safety and efficiency of the wireless communication systems. The dataset includes wireless communication signals from both simulated signals and real-world communication systems. The model is trained with the I/Q signal data, which is a digitalized signal, and the model classifies the signal type, channel state information, and jamming. The study uses deep neural networks and convolutional neural networks, for classification tasks and monitoring model's performances with accuracy and mean squared error. The key findings of the study show that deep learning models performed better than traditional signal processing methods, especially when there is noise and interference in the environment. This study is presented as a guide for future research. As a result, the study shows that deep learning can be used in many areas such as jammer classification, signal classification and channel status in wireless communication systems and that the models used increase the resistance of communication systems against jamming attacks, close security gaps and increase their performance. (Erpek et al., 2019)

The study of Sun et al. (2019) provides a review of machine learning techniques applied to wireless networks, such as resource management, networking, mobility management, and localization. The study has discussed in detail the problems and open issues that will be encountered with the use of supervised, unsupervised, and reinforcement learning models from the network layer to the physical layer and even to the application layer of wireless communication systems. As a result of this study, it has been shown that machine learning can be used in many areas such as spectrum management, beamforming, power management, indoor positioning, network traffic prediction. In addition, it has been presented that there are no standardized datasets for the difficulties encountered in using machine learning in wireless communication systems, that more work needs to be done on network slicing, and that models should be trained by combining data collected from different layers. (Sun et al., 2019)

In the study of Zhou et al. (2021), the use of machine learning algorithms to optimize advanced 5G wireless systems, focusing on tasks such as adaptive modulation selection, channel equalization, channel coding, beamforming, load prediction, and trajectory prediction is examined. The main purpose of the study is showing the effects of machine learning on network efficiency, performance, QoS, resource management and energy consumption. In this study, performance metrics for different tasks are given as follows, accuracy for the load prediction, error correction for channel coding, and resource allocation efficiency for beamforming. As a result of this study, it has been presented that the use of machine learning in 5G wireless systems has many benefits, but it can be difficult due to long training times and variable conditions. The study suggests using of distributed machine learning in future studies. (Zhou et al., 2021)

The study of Ahmad and Agarwal (2024) presents the implementation of a machine learning algorithm for a Multiple-Input-Single-Output system using software-defined radios. It focuses on signal detection in MISO wireless communication systems without knowing the channel state in the transmitter side. Model is trained with the RF signals and predicting transmitted symbols. The dataset comprises combinations of signal and noise under different fading, noise, and channel distortions. The machine learning approach is outperformed traditional signal processing methods like maximum ratio transmission by improving the BER, energy consumption and the performance with a 10 dB gain. This study presents a method that increases system efficiency by eliminating the need to know the channel state on the transmitter side using machine learning. (Ahmad & Agarwal, 2024)

In the study of Pan et al. (2018), supervised machine learning algorithms were used to improve the performance of radar systems under complex electromagnetic environment. The inputs to the model are the number of reference and guard cells in the constant false alarm rate detection process, jamming signal power, and the frequency modulation index of noise. The output is radar range estimation performance, which is compared with RMSE. Linear Regression, KNN, Support Vector Machine, Random Forest, Gradient Boosting, and Multilayer Perceptron models are trained on 90 experimental samples generated using uniform design methodology to predict radar performance. The analysis is conducted in scenarios involving jamming and noise, with features of both the radar and electromagnetic interference considered. MLP outperformed other models used in the study by RMSE (1.77) metric. As a result of this study, it has been shown that radar performance can be successfully estimated with MLP without the need for large amounts of data. (Pan et al., 2018)

The study of McCaskey et al. (2018) explores the use of neural networks to model and optimize node-to-node RF communication channels. In this framework, the inputs and outputs represent transmitted and received bits, while the auto-encoder's middle layer simulates a phase-modulated RF signal. The aim of the model is to minimize BER under AWGN channel. The key finding of the study is that the auto-encoder can effectively learn modulation schemes in noisy environments. The auto-encoder shows better performance in simulations, particularly when trained on noisy data. This study shows that although the traditional BPSK modulation provides better results than the presented auto-encode modulation in the experiments, the presented methodology can be used with future improvements. (McCaskey et al., 2018)

The study conducted by Younes et al. (2023) presents machine learning algorithms to predict the frequency of the laser in an Orthogonal Frequency Division Multiplexing-Free Space Optics (OFDM-FSO) system under varying weather conditions. The dataset used in the study was collected with the Optisystem v.15 software, a simulation tool used for optical communication systems such as fiber optics, freespace optics and photonics, developed by

OptiWave (OptiWave, 2025). The dataset includes distance between the optic communication systems and climatic conditions such as fog, rain, and clear conditions, with wavelengths of 1550nm, 1250nm, and 850nm. For longer wavelengths (lower frequency), the atmospheric losses are lower, and the longer distances can be obtained with the laser. Random Forest and Linear Regression are trained to predict the best wavelengths based on weather conditions. The models take attenuation values as input under different weather conditions, while the output focuses on distance achieved by each wavelength. Random forest is outperformed Linear Regression by accuracy in this study. This study shows the use of machine learning to optimize FSO communication systems by selecting wavelengths based on climatic conditions. (Younes et al., 2023)

The study of Menu et al. (2023), presents machine learning algorithms with a MIMO indoor visible light communication system to improve the reliability of data transmission. In the optical communication systems, a combination of LEDs is used for transmission the light like the transmitter antenna in the RF communication systems. Photodiodes are used as the receiver antenna like in the RF communication systems. The study applies Random Forest, Decision Tree, and Support Vector Machine algorithms to classify and predict transmitted messages. Performance metrics indicate that both Decision Tree and Support Vector Machine achieved 100% accuracy, while Random Forest achieved 97.4% accuracy. The study shows that machine learning algorithms can be used for classifying and predicting the transmitted messages in VLC with very high accuracy. (Menu et al., 2023)

The study of (Aghabeiki et al., 2021), introduces a machine learning methodlogy to spectrum sensing for software-defined radio. Machine learning algorithms are used to detect signals with low level SNR in cognitive radio networks. The study compares four machine learning algorithms: Naïve Bayes, Support Vector Machine, Gradient Boosting Machine, and Distributed Random Forest. The dataset includes signal characteristics received under different noise levels, and the models are trained to classify signals. Principal component analysis is done to reduce dataset dimensionality. Performance of the models are compared by using ROC curves. Naïve Bayes and SVM models are the best performed models according to the research for both simulation and real-world tests, especially for low SNR conditions. This study has shown that machine learning models, especially NB and SVM, are effective and achieve high success rates in the field of spectrum sensing. (Aghabeiki et al., 2021)

The study of Valieva et al. (2019), compares machine learning algorithms for classifying modulation types of a radio frequency communication system. The dataset consists of I/Q data and SNR values simulated various SNR conditions ranging from 1 to 30 dB by using Simulink model with AWGN for a software defined radio. Twenty-three supervised ML algorithms are compared in this study. Since the study will be used for real-time cognitive radio applications performance metrics are accuracy and prediction speed. The best accuracy is obtained with Fine Gaussian SVM achieved but it was too slow for real-time use. Decision Trees and Ensemble Boosted Trees models can be used for real time applications because their accuracy is high enough and they are fast. This study has shown that the modulation type of signals with high SNR values can be predicted with a high success rate (97% accuracy at SNR levels above 27 dB) using machine learning models (Valieva et al., 2019).

The study of Senthilkumar et al. (2022), compares two machine learning algorithms (Multilayer Perceptron and Random Forest) to predict frequency bands and path loss in wireless communication systems. The study focuses on frequency bands used in 5G and will be used in future networks, aiming to improve the base station's spectrum allocation and path loss prediction. The dataset includes channel state information and environmental parameters.

Regression models were used to predict path loss and suitable frequency bands across a spectrum ranging from 1 to 100 GHz. The model's performance is evaluated with metrics; mean absolute error, mean squared error, and R-squared. The Random Forest model achieved nearly 90% accuracy in frequency band prediction and an R-squared of 89% in path loss prediction. Results shows that combining supervised and unsupervised learning methods significantly improves prediction accuracy for higher frequency bands. In addition, it has been presented that when Random Forest and PCA are used together, high success is achieved in path loss and frequency band estimation in 5G wireless communication systems and Random Forest can be used for future studies (Senthilkumar et al., 2022).

The study of Al-Amodi et al. (2022) is about machine learning based Convolutional Neural Networks, to estimate and predict parameters of the channel for underwater communication systems. The underwater channel is different from the free space and the frequency of underwater communication systems uses lower frequency bands, but the attribute of the signal is very likely to the wireless radio frequency communication systems. The study presents challenges due to signal attenuation caused by absorption and scattering. The model is trained to estimate temperature gradients and air bubble levels, both of which impact channel performance. The CNN architecture is trained to estimate the channel parameters based on received signal data and signal samples. The dataset is both collected from real world applications and simulated data. The CNN's performance metrics are mean squared error and normalized MSE. The results shows that the CNN can predicts channel parameters with a good performance. (Al-Amodi et al., 2022)

CHAPTER 3 RESEARCH METHODOLOGY

According to the literature review, machine learning and deep learning can be used in many different areas of RF Communication. In this study, the classification of jammer attacks against RF Communication systems used in autonomous vehicles will be provided with machine learning. For this reason, a total of 3 different machine learning models were trained with different combinations of two datasets and their performances were compared. This chapter explains how the study was done.

Dataset Description

The dataset used in this study is the dataset used in a study previously conducted by Kosmanos et al. (Kosmanos et al., 2019). The dataset is available via IEEE Dataport. The dataset used in this study was created to classify jamming methods used to prevent the operation of autonomous vehicles using machine learning. The dataset was divided into two by estimating the speed of the jammer relative to the autonomous vehicle, which was estimated from the I/Q Data analyzed at the physical level of the signal. One dataset was prepared with a maximum estimated relative speed of 15 m/s and the other dataset was prepared with an estimated relative speed of 25 m/s. The dataset includes Time, RSSI, SNR, PDR, Speed, Relative Speed and Scenario parameters.

- **Time:** The time stamp of the measurements taken by the autonomous vehicle.
- **Received Signal Strength Indicator (RSSI):** The parameter that shows the received signal level in dBm scale.
- **Signal to Noise Ratio (SNR):** The ratio of the received signal to the noise level.
- **Package Delivery Ratio (PDR):** A parameter that shows what percentage of packets sent by another vehicle were successfully received at the application level.
- **Speed:** The speed of the autonomous vehicle in m/s at the time of measurement.
- **Relative Speed:** The speed difference between the jammer and the autonomous vehicle in m/s.
- **Scenario:** The type of attacks made by the jammer.

RSSI is the parameter that expresses the power level perceived by the receiver of a signal sent by a transmitter in wireless communication systems. Since RSSI is a power level, it is expressed in dBm, relative to a reference power of 1 milliwatt (mW).

SNR is the ratio of the received signal level to the ambient noise in wireless communication systems. It is generally expressed in dB. It is a parameter frequently used to understand the signal quality and whether there is any multipath or jammer in the environment. However, SNR alone is not enough to make an inference about why the noise floor is increasing.

PDR is a parameter that shows what percentage of the packet from the transmitter side can be decoded on the receiver side in wireless communication systems. It is controlled at the upper level of communication systems. It is inversely proportional to the packet error rate.

Speed indicates the speed of the autonomous vehicle. The speed of the vehicle at the time the measurement is taken is recorded.

Relative Speed is an estimated parameter. It is the difference in speed between the jammer and the autonomous vehicle. This estimation is estimated using I/Q Data, which is the output of the Analog to Digital Converter circuit used in the physical level resolution of the signal. An example block diagram for Receiver Circuit is given at Figure 1 (Takahashi et al., 2007). It can

be explained as the main signal received from the antenna is multiplied with a Local Oscillator and the frequency of the signal is down converted to Intermediate Frequency. The reason behind the multiplying operation, the ADC has a sampling capacity, therefore when the signal is digitalized with a higher frequency, there can be some data loss, therefore its frequency is down converted initially. After that operation signal is digitalized, and I/Q data can be read by the FPGA or processor for further calculations. In our case it is used for both decoding the packages and estimating the Relative Speed of the Jammer. However, this parameter has already been estimated in the dataset used in this study. While doing this, the maximum estimated relative speed was determined as 15 and 25 m/s and 2 datasets were simulated accordingly.

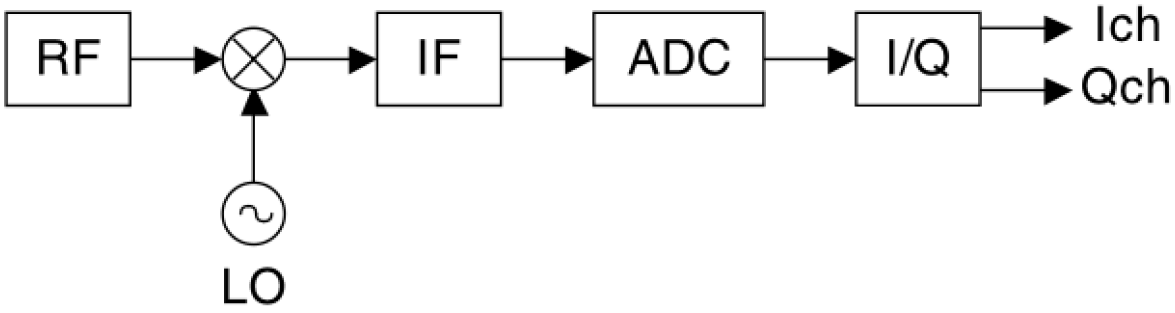

Figure 1 Example Circuit Diagram for Receiver (Takahashi et al., 2007)

The scenarios in the dataset are classified into three types: No Attack, Reactive Attack and Constant Attack. "No Attack" is the scenario where there is no jammer exists in the environment, only the negative effects of multipath on communication, which occurs in areas where the signal reflection is high, such as in urban areas. In the "Reactive Attack" scenario, there is a jammer in the environment, but the jammer does not attack the target vehicle before approaching it but starts the jamming process when it is near the vehicle. Finally, in the "Constant Attack" scenario, there is a jammer in the environment, and it attacks continuously.

In creating the dataset, 1000 measurements were taken in a simulation environment in accordance with the IEEE 802.11p standard for each scenario. The effects of environmental factors such as signal attenuation and interference were included using the Rician fading model. Each scenario has specific parameters designed to evaluate the attack style and effect of a jammer. In particular, the inclusion of the relative speed metric contributes to distinguishing between interference and jamming cases for classification.

Data Simulation

The dataset used in this study was simulated for use in another study conducted by Kosmanos et al. The simulation environment, which can be seen from Figure 2, is designed to model autonomous vehicle communication scenarios and the effects of jammer attacks on these scenarios. The simulation operates in an urban topology with one transmitter and one receiver. In order to simulate realistic channel conditions, a Rician fading model is used, which includes on-line and off-line signal paths caused by reflections. This simulation model combines free space loss, Rayleigh effects and Doppler frequency shift to accurately reflect signal distortions caused by the movement of the autonomous vehicle and objects in the environment that may cause reflections. For the simulation to be adapted to real life, basic parameters such as output power, frequency and Doppler shifts are set in accordance with IEEE 802.11p standards. (Kosmanos et al., 2019)

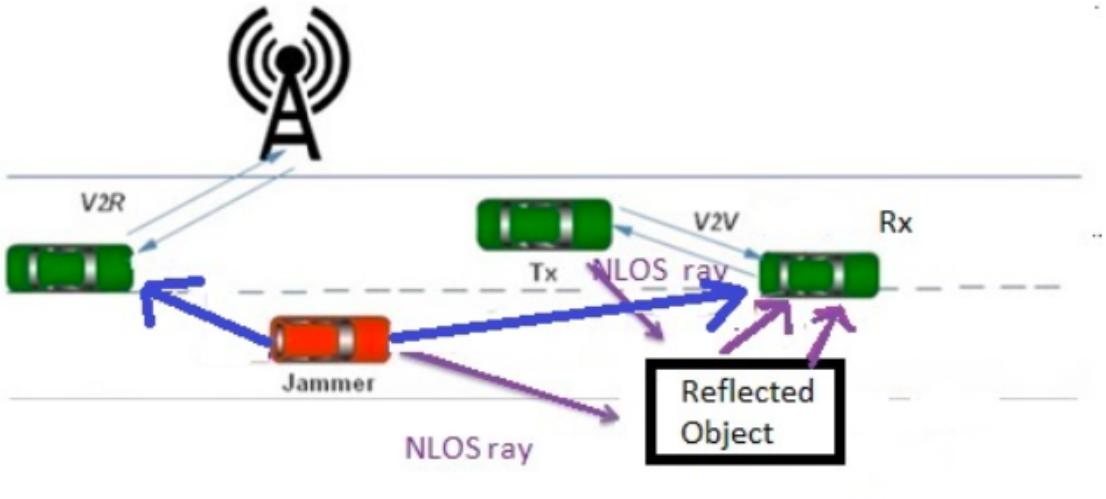

Figure 2 Simulation Setup (Kosmanos et al., 2019)

The study is taken with three different jamming scenarios. In the No Attack scenario, since there is no jammer in the network, only reflections, Doppler shift and free space loss are intended to be included. The Reactive Attack scenario refers to a situation where the jammer changes its output power and timing to avoid detection. In the Constant Attack scenario, the jammer continuously sends a polluting signal at maximum power, causing serious communication disruptions.

Exploratory Data Analysis

Exploratory Data Analysis (EDA) is applied to detect the distributions of the data in the dataset, the relationships between the parameters and the outliers. During EDA, it is also checked whether there is missing or misleading data in the dataset. If there is missing data, additions are made according to the type or cyclicality of the data. However, there is no need for any additional steps because no missing data is in the dataset used in this study.

EDA is a couple of analyses such that correlation analysis, visualization and distribution of the data, to determine the relations between the features in the dataset. Visualization methods such as histograms, scatter plots, QQ plots, and heatmaps are very useful to understand the distributions in the data and the correlations between the variables. Exploratory data analysis is very important step in the training processes of the models, aiming to understand, organize and clean the data to increase the performance of the model.

The histogram plots of the first dataset, the maximum relative speed estimated at 25 m/s, is given in Figure 3. When the histogram plots are examined, it can be seen that the samples are taken at regular intervals and the SNR, Speed, PDR and Relative Speed metrics are generally concentrated in the region. Apart from this, RSSI is concentrated in only one region. The scenarios are numbered 1, 2, 3 respectively as No Attack, Reactive Attack and Constant Attack.

QQ Plots of the first dataset can be seen from Figure 4, since Scenario is a categorical variable, it's not shown on the QQ Plots. It can be understood from QQ Plots that the variables

used in the study are far from normal distribution. Detailed explanations of the plots are given below.

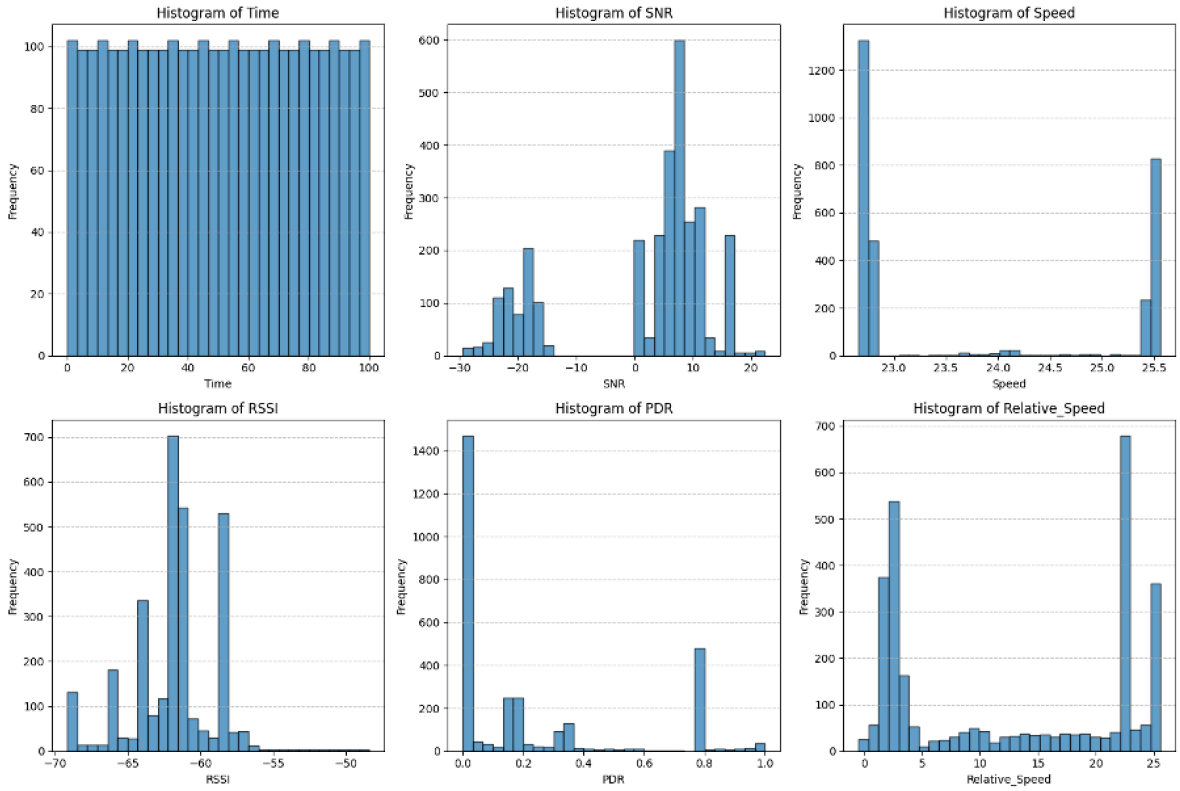

Figure 3 Histogram Plots of Dataset 1

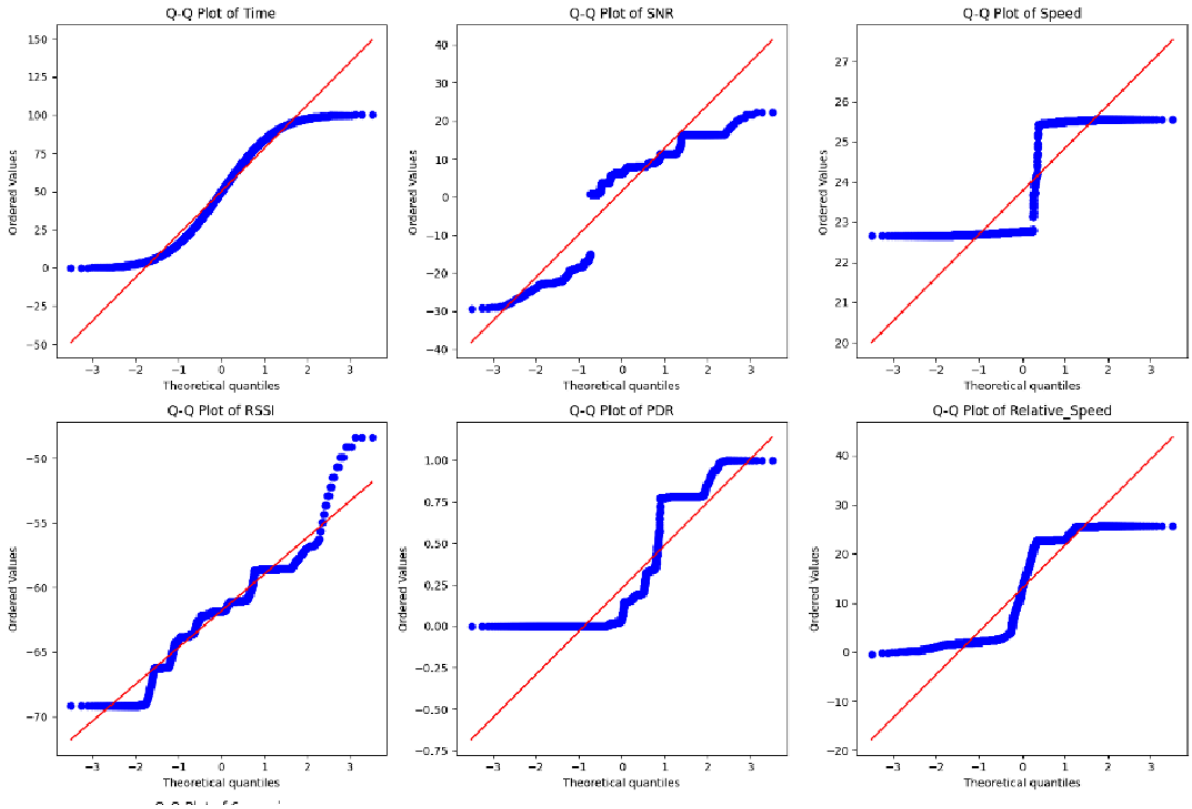

Figure 4 QQ Plots of Dataset 1

- **Time:**

The time parameter is distributed regularly. From here, it is seen that measurements are taken at regular intervals and recorded in the dataset.

- **SNR:**

The SNR variable is generally distributed between -30 dB and 20 dB. As the SNR level decreases, the possibility of an interference or jammer in the environment increases because the noise in the environment will increase in both conditions. For this reason, when the SNR value is below zero, it can be said that the environment is dirty in terms of RF signal. In general, the SNR values that can be measured in a communication system are also like this.

- **Speed:**

The speed variable of the autonomous vehicle generally varies between 22 and 25 m/s. This parameter also shows that it is simulated with a fixed speed in the simulation environment.

- **RSSI:**

RSSI is concentrated between -70 dBm and -50 dBm. When free space loss is taken into account, the signal levels obtained here make sense. Since the main factor affecting RSSI is the distance between two autonomous vehicles, RSSI of two vehicles may vary depending on the distance.

- **PDR:**

The PDR graph shows that the values are distributed between 0 and 1. A value of 1 indicates that no packet loss is encountered, while 0 indicates that all packets are lost. The high number of 0 values in the dataset indicates that the RF pollution in the environment affects the waveforms of the communication systems and that the packets cannot be recovered.

- **Relative Speed:**

The relative speed is distributed between 0 and 25, but there is more concentration at the values of 0 and 25 m/s. Since the maximum estimated relative speed parameter in this dataset is 25 m/s, a maximum of 25 m/s is also seen in the visuals.

