Αξιολόγηση τεχνικών μηχανικής μάθησης για πρόβλεψη ποιότητας ασύρματων ζεύξεων

Evaluation of machine learning techniques for wireless link quality prediction

ΚΩΝΣΤΑΝΤΙΝΟΣ ΓΚΙΟΥΛΗΣ

KONSTANTINOS GKIOULIS

Διπλωματική Εργασία

THESIS

Επιβλέπων: Χρήστος Λιάσκος

Supervisor: Christos Liaskos

Ιωάννινα, Σεπτέμβριος, 2024

Ioannina, September, 2024

ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

Contents:

Περίληψη

Η παρούσα διπλωματική εργασία διερευνά την εφαρμογή τεχνικών μηχανικής μάθησης για την πρόβλεψη της ποιότητας ασύρματων ζεύξεων σε πολύπλοκα περιβάλλοντα, χρησιμοποιώντας προσομοιώσεις με τη μέθοδο ray tracing. Με την αυξανόμενη ανάγκη σε συστήματα ασύρματης επικοινωνίας, η ακριβής πρόβλεψη μετρήσεων ποιότητας σήματος όπως το Bit Error Rate (BER), οι διασπορές καθυστέρησης (delay spreads) και τα προφίλ καθυστέρησης ισχύος (PDP) είναι κρίσιμη για την ανάπτυξη δικτύων επόμενης γενιάς. Τα παραδοσιακά εμπειρικά μοντέλα συχνά αδυνατούν να λάβουν υπόψη την πολυπλοκότητα αστικών ή εσωτερικών περιβαλλόντων, όπου οι ανακλάσεις, οι διαθλάσεις και οι σκεδάσεις επηρεάζουν την διάδοση του σήματος. Η διπλωματική αυτή εργασία αξιολογεί την αποτελεσματικότητα των μοντέλων μηχανικής μάθησης, που έχουν εκπαιδευτεί με δεδομένα τα οποία έχουν ληφθεί με προσομοιώσεις χρησιμοποιώντας ray tracing από το MATLAB για την πρόβλεψη της ποιότητας ασύρματων ζεύξεων σε συστήματα πολλαπλής εισόδου και εξόδου (MIMO) με τεχνολογία ορθογώνιας πολυπλεξίας διαίρεσης συχνότητας (OFDM).

Χρησιμοποιήθηκαν διάφοροι αλγόριθμοι μηχανικής μάθησης μέσω του AutoGluon για την αυτοματοποίηση της δημιουργίας μοντέλων πρόβλεψης. Τα μοντέλα αυτά δοκιμάστηκαν σε μια σειρά από σενάρια, εστιάζοντας στην ακρίβεια μετρήσεων, όπως στη διασπορά καθυστέρησης RMS, το BER, την απώλεια διαδρομής (path loss), το πλάτος και το προφίλ καθυστέρησης ισχύος (PDP) σε σχέση με τα αρχικά δεδομένα της προσομοίωσης. Τα αποτελέσματα δείχνουν ότι τα μοντέλα μηχανικής μάθησης μπορούν να παρέχουν πολύ ακριβείς προβλέψεις για μετρήσεις ασύρματων ζεύξεων, ενώ μειώνουν σημαντικά τον χρόνο υπολογισμού σε σύγκριση με τις παραδοσιακές μεθόδους προσομοίωσης. Αυτή η προσέγγιση έχει εφαρμογές στον σχεδιασμό και τη βελτιστοποίηση δικτύων, ιδίως σε σενάρια που απαιτούν ταχείες και αξιόπιστες προβλέψεις χωρίς εκτεταμένες προσομοιώσεις.

Abstract

This thesis explores the application of machine learning techniques to predict wireless link quality in complex environments, using ray tracing simulations. With the increasing reliance on wireless communication systems, accurate predictions of signal quality metrics such as Bit Error Rate (BER), delay spreads and power delay profiles (PDP) are critical for the development of next-generation networks. Traditional empirical models often fail to account for the complexities of urban or indoor environments where reflections, diffraction, and scattering influence signal propagation. This thesis evaluates the effectiveness of using machine learning models trained with simulation data from MATLAB's ray tracing tool in predicting wireless link quality in multiple-input and multiple-output (MIMO) orthogonal frequency-division multiplexing (OFDM) systems.

A variety of machine learning algorithms were employed via AutoGluon to automate the creation of predictive models. These models were tested across a range of scenarios, focusing on the accuracy of metrics such as RMS delay spread, BER, path loss, amplitude and PDP over the simulation data. The results show that machine learning models can provide highly accurate predictions of wireless link metrics while significantly reducing computational time compared to traditional simulation methods. This approach has potential applications in network planning and optimization, particularly in scenarios requiring rapid, reliable predictions without extensive simulations.

Chapter 1: Introduction

Wireless communication has become essential in all aspects of society, facilitating seamless connectivity and real-time interaction, from smartphones and laptops to metropolitan networks and satellite networks. The continuously increasing reliance on wireless systems and the demand for better systems to facilitate new innovations requires a corresponding ever-growing need for development in the sector of telecommunications and wireless communications. However, there are obstacles in this continuous development of wireless communication, because of physical constraints in the environment where signal interference, multipath propagation and complex electromagnetic interactions degrade the quality of wireless links making the process of simulating such systems extremely complex. We used an approach of predicting the wireless link quality with machine learning (ML) techniques which can model complex, nonlinear relationships between variables, because of the ray tracing simulation used to train them, that traditional empirical or theoretical models struggle to do. The goal is to evaluate machine learning algorithms in the context of wireless link quality prediction using ray tracing techniques, with a particular focus on multiple-input and multiple-output (MIMO) systems and orthogonal frequency-division multiplexing (OFDM) technology. Traditional wireless link quality prediction models, including empirical models like the Okumura-Hata model, focus on predicting path loss, signal degradation and propagation delay but are limited in terms of accuracy and applicability to specific environments. These models often fail to account for the complexities of real-world scenarios, particularly in indoor environments or urban areas where reflections, diffraction, and scattering significantly impact signal quality. In recent years, ray tracing is used to simulate the propagation of electromagnetic waves in complex environments. Ray tracing methods calculate the multiple paths that a signal may take as it reflects, diffracts, and scatters off objects in its environment and provide accurate representations of how these signals propagate in 3D environments. Because ray tracing simulates rays propagating through 3D environments by reflecting, diffracting, and especially scattering off objects, it is computationally intensive. That is the reason machine learning was used. To provide results fast and

reliably. The primary objective was to create machine learning models which predict wireless link quality, specifically focusing on metrics such as Bit Error Rate (BER), delay spreads, and power delay profiles (PDP) and assess their accuracy and time efficiency . Using MATLAB's ray tracing capabilities, this study simulates wireless communication links in an indoor environment using MIMO-OFDM systems [1]. The simulation generates data on various aspects of signal quality, such as path loss, propagation delay, for a number of rays in different receiver and transmitter positions. After the data extraction by the simulation, the data was used to train ML algorithms. To do that we used AutoGluon, an open source tool which automates the process of creating accurate ML models. Once trained, the machine learning models were used to predict metrics based on new transmitter and receiver positions. The accuracy of the machine learning models was evaluated by comparing the predicted results with the actual simulation outputs.

The ability to accurately predict wireless link quality is critical for the design and deployment of next-generation wireless networks as systems become more complex and the needs for reliability increase. These predictive models can estimate signal performance without the need for extensive real-time simulations or empirical measurements have the potential to significantly reduce the time and cost associated with network planning and optimization.

Chapter 2: Wireless link quality prediction with ray tracing techniques

In this chapter we will present a brief history of electromagnetism , wireless transmission, ray tracing, multiple-input multiple-output (MIMO) technology and wireless communications in general. The chapter will analyse the Ray tracing techniques that are used in indoor MIMO communications and will assess the quality of their results.

2.1 History

2.1.1 Electromagnetism

The origins of wireless communication are rooted in the study of electricity and magnetism, which began its contemporary appearance in the late 18th century. Key discoveries in electromagnetism, started from Benjamin Franklin's electricity kite experiments and Alessandro Volta's invention of the battery in 1800. In 1820, Danish physicist Hans Christian Ørsted demonstrated that electric currents produce magnetic fields, he made that observation by using a compass needle, which was deflected at right angles by a current-carrying wire thus linking electricity and magnetism for the first time.Later, physicists such as Michael Faraday, Andre-Marie Ampere, and Joseph Henry tried to define the mathematical relationship between electricity and magnetism. However, the formal unification of electricity and magnetism came through James Clerk Maxwell, whose equations, presented between 1861 and 1864, are now the cornerstone of electromagnetic theory [6],[9].

2.1.2 The Hertz Transmitter

Building on Maxwell's theory, the German physicist Heinrich Hertz was the first to physically demonstrate the existence of electromagnetic waves. Between 1885 and 1889, Hertz conducted a series of experiments using a transmitter consisting of a Ruhmkorff-type coil connected to a centred-fed dipole equipped with a spark gap (Image 1) .

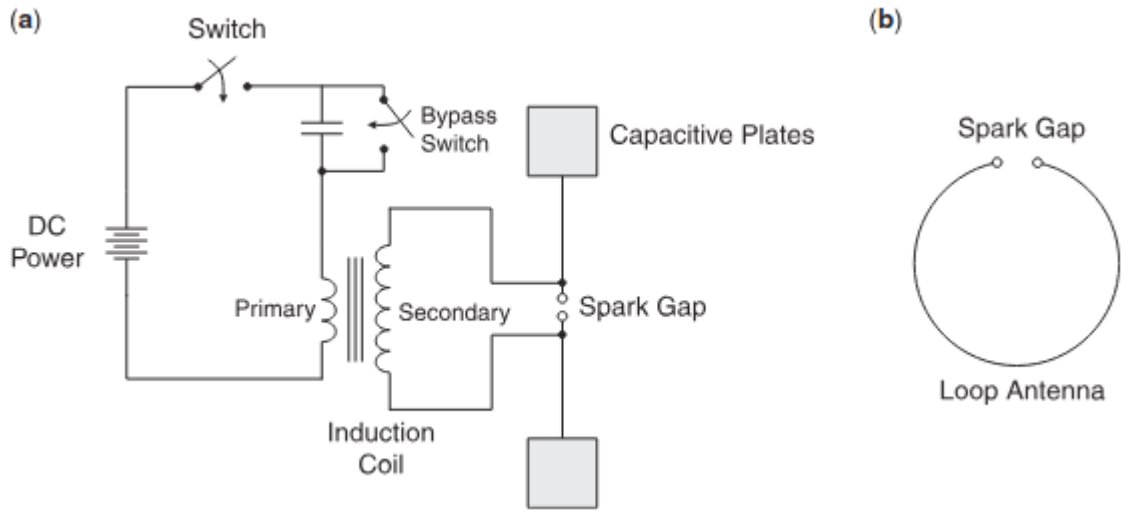


Image 1. Hertz's spark-gap experimental setup showing (a) transmitter and (b) receiver.

Hertz successfully generated and detected electromagnetic waves, proving they travel at the speed of light. This discovery not only confirmed Maxwell's theory but also laid the groundwork for practical wireless communication systems [6],[9].

2.1.3 Tesla and Wireless Power

Nikola Tesla, took Hertz's discovery further by exploring the transmission of wireless power. In 1891, he used alternators to wirelessly power phosphorus-lined vacuum tubes (Geissler tubes) and cause them to glow. In 1892, at a meeting of the Institution of Electrical Engineers of London, Tesla stated that this technology could be used to convey information from one point to another wirelessly [3]. As early as 1896, Tesla transmitted a continuous-wave signal from his Houston St. lab in New York City to West Point, a distance of approximately 48 km [4],[5], [9].

2.1.4 Lodge and Tunable Circuits

Oliver Lodge also in 1894, demonstrated wireless transmission over short distances with Morse code, using Edouard Branly's radio detector. Later on, he showed that untuned circuits would respond to electromagnetic waves at almost any frequency, but that their response would fall off quickly. Conversely, a tuned circuit would respond to waves whose frequency corresponded with the circuit's natural frequency. Thus, the response would not decay as quickly as for an untuned circuit [7],[9] .

2.1.5 Marconi and Trans-Atlantic Communication

Guglielmo Marconi is perhaps the most famous figure associated with the birth of wireless communication. In 1895, Marconi demonstrated the transmission of Morse code over a distance of 1.5 kilometres. By 1901, Marconi had achieved the first trans-Atlantic wireless transmission, sending a signal from England to Canada.
Marconi's innovations, particularly in antenna design and signal tuning, led to the establishment of long-distance wireless communication systems. He successfully commercialised the technology, founding companies that would provide wireless telegraphy services across the Atlantic[7],[9].

2.1.6 The Evolution of Wireless Technology in 20th century

The early 1960s saw resonant inductive wireless energy transfer being applied successfully in medical devices such as pacemakers and artificial hearts. The first passive RFID (Radio Frequency Identification) technologies were introduced by Mario Cardullo in 1973 and further developed by Koelle in 1975. By the 1990s, RFID was widely used in proximity cards and contactless smartcards. In recent years the advent of wireless networks,cell phones and smartphones has made critical the need for wireless communications as we know them today [2],[9].

2.1.7 Ray Tracing History

The idea of ray tracing comes from the 16th century painter Albrecht Dürer, who at the time described methods for projecting 3D images to a planar surface, similar to rasterization and others determining what geometry is visible along a given ray, similar to ray tracing.
Computer generated shaded pictures using ray tracing first appeared in 1968 by Arthur Appel, he used a non-recursive ray tracing-based rendering algorithm today called "ray casting" . Turner Whitted was the first to demonstrate recursive ray tracing, which allowed for the simulation of mirror reflections and refraction through translucent objects based on the object's index of refraction. Whitted used ray tracing for anti-aliasing and showed how ray-traced shadows could enhance visual realism [10].

2.1.8 Multiple-Input and Multiple-Output (MIMO) History

Bandwidth is the most critical resource in a radio channel. The range of frequencies is heavily regulated and distributed by governments, with the exception of some private satellite networks. This is the reason private entities wanted to make the most of their given frequencies. The invention of multiple-input and multiple-output (MIMO) made, possible to increase the capacity and reliability of the channel by minimising errors. Multiple antennas in wireless communication systems can be traced back to early research on multi-channel transmission and interference in wireline systems during the 1970s. AR Kaye and DA George (1970), Branderburg and Wyner (1974), and W. van Etten (1975, 1976) investigated digital transmission over multiple channels, with a focus on mitigating crosstalk in cable bundles. These early works laid the groundwork by providing mathematical techniques later applied to MIMO systems, even though they didn’t exploit multipath propagation to transmit multiple information streams. In the mid-1980s, Jack Salz from Bell Laboratories extended this research to multi-user systems, considering the challenges of cross-coupled networks and additive noise. This work set the stage for spatial techniques like Space-Division Multiple Access (SDMA), which enhanced cellular radio network performance by enabling more aggressive frequency reuse. In 1991,

Richard Roy and Björn Ottersten proposed an SDMA system using directional antennas to increase capacity in wireless networks, a method patented in 1996.