- **Scenario:**

It is seen that the scenarios are distributed uniformly. As stated in the simulation, 1000 samples were taken for each scenario.

After examining the histogram and QQ Plots, in order to check whether there is a jammer attack in regions with low packet success rate and low SNR, and whether there is no jammer attack in regions with high SNR and high packet success rate, the SNR and PDR distributions were checked according to the scenarios given in Figure 5. It was seen from the visuals that there is a similar distribution with minor differences in each scenario.

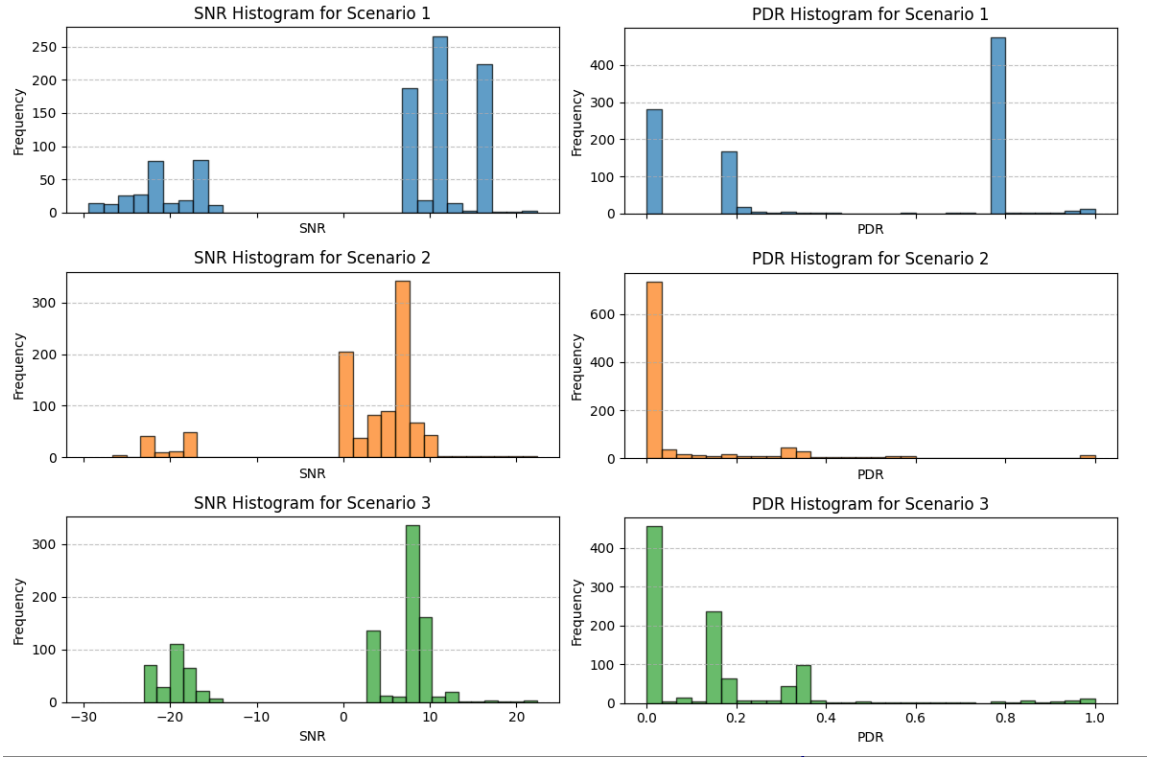

Figure 5 SNR and PDR Distribution According to the Scenario of Dataset 1

The correlation matrix, which can be seen from Figure 6, was used to analyze the relationships between the parameters in the dataset. There is a negative correlation (-0.85) between Time and Speed, and in the simulation conducted considering this information, we can see that the speed of the autonomous vehicle decreases as time progresses. Although not as strong as Speed and Time, there is also a negative correlation (-0.47) between Time and Relative Speed. From here, we can see that the relative speed between the autonomous vehicle and the jammer decreases as time progresses and is simulated.

There is a positive correlation (0.62) between SNR and PDR. This is expected because the signal received by the communication system approaches the noise even more at low SNR values and even remains below the noise at negative SNR values. For this reason, packet loss is expected as the signal deteriorates. Similarly, there is a similar correlation (0.68) between RSSI and PDR. As the signal level decreases, the signal approaches the noise floor and the probability of the system making an error increase.

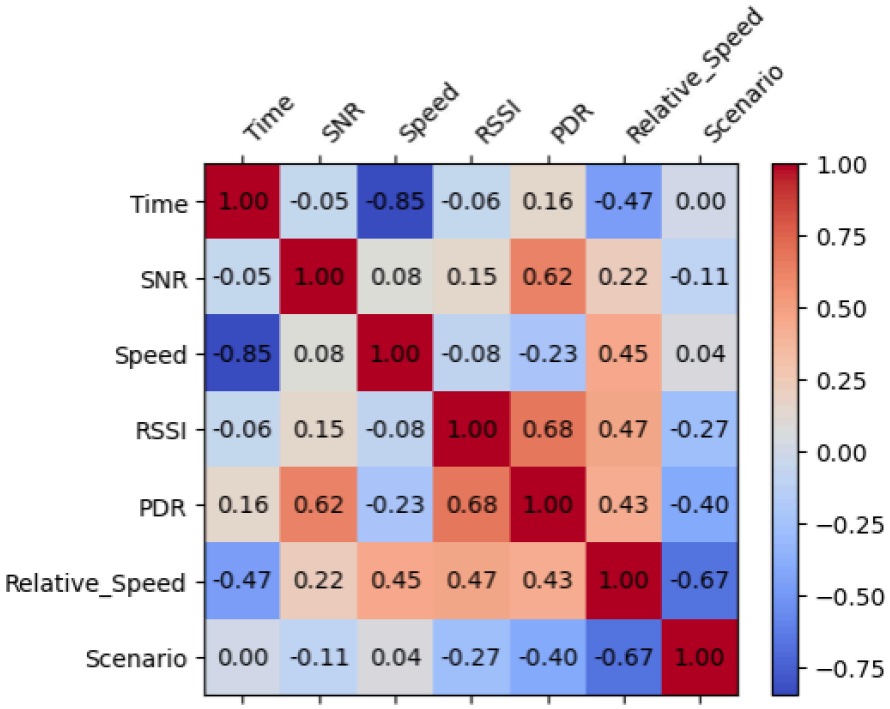

Figure 6 Pearson Correlation Matrix of Dataset 1

The histogram plots of the second dataset, the maximum relative speed estimated at 15 m/s, is given in

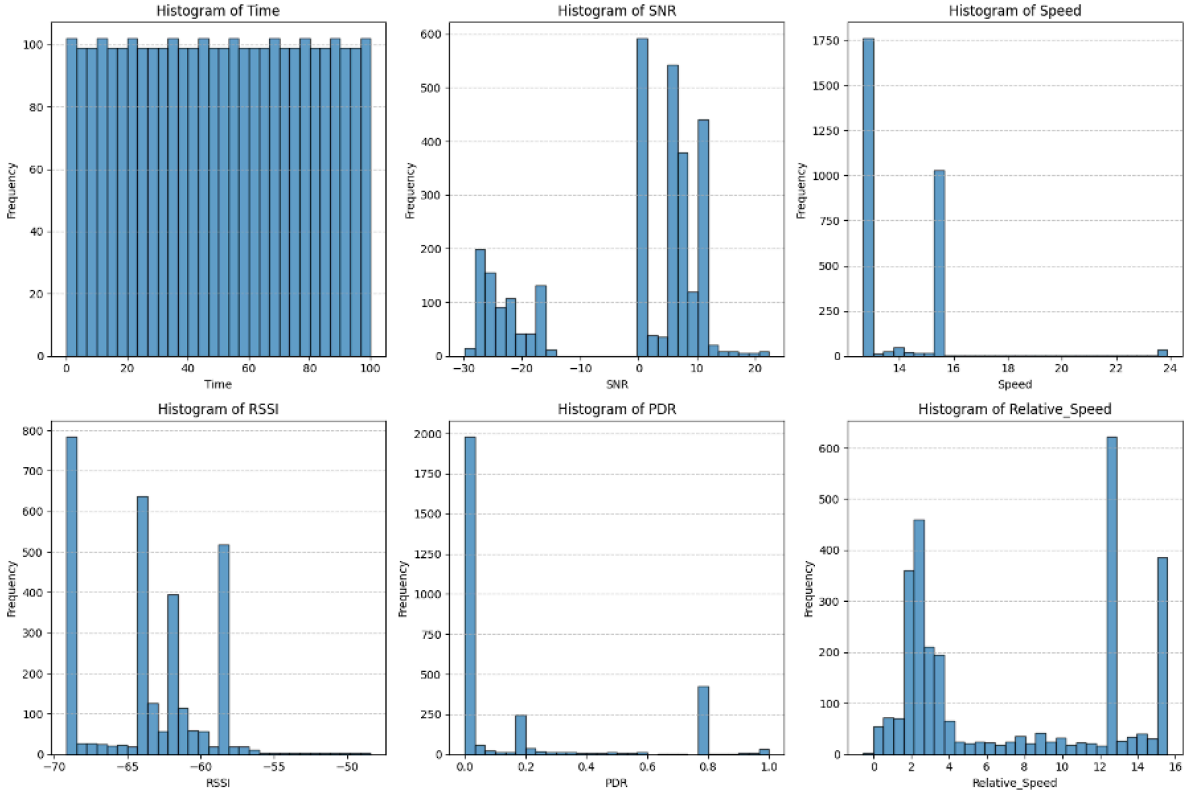

Figure 7. QQ Plots of the first dataset can be seen from Figure 8. There is not a significant difference between the first and the second dataset. The only difference is speed and relative speed because this simulation is done for 15 m/s.

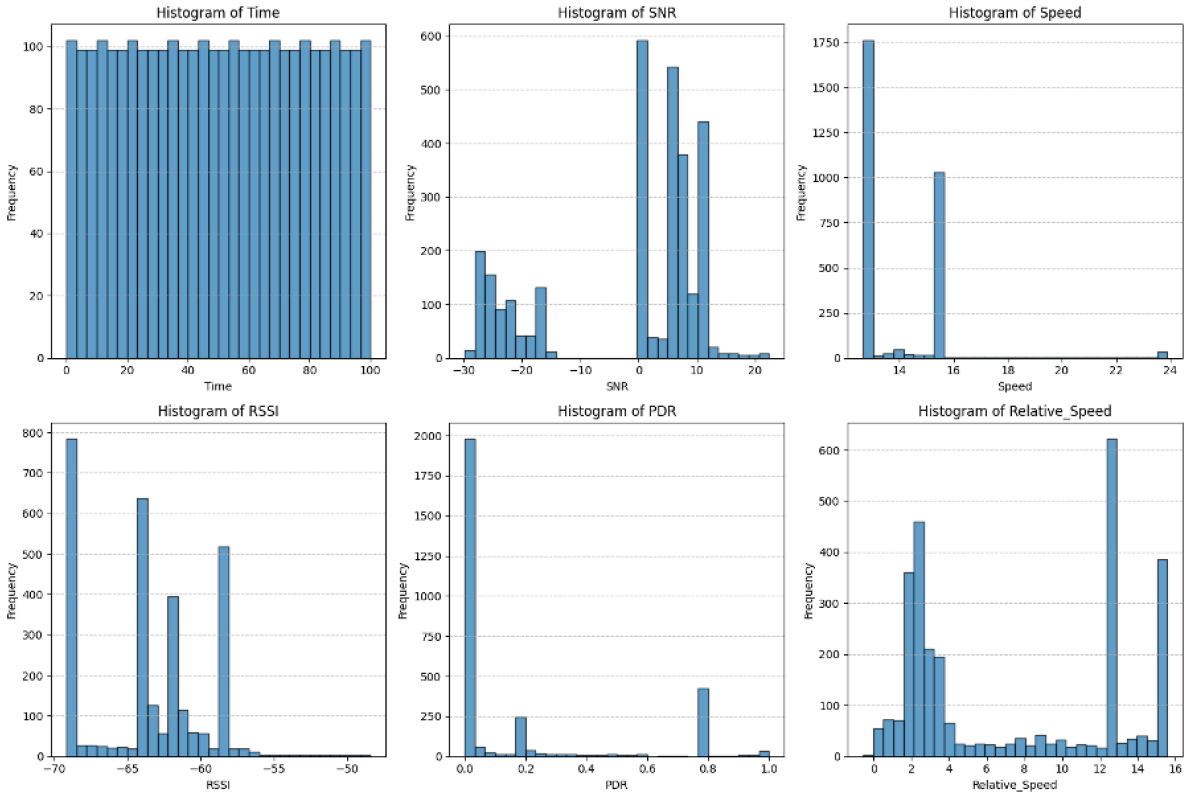

Figure 7 Histogram Plots of Dataset 2

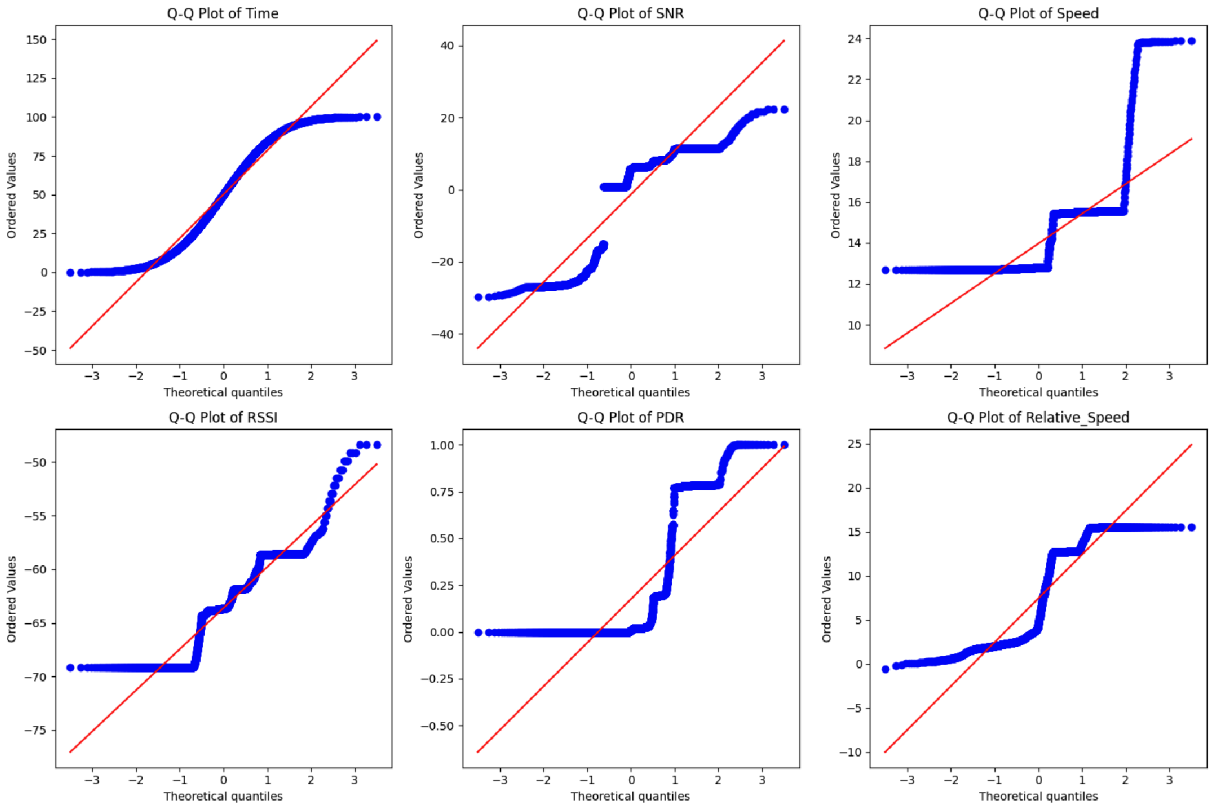

Figure 8 QQ Plots of Dataset 2

After examining the histogram and QQ Plots, in order to check whether there is a jammer attack in regions with low packet success rate and low SNR, and whether there is no jammer attack in regions with high SNR and high packet success rate, the SNR and PDR distributions were checked according to the scenarios given in Figure 9. It was seen from the visuals that there is a similar distribution with minor differences in each scenario for the second dataset.

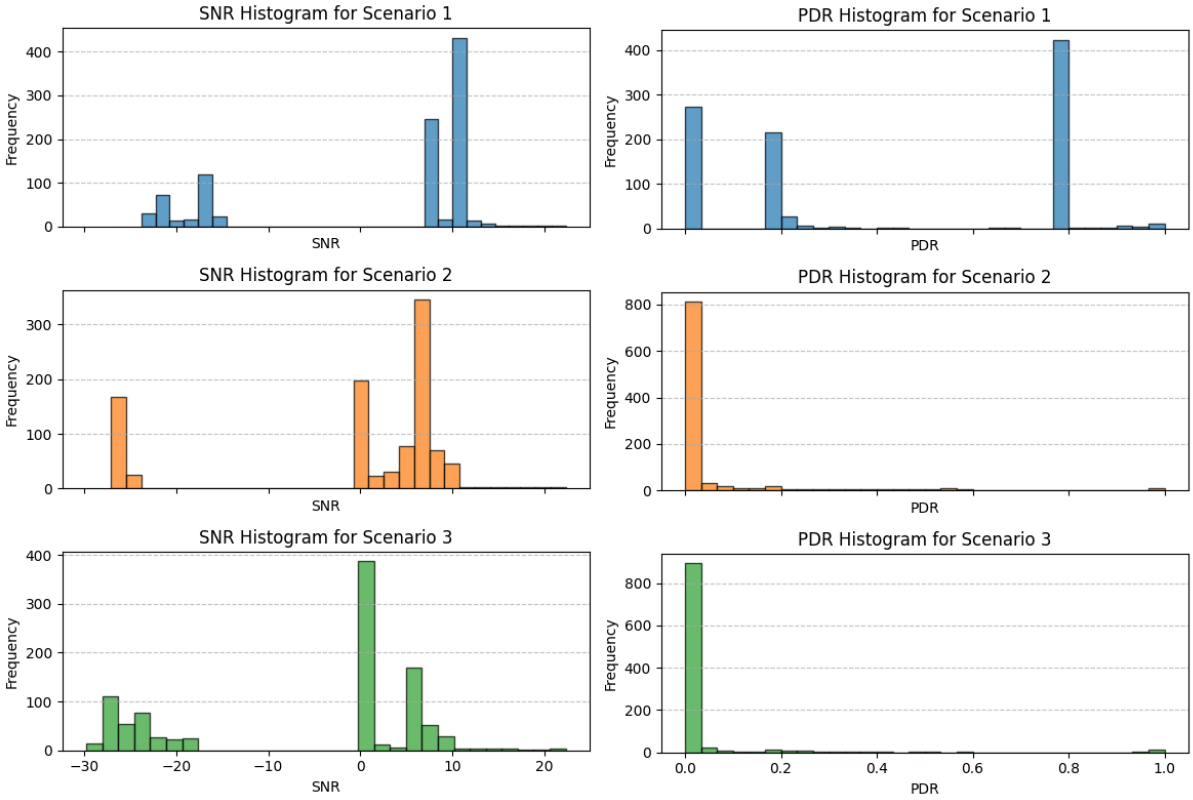

Figure 9 SNR and PDR Distribution According to the Scenario of Dataset 2

The correlation matrix, which can be seen from

Figure 10. In this Pearson Correlation Matrix, it is seen that the correlation between speed and relative speed is lower over time compared to the first dataset. As in the first dataset, there is a positive relationship between SNR and PDR and RSSI and PDR. In this dataset, unlike the first dataset, there is a strong negative relationship between Scenario and RSSI and Scenario and PDR. From this situation, it is understood that there is usually no jammer when high RSSI and high packet performance ratio are obtained.

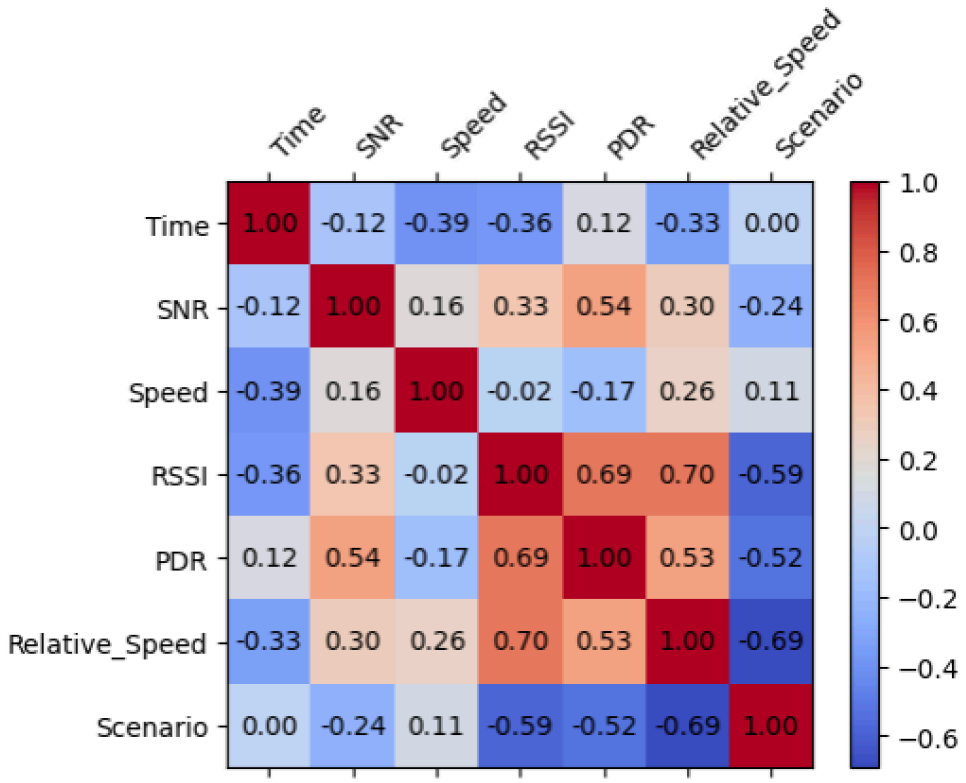

Figure 10 Pearson Correlation Matrix of Dataset 2

Data Preprocessing

In the data preprocessing process, two data sets containing 15 m/s and 25 m/s relative speeds were studied. In order to increase the performance of the model, a parameter was derived that shows whether there is any change in the relative speed. The VRS metric used in the study conducted by Kosmanos et al. is a metric that controls the change in the relative speed of the jammer with the autonomous vehicle. In each measurement taken, the previous and next Relative Speed values are compared, and it is determined whether there is a change. If there is a change, the VRS parameter is set to 1, if there is no change, the VRS parameter is set to 0. The algorithm processes each measurement in order and returns the VRS result of each measurement as a list. (Kosmanos et al., 2019). It is evaluated that the VRS parameter will play an important role in the estimation of the No Attack scenario in the modeling process.

When separating the data set into training and test data, care was taken to ensure that both sets were uniform. A balanced separation of the training and test sets ensured that different speeds and scenario conditions were sufficiently represented in both sets. As a result of the splitting process, 30% of the data set was separated as training and 70% as test data. This ratio was chosen to prevent the model from memorizing or overfitting. For all the KNN models, models are trained with a normalized data by Standard Scaler. Because RSSI and SNR values are high integer values compared to VRS and PDR parameters, they affect the performance of models based on distance calculation such as KNN. However, since decision tree-based machine learning algorithms such as Random Forest and XGBoost do not calculate distance, there is no need to normalize the data. For these two models, an experiment was conducted to check whether there was a change in performance when data is normalized. Therefore, Standard Scaler was not used in all Random Forest and XGBoost models.

A third test and train set were created to see what the performance of the model would be when trained for one relative speed and tested with data from another relative speed. In this case, the train set was set to 15 m/s and the test set was set to 25 m/s.

The 3D scatters of train and test set obtained from the first dataset (25 m/s) can be seen from

Figure 11. 3D When 3D scatters are examined, the RSSI, SNR and VRS distributions in 3D space are shown in the first row. It is seen that in the scenario where there is no jammer in the environment, the VRS metric is always 0, and if there is a jammer in the environment, it is always 1. When there is a jammer, more complex analyses should be performed to classify the attacking scenario from RSSI and SNR. In the plots in the second row, the RSSI, SNR and PDR distributions in 3D space can be seen. It is seen that the distributions of the data for each scenario are intertwined and cannot be easily classified.

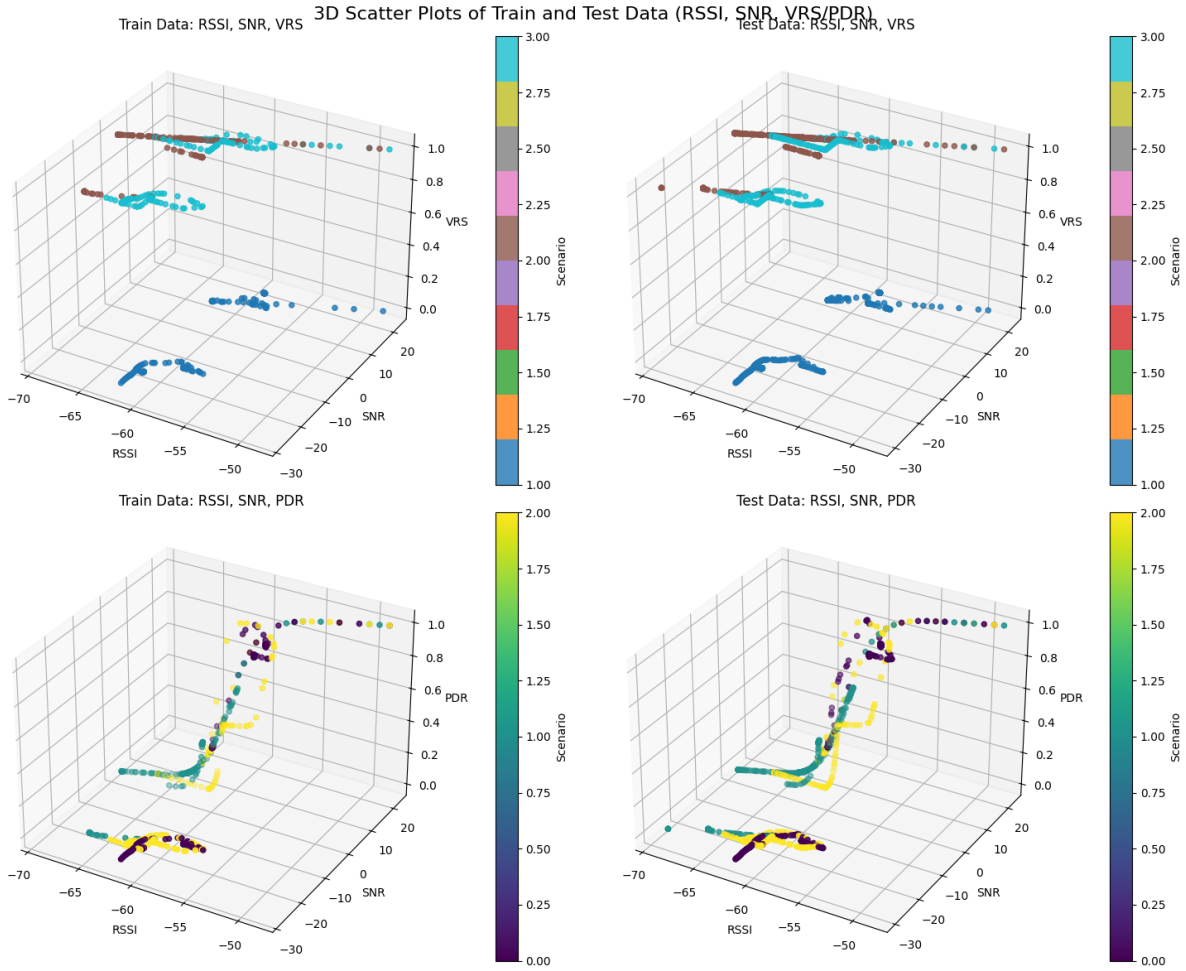

Figure 11 3D Scatters of Train and Test Set Obtained from Dataset 1

The uniform distribution of the test and train set according to the scenario can be seen from Table 1.

Table 1 Train and Test Set size for Dataset 1

Scenario	Train Size	Test Size
1	300	700
2	300	700
3	300	700

The 3D scatters of original train and test set obtained from the second dataset (15 m/s) can be seen from

Figure 12. When the plots are examined, it is seen that the distribution of the second dataset in 3D space is similar to the first dataset and there is no major difference between the distributions of the datasets.

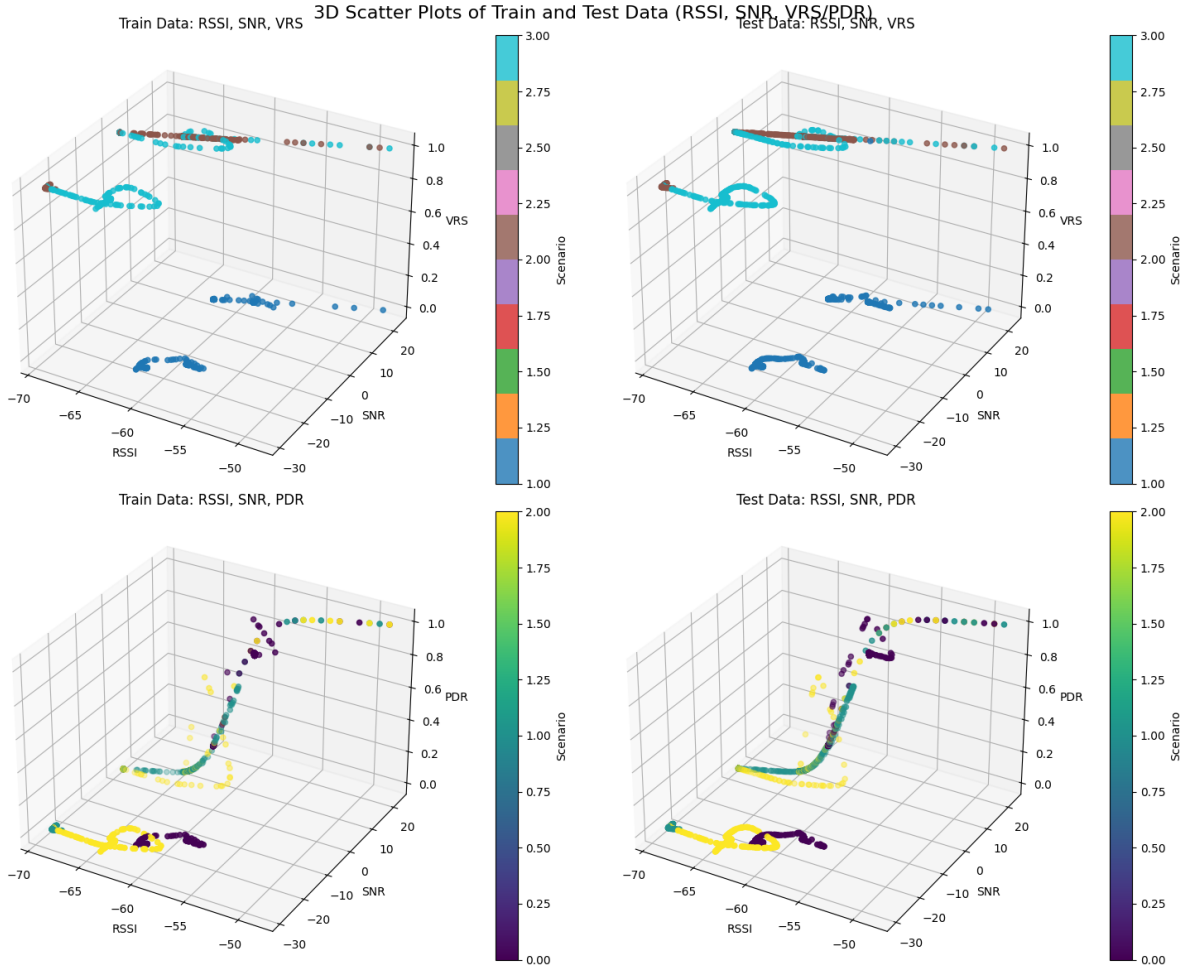

Figure 12 3D Scatters of Train and Test Set Obtained from Dataset 2

The uniform distribution of the test and train according to the scenario set can be seen from Table 2.

Table 2 Train and Test Set size for Dataset 2

Scenario	Train Size	Test Size
1	300	700
2	300	700
3	300	700

The uniform distribution of the train set obtained from the second dataset and test set obtained from first dataset can be seen from Table 3.

Table 3 Train Set size for Dataset 1 and Test Set size for Dataset 2

Scenario	Train Size	Test Size
1	1000	1000
2	1000	1000
3	1000	1000

Models

The models used in this study are the KNN, Random Forest and XGBoost models described below.

- **K-Nearest Neighbors (KNN)**

The foundations of the KNN algorithm were laid in a study conducted by Evelyn Fix and Joseph Hodges (Fix & Hodges, 1989). In addition to this study, Thomas Cover expanded the work of Fix and Hodges with his Nearest Neighbor Pattern Classification study (Cover & Hart, 1967). KNN is a supervised machine learning algorithm which is used for both regression and classification tasks. Unlike decision trees and random forests which are model based learning methods, KNN is lazy or instance-based learning algorithm. In instance-based learning, there is no build explicit model from the training data, it stores the training dataset and based on that dataset predictions are made with the unseen data by comparing to its nearest training examples in the space. So, computation is halted until it needs to make predictions (lazy learning) and it uses entire dataset for the predictions (instance-based learning). Also, KNN does not make any assumption about the underlying data distribution.

KNN in classification process is like the following:

- **Distance Calculations**: In this phase, the distance between unseen data point and every point in the dataset is calculated.
- **Find Nearest Neighbors:** Based on the chosen metric which refers to K, identifying the K closest neighbors to that of unseen datapoint. In this phase, distance metric decision is very important.
- **Voting**: Class of the unseen data point is determined by majority voting by investigating the K nearest neighbors.

KNN in regression process is like the following:

- **Distance Calculations**: In this phase, the distance between unseen data point and every point in the dataset is calculated.

- **Find Nearest Neighbors:** Based on the chosen metric which refers to K, identifying the K closest neighbors to that of unseen datapoint. In this phase, distance metric decision is very important.
- **Prediction**: Prediction of the unseen data point calculated using target values of the K nearest neighbors. Usually average of the items are taken.

The most important thing in KNN is the distance metrics. Usually following three distance metrics are used:

- **Eucledian Distance:** Euclidean distance between two points in Euclidean space is the length of the line segment between them. This is also referred as L2 norm. It is the shortest distance to go from one point to another. This tends to penalize heavier on larger differences. This distance metric is very sensitive to outliers. The formula is as follows:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- **Manhattan Distance:** Manhattan distance (L1 Norm) is the sum of the magnitudes of the vectors in a space. It is the most natural way of measure distance between vectors, that is the sum of absolute difference of the components of the vectors. In this norm, all the components of the vector are weighted equally. This metric is less sensitive to outliers.

$$\boldsymbol{d(x, y) = \sum_{i=1}^{n} |x_i - y_i|}$$

- **Cosine Similarity:** This metric measures the similarity between two vectors of an inner product space. This metric is less sensitive to magnitude differences between features.

$$\boldsymbol{cos(\theta) = \frac{A.B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{N} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i{}^2} \sqrt{\sum_{i=1}^{n} B_i{}^2}}}$$

Hyperparameter tuning is important when developing machine learning models to investigate the best parameter combination. Generally, following hyperparameters are tuned while developing KNN:

1. **Number of Neighbors**
    - K is the most important hyperparameter.
    - If K is too large, distance is calculated with too many points which are far away from each other which causes other classes to play role. This causes model to become too smooth and possibly underfit the data.
    - If K is too small, the model become too sensitive to outliers and noise. Model has a risk to overfit.

2. **Distance Metric**
    - Euclidean distance is the most common choice for continuous numerical data.
    - Decision is based on domain knowledge.
3. **Weighting Scheme**
    - **Uniform weighting:** Each of the K values affect the calculations equally to the final prediction.
    - **Distance-weighted:** Closest neighbors have a higher weight than the far neighbors.
4. **Neighbor search algorithm**
    - **Brute Force:** This computes the distance to all points in the training set.
    - **Tree Based:** If dataset is too large, it can speed up neighbor searches.

- **Random Forest**

Random Forest is a Decision Tree based classification and regression algorithm which is introduced by Leo Breiman (Breiman, 2001). It consists of multiple decision trees at training time and outputs either the mode of classes for classification or the mean prediction for regression. It uses bagging method which is an ensemble method which combines multiple learners to produce the final learned model. These learners are weak learners and used as a base estimator. For each tree random forests use random subset of the data also called as bootstrap samples for each bagging. By introducing randomness, trees become uncorrelated which reduces the variance the final prediction.

To understand random forest better, decision trees must be investigated first. Decision tree composes of internal nodes which is also called as decision points and leaf nodes which is also called as final predictions. At each node, decision tree algorithm selects a feature and a threshold which separates the data with homogenous labels best in classification tasks. For regression tasks, values are splitted rather than categories. For loss function, Gini impurity or entropy is used for classification and mean squared error tried to minimize for regression. A stopping criterion must be set to avoid tree to keep growing and possibly overfit. A single decision tree has a severe risk to overfit that is why random forests are introduced by averaging across many decision trees to stabilize the final prediction.

Random forests work by combining the bagging with random forest selection.

1. **Bagging (Bootstrap Sampling):**
    - For each tree, training set sampling is done with replacement from the original dataset. This means some instances will appear multiple times in tree's training subset where as some will not appear at all.
2. **Random Feature Selection:**
    - When splitting a node in the tree, random forest considers only the subset of the features rather than selecting all the features.
    - With this feature decorrelates the trees because each tree sees different subset of features at each split.
3. **Tree Construction:**
    - Each tree is grown to a large depth. The randomness introduced helps reduce the risk of overfitting that would otherwise happen with a single deep tree.

4. **Prediction Combination**:
    - Each tree votes for a class. The final class is based on majority vote across all trees.
    - The final prediction value is the average of each tree numeric prediction.

Hyperparameter tuning is important when developing machine learning models to investigate the best parameter combination. Generally, following hyperparameters are tuned while developing Random Forests:

1. **Number of trees:**
    - This parameter controls the number of trees
    - More trees generally improve performance however training time is increased.
    - Larger values can stabilize the predictions.
2. **Max Features:**
    - For classification tasks, default is the square root of the total number of features.
    - For regression tasks common default option is total number of features divided by 3.
    - It controls how random each tree's split is.
3. **Max depth:**
    - It controls how deep each tree can grow.
    - To capture more complex dependencies, deeper trees must be used but there is a risk of overfitting.
4. **Min Samples Split:**
    - This is the minimum number of samples required to split an internal node.
    - This parameter prevents trees from becoming to deep and overfit.
    - Larger values prevent make the trees less prone to overfitting.
5. **Min Samples Leaf:**
    - This is the minimum number of samples in a leaf node.
    - This ensures each leaf has at least a certain number of samples.
6. **Bootstrap:**
    - This parameter decides whether to use bootstrap or not. If set to false, the entire dataset is used for each tree.
7. **Criterion:**
    - This is for criterion metric. For classification, gini impurity or entropy is used. For regression, mean squared error or mean absolute error is used.

Random forests provide high accuracy and robustness and can handle large feature spaces. It also prevents overfitting. However, they are less interpretable and has computational and memory cost.

- **XGBoost**

Extreme Gradient Boosting is a frequently used and effective machine learning algorithm for classification presented by Tianqi Chen (Chen & Guestrin, 2016). XGBoost is a gradient boosting algorithm designed for speed and performance. The main idea is as follows:

- Start with simple model. This can be like a constant value.

- Compute the gradients. For classification tasks, make the calculations using the gradient of the loss function with respect to the predictions. For regression tasks, residual is the difference between the current prediction and the target.
- Based on the gradients, train a new model
- Add the new model to the ensemble with a certain weight and learning rate.
- Repeat the process until the stopping criteria is reached.

XGBoost uses second order derivative of the loss function instead of just first order gradients to provide more accurate approximations. It includes L1 and L2 regularization to reduce overfitting in the trees. By this way, it encourages smaller simpler trees. It uses a concept called "gain" to measure how good a split is.

Hyperparameter tuning is important when developing machine learning models to investigate the best parameter combination. Generally, following hyperparameters are tuned while developing XGBoost:

1. **Number of estimators:**
    - Number of trees used in the algorithm.
    - More trees can improve performance however this leads to overfitting.
2. **Learning rate:**
    - Determines the contribution of each tree by a factor. Generally, between 0 and 1.
    - Smaller learning rates requires more trees but can lead to better generalization.
3. **Max depth:**
    - Maximum depth of each tree.
4. **Subsample:**
    - Fraction of training samples used to grow each tree.
    - A value smaller than 1 act like bagging.
5. **Subsample ratio of columns:**
    - Fraction of features used in each tree or at each split.
    - It is like random forest feature subsampling.

In order to examine their performance under different datasets, the models were trained with the train and test set combinations given in Table 4.

Table 4 Model Combinations

Models	Source of Train Set	Source of Test Set	Standard Scaler	VRS
KNN	**Dataset-2** (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	Yes	No
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	Yes	Yes
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	Yes	No
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	Yes	Yes
	Dataset-1 (Rel. Speed: 25 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	Yes	No
	Dataset-1 (Rel. Speed: 25 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	Yes	Yes
Random Forest	**Dataset-2** (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	No	No
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	No	Yes
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	No	No
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	No	Yes
	Dataset-1 (Rel. Speed: 25 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	No	No
	Dataset-1 (Rel. Speed: 25 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	No	Yes
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	Yes	No
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	Yes	Yes
XGBoost	**Dataset-2** (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	No	No
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	No	Yes
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	No	No
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	No	Yes
	Dataset-1 (Rel. Speed: 25 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	No	No
	Dataset-1 (Rel. Speed: 25 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	No	Yes
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	Yes	No
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	Yes	Yes

Hyperparameter Tuning

Hyperparameter tuning process was performed to optimize model performance. In this process, it was aimed to obtain the best performance by trying various hyperparameter combinations for different models. The tuned hyperparameters can be seen from Table 5. For cross validation, "GridSearchCV" function was used within the scope of hyperparameter tuning. In this way, cross validation sets were created at a ratio of 4 to 1 from the given hyperparameter set and the best hyperparameter set was selected according to the average metrics for hyperparameters. Jammer attacking scenarios were predicted over the test set with the selected parameter set.

Weights & Biases is an online tool used to track and manage the outputs of many functions such as hyperparameter tuning and tracking performance metrics in the implementation of machine learning algorithms. It enables visualization to better understand the impact of metrics, so that the performance metrics, hyperparameters and model conditions can be easily followed by the user (Guides, n.d.). Weights & Biases integration to the implementation of the study is done. With this integration, effects of hyperparameter tuning can be seen online. The results of hyperparameter tuning can be seen at Appendix B.

Table 5 Hyperparameters

Model	Hyperparameter	Values
KNN	Number of Neighbors	1,2,3,…,50
	Weights	'uniform', 'distance’
	Metric	‘euclidean', 'manhattan', 'minkowski'
Random Forest	Number of Estimators	10, 20,30,…,200
	Max Depth	5,10,15,20
	Minimum Samples Split	2, 5, 10
	Minimum Samples Leaf	1, 2, 4
	Bootstrap	True, False
XGBoost	Number of Estimators	50,100,150,200,250
	Max Depth	3, 5, 7, 9
	Learning Rate	0.01, 0.05, 0.1, 0.2
	Subsample	0.6, 0.8, 1.0
	Subsample ratio of columns	0.6, 0.8, 1.0

Performance Metrics

To evaluate the model performance, metrics such as accuracy, precision, recall, F1-score and support were used. Thanks to these metrics, the performance of the model under different scenarios was observed.

- **Accuracy**

Accuracy is the ratio of correct predictions made to total predictions.

- **Precision**

Precision is the ratio of predictions that correctly labeled positively to all labeled positive targets.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall**

Recall is the ratio of predictions that correctly labeled as positive to the targets that are positive.

$$Recall = \frac{TP}{TP + FN}$$

- **F1 Score**

F1 Score is harmonic mean of precision and recall. F1 score gives equal importance to precision and recall.

$$F1Score = \frac{2}{\frac{1}{recall} + \frac{1}{precision}}$$

In addition to these metrics, a confusion matrix is also extracted for the models trained with each dataset combination. The confusion matrix visualizes the distribution of each correctly and incorrectly classified scenario. In this matrix, the relationships between the true classes and the predicted classes are clearly shown and it is analyzed whether the model tends to confuse certain classes.

CHAPTER 4 RESULTS

In this section, the results obtained in the study are presented in detail. This study examines the performance of models developed with different train and test sets for the detection and classification of jammer attacks against communication systems used in autonomous vehicles and the effect of optimized hyperparameters. In order to evaluate the performance of the models, primarily the Accuracy metric, as well as Precision, Recall, F1 Score and Support metrics were recorded. In addition, feature importance was checked in order to understand which parameter had the most effect in the training of the models. Accuracy results of the models trained in each combination are given in Table 6.

Table 6 Accuracy Results of Each Model

Models	Source of Train Set	Source of Test Set	Standard Scaler	VRS	Accuracy
KNN	**Dataset-2** (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	Yes	No	0.79
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	Yes	Yes	0.81
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	Yes	No	0.62
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	Yes	Yes	0.73
	Dataset-1 (Rel. Speed: 25 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	Yes	No	0.89
	Dataset-1 (Rel. Speed: 25 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	Yes	Yes	0.93
Random Forest	**Dataset-2** (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	No	No	0.78
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	No	Yes	0.82
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	No	No	0.60
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	No	Yes	0.68
	Dataset-1 (Rel. Speed: 25 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	No	No	0.90
	Dataset-1 (Rel. Speed: 25 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	No	Yes	0.95
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	Yes	No	0.78
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	Yes	Yes	0.82
XGBoost	**Dataset-2** (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	No	No	0.78
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	No	Yes	0.83
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	No	No	0.60
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	No	Yes	0.68
	Dataset-1 (Rel. Speed: 25 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	No	No	0.91
	Dataset-1 (Rel. Speed: 25 m/s)	**Dataset-1** (Rel. Speed: 25 m/s)	No	Yes	0.94
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	Yes	No	0.79
	Dataset-2 (Rel. Speed: 15 m/s)	**Dataset-2** (Rel. Speed: 15 m/s)	Yes	Yes	0.82

When the results are examined, it is seen that KNN, Random Forest and XGBoost models give simmilar results. When the VRS parameter is used, the accuracy is increased, because the presence of the VRS parameter almost eliminates the incorrect No Attack classification. While the VRS parameter is zero in the No Attack case, it is one in the Reactive Attack or Constant Attack scenarios. For this reason, it increases the performance of the model. When the model is trained with one dataset and tested with the other dataset, it is seen that its performance is lower. It is seen that the reason for this is that there are only simulations made in one way in the training dataset, and the model is not sufficient when different conditions occur. In order to increase the performance of the model under different conditions, the model can be trained with data created under different conditions.

The other performance metrics (Precision, Recall, F1 Score and Support) can be seen from below section.

KNN Train: 15 m/s Test 15 m/s, No VRS, Normalization

Table 7 Performance Metrics of KNN Train: 15 m/s Test 15 m/s, No VRS, No Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	0.93	0.96	0.94	700
Reactive Attack	0.70	0.70	0.70	700
Constant Attack	0.73	0.71	0.72	700

KNN Train: 15 m/s Test 15 m/s, VRS, Normalization

Table 8 Performance Metrics of KNN Train: 15 m/s Test 15 m/s, VRS, No Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	1.00	1.00	1.00	700
Reactive Attack	0.72	0.73	0.72	700
Constant Attack	0.73	0.71	0.72	700

KNN Train: 15 m/s Test 25 m/s, No VRS, Normalization

Table 9 Performance Metrics of KNN Train: 15 m/s Test 25 m/s, No VRS, No Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	0.68	0.94	0.79	1000
Reactive Attack	0.59	0.81	0.69	1000
Constant Attack	0.41	0.10	0.16	1000

KNN Train: 15 m/s Test 25 m/s, VRS, Normalization

Table 10 Performance Metrics of KNN Train: 15 m/s Test 25 m/s, VRS, No Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	1.00	1.00	1.00	1000
Reactive Attack	0.57	0.84	0.68	1000
Constant Attack	0.69	0.37	0.48	1000

KNN Train: 25 m/s Test 25 m/s, No VRS, Normalization

Table 11 Performance Metrics of KNN Train: 25 m/s Test 25 m/s, No VRS, No Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	0.89	0.93	0.91	700
Reactive Attack	0.91	0.86	0.88	700
Constant Attack	0.86	0.87	0.87	700

KNN Train: 25 m/s Test 25 m/s, VRS, Normalization

Table 12 Performance Metrics of KNN Train: 25 m/s Test 25 m/s, VRS, No Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	1.00	1.00	1.00	700
Reactive Attack	0.92	0.88	0.90	700
Constant Attack	0.88	0.93	0.90	700

RF Train: 15 m/s Test 15 m/s, No VRS, No Normalization

Table 13 Performance Metrics of RF Train: 15 m/s Test 15 m/s, No VRS, No Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	0.92	0.95	0.94	700
Reactive Attack	0.70	0.73	0.71	700
Constant Attack	0.72	0.67	0.70	700

RF Train: 15 m/s Test 15 m/s, VRS, No Normalization

Table 14 Performance Metrics of RF Train: 15 m/s Test 15 m/s, VRS, No Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	1.00	1.00	1.00	700
Reactive Attack	0.72	0.76	0.74	700
Constant Attack	0.75	0.71	0.73	700

RF Train: 15 m/s Test 25 m/s, No VRS, No Normalization

Table 15 Performance Metrics of RF Train: 15 m/s Test 25 m/s, No VRS, No Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	0.72	0.94	0.81	1000
Reactive Attack	0.55	0.70	0.62	1000
Constant Attack	0.39	0.17	0.24	1000

RF Train: 15 m/s Test 25 m/s, VRS, No Normalization

Table 16 Performance Metrics of RF Train: 15 m/s Test 25 m/s, VRS, No Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	1.00	1.00	1.00	1000
Reactive Attack	0.52	0.69	0.59	1000
Constant Attack	0.54	0.36	0.43	1000

RF Train: 25 m/s Test 25 m/s, No VRS, No Normalization

Table 17 Performance Metrics of RF Train: 25 m/s Test 25 m/s, No VRS, No Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	0.91	0.93	0.92	700
Reactive Attack	0.93	0.86	0.90	700
Constant Attack	0.86	0.92	0.89	700

RF Train: 25 m/s Test 25 m/s, VRS, No Normalization

Table 18 Performance Metrics of RF Train: 25 m/s Test 25 m/s, VRS, No Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	1.00	1.00	1.00	700
Reactive Attack	0.95	0.89	0.92	700
Constant Attack	0.89	0.96	0.92	700

RF Train: 15 m/s Test 15 m/s, No VRS, Normalization

Table 19 Performance Metrics of RF Train: 15 m/s Test 15 m/s, No VRS, Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	0.92	0.96	0.94	700
Reactive Attack	0.70	0.72	0.71	700
Constant Attack	0.72	0.67	0.70	700

RF Train: 15 m/s Test 15 m/s, VRS, Normalization

Table 20 Performance Metrics of RF Train: 15 m/s Test 15 m/s, VRS, Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	1.00	1.00	1.00	700
Reactive Attack	0.72	0.77	0.74	700
Constant Attack	0.75	0.70	0.73	700

XGBoost Train: 15 m/s Test 15 m/s, No VRS, No Normalization

Table 21 Performance Metrics of XGBoost Train: 15 m/s Test 15 m/s, No VRS, No Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	0.93	0.93	0.93	700
Reactive Attack	0.70	0.72	0.71	700
Constant Attack	0.70	0.67	0.69	700

XGBoost Train: 15 m/s Test 15 m/s, VRS, No Normalization

Table 22 Performance Metrics of XGBoost Train: 15 m/s Test 15 m/s, VRS, No Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	1.0	1.0	1.0	700
Reactive Attack	0.73	0.76	0.74	700
Constant Attack	0.75	0.72	0.74	700

XGBoost Train: 15 m/s Test 25 m/s, No VRS, No Normalization

Table 23 Performance Metrics of XGBoost Train: 15 m/s Test 25 m/s, No VRS, No Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	0.73	0.96	0.83	1000
Reactive Attack	0.53	0.67	0.59	1000
Constant Attack	0.39	0.16	0.23	1000

XGBoost Train: 15 m/s Test 25 m/s, VRS, No Normalization

Table 24 Performance Metrics of XGBoost Train: 15 m/s Test 25 m/s, VRS, No Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	1.0	1.0	1.0	1000
Reactive Attack	0.52	0.70	0.59	1000
Constant Attack	0.53	0.35	0.42	1000

XGBoost Train: 25 m/s Test 25 m/s, No VRS, No Normalization

Table 25 Performance Metrics of XGBoost Train: 25 m/s Test 25 m/s, No VRS, No Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	0.93	0.93	0.93	700
Reactive Attack	0.95	0.88	0.91	700
Constant Attack	0.86	0.92	0.89	700

XGBoost Train: 25 m/s Test 25 m/s, VRS, No Normalization

Table 26 Performance Metrics of XGBoost Train: 25 m/s Test 25 m/s, VRS, No Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	1.0	1.0	1.0	700
Reactive Attack	0.95	0.88	0.91	700
Constant Attack	0.89	0.96	0.92	700

XGBoost Train: 15 m/s Test 15 m/s, No VRS, Normalization

Table 27 Performance Metrics of XGBoost Train: 15 m/s Test 15 m/s, No VRS, Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	0.93	0.95	0.94	700
Reactive Attack	0.70	0.73	0.72	700
Constant Attack	0.72	0.69	0.70	700

XGBoost Train: 15 m/s Test 15 m/s, VRS, Normalization

Table 28 Performance Metrics of XGBoost Train: 15 m/s Test 15 m/s, VRS, Normalization

Scenario	Precision	Recall	F1-Score	Support
No Attack	1.0	1.0	1.0	700
Reactive Attack	0.72	0.77	0.74	700
Constant Attack	0.75	0.71	0.73	700

The feature importances can be seen from Table 29. This table shows the most important parameters that affect the performance of KNN, Random Forest and XGBoost when models are trained with different combinations. For Random Forest and XGBoost, no normalization was performed except for the lines containing the Normalization statement given in the table. Since the distance calculation was performed for KNN, normalization was performed for each training and test set combination. In general, RSSI parameter is the most important feature. If the VRS parameter is used, the parameter that has the most effect on the model is VRS. VRS has been the feature with the highest importance scores, especially in the XGBoost model. Since VRS has a great effect in the No Attack scenario, this situation is expected to be encountered. The PDR and SNR parameters are less important for overall models.