In 1993, Arogyaswami Paulraj and Thomas Kailath proposed a technique using multiple antennas at both the transmitter and receiver to broadcast high-rate signals split into several low-rate streams.They were the first to introduce a MIMO system and take advantage of the dramatic increase in capacity. In 1996, Greg Raleigh suggested that natural multipath propagation could be leveraged to transmit independent data streams using co-located antennas. He developed practical solutions for modulation, coding, and channel estimation, which were foundational to MIMO’s development. Around the same time, Gerard J. Foschini at Bell Labs introduced his "layered space-time architecture," which showed that it was feasible to multiply the capacity of a wireless link using multiple antennas. Foschini's work led to the development of the Bell Labs Layered Space-Time (BLAST) algorithm, a major advance that greatly increased channel capacity. MIMO technology reached a turning point in the early 2000s with its inclusion in wireless standards. Airgo Networks, founded by Raleigh and V.K. Jones, was the first that made MIMO-OFDM products in 2004, which played a key role in developing the IEEE 802.11n standard for wireless LANs. The 802.11n standard, finalised in 2009, allowed wireless LANs to support speeds up to 600 Mbit/s using four data streams. MIMO's role in cellular networks also grew as it was incorporated into 3G and 4G standards [8],[9].

2.2 MIMO Communications

The logic of this system is to increase the available capacity by introducing a correlation between the signals transmitted by the antennas and the time that those signals were transmitted. This happens by using multiple antennas in the receiver and transmitter and using the spatial dimension in addition to the time and frequency ones, without changing the bandwidth requirements of the system. As we know a signal that propagates from a transmitter to a receiver in an urban environment will reflect, diffract and scatter at different objects in which case will create multipaths, these multipaths reach the receiver at different phases and amplitudes than the original signal and have an effect of interference in the receiver. To combat the multipath effect many transmitters and receivers are placed with some separating space to reduce the multipath effect by increasing the change of independent paths in the channel.The capacity of the MIMO channel is given by Foschini's formula:

$$C = \log_2 \det\left[\mathbf{I}_{M_{\mathrm{Rx}}} + \frac{\rho}{M_{\mathrm{Tx}}}\mathbf{H}\mathbf{H}^{H}\right] \mathrm{b/s/Hz},$$

where:

C: is the instantaneous narrowband capacity (bps/Hz).

H: is the MIMO channel matrix (with complex gains between antenna pairs).

$\mathbf{H^H}$: is the Hermitian transpose of H.

$\mathbf{M_{Tx}}$ and $\mathbf{M_{Rx}}$ are the number of transmit and receive antennas, respectively.

$\mathbf{I_{MRx}}$: is the identity matrix

ρ : represents the signal-to-noise ratio (SNR)

The MIMO system is mathematically represented by an H-matrix, which captures the complex gains between each transmitter-receiver antenna pair. In a 2x2 MIMO system, for instance: H= $\begin{pmatrix} h11 & h12 \\ h21 & h22 \end{pmatrix}$,each h_{mn} is a complex number representing the gain between the m_{th} receiver and n_{th} transmitter. The matrix encapsulates the effects of multipath propagation.

MIMO can also leverage spatial multiplexing to increase throughput or spatial diversity to improve reliability:

Spatial Multiplexing transmits independent data streams over multiple antennas, increasing the data rate.

Diversity Gain refers to transmitting the same signal across multiple antennas, which enhances signal robustness in fading channels. This is known as the diversity-multiplexing trade-off [8],[9].

2.3 Ray Tracing in wireless communication

Solving Maxwell's equations is an approach for simulating wireless networks,however it is not possible in general to have an analytical solution for the EM field in a realistic propagation environment, it is computationally intensive and can consume significant time and resources. This has driven the development of theoretical and empirical models as more efficient alternatives such as the free space model, the okumura-hata model and many others. These models estimate the path loss and propagation delay based on range, and are valid only for those environments that resemble the modelling environment. As a result, these models usually do not provide accurate temporal or spatial information, for the real world. That's the reason ray tracing, originally a technique used in computer graphics for simulating the behaviour of light, has evolved into a tool in the field of wireless communications. Unlike these models, ray tracing models are specific to the 3-D environment, and are therefore appropriate for scenarios such as urban environments.

2.3.1 How Ray Tracing Works in Wireless Communication

Ray tracing simulates the travel of electromagnetic waves (as rays) as they interact with their surroundings. Each ray represents a potential path for the signal, and the method computes multiple propagation paths from the transmitter to the receiver. There are several methods that implement ray tracing in wireless communication, we used the method provided by MATLAB MATHWORKS [11-12] and is based on Ray Tracing for Radio Propagation Modeling: Principles and Applications [13]

For propagation modelling, a *ray* is an individual radio signal that:

1) A ray travels in a straight line in a homogeneous medium.

2) It obeys the laws of reflection and refraction, as well as the law of diffraction.

3) A ray carries energy. It is more intuitive to treat a ray as a tube (surrounding this central ray) in which the energy is contained and propagated

We have several types of rays :

A. Line of sight (LOS): The ray travels directly from the transmitter to the receiver.
B. Reflection and Transmission: When a ray hits a surface, it gets reflected in another direction based on the Laws of Reflection. The magnitude of the reflected (transmitted) field is determined by Fresnel's equations for different polarizations.
C. Diffraction: When a ray encounters sharp edges of an object (e.g., the corner of a building), it bends or interferes around the edge, allowing signals to reach areas that might otherwise be in shadow. The calculation of diffracted rays is more complicated than reflected and requires a method like the one described in [14]
D. Scattering: When rays encounter rough surfaces or small obstacles, they scatter in multiple directions (e.g., building facade)

We have several types of basic ray tracing algorithms :

A. Fermat's Principle of Least Time: It states that the ray will take a route which consumes the least time possible travelling from the source to the destination.
B. Image Method: The image of the transmitter (Tx') is created as if the reflection surface were a mirror. Tx' is positioned symmetrically on the opposite side of the planar reflection surface relative to Tx.A straight line is drawn from the image (Tx') to the receiver (Rx). This line represents the hypothetical direct path of the ray if no obstacles existed between Tx' and Rx.If this line intersects the planar reflection surface at any point (denoted as R), then a valid reflected path exists from Tx to Rx via that intersection point R.

This point represents the location where the actual reflection occurs.To find paths involving multiple reflections, the image method can be applied recursively. New images of Tx are generated relative to successive reflection surfaces, and the process is repeated to determine potential ray paths for each reflection.

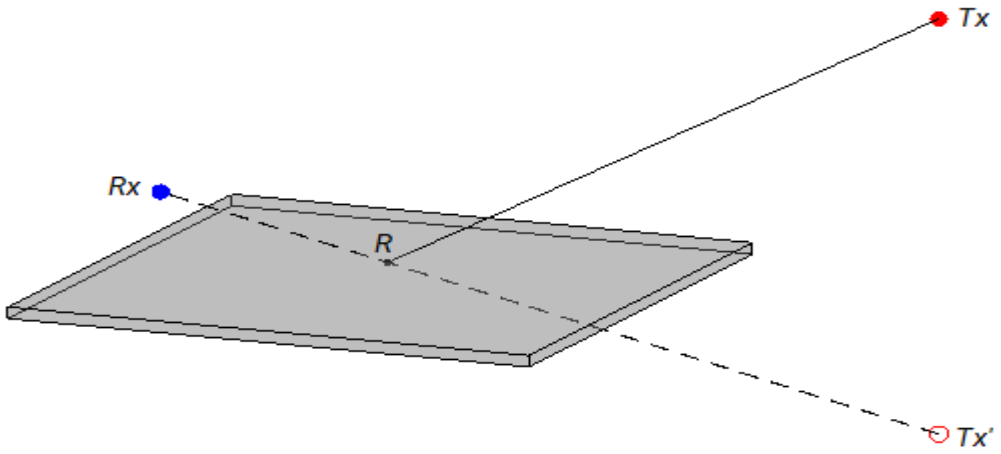


Image 2 : The image method. Ri is the image of Rx with respect to the reflection plane

C. Shooting and Bouncing Ray (SBR) Method:

The basic idea of SBR is to trace every ray launched from a source location to determine if they arrive at the destination location.When a ray is traced from the source location, there are two possible outcomes for how it reaches the destination point:

Direct Path: The ray can travel straight from the source to the field point without any obstructions or interactions.