Table 29 Feature Importances

Input Combination	KNN (Always Normalized)	Random Forest	XGBoost
Train Set: Dataset-2 (15 m/s) Test Set: Dataset-2 (15 m/s)	RSSI: 0.322476 SNR: 0.231429 PDR: 0.229143	RSSI: 0.244000 PDR: 0.094857 SNR: 0.090095	RSSI: 17.715017 SNR: 11.674231 PDR: 5.715641
Train Set: Dataset-2 (15 m/s) Test Set: Dataset-2 (15 m/s) VRS	VRS: 0.348667 SNR: 0.141619 RSSI: 0.129048 PDR: 0.052000	VRS: 0.382857 RSSI: 0.108667 SNR: 0.082381 PDR: 0.074476	VRS: 23.264172 RSSI: 4.049105 SNR: 1.583684 PDR: 1.513139
Train Set: Dataset-2 (15 m/s) Test Set: Dataset-1 (25 m/s)	PDR: 0.233333 RSSI: 0.190600 SNR: 0.052267	RSSI: 0.092067 PDR: 0.041200 SNR: -0.003400	RSSI: 2.112059 SNR: 0.651852 PDR: 0.595214
Train Set: Dataset-2 (15 m/s) Test Set: Dataset-1 (25 m/s) VRS	VRS: 0.286933 RSSI: 0.097333 PDR: 0.042333 SNR: 0.035800	VRS: 0.284800 RSSI: 0.018400 PDR: -0.006267 SNR: -0.042400	VRS: 402.185913 RSSI: 21.852936 PDR: 17.722486 SNR: 16.116278
Train Set: Dataset-1 (25 m/s) Test Set: Dataset-1 (25 m/s)	RSSI: 0.498381 PDR: 0.354952 SNR: 0.273238	RSSI: 0.345143 SNR: 0.157714 PDR: 0.118095	RSSI: 1.899271 PDR: 1.107933 SNR: 0.912065
Train Set: Dataset-1 (25 m/s) Test Set: Dataset-1 (25 m/s) VRS	VRS: 0.411143 RSSI: 0.252857 PDR: 0.158952 SNR: 0.128190	VRS: 0.357143 RSSI: 0.253333 PDR: 0.095048 SNR: 0.080381	VRS: 22.303427 RSSI: 3.531750 SNR: 1.750347 PDR: 1.580866
Train Set: Dataset-2 (15 m/s) Test Set: Dataset-2 (15 m/s) Normalization	N/A	RSSI: 0.227524 SNR: 0.109143 PDR: 0.068286	RSSI: 9.579320 PDR: 3.202416 SNR: 3.181648
Train Set: Dataset-2 (15 m/s) Test Set: Dataset-2 (15 m/s) VRS Normalization	N/A	VRS: 0.391143 RSSI: 0.112571 SNR: 0.112190 PDR: 0.073048	VRS: 39.290897 RSSI: 10.007263 PDR: 4.358130 SNR: 3.658278

CHAPTER 5 CONCLUSION

In this study, the use of machine learning for scenario detection in jammer attacks against communication systems used in autonomous vehicles is discussed. Since the vulnerability created by the weak points of autonomous vehicles' wireless communication networks can cause operational failures, an attempt has been made to classify jammer scenarios. In this context, developing effective and fast detection methods for jammer scenarios is critical for autonomous vehicle systems.

Jammer detection was performed using XGBoost, Random Forest and k-Nearest Neighbors (KNN) algorithms. In the study conducted, the similar performances were obtained with KNN, Random Forest and XGBoost models. Hyperparameter tuning was performed to increase the performance of the models, and the most ideal parameters were determined. In particular, all of the models provided a similar and high performance in terms of accuracy and other performance metrics. RSSI and VRS are very important parameters in terms of the performance of the models. The results of the project revealed that KNN, Random Forest and XGBoost algorithms can be used for jammer detection.

As a future work, more complex relationships can be learned from signal features using more complex deep learning models such as CNN or LSTM. In addition, models can be trained by collecting more data, both in number and variety, from the physical level of the wireless communication system in addition to the data in the dataset used.The developed models can be tested in real-time scenarios and updated against new requirements.

In conclusion, this study has shown that machine learning-based approaches offer a solution for jammer detection.

REFERENCES

Jagannath, J., Polosky, N., Jagannath, A., Restuccia, F., & Melodia, T. (2019b). Machine learning for wireless communications in the Internet of Things: A comprehensive survey. *Ad Hoc Networks*, *93*, 101913. https://doi.org/10.1016/j.adhoc.2019.101913

Erpek, T., O'Shea, T. J., Sagduyu, Y. E., Shi, Y., & Clancy, T. C. (2019). Deep learning for wireless communications. In *Studies in computational intelligence* (pp. 223–266). https://doi.org/10.1007/978-3-030-31764-5_9

Sun, Y., Peng, M., Zhou, Y., Huang, Y., & Mao, S. (2019). Application of Machine learning in wireless networks: key techniques and open issues. *IEEE Communications Surveys & Tutorials*, *21*(4), 3072–3108. https://doi.org/10.1109/comst.2019.2924243

Zhou, Y., Chen, J., Zhang, M., Li, D., & Gao, Y. (2021). Applications of machine learning for 5G advanced wireless systems. *2022 International Wireless Communications and Mobile Computing (IWCMC)*. https://doi.org/10.1109/iwcmc51323.2021.9498754

Ahmad, A., & Agarwal, S. (2024). Demonstration of Machine Learning Based Receiver for MISO System Using Software-Defined Radios. *2024 IEEE International Conference on Machine Learning for Communication and Networking*, 1–2. https://doi.org/10.1109/icmlcn59089.2024.10624922

Pan, Y., Zhang, J., Luo, G. Q., & Yuan, B. (2018). Evaluating Radar Performance Under Complex Electromagnetic Environment Using Supervised Machine Learning Methods: A Case Study. *2018 8th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, 206–210. https://doi.org/10.1109/iceiec.2018.8473520

McCaskey, M., Kukura, E., Corrigan, R., Bhasin, K., & Chelmins, D. (2018). Machine Learning Applied to an RF Communication Channel. *NAECON 2018 - IEEE National Aerospace and Electronics Conference*, 179–185. https://doi.org/10.1109/naecon.2018.8556708

Optiwave. (2025). *OptiSystem*. Retrieved from https://optiwave.com, Last accessed date: 14.01.2025

Younes, R., Nassr, M., Anbar, M., Ghosna, F., Alasadi, H. a. A., & Voronkova, D. K. (2023). Machine Learning approach for predicting suitable wavelengths in OFDM-FSO system. *2022 4th International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE)*, 1–5. https://doi.org/10.1109/reepe57272.2023.10086914

Menu, K. V., Leke, C. A., & Ndjiongue, A. R. (2023). Machine learning assisted indoor visible light communication system. *2022 International Telecommunications Conference (ITC-Egypt)*, 440–445. https://doi.org/10.1109/itc-egypt58155.2023.10206067

Aghabeiki, S., Hallet, C., Noutehou, N. E., Rassem, N., Adjali, I., & Mabrouk, M. B. (2021). Machine-learning-based spectrum sensing enhancement for software-defined radio applications. *2021 IEEE Cognitive Communications for Aerospace Applications Workshop*, 1–6. https://doi.org/10.1109/ccaaw50069.2021.9527294

Valieva, I., Bjorkman, M., Akerberg, J., Ekstrom, M., & Voitenko, I. (2019). Multiple machine learning algorithms comparison for modulation type classification for efficient cognitive radio. *MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM)*, 318–323. https://doi.org/10.1109/milcom47813.2019.9020735

Senthilkumar, C., Nirmala, P., Ahila, S. S., Geetha, M., & Ramesh, S. (2022). Predicting the Frequency Bands and the Path Loss in Wireless Communication Systems using Random Forests. *2022 3rd International Conference on Smart Electronics and Communication (ICOSEC)*, 669–674. https://doi.org/10.1109/icosec54921.2022.9951963

Al-Amodi, A., Masood, M., & Khan, M. Z. M. (2022). Underwater Wireless Optical Communication Channel Characterization Using Machine Learning Techniques. *7th Optoelectronics Global Conference (OGC 2022)*. https://doi.org/10.1109/ogc55558.2022.10050890

Kosmanos, D., Karagiannis, D., Argyriou, A., Lalis, S., & Maglaras, L. (2019). RF Jamming Classification Using Relative Speed Estimation in Vehicular Wireless Networks. *Security and Communication Networks*, *2021*, 1–16. https://doi.org/10.1155/2021/9959310

Takahashi, Y., Sekine, T., & Yokoyama, M. (2007). A 70 MHz Multiplierless FIR Hilbert Transformer in 0.35 m Standard CMOS Library. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, *E90-A*(7), 1376–1383. https://doi.org/10.1093/ietfec/e90-a.7.1376

Fix, E., & Hodges, J. L. (1989). Discriminatory anXalysis. Nonparametric discrimination: Consistency properties. *International Statistical Review*, *57*(3), 238. https://doi.org/10.2307/1403797

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*(1), 21–27. https://doi.org/10.1109/tit.1967.1053964

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/a:1010933404324

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Guides. (n.d.). Weights & Biases Documentation. https://docs.wandb.ai/guides/, Last accessed date: 14.01.2025

APPENDIX A CONFUSION MATRICES

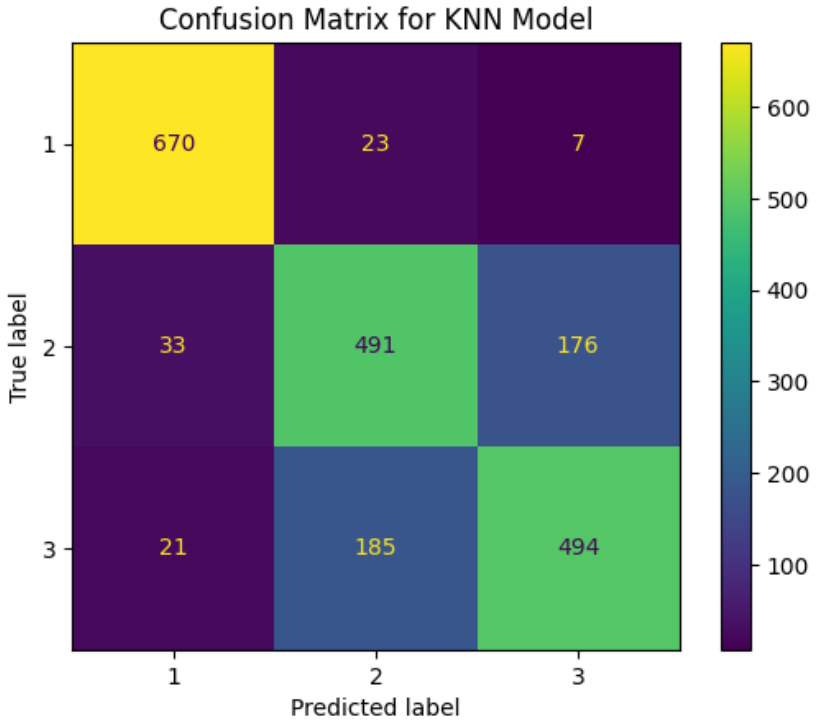

Figure 13 Confusion Matrix: KNN, Train: Dataset-2, Test: Dataset-2, No VRS

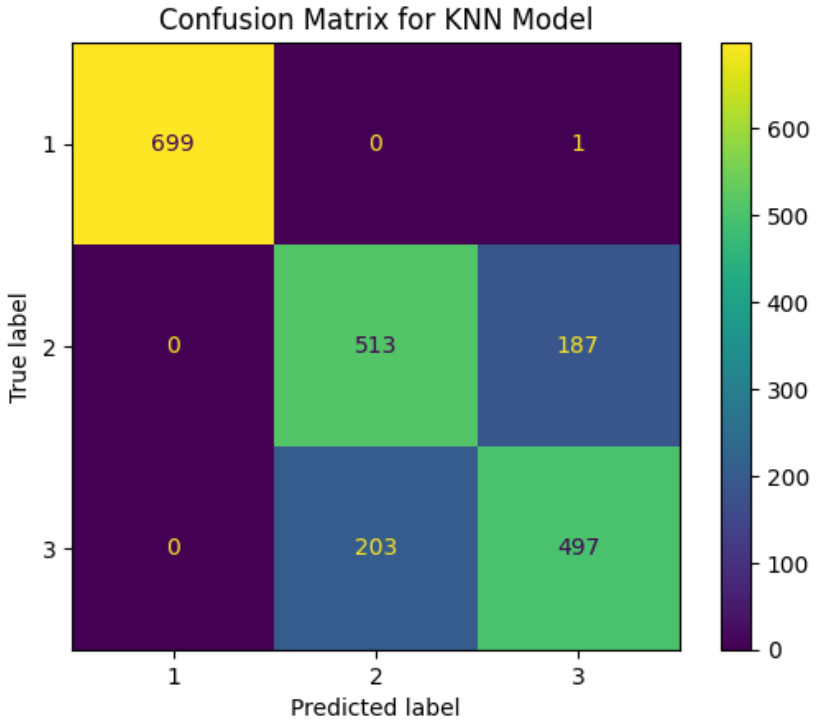

Figure 14 Confusion Matrix: KNN, Train: Dataset-2, Test: Dataset-2, VRS

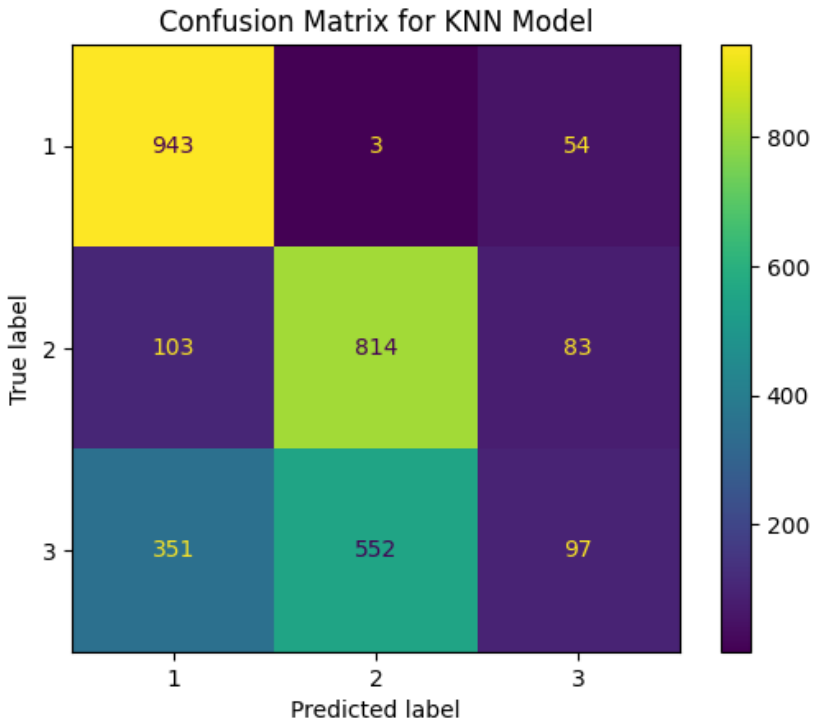

Figure 15 Confusion Matrix: KNN, Train: Dataset-2, Test: Dataset-1, No VRS

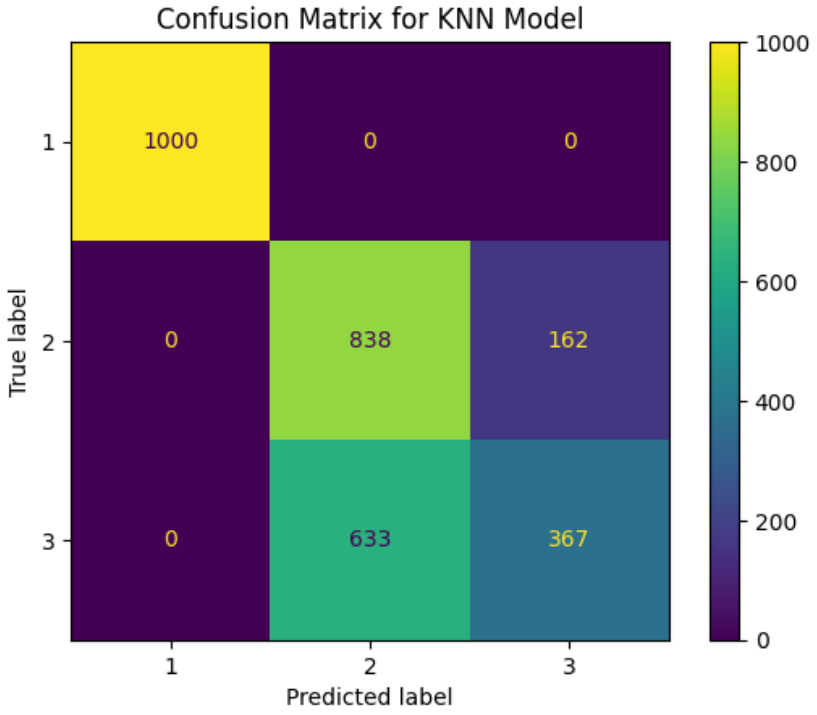

Figure 16 Confusion Matrix: KNN, Train: Dataset-2, Test: Dataset-1, VRS

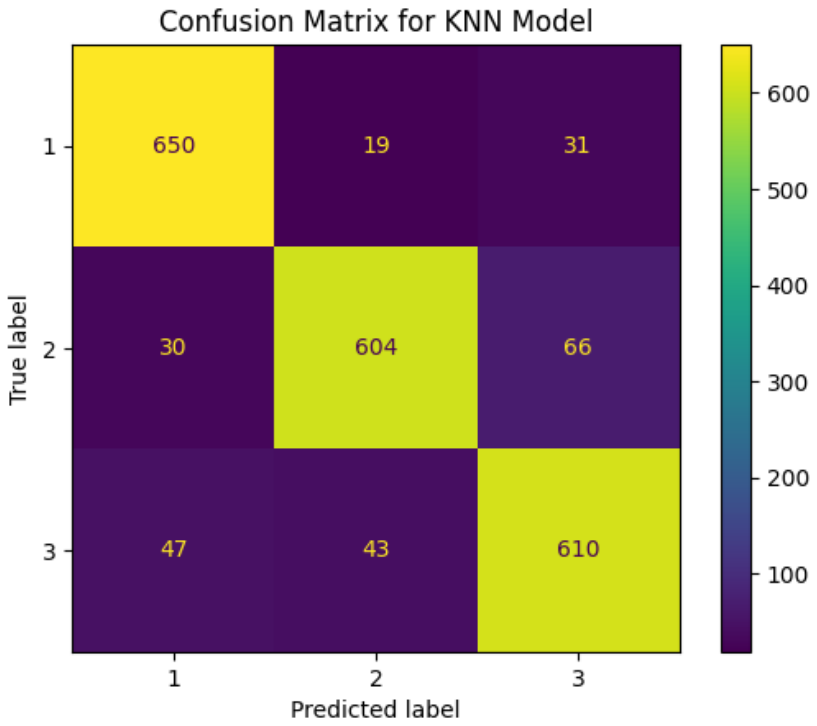

Figure 17 Confusion Matrix: KNN, Train: Dataset-1, Test: Dataset-1, No VRS

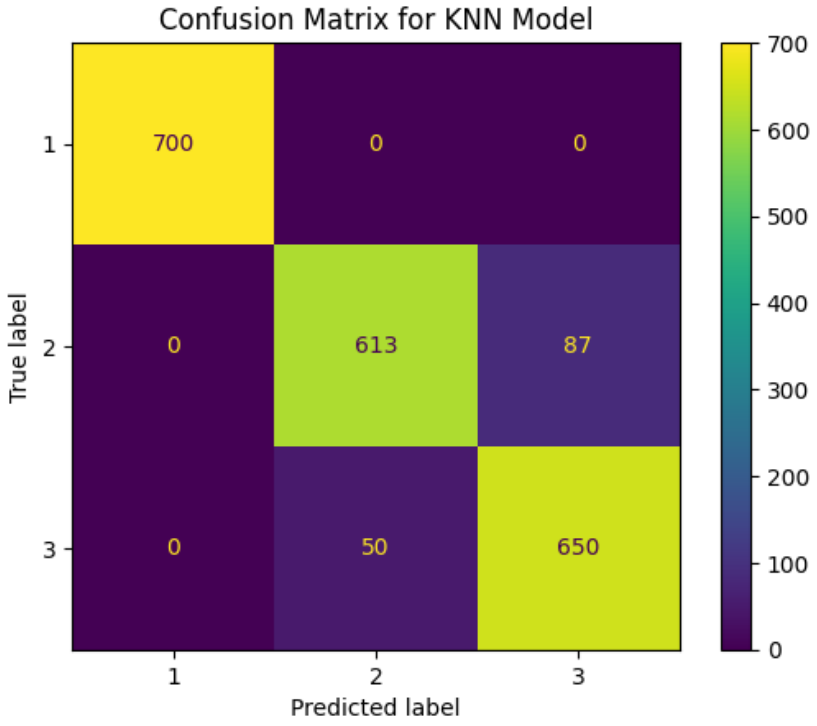

Figure 18 Confusion Matrix: KNN, Train: Dataset-1, Test: Dataset-1, VRS

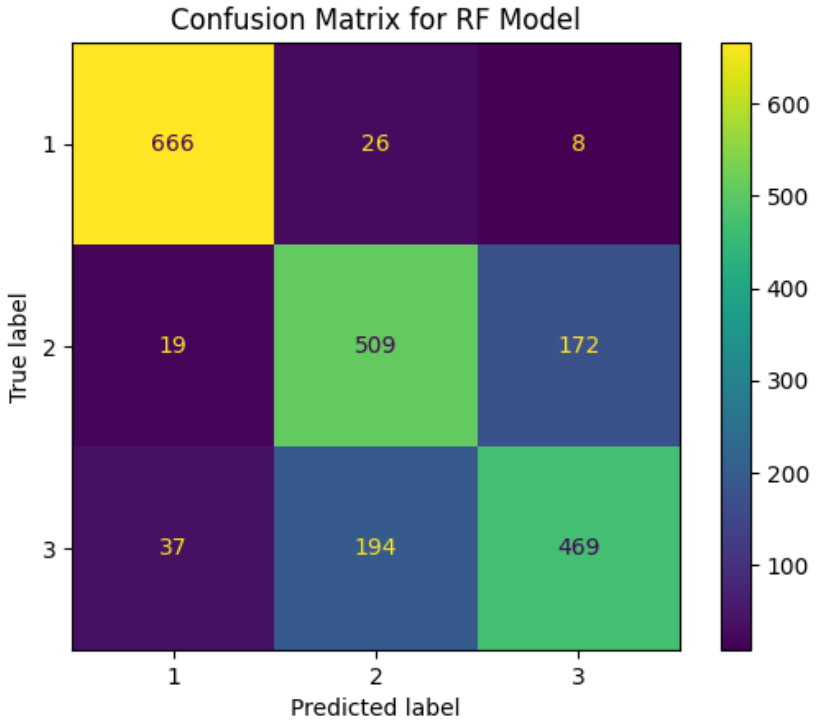

Figure 19 Confusion Matrix: RF, Train: Dataset-2, Test: Dataset-2, No VRS

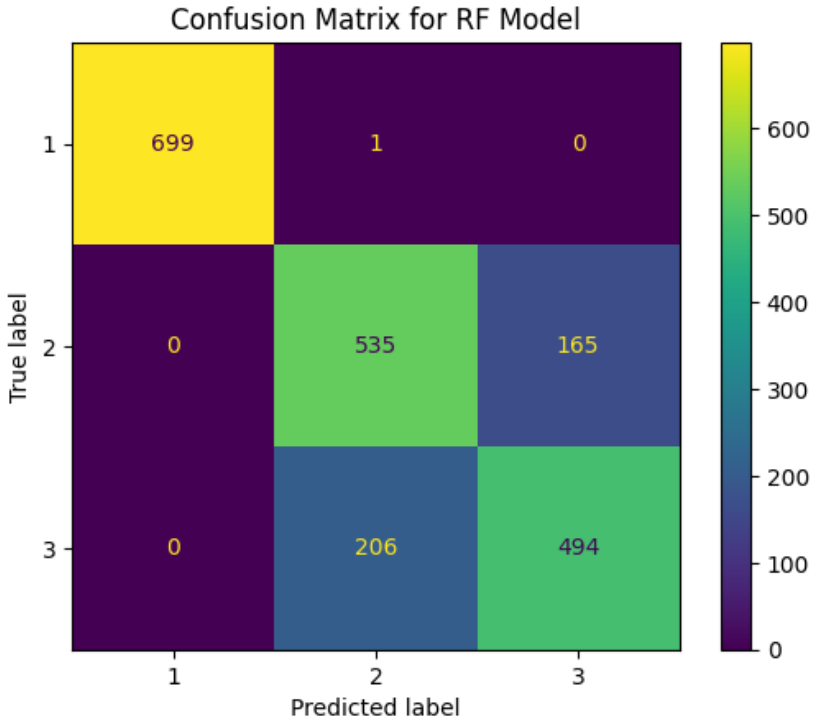

Figure 20 Confusion Matrix: RF, Train: Dataset-2, Test: Dataset-2, VRS

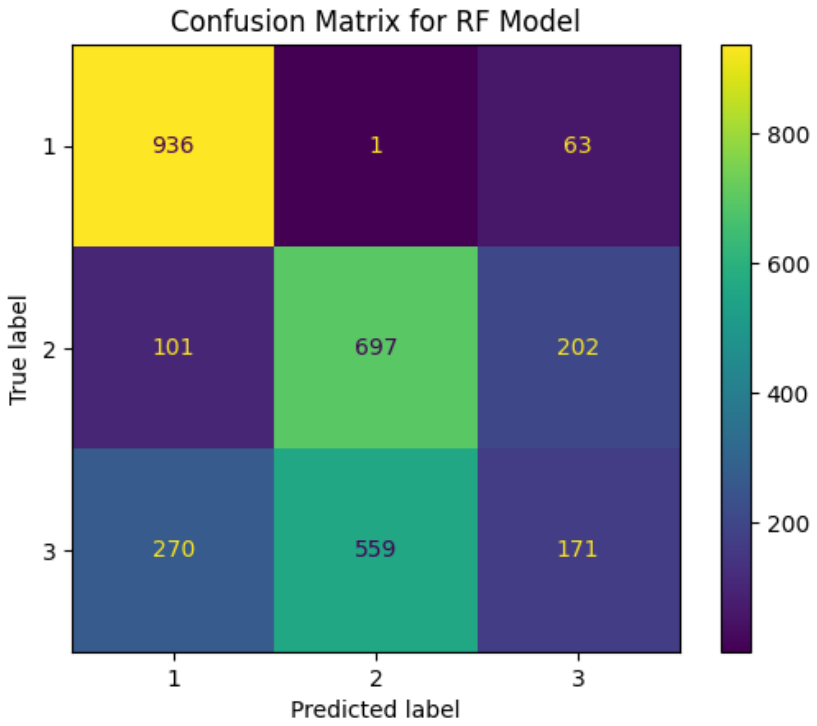

Figure 21 Confusion Matrix: RF, Train: Dataset-2, Test: Dataset-1, No VRS

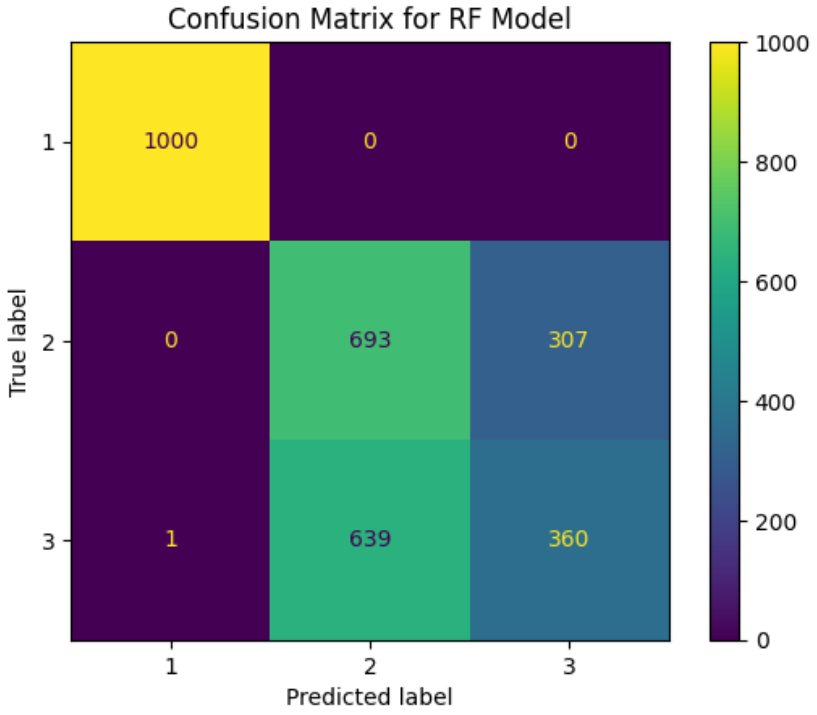

Figure 22 Confusion Matrix: RF, Train: Dataset-2, Test: Dataset-1, VRS

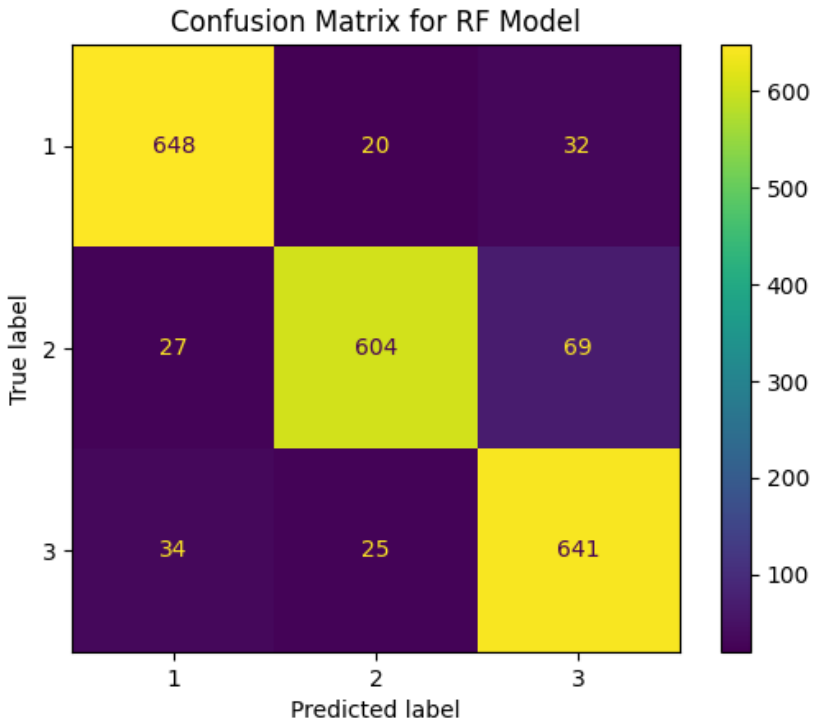

Figure 23 Confusion Matrix: RF, Train: Dataset-1, Test: Dataset-1, No VRS

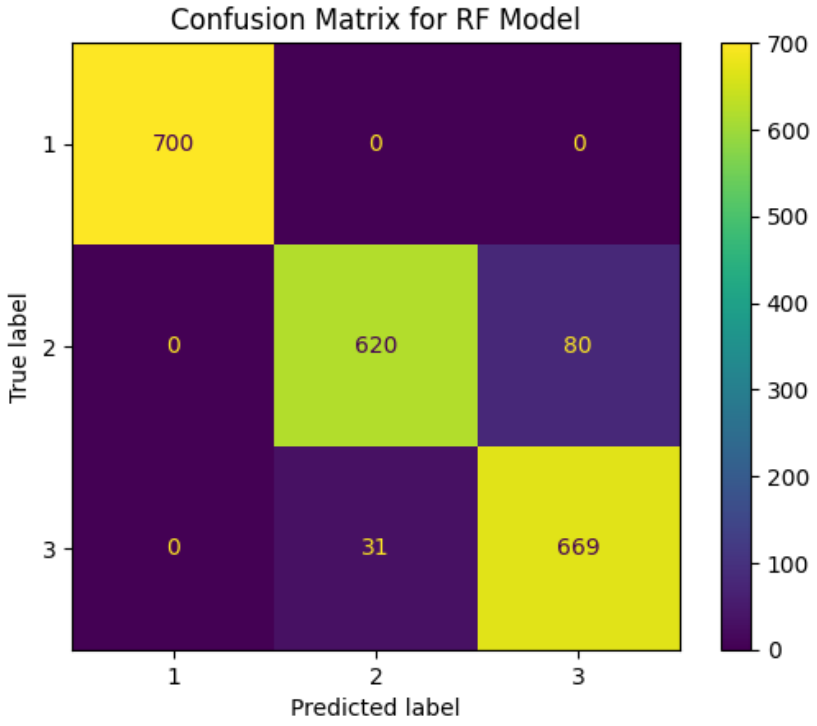

Figure 24 Confusion Matrix: RF, Train: Dataset-1, Test: Dataset-1, VRS

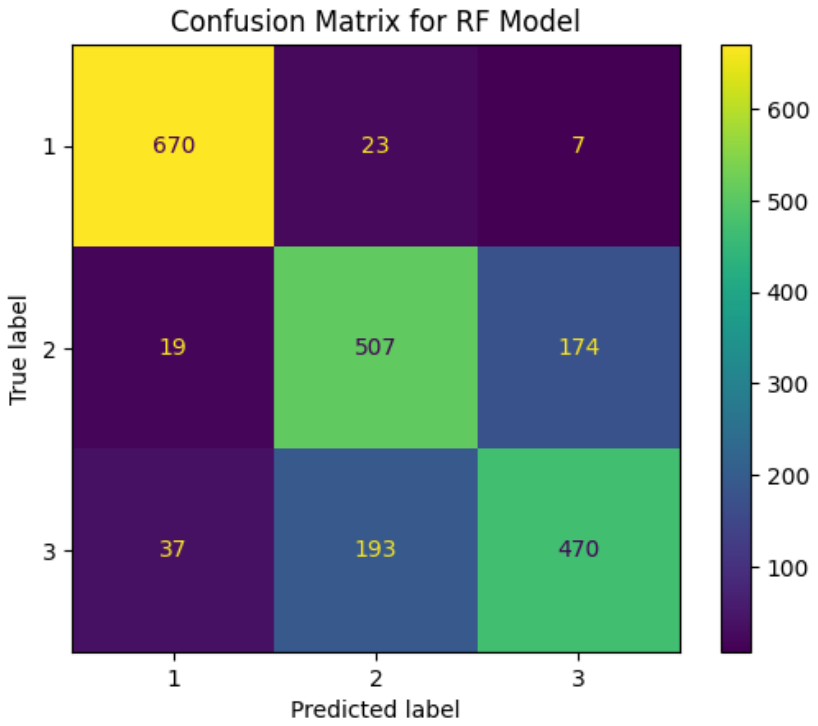

Figure 25 Confusion Matrix: RF, Train: Dataset-2, Test: Dataset-2, No VRS, Normalization

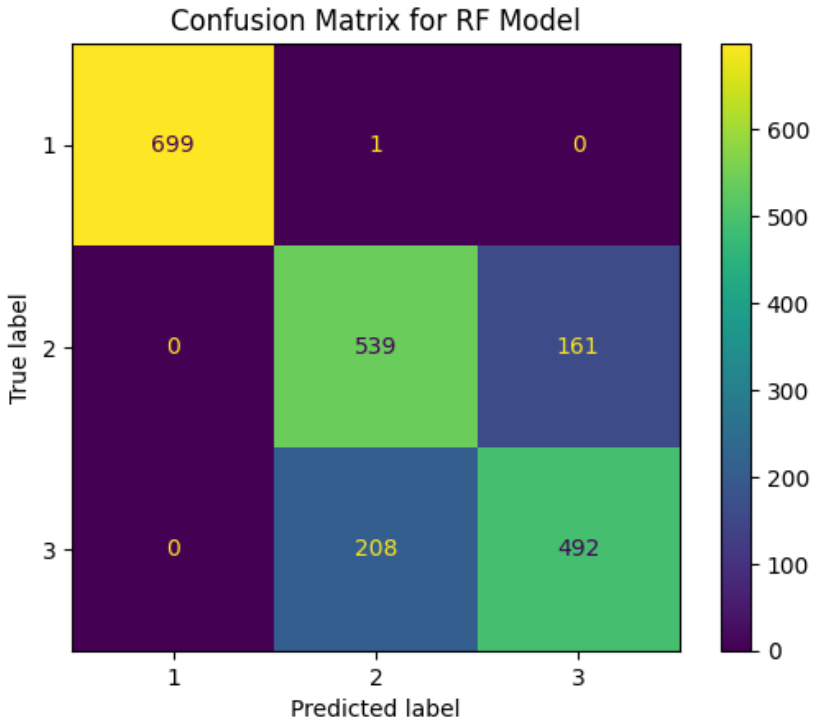

Figure 26 Confusion Matrix: RF, Train: Dataset-2, Test: Dataset-2, VRS, Normalization

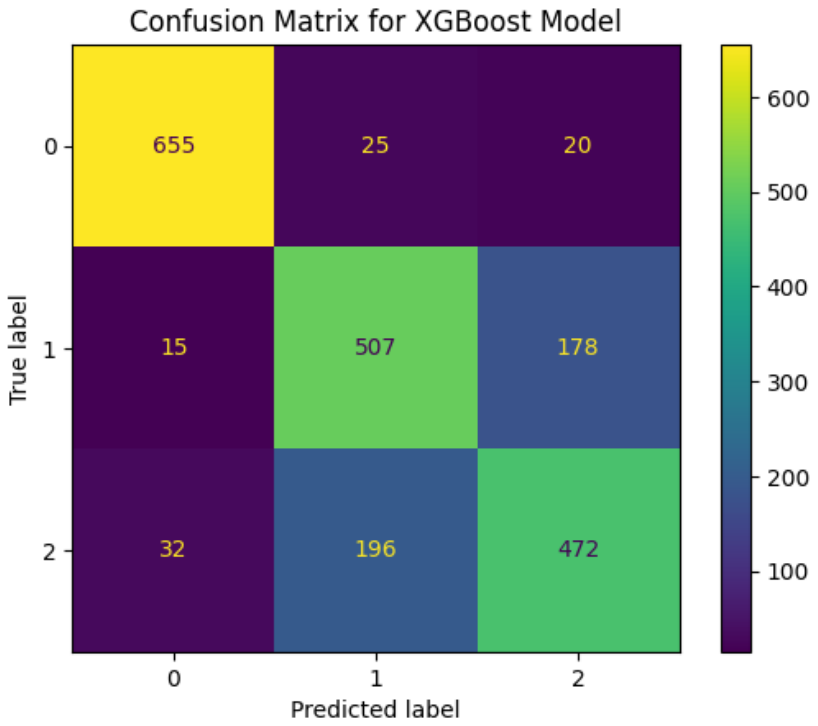

Figure 27 Confusion Matrix: XGBoost, Train: Dataset-2, Test: Dataset-2, No VRS

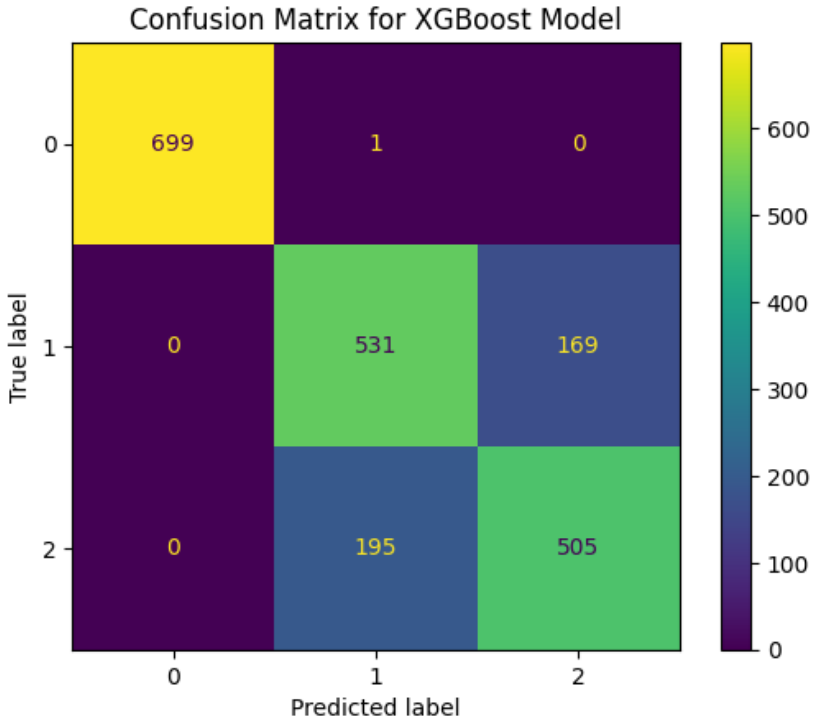

Figure 28 Confusion Matrix: XGBoost, Train: Dataset-2, Test: Dataset-2, VRS

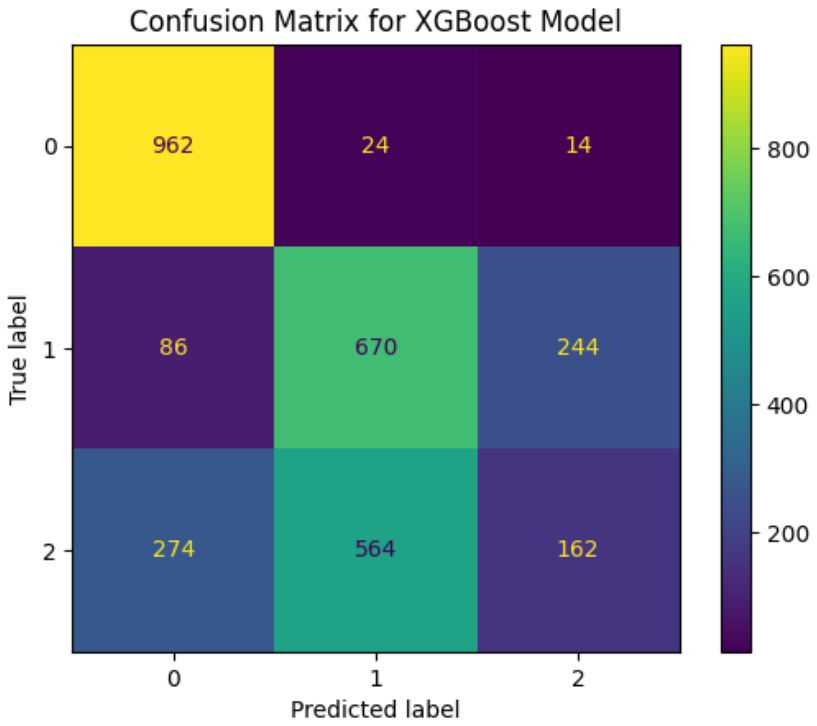

Figure 29 Confusion Matrix: XGBoost, Train: Dataset-2, Test: Dataset-1, No VRS

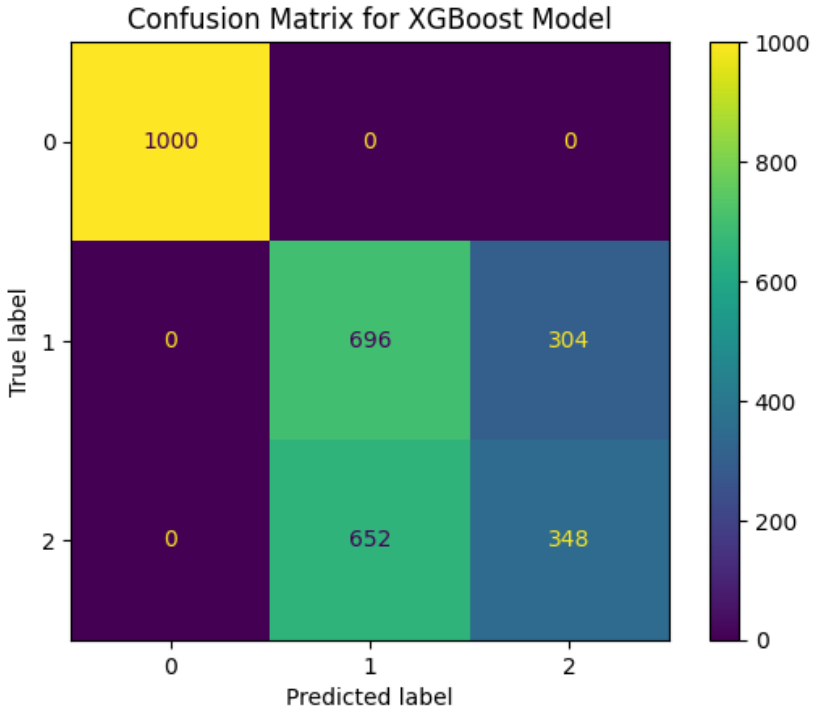

Figure 30 Confusion Matrix: XGBoost, Train: Dataset-2, Test: Dataset-1, VRS

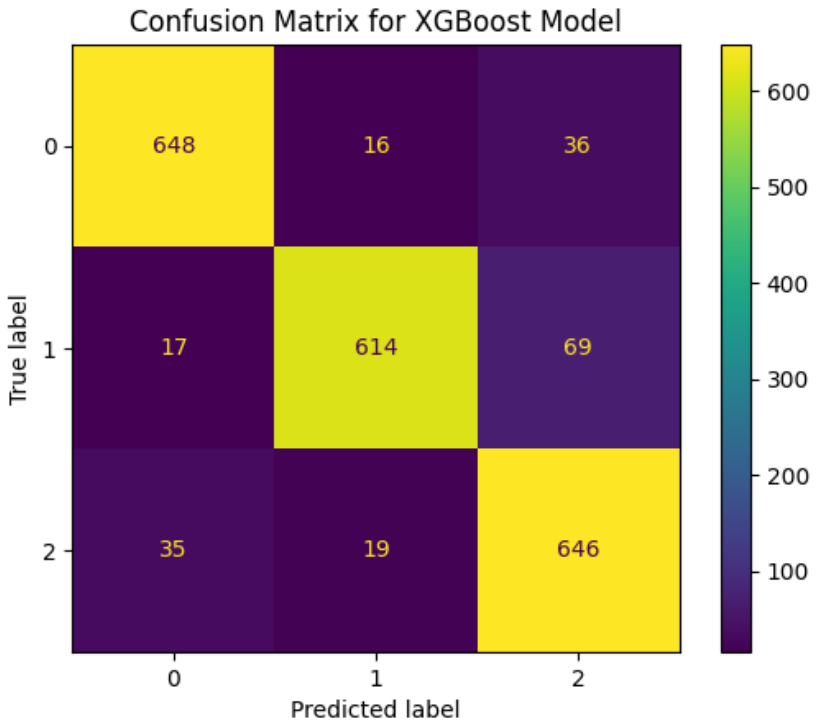

Figure 31 Confusion Matrix: XGBoost, Train: Dataset-1, Test: Dataset-1, No VRS

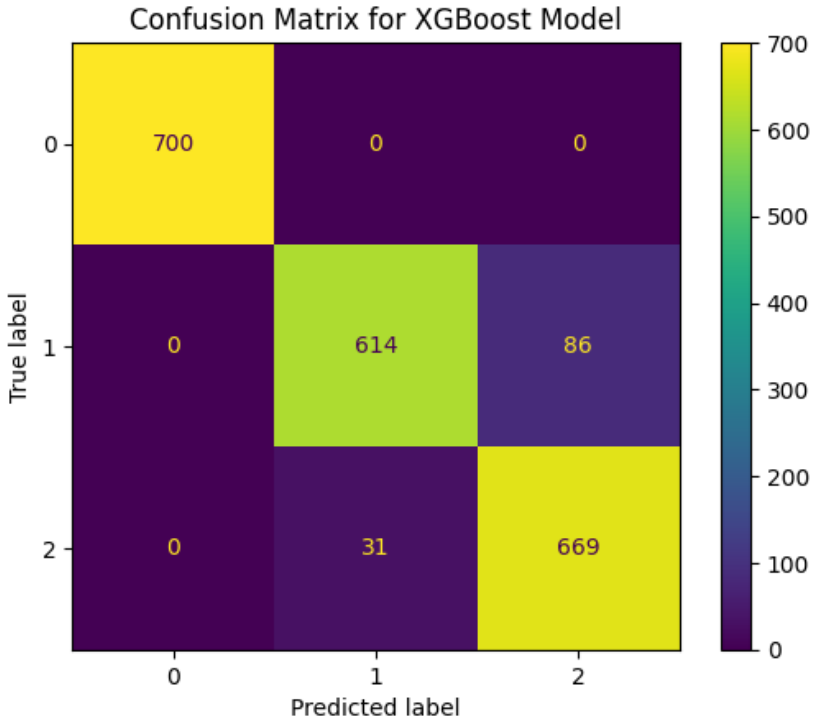

Figure 32 Confusion Matrix: XGBoost, Train: Dataset-1, Test: Dataset-1, VRS

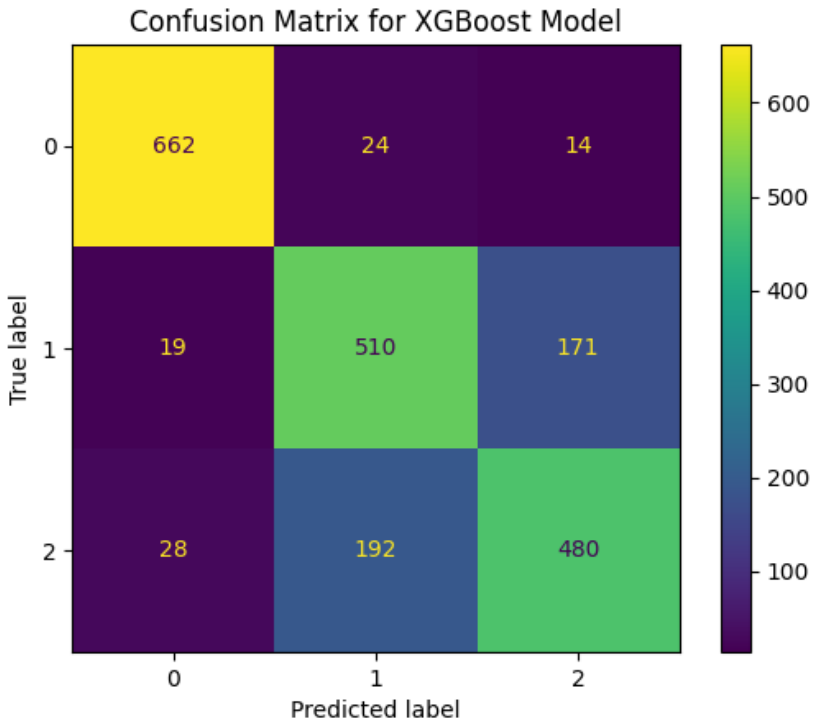

Figure 33 Confusion Matrix: XGBoost, Train: Dataset-2, Test: Dataset-2, No VRS, Normalization