Reflections and Diffractions: In more complex environments like urban ones, the ray may encounter various objects, such as buildings, walls, or other obstacles. If the ray hits an object, a reflected or diffracted ray will be spawned.

At the destination point we have the reception test of the ray, this test ensures that the ray has reached the receiving point, using expanding ray tubes to model the spread of energy. Circular tubes are simpler but can lead to inaccuracies, while polygonal tubes are more precise but require more computational power.

D. Hybrid Method:

The hybrid method leverages the speed of SBR for large-scale ray tracing and the precision of the image method to fine-tune ray trajectories. At first it uses the SBR method to determine a valid ray and then the image method to adjust the ray trajectory.

The Propagation Losses are calculated using the sum of Free space loss, reflection loss, and diffraction loss. The Effects of Surface Materials are also taken into account in the simulation, the way the reflection loss, diffraction loss and the effects of surface materials are calculated are explained in [11]. For a much more detailed and mathematical analysis of ray tracing in wireless communications see Chapter 17 in [15]

2.4 Quality of Ray Tracing Techniques in Wireless Communications

The quality of a propagation simulation depends on several factors. One of the most important is the accuracy of the geometrical modelling. Buildings, obstacles, terrain and material properties affect the accuracy of a simulation greatly. Ray tracing techniques provide very accurate and efficient results because they compute the actual paths that EM waves would take in the environment, following Fermat's principle. That's where this technique excels in regard to other theoretical or empirical techniques. The ray tracing techniques accuracy depend on the method of ray tracing used see chapter 17.4 [15] and [16] .

There are two key types of algorithms used in ray tracing direct and inverse. The direct algorithm tracks all possible rays but may face accuracy issues due to quantization errors (rays that don't precisely converge at the receiver). The inverse algorithm, which is more accurate, directly computes the exact paths that follow Fermat's principle. However, its complexity increases significantly with the number of objects and reflections in the environment .

The Image method is an inverse ray tracing algorithm that focuses on finding the exact propagation paths between the transmitter and receiver by creating virtual (or mirror) images of objects in the environment. When an electromagnetic wave reflects off a surface, it can be thought of as coming from the image of the transmitter (or receiver) reflected across that surface. The algorithm traces the straight-line path between

these image points and the actual receiver, accounting for reflections.In the case of the image method once the maximum number of bouncing is fixed, the image method provides all the possible rays that connect exactly the source and destination.

The SBR method belongs to the class of direct algorithms, where rays are "shot" or launched from the transmitter in multiple directions. These rays propagate in straight lines until they hit an object, after which they either reflect or diffract, depending on the properties of the surface they encounter. The reflected or diffracted rays continue to bounce off other objects, and this process repeats until some of the rays reach the receiver.In the case of the shooting and bouncing method, the number of rays reaching the destination point depends on the accuracy of the angular sampling of launching rays from the source. Moreover, the SBR simply applies Snell’s law to compute the refraction angle and to trace the refracted (transmitted) rays inside an obstacle. In the IM, Snell’s law cannot be used to trace exactly a refracted ray since the starting and ending points of the ray are fixed.For this reason many ray-tracing algorithms ignore the offset between the refracted ray and the incident ray since this offset is very small in many practical cases.

Chapter 3: Prediction Algorithms and Application to Wireless Link Quality Prediction

3.1 Simulation

We used matlab to simulate ray tracing in an indoor environment and used the results to build a channel model for a link level simulation with the MIMO-OFDM technique. That is a matlab simulation named Indoor MIMO-OFDM Communication Link Using Ray Tracing [12]. This simulation is a ray tracing analysis between one transmitter site and one receiver site in a 3-D conference room. Computed rays are used to construct a deterministic channel model which is specific for the two sites. The channel model is used in the simulation of a MIMO-OFDM communication link. This diagram characterises the communication link [1].

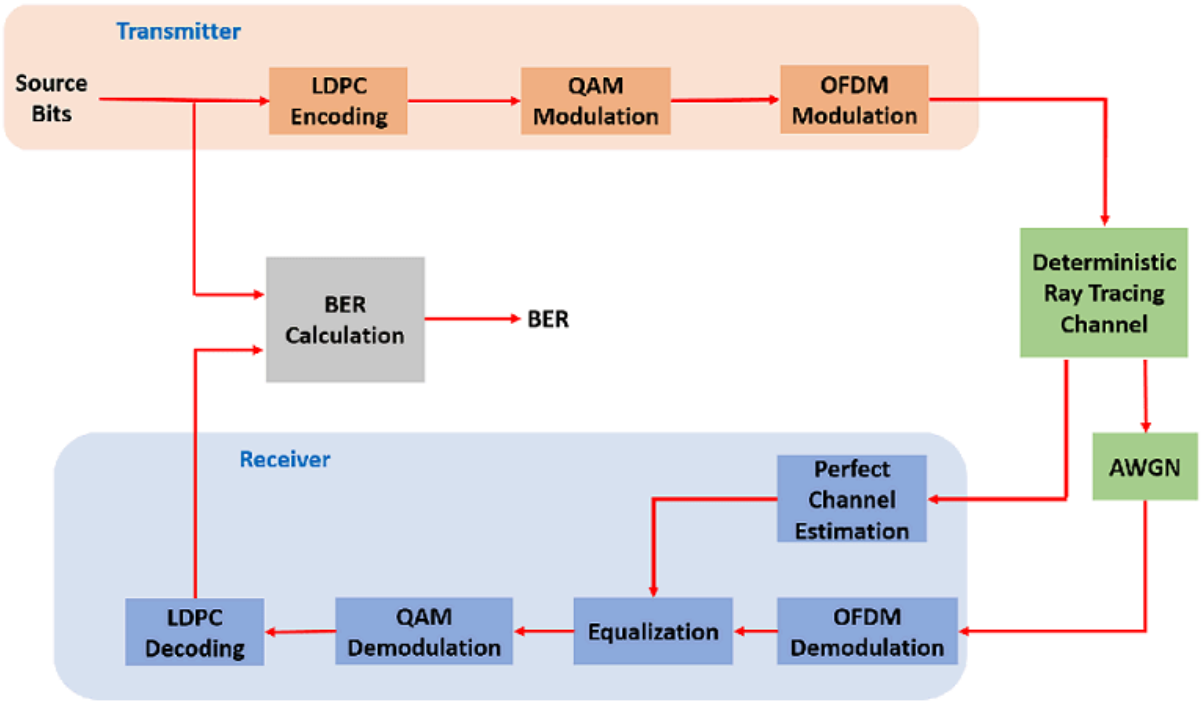


Image 4: Communication link between a transmitter and a receiver

3.2 Simulation Added Code

The simulation was changed in order to facilitate multiple random runs without the need to be rerun, essentially the simulation run in a for loop with randomised transmitter and receiver positions within their respective boundaries in the map. Some minor changes also were necessary in order to facilitate multiple runs in the helperIndoorRayTracingSimulationLoop.m file. The extended code also included the writing of simulation data in a csv file. The data collected included information for each ray produced such as propagation Delay,pathLoss, propagationDelay, PhaseShift, TransmitterLocation, ReceiverLocation, LineOfSight, AngleOfArrival, AngleOfDeparture, NumInteractions,numRays as well as information for the general summary result such as max,min,rms delay.