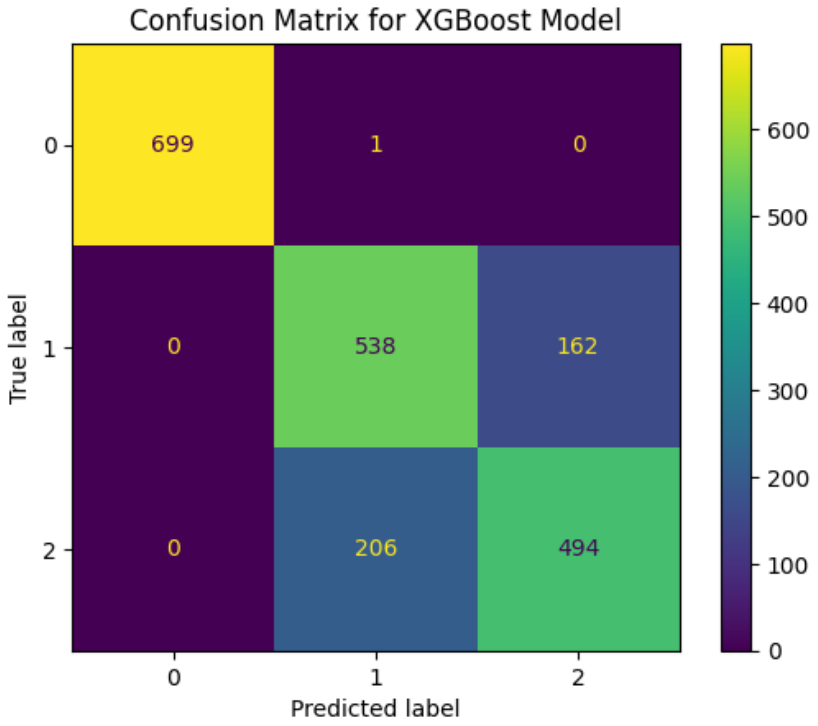

Figure 34 Confusion Matrix: XGBoost, Train: Dataset-2, Test: Dataset-2, VRS, Normalization