3.3 Data Cleaning

The results were later cleaned and formatted in order to fix some indexing issues later on in the training of the models. This process of cleaning and formatting was done using python scripts. The data has been collected all in a single file and organised in blocks with the BER, Total Errors, Total Bits, maxDelay, minDelay, rmsDelay written in the first line of each simulation and then followed in the next line by an index of the ray followed by pathLoss, propagationDelay, PhaseShift, TransmitterLocation, ReceiverLocation, LineOfSight, AngleOfArrival, AngleOfDeparture, NumInteractions, numRays for each ray generated. Because of the strange formatting the cleaning steps by the scripts were: For each block:

- First Line Extraction: From the first line of each block the following metrics are extracted:Bit Error Rate (BER),Maximum, Minimum, and RMS Delay Spreads.(the Total Errors, Total Bits were discarded)
- Next Lines Processing: The subsequent lines in the block containing the data for individual transmission rays are saved for further processing, and the 3D coordinates of the transmitter and receiver are extracted.

Output Generation:
The data then was split into two files.

- A new CSV file with each row corresponding to one block and containing the following fields:
  Bit Error Rate (BER),
  Maximum, Minimum, and RMS Delay Spreads,
  Average transmitter (Tx) position in 3D space (x, y, z),
  Average receiver (Rx) position in 3D space (x, y, z).

- The ray-specific data for each block is saved into a separate CSV file . Each row contains the following fields:
  Ray ID, Path Loss, Propagation Delay, Phase Shift,
  Transmitter (Tx) and Receiver (Rx) positions (x, y, z),
  Line of Sight information and other angular parameters (Angles of Arrival and Departure).

Then the two splitted files are merged creating a new CSV. The merging of the two files needed to be in such a way in order for each ray specific data to also have associated with it, the BER and min,max,RMS Delay Spreads (meaning copying the BER and min,max,RMS Delay Spreads in each ray of an individual simulation). Blocks begin when the value in the "Ray ID" column (index) resets to 1. The script identifies the starting point of each block, then expands the summary data to match the size of the block. For each block, the summary data corresponding to that block are repeated for the number of rows in the block. Finally concatenation is applied and the saving of the data in the new file. A transformation of the path loss to amplitude happened as well to have the same results as the simulation.

3.4 AutoGluon

3.4.1 Machine Learning

Machine learning (ML) is concerned with statistical algorithms that can learn from data and generalise to unseen data and thus perform tasks without explicit instructions, so it's perfect for our needs to predict the results of a specific simulation given a big enough dataset. This use of machine learning is also known as predictive analytics as it generates models for specific applications [20] . The core objective is to generalise. Given a specific learning dataset, the ML algorithm needs to be able to predict accurately on new data. There are 3 types of machine learning:
Supervised Learning: Learning through supervised (labelled) data, given by a "tutor", in order to find a way to map the inputs to the respective outputs. The training data is represented by a matrix and gradually improves the input by repeatedly adjusting its parameters to minimise a function that measures its performance. The process is driven by an optimization algorithm that, finds the best parameters to achieve the goal of accurate predictions. Types of supervised-learning algorithms include active learning, classification and regression. Active Learning algorithms work by interaction with a user, who labels the respective data. Classification works by classifying the data based on an observation (instance) into categories (classes). Classification algorithms are used when the outputs are restricted to a limited set of values. Regression is a set of statistical processes for estimating the relationships between a label and the input variables (features) by most closely fitting the data according to a specific mathematical criterion. Regression algorithms are used when the outputs may have any numerical value within a range.
Unsupervised Learning: Identifying hidden patterns in data without explicit labels. It identifies commonalities in the data and reacts based on the presence or absence of such commonalities in a new piece of data.
Reinforcement Learning: Learning optimal actions through trial and error to maximise rewards, by interacting with a dynamic environment in which it must perform a certain goal.
There is also an in-between type known as Semi-supervised learning in which some of the data are labelled and some not.

There are several algorithms that are being used in the models above.Some of them are :

Artificial Neural Networks modelling themselves of biological brains, follow the same principles. They have artificial neurons which are connected to each other by artificial synapses. In practice these artificial neurons are mathematical functions which receive inputs, apply weights to them and summing them to produce outputs. The artificial synapses are the edges, which can also have weights and connect the artificial neurons and propagate the signals.

Decision trees which usually are used in supervised learning models, are used for both classification and regression. They work by splitting the data into subsets, leading to a tree structure where each internal node represents a decision on an attribute, each branch represents the outcome of the decision, and each leaf node represents a final prediction or decision.

Support-vector machines (SVMs), which usually are used in supervised learning models, are used for both classification and regression. SVMs construct a hyperplane (or multiple hyperplanes) in a high-dimensional space that separates data into different classes. The goal is to find the hyperplane that maximises the margin between two classes, making it a binary linear classifier.

Regression analysis estimates the relationships between input variables and an output. The simplest form and most common of regression methods, where a straight line is fit to the data, is linear regression.

Bayesian networks represent probabilistic relationships between variables using a directed acyclic graph (DAG) [22]. They can be used to infer the likelihood of certain outcomes based on given evidence.

Gaussian processes use a process called multivariate normal distribution [23] on random variables and make predictions for new data points based on the covariances between known data points and the new points.

Genetic algorithms mimic natural selection, by iteratively using methods such as mutation [24] and crossover [25] to generate new genotypes [26] in the hope of finding good solutions to a given problem.

We used machine learning algorithms to automate an Indoor MIMO-OFDM Communication Link Using Ray Tracing Simulation through AutoGluon which is a tool which automates the ML workflow.[19]

3.4.2 How Autogluon Works

AutoGluon automates machine learning tasks. It does that by training a variety of different models, using bagging when training those models, and stack-ensembling those models to combine their predictive power into a “super” model [27].

3.4.2.1 AutoGluon Training

AutoGluon trains multiple different types of models (discussed in 3.4.1), in order to be robust in different machine learning tasks. The training process in autogluon involves several steps where various models are utilised. All models in AutoGluon inherit from the AbstractModel class (Base Model). It serves as a base template for the various model types. This class defines the basic structure for models, including where to store model outputs, what the problem type is (classification or regression, etc), and based on what metric to evaluate the model. Each model is built with customizable hyperparameters, allowing flexibility depending on the model type. [32]
Some of the models Autogluon uses are :
LGBModel (LightGBM): A gradient boosting model that uses tree-based learning algorithms.
CatBoostModel (CatBoost): A gradient boosting algorithm designed to handle categorical features.
XGBoostModel (XGBoost): A gradient boosting algorithm.
RFModel (Random Forest): An ensemble method that constructs multiple decision trees and aggregates their predictions.
XTModel (Extra Trees): An ensemble method similar to Random Forest but uses a more randomised approach in constructing decision trees.
KNNModel (K-Nearest Neighbors): An algorithm that classifies data points based on the majority label of their nearest neighbours.
LinearModel (Linear Regression, Logistic Regression): Models that predict outcomes based on a linear relationship between input features and target variables.

TabularNeuralNetTorchModel (PyTorch Neural Network for tabular data): A neural network architecture built with PyTorch for handling tabular data.
FastAI Models (Neural Networks): A high-level library built on PyTorch used for building and training neural networks.
VowpalWabbitModel: An online learning algorithm designed for reinforcement learning and supervised learning.
MultiModalPredictorModel: A model that can handle and combine multiple types of input data, such as text and images, to make predictions.
TextPredictorModel: A multimodal model focused on text data, excluding image features.
ImagePredictorModel: A multimodal model designed to work exclusively with image data.
BaggedEnsembleModel: An ensemble method that trains a base model multiple times on different folds of the training data.
StackerEnsembleModel: An ensemble model that combines predictions from multiple base models to generate a final stacked model.
WeightedEnsembleModel: An ensemble method that assigns weights to different base models.
FTTransformerModel: A transformer-based model designed for tabular data.
TabPFNModel: A model that uses the TabPFN approach for quick classification of small tabular datasets with a limited number of classes and features.
FastTextModel: A model optimised for text classification using the FastText approach.
You can find more information on each model in the [sources] in [32].

3.4.2.2 AutoGluon Bagging

It then uses Bagging or bootstrap aggregation to reduce overfitting [30]. Overfitting is the behaviour that occurs when the machine learning model gives accurate predictions for training data but not for new data. Bagging starts by generating different training sets by randomly sampling (with replacement) from the original dataset. This means some data points will be included multiple times in one sample, while others might be left out. Each of these bootstrapped datasets is used to train a separate “weak” model, independently and in parallel. After training, we aggregate the predictions from all the models by combining them to produce the final output. For example, in regression tasks, the predictions are averaged (soft voting), and for classification tasks, the majority vote decides the class (hard voting). [29] However, AutoGluon performs bagging by combining it with cross-validation. Cross-validation allows us to train and validate multiple models using all the training data. In the image below we see the workflow used in Autogluon’s k-fold cross-validation with bagging. The dataset is partitioned into 5 folds. For each fold a model is trained with all the data except the fold, meaning that the 5 model instances are trained with different portions of the data. This is known as a bagged model. After each fold's training, the model generates predictions on the validation data. These are called out-of-fold predictions because they are generated using the data that was not seen by the folds during the training. The predictions from each iteration are then compared with the ground truth (actual labels), allowing for a cross-validation score to be computed. The individual model predictions on the test data are then aggregated by averaging their predictions. This is the "bagging" part, where multiple models' outputs are combined. This process is repeated for all the models Autogluon uses.Thus, the number of models that AutoGluon trains during this process is N x K, where N is the number of models and K is the number of folds.

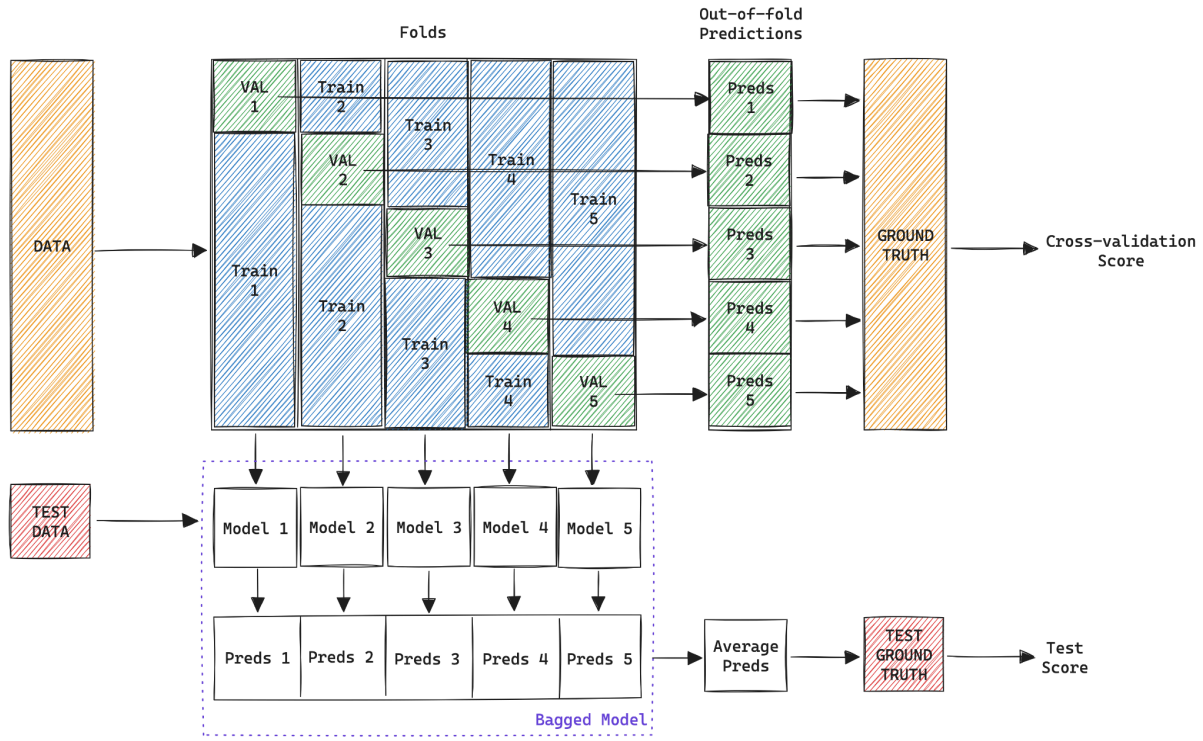


Image 5 : K- fold cross validation mapping with bagging.

3.4.2.3 AutoGluon Stacked Ensembling

Autogluon also uses Stacked Ensembling, a process which also improves the accuracy of predictions by combining the strengths of multiple models. Below you can see (image 6) describing the process:

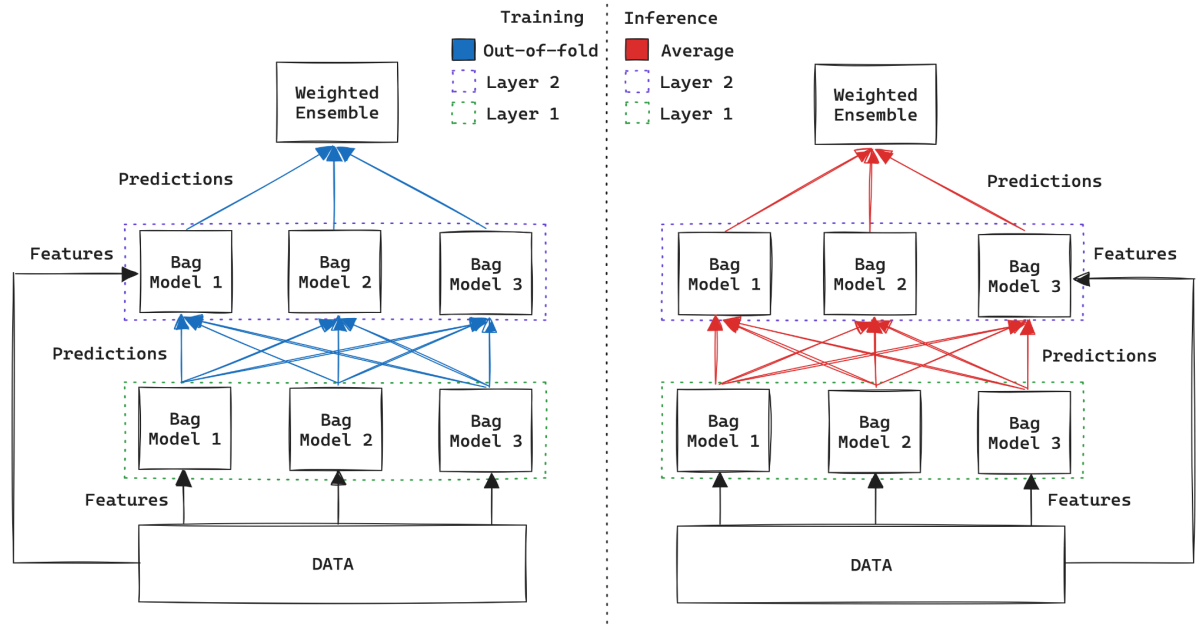


Image 6 : Process of creating a 2-layer stacked ensemble model during training (left) and inference (right).