APPENDIX B WEIGHTS & BIASES

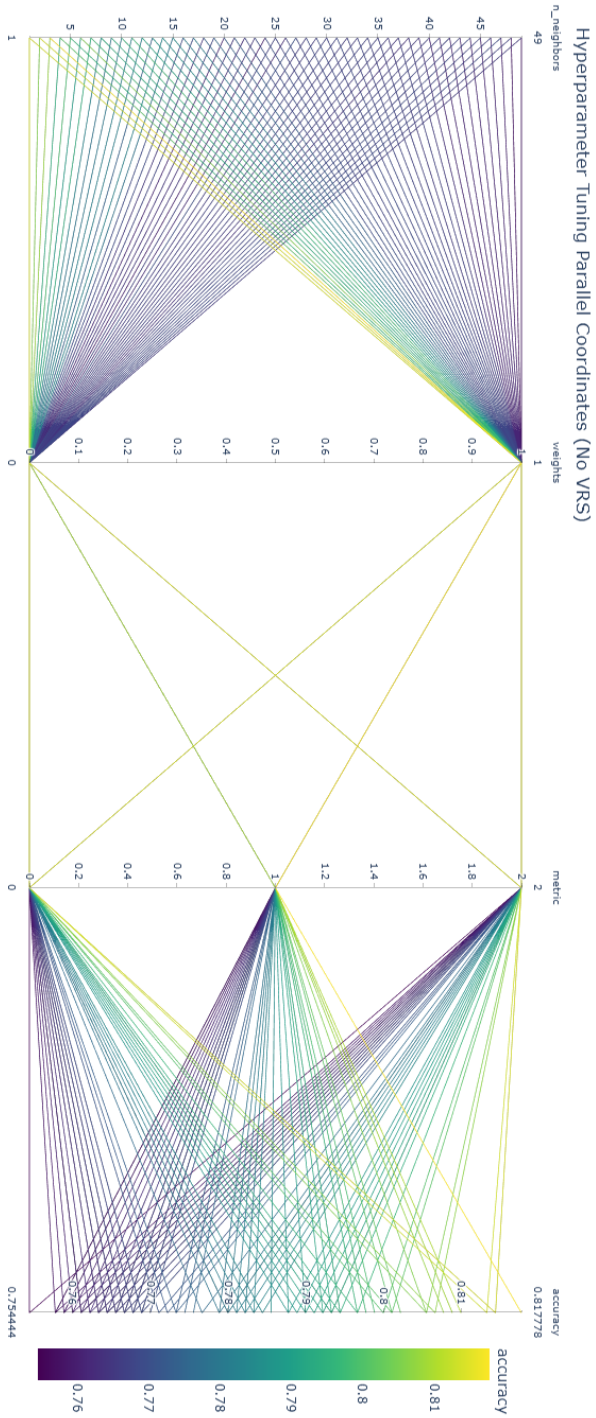

Figure 35 Hyperparameter Graph: KNN, Train: Dataset-2, Test: Dataset-2, No VRS

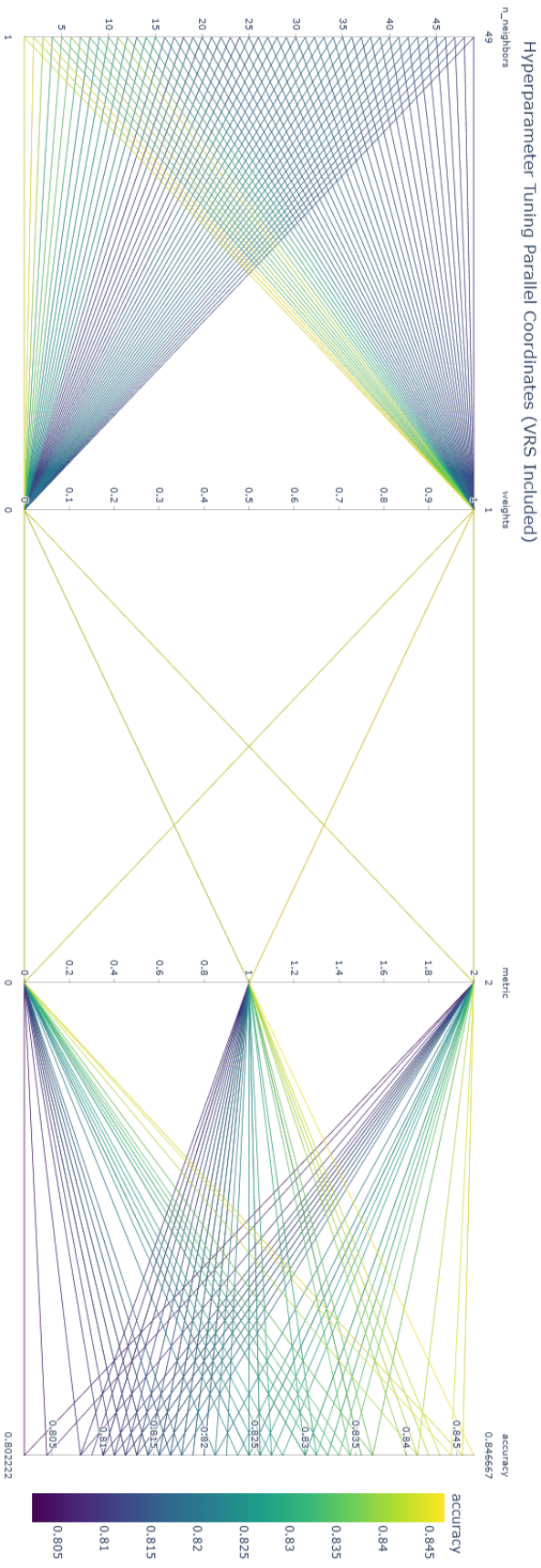

Figure 36 Hyperparameter Graph: KNN, Train: Dataset-2, Test: Dataset-2 VRS

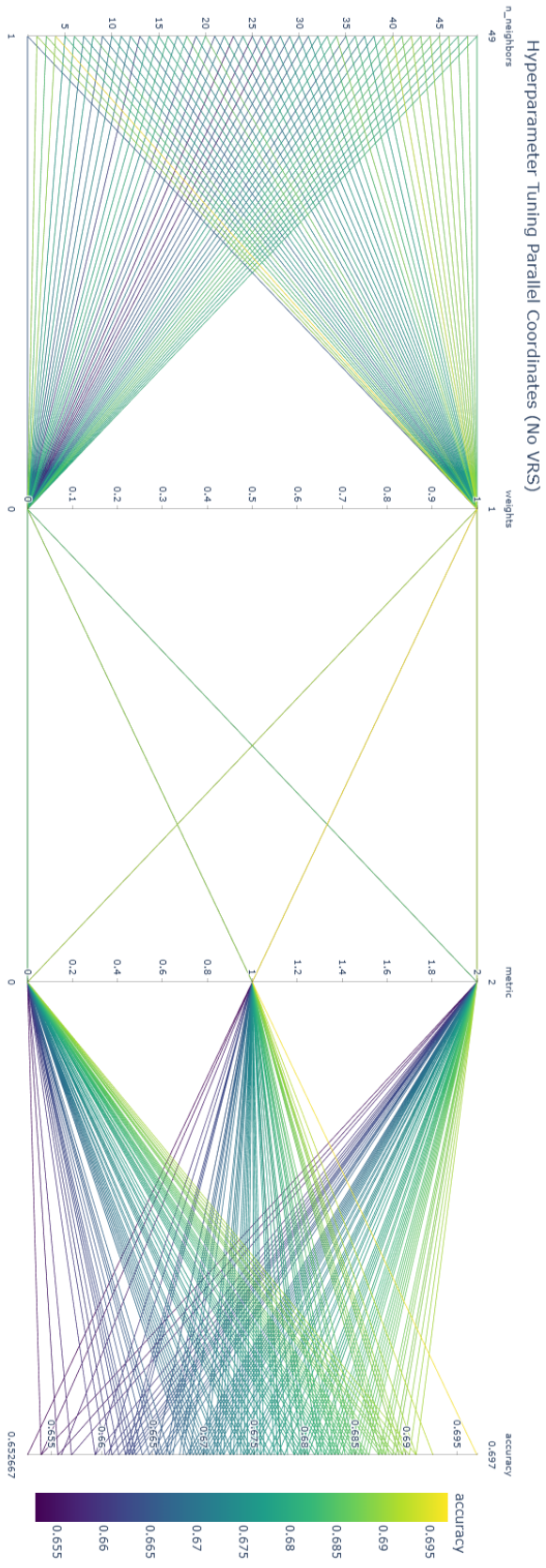

Figure 37 Hyperparameter Graph: KNN, Train: Dataset-2, Test: Dataset-1, No VRS

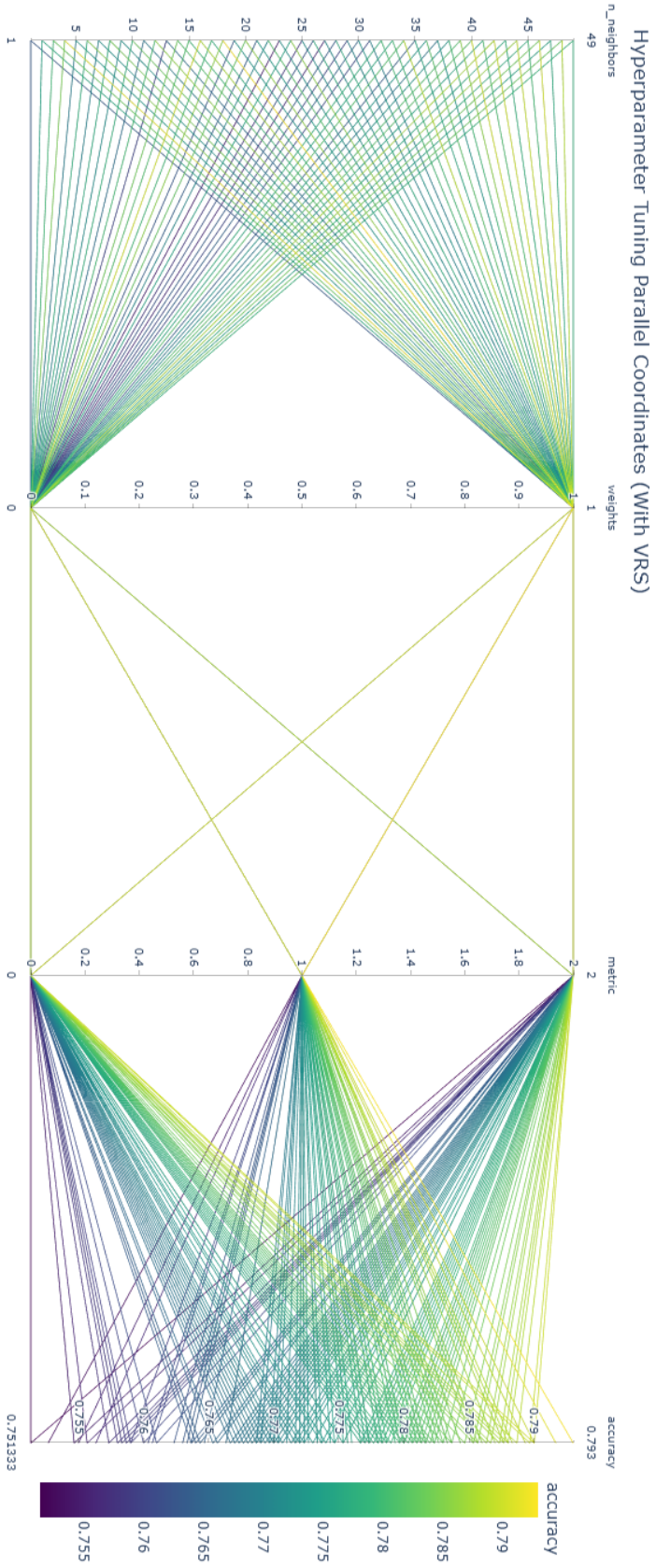

Figure 38 Hyperparameter Graph: KNN, Train: Dataset-2, Test: Dataset-1, VRS

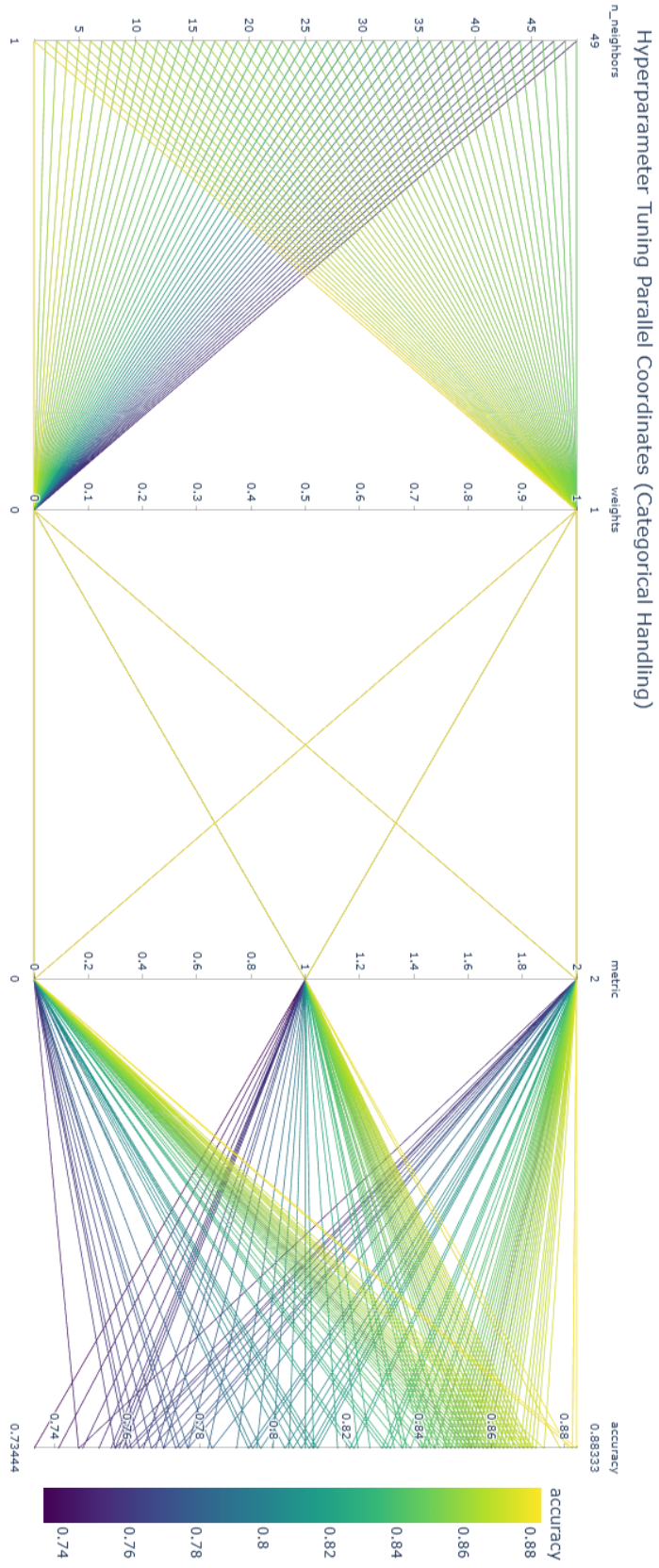

Figure 39 Hyperparameter Graph: KNN, Train: Dataset-1, Test: Dataset-1, No VRS

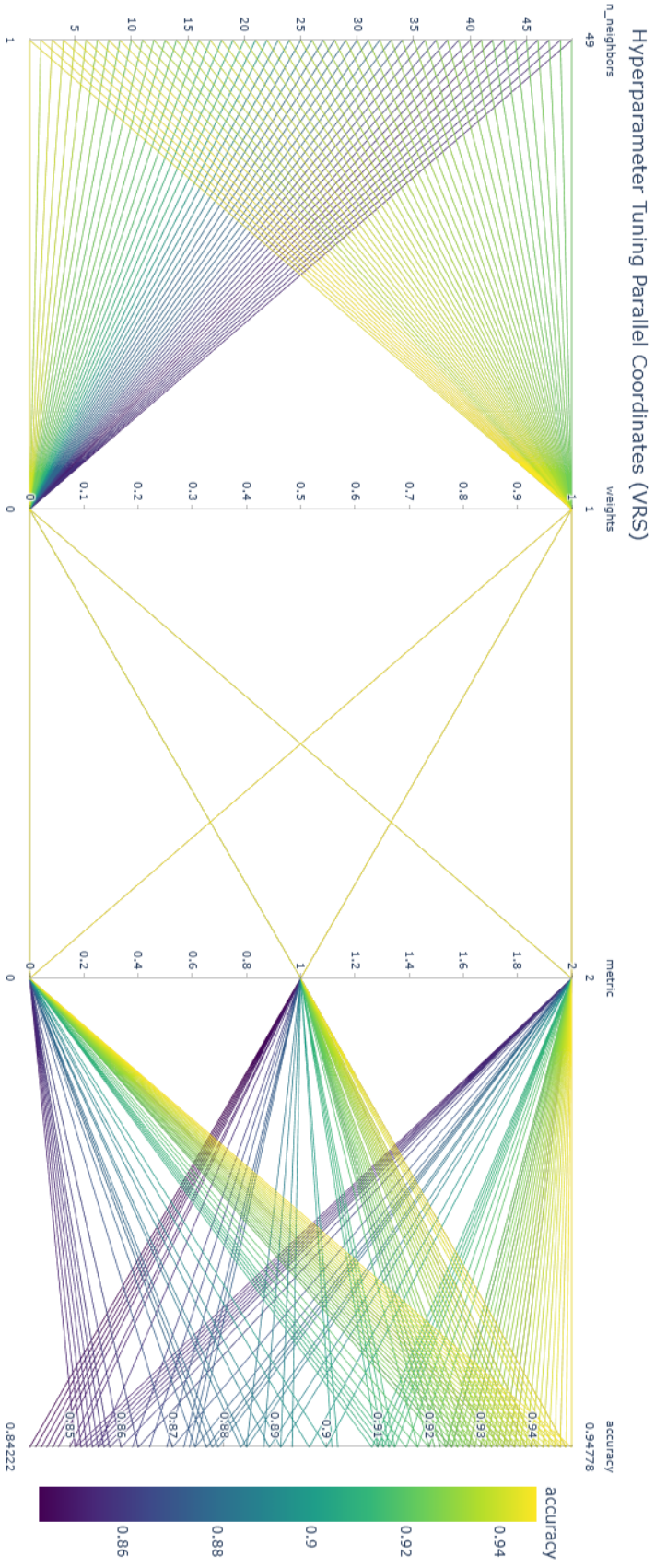

Figure 40 Hyperparameter Graph: KNN, Train: Dataset-1, Test: Dataset-1, VRS

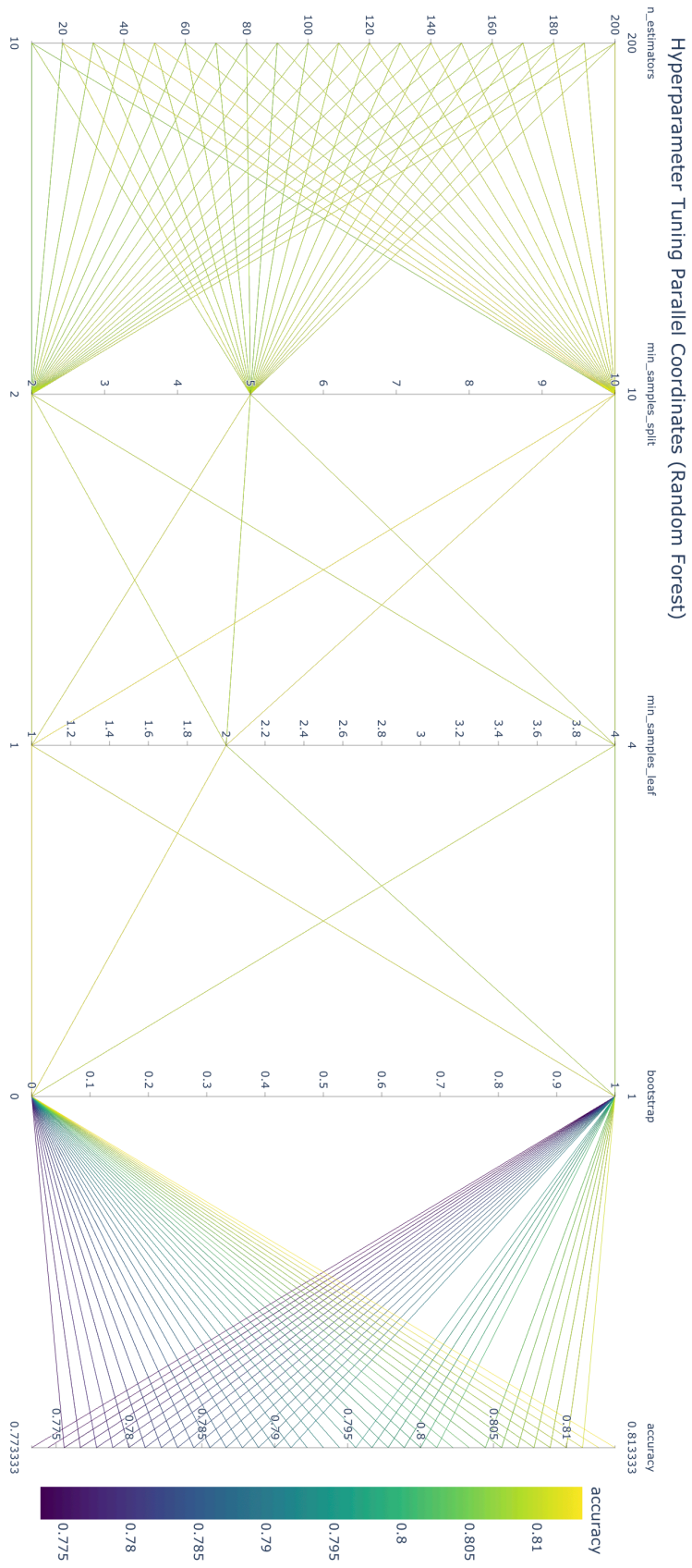

Figure 41 Hyperparameter Graph: RF, Train: Dataset-2, Test: Dataset-2, No VRS

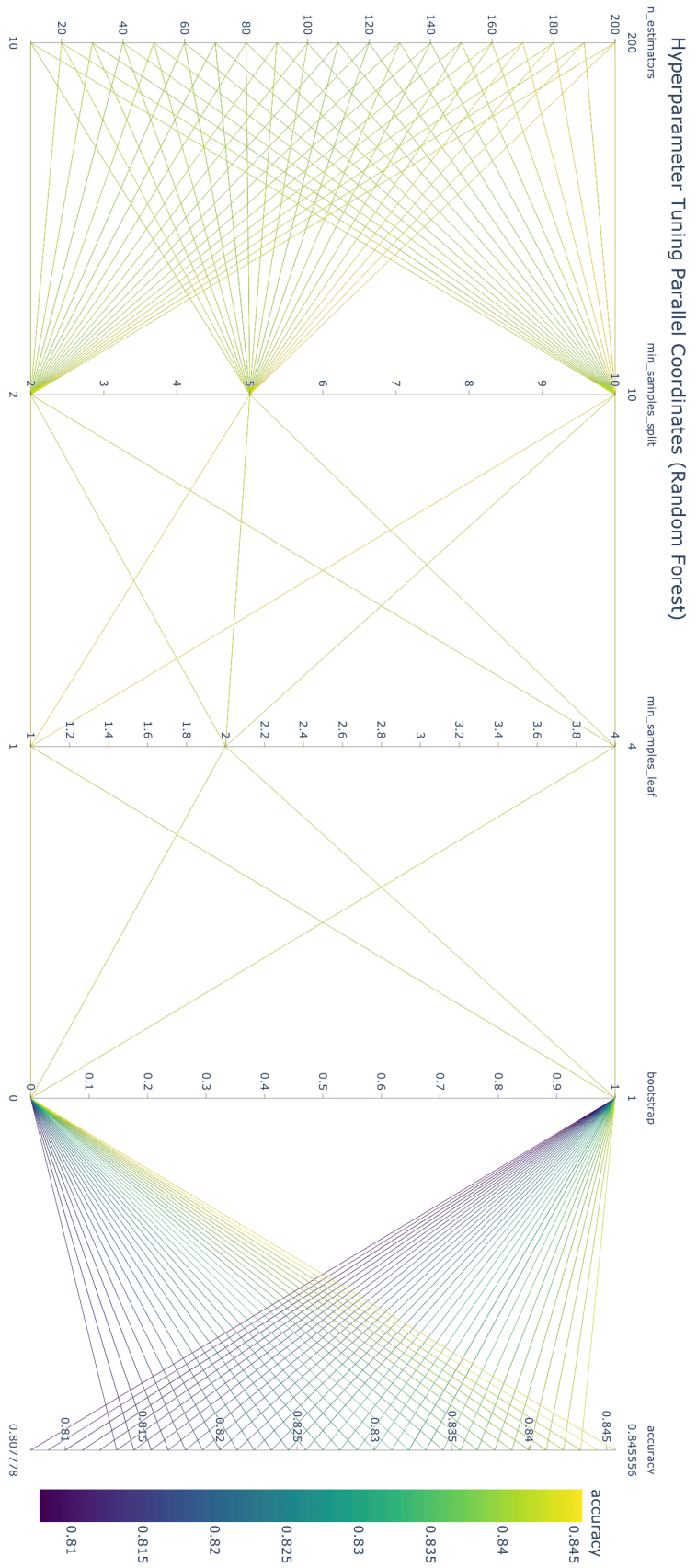

Figure 42 Hyperparameter Graph: RF, Train: Dataset-2, Test: Dataset-2, VRS

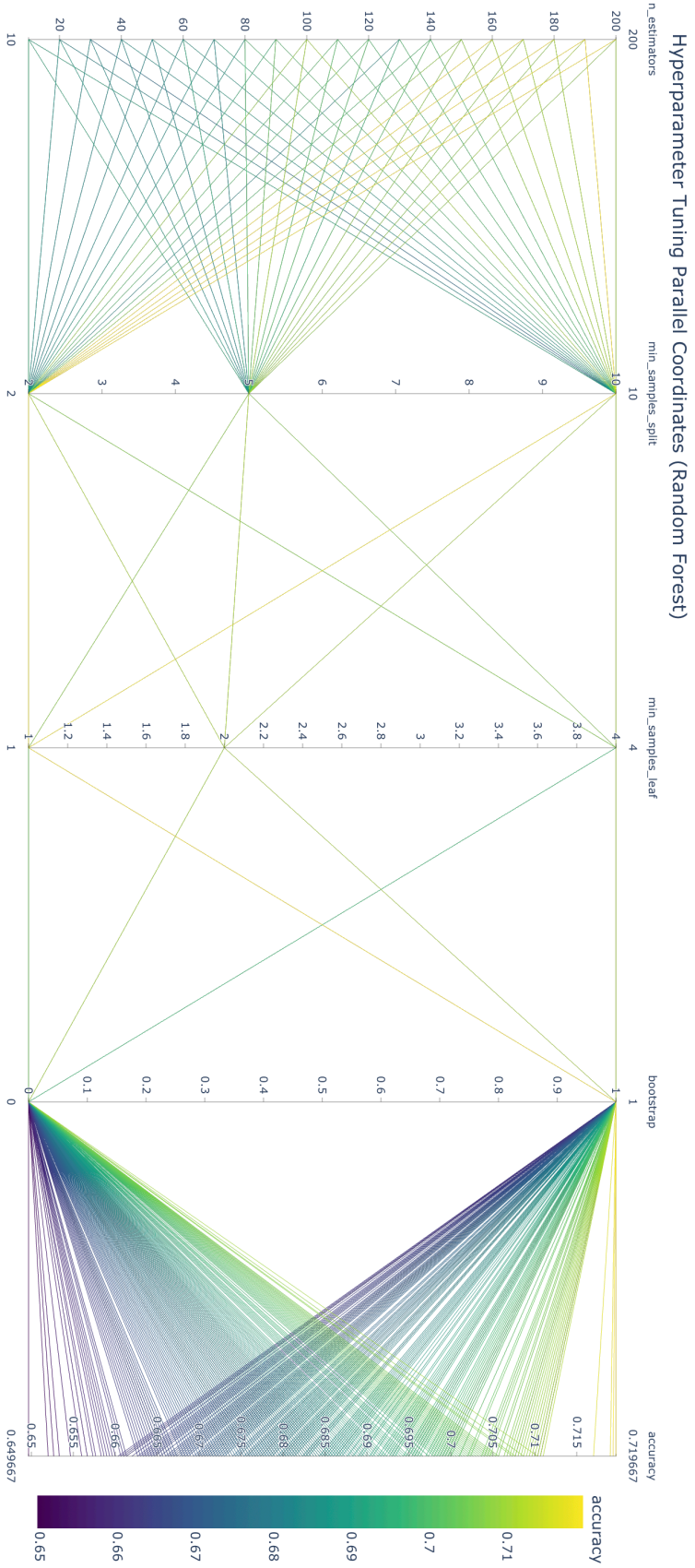

Figure 43 Hyperparameter Graph: RF, Train: Dataset-2, Test: Dataset-1, No VRS

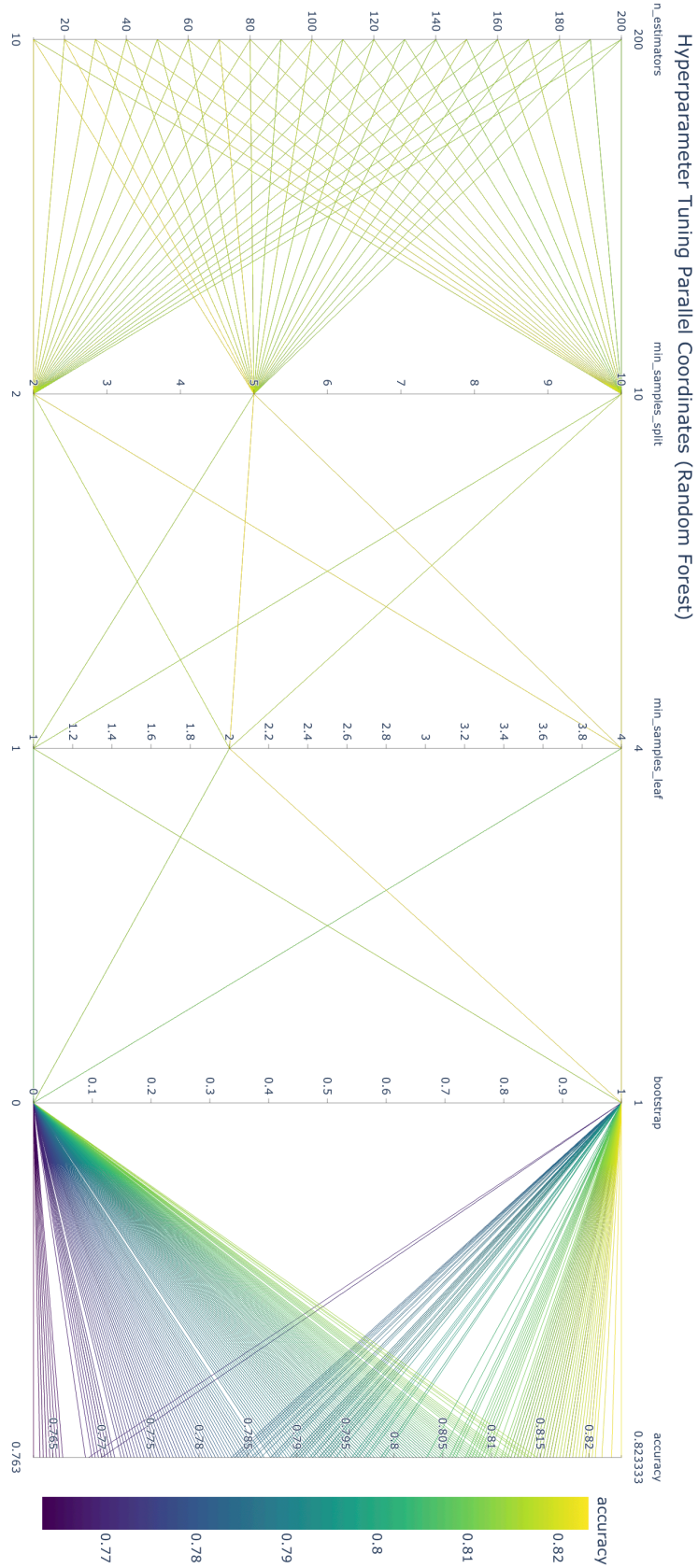

Figure 44 Hyperparameter Graph: RF, Train: Dataset-2, Test: Dataset-1, VRS

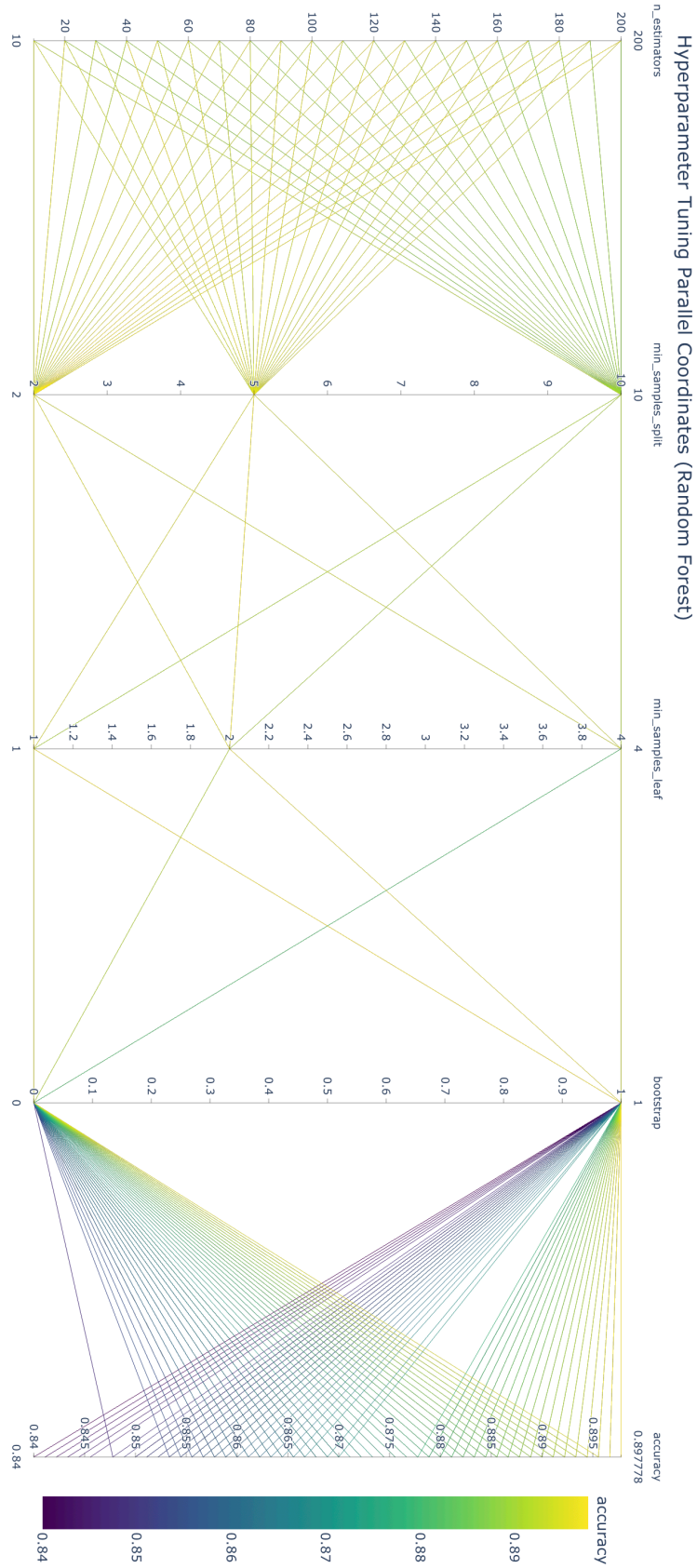

Figure 45 Hyperparameter Graph: RF, Train: Dataset-1, Test: Dataset-1, No VRS

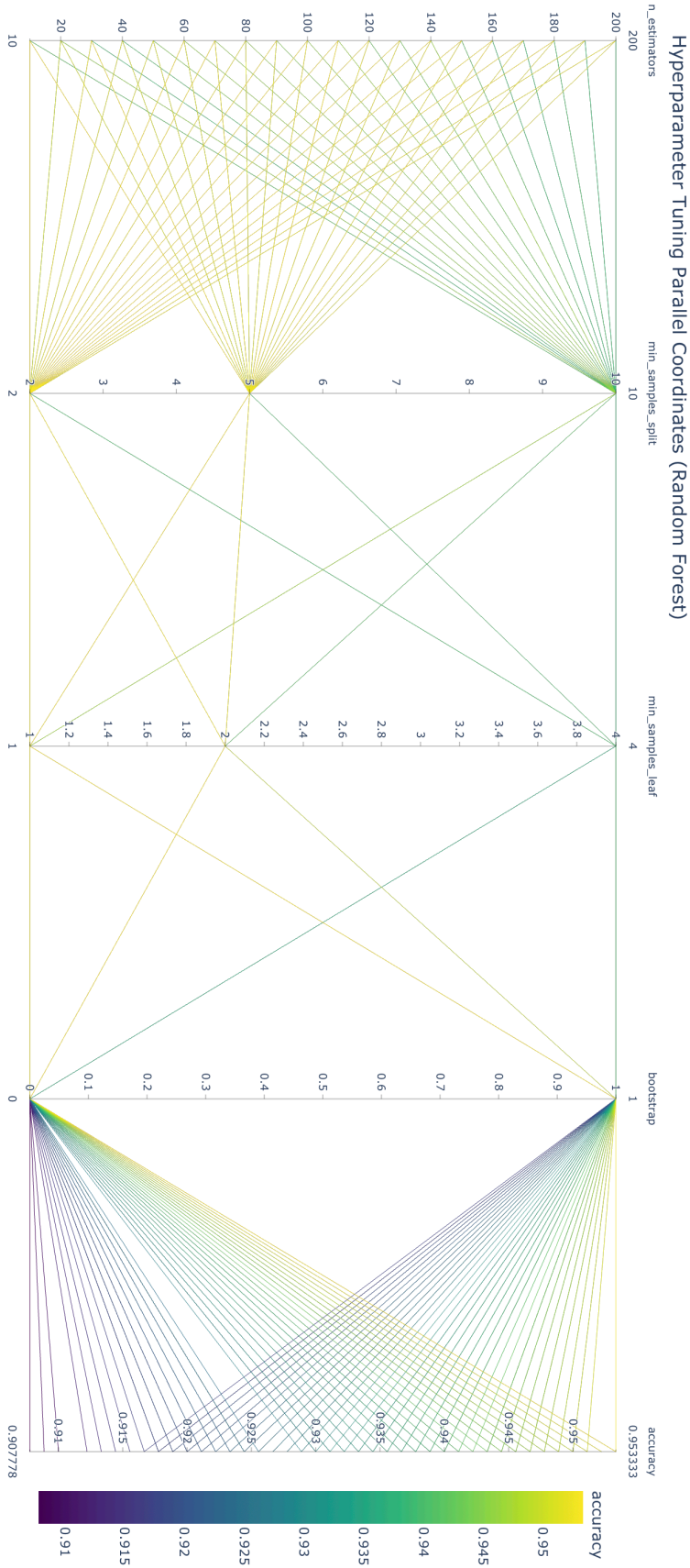

Figure 46 Hyperparameter Graph: RF, Train: Dataset-1, Test: Dataset-1, VRS

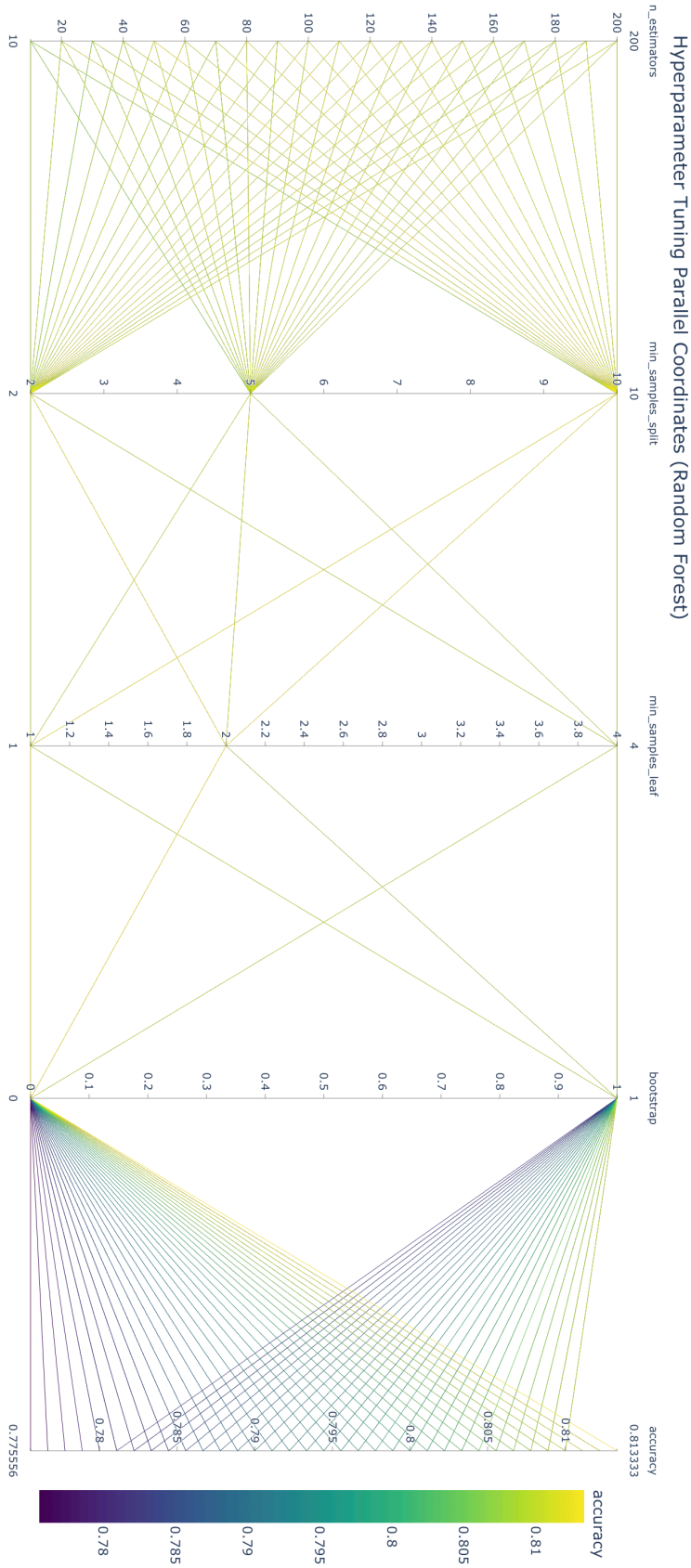

Figure 47 Hyperparameter Graph: RF, Train: Dataset-2, Test: Dataset-2, No VRS, Normalization

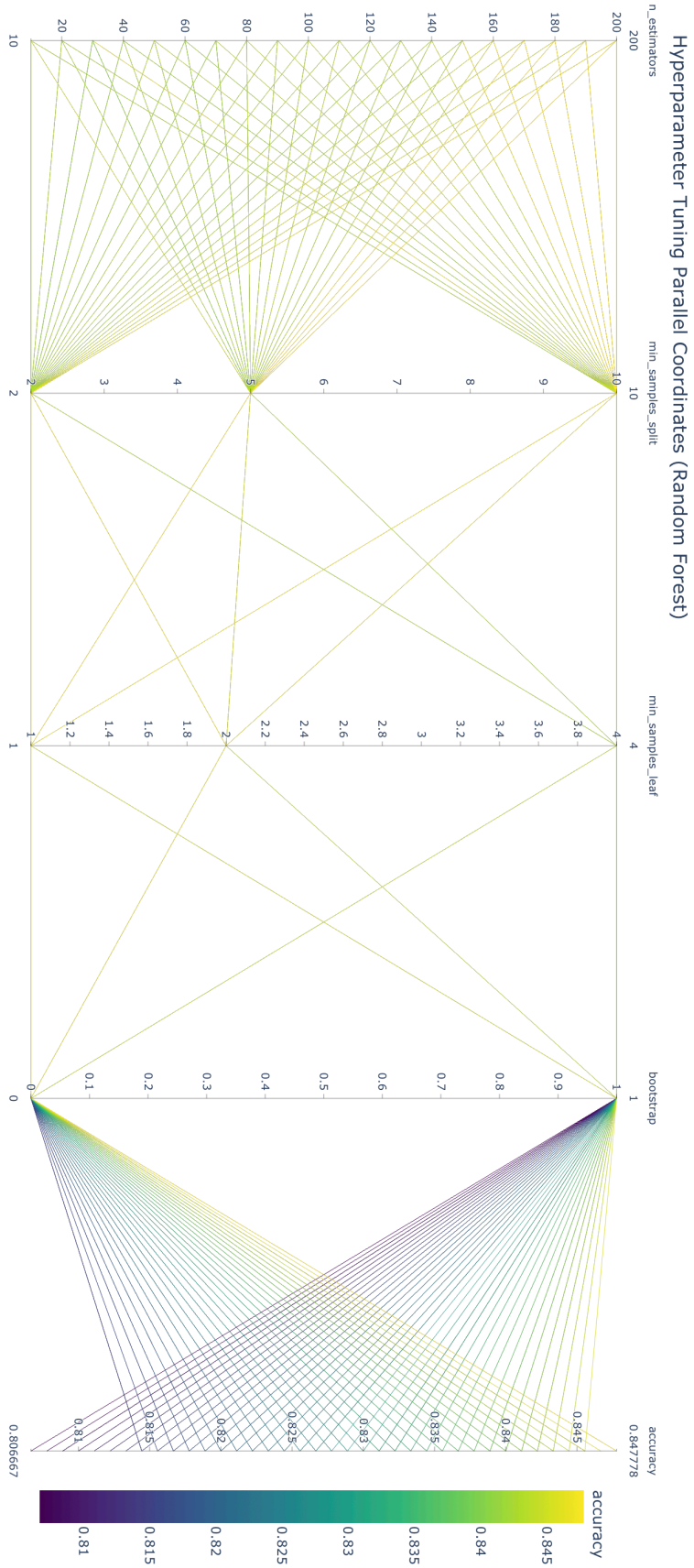

Figure 48 Hyperparameter Graph: RF, Train: Dataset-2, Test: Dataset-2, VRS, Normalization

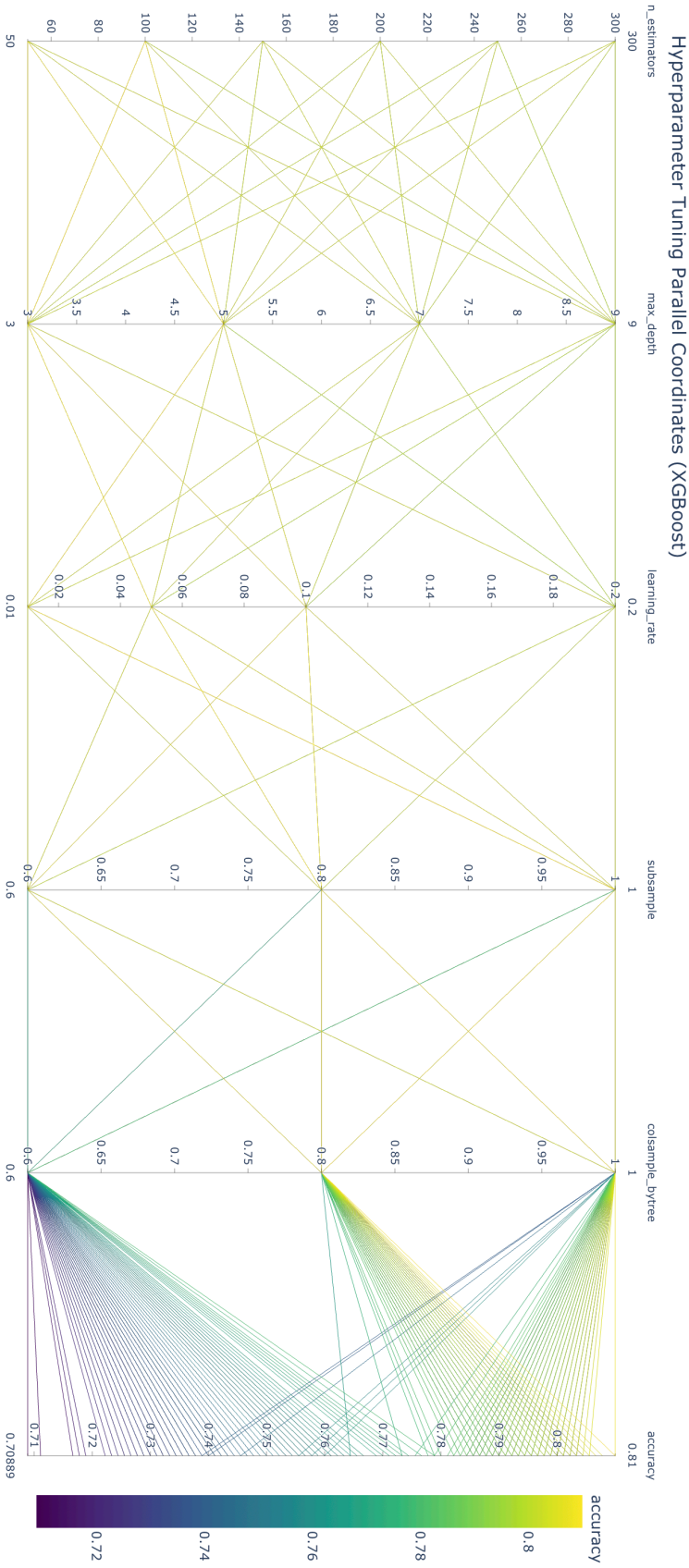

Figure 49 Hyperparameter Graph: XGBoost, Train: Dataset-2, Test: Dataset-2, No VRS