This process consists of multiple layers of bagged models combined into a weighted ensemble. It works by creating layers of different bagged models that use the predictions from the previous layer as inputs (features) in addition to the original features from the training data. Then it uses a greedy weighted ensemble algorithm [31] to produce a final meta-model. This meta-model learns how to combine the outputs from all the second to last models to make a final prediction. Then, as in bagging, out-of-fold predictions are used to ensure that predictions for each data point are made by models that did not train on that particular data point, preventing data leakage. Lastly, the meta-model averages the predictions of all models to make predictions. M x N + 1 is the number of models autogluon creates for the stacked ensemble where M is the number of layers in the ensemble, including the base and excluding the last layer and N is the number of models per layer and 1 is the last meta-model. In image 6 we see how this is done in the training and inference phases. The process starts by taking the original features from the data and using them to train three different bagged models in Layer 1. Each of these models is trained on random subsets of the data using bagging. During this stage, out-of-fold (OOF) predictions are used. Then in layer 2, the predictions from Layer 1 are then used as input features for the next layer of models. This layer also incorporates the original features (skip connection) as input, so the models in Layer 2 use both the original data features and the predictions from Layer 1. Here OOF predictions are again used in this layer to avoid data leakage. Finally, the weighted ensemble (meta-model) is trained to combine the predictions from the second layer of models. This meta-model learns to assign different weights to the outputs of the Layer 2 bagged models based on their performance. In the inference layer a similar procedure is followed with the distinction that the test data is fed into each model directly , because we don’t have data leakage during inference. Also in the end of the inference phase the predictions of all models are averaged to create the final meta-model. [27]

3.4.3 AutoGluon Training

The time of almost 1300 runs in an Intel Core i7 (4th Gen) 4510U / 2GHz was around 8 days. That's the reason we decided to use AutoGluon [18], a machine learning tool in order to automate the simulation and get accurate and reliable results instantly. The above process (chapter 2.3) was necessary because in order to train the model to predict a given output based on the positions we needed the positions as well as the data in the same file. The 2 kinds of data did not necessarily need to be merged in the same file but it was chosen that way for convenience.

The training process involved loading the data, normalising them and then defining which were the features (input variables) and which the targets (prediction targets). Then splitting the data into training and valid data and by using a loop, training each model and evaluating the results by using mean square error as a metric and best quality as a preset. For the training of path loss, amplitude, etc, a new column is created to identify different simulation blocks. It increments every time the "Ray ID" resets to 1. This ensures that the SimulationID can be used to help the model understand that the data comes from distinct simulation runs.

3.4.4 AutoGluon Prediction

The prediction process involved loading test data (i.e. transmitter and receiver positions) based on which the prediction will be made. Defining features and loading models. Making predictions. Creating a SimulationID by identifying when Ray ID restarts from 1 and grouping the data by the SimulationID for the visualisation. Finally plotting the data.

3.5 Error Calculation

The error calculation process involves similar steps with the prediction process but the prediction happens and then a calculation of mean absolute error and the mean squared error based on the dataset, as well as saving and visualising the errors.

Chapter 4: Results and Discussion

4.1.Predicted Results

4.1.1.Predicted Results for RMS,Max,Min,BER

At first we predicted results for RMS (Root Mean Square) Delay Spread, Max Delay Spread, Min Delay Spread & BER (Bit Error Rate). For a specific transmitter and receiver location the trained models provide the respective results for the above metrics. For example for random positions given to the predictor script we get results like below:

```
python predictor_for_ber_max_min_rms.py
Predicted RMS Delay Spread:
0    9.708429e-09
1    9.498983e-09
2    1.035317e-08
Name: allRMSDelaySpead, dtype: float32
Predicted Max Delay Spread:
0    1.412324e-08
1    1.390117e-08
2    1.450742e-08
Name: allMaxDelaySpread, dtype: float32
Predicted Min Delay Spread:
0    3.487222e-09
1    2.678811e-09
2    3.413289e-09
Name: allMinDelaySpread, dtype: float32
Predicted BER:
0    0.446374
1    0.316523
2    0.048628
Name: allBERs, dtype: float32
```

Where the first column is an index for each position given in the script and for each index we get their respective RMS Delay Spread,Max Delay Spread, Min Delay Spread, BER , for this instance we provided three positions given below:

```
new_data = pd.DataFrame({
     'Tx_position_x': [0.1, 0.2,0.2],
     'Tx_position_y': [0.1, 0.2,0.1],
     'Tx_position_z': [1.0, 1.5,1.8],
     'Rx_position_x': [-0.1, -0.2,-0.5],
     'Rx_position_y': [-0.1, -0.2,-0.5],
     'Rx_position_z': [1.0, 1.5,1.8]
})
```

For the above positions we provide the actual results from the simulation:

Index 0:

```
Tx = [0.1;0.1;1.0]
Rx = [-0.1;-0.1;1.0]
```

```
RMS Delay Spread: 7.994277e-09 s
Max Delay Spread: 1.005130e-08 s
Min Delay Spread: 9.434617e-10 s
BER: 0.497924
```

Index 1:

```
Tx = [0.2;0.2;1.5]
Rx = [-0.2;-0.2;1.5]
```

```
RMS Delay Spread: 8.181269e-09 s
Max Delay Spread: 1.009548e-08 s
Min Delay Spread: 1.886923e-09 s
BER: 0.499954
```

Index 2:

```
Tx = [0.2;0.1;1.8]
Rx = [-0.5;-0.5;1.8]
```

```
RMS Delay Spread: 1.024985e-08 s
Max Delay Spread: 1.499555e-08 s
Min Delay Spread: 3.075309e-09 s
BER: 0.064931
```

4.2.Error Analysis

4.2.1 Error Analysis for RMS,Max,Min,BER

We calculated the error of the predicted metrics based on the training data:

```
Errors for allRMSDelaySpead:
  Mean Absolute Error: 5.241864282718379e-10
  Mean Squared Error: 6.459919628935503e-19
```