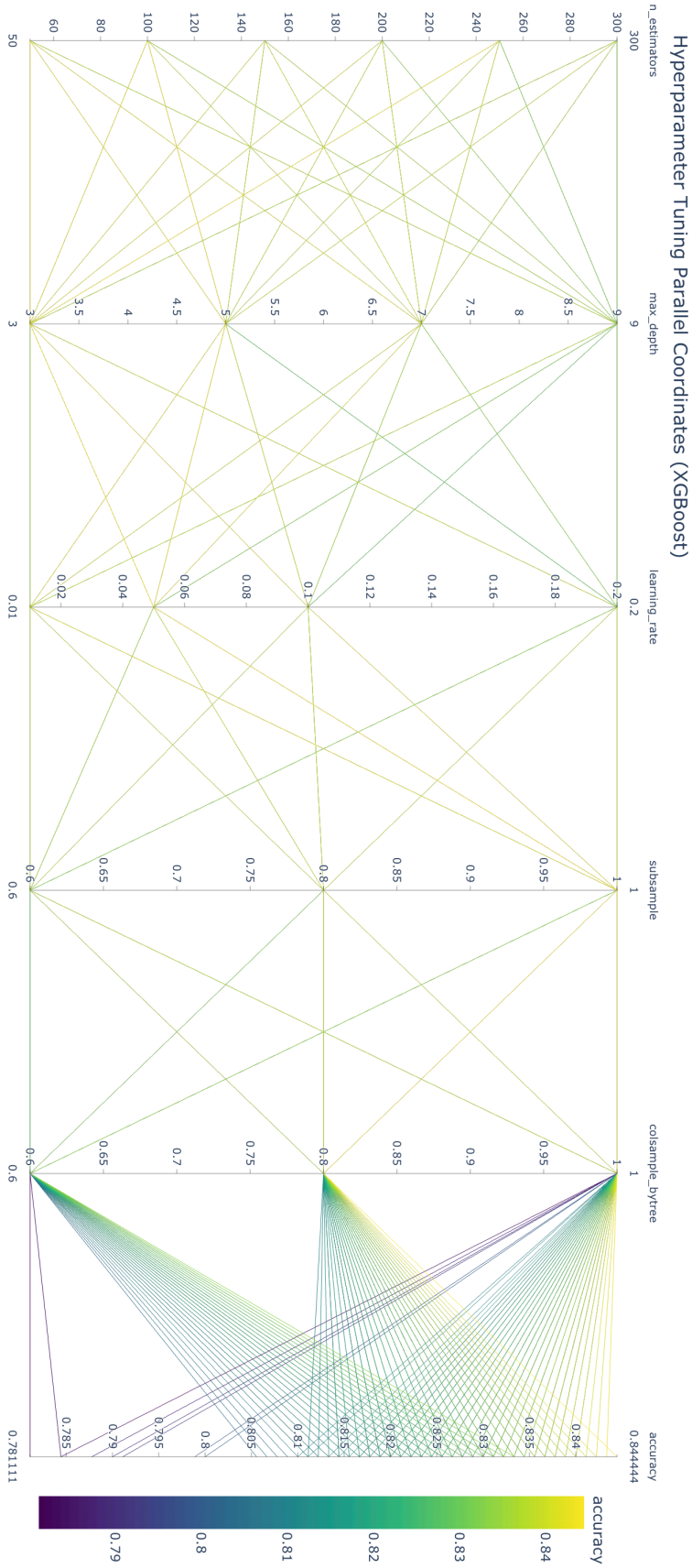

Figure 50 Hyperparameter Graph: XGBoost, Train: Dataset-2, Test: Dataset-2, VRS

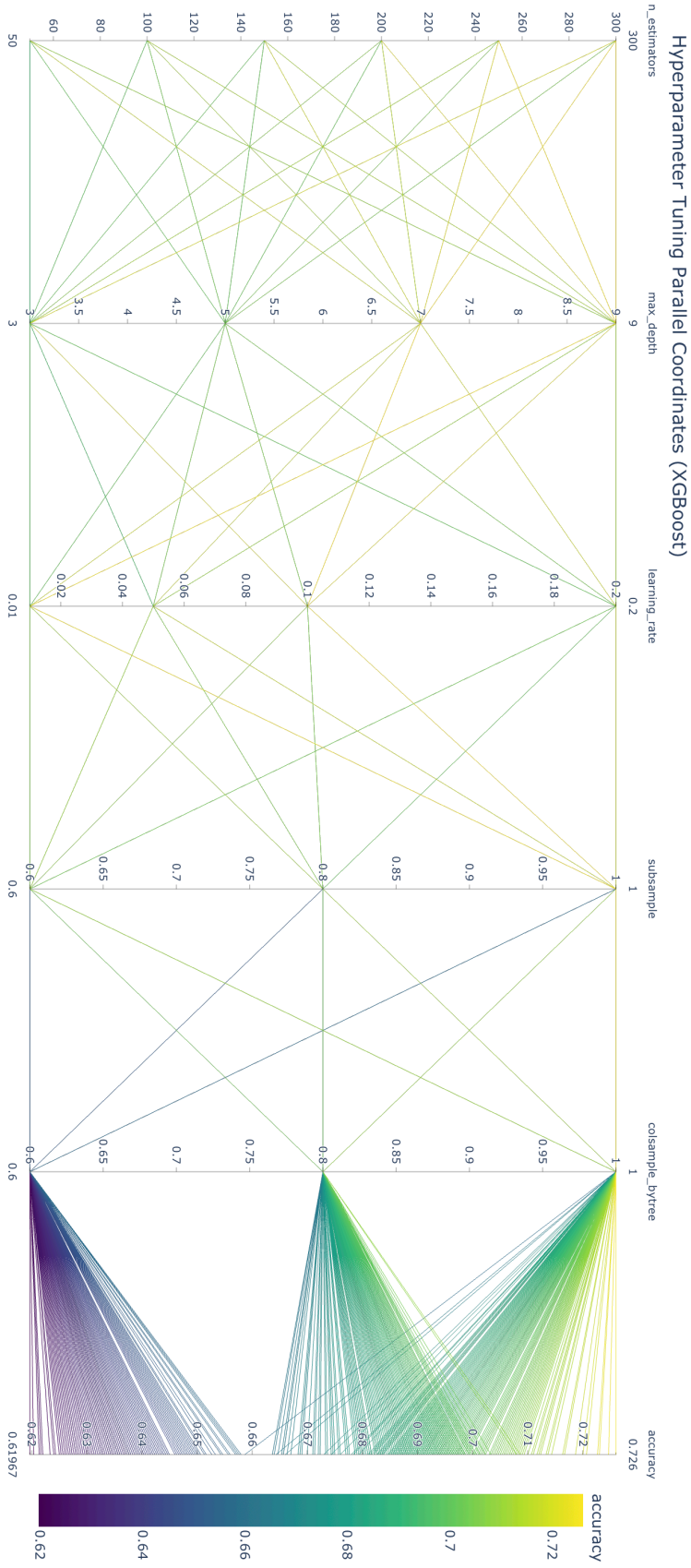

Figure 51 Hyperparameter Graph: XGBoost, Train: Dataset-2, Test: Dataset-1, No VRS

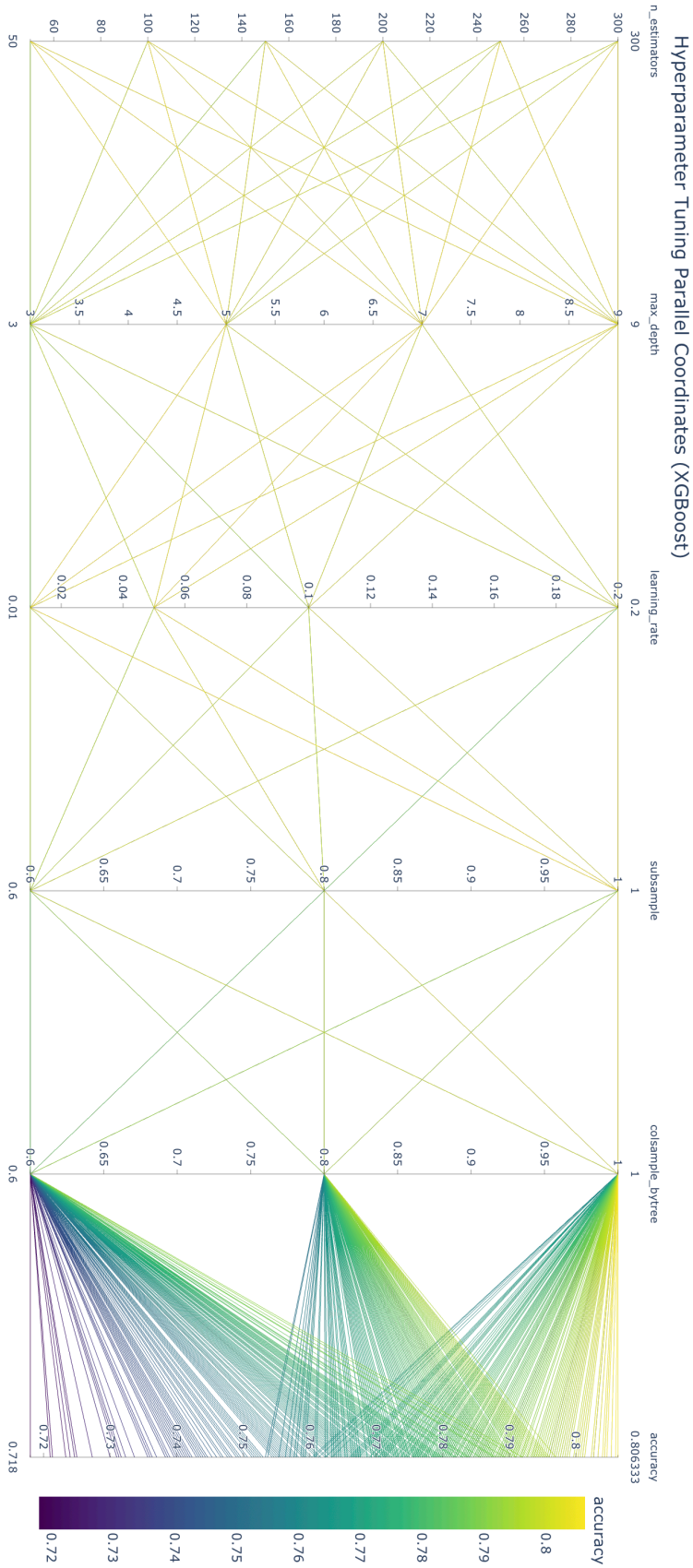

Figure 52 Hyperparameter Graph: XGBoost, Train: Dataset-2, Test: Dataset-1, VRS

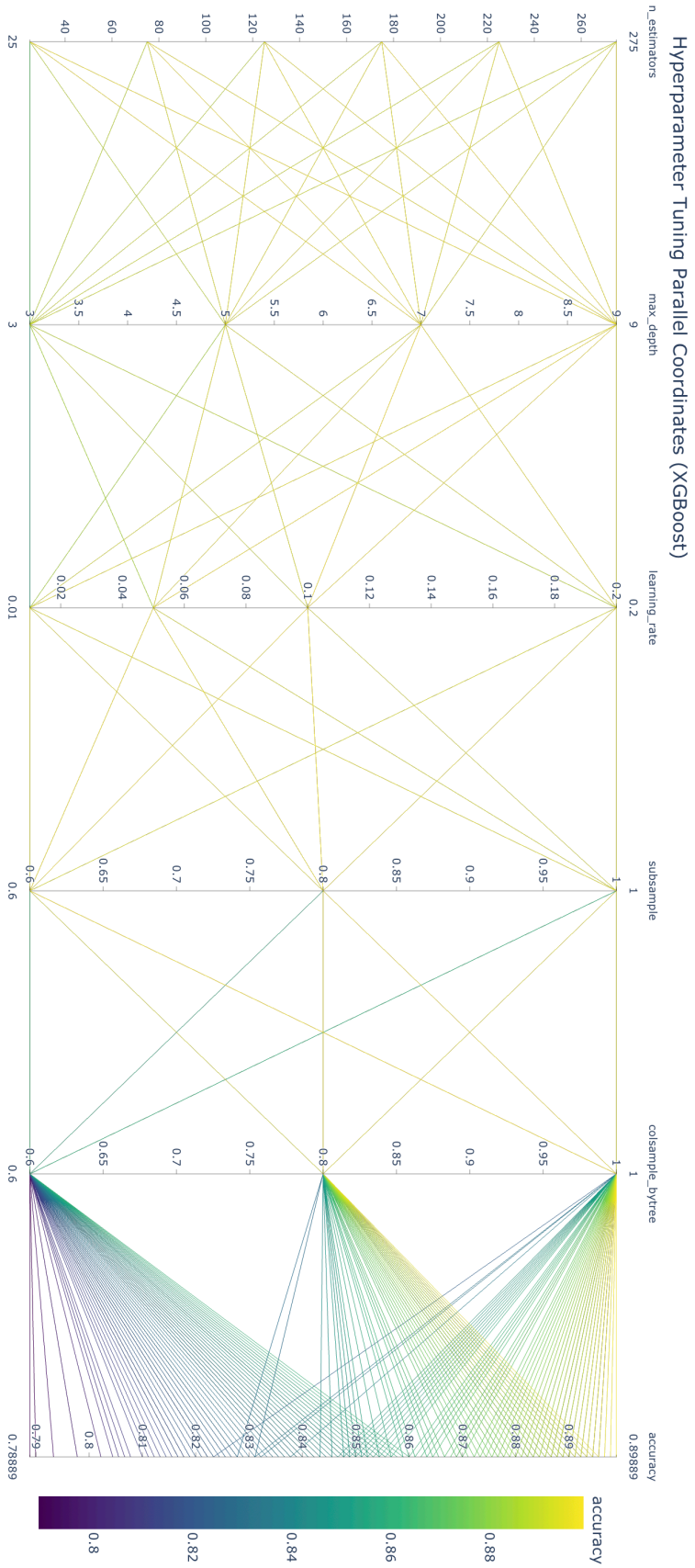

Figure 53 Hyperparameter Graph: XGBoost, Train: Dataset-1, Test: Dataset-1, No VRS

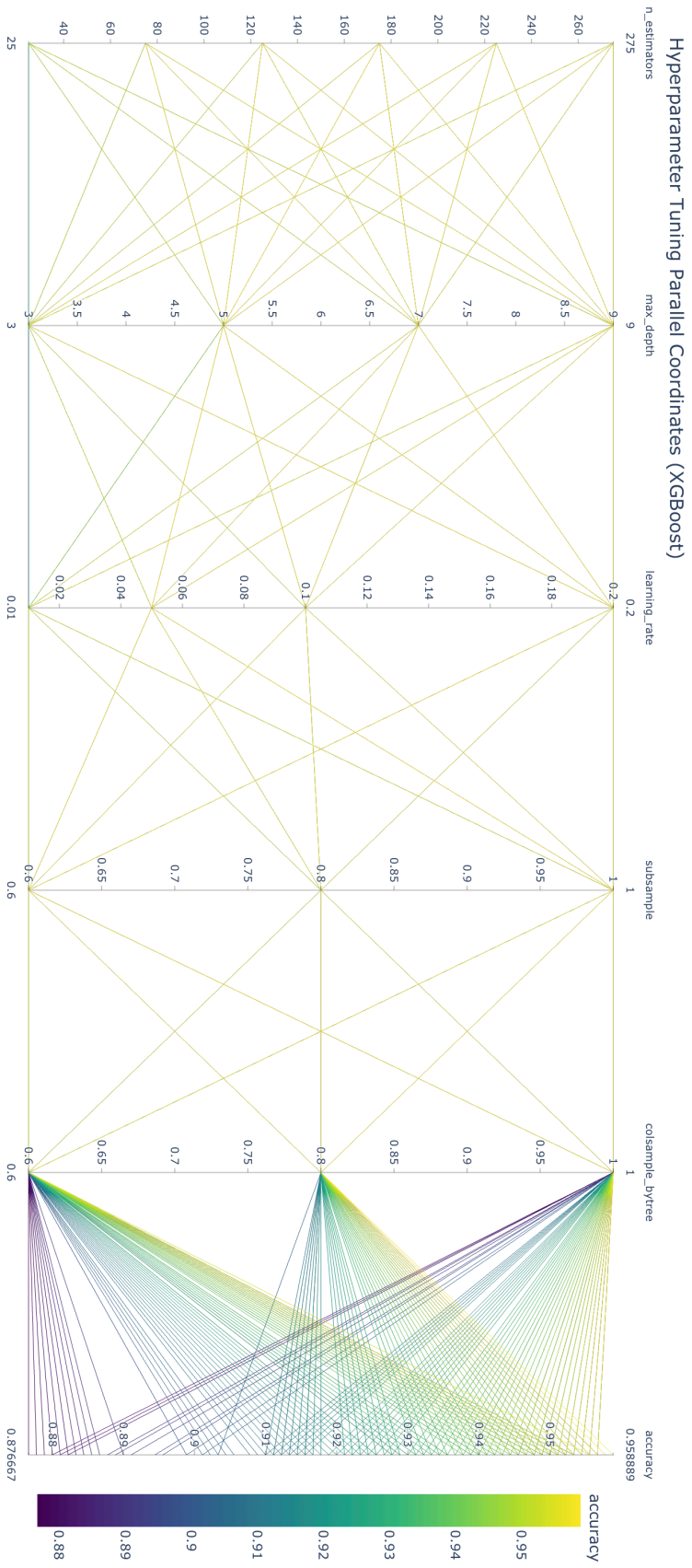

Figure 54 Hyperparameter Graph: XGBoost, Train: Dataset-1, Test: Dataset-1, VRS

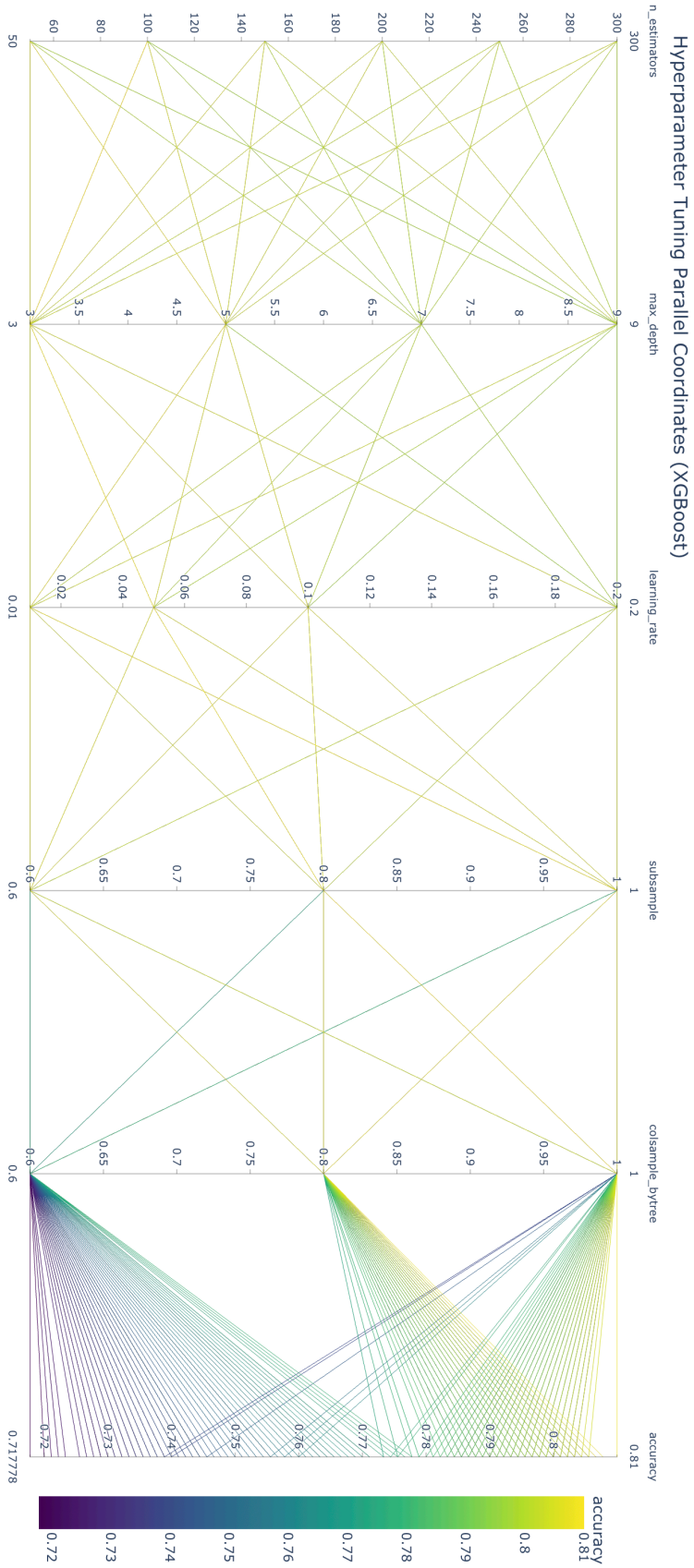

Figure 55 Hyperparameter Graph: XGBoost, Train: Dataset-2, Test: Dataset-2, No VRS, Normalization

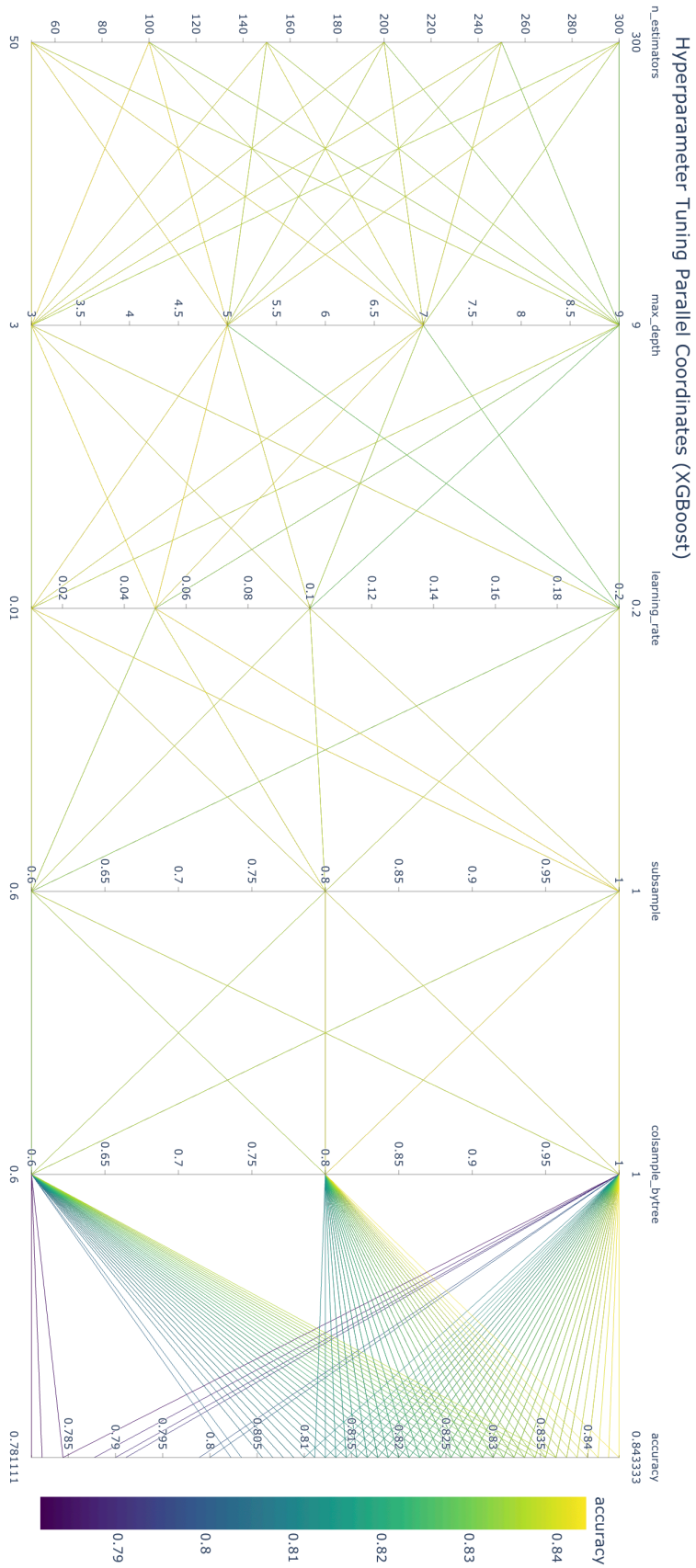

Figure 56 Hyperparameter Graph: XGBoost, Train: Dataset-2, Test: Dataset-2, VRS, Normalization