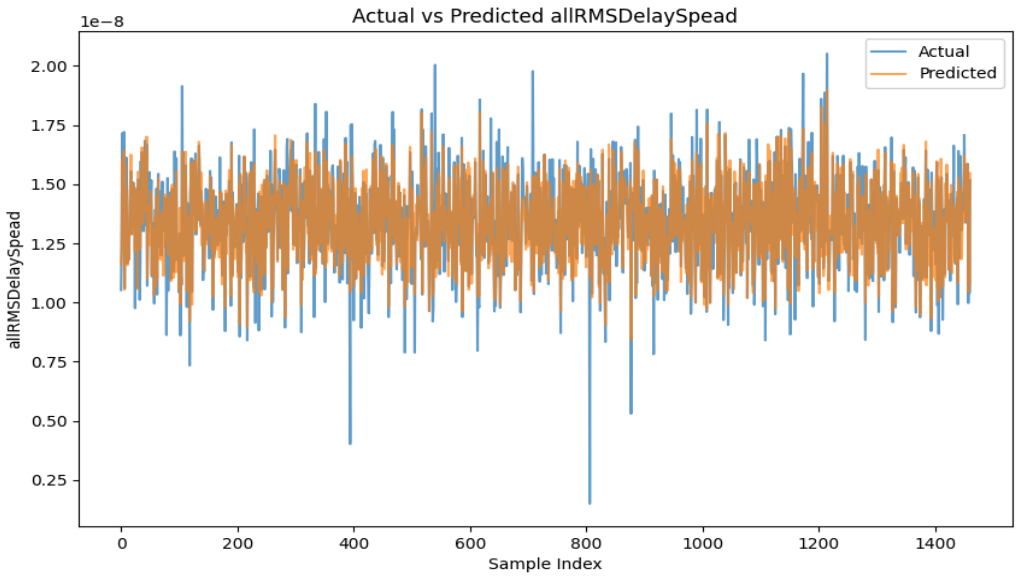


Figure 1

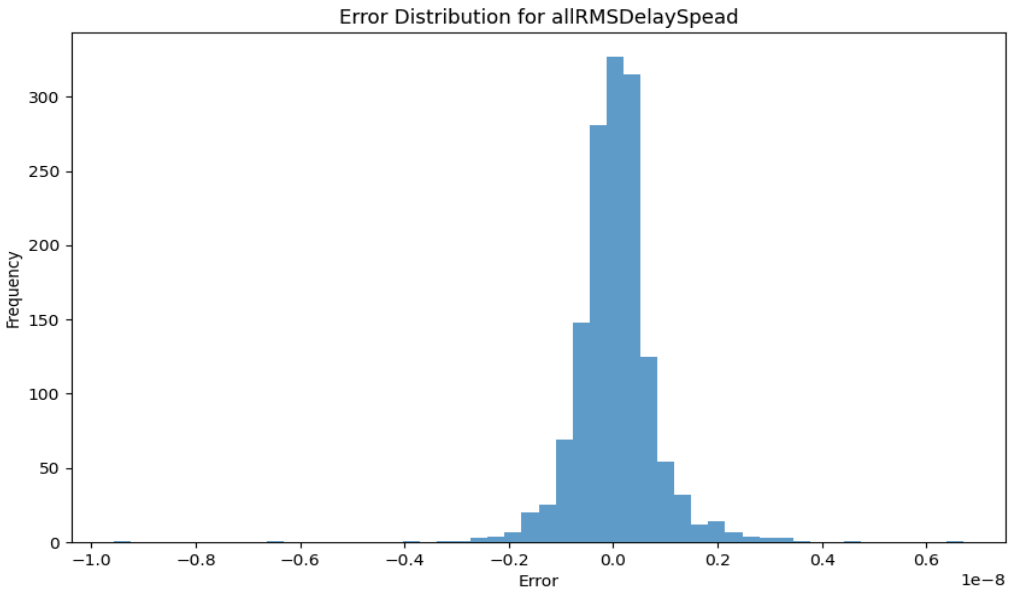


Figure 2

```
Errors for allMaxDelaySpread:
  Mean Absolute Error: 1.2937923066779813e-09
  Mean Squared Error: 3.40077314810215e-18
```

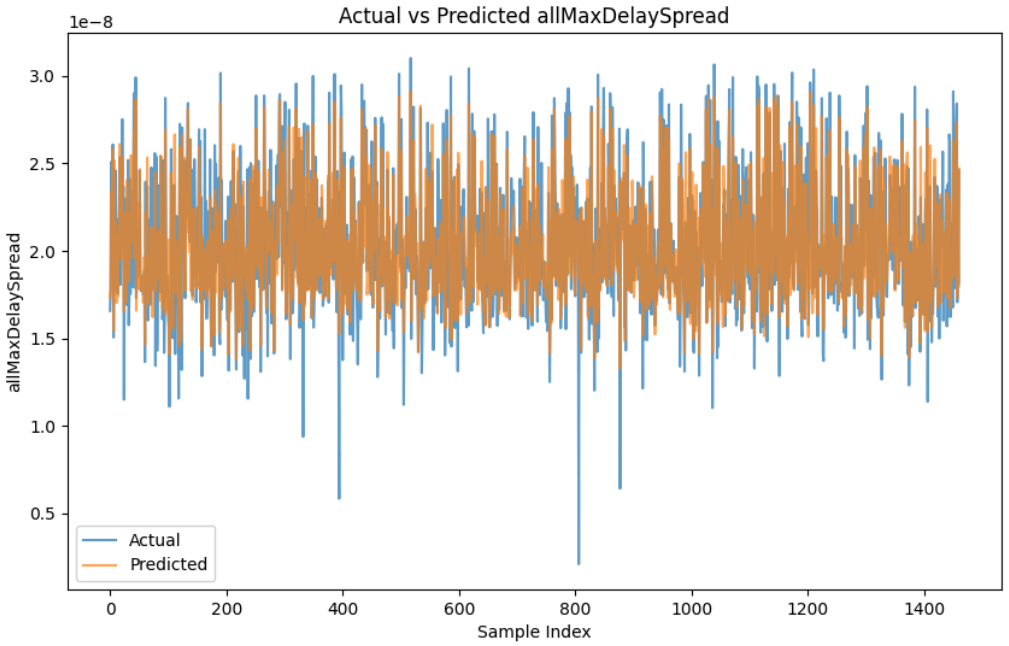


Figure 3

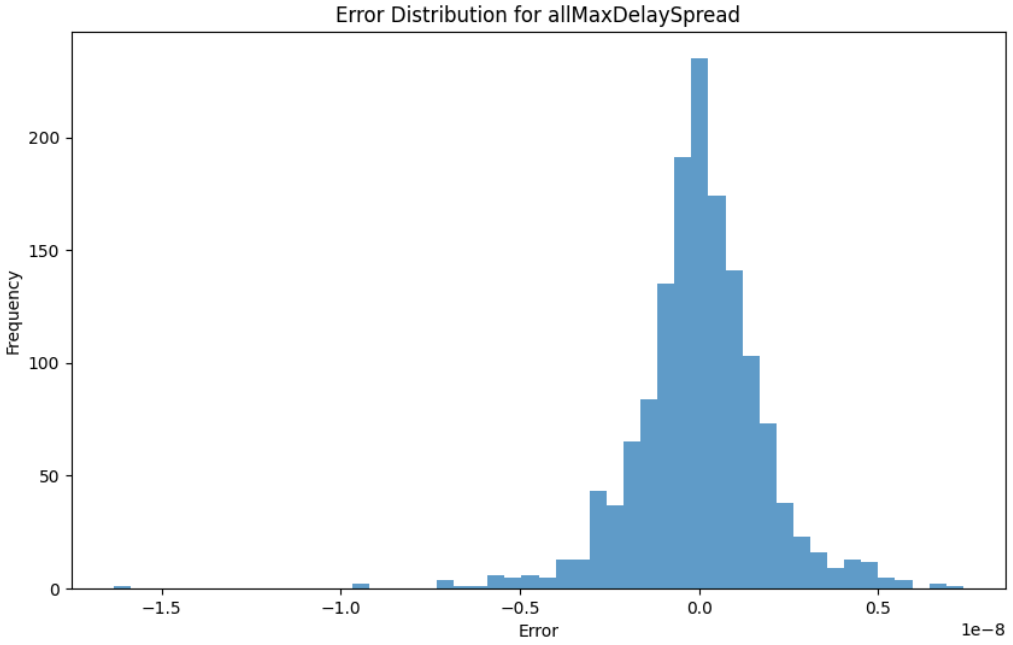


Figure 4

```
Errors for allMinDelaySpread:
  Mean Absolute Error: 4.1532811431906516e-10
  Mean Squared Error: 4.839939379547288e-19
```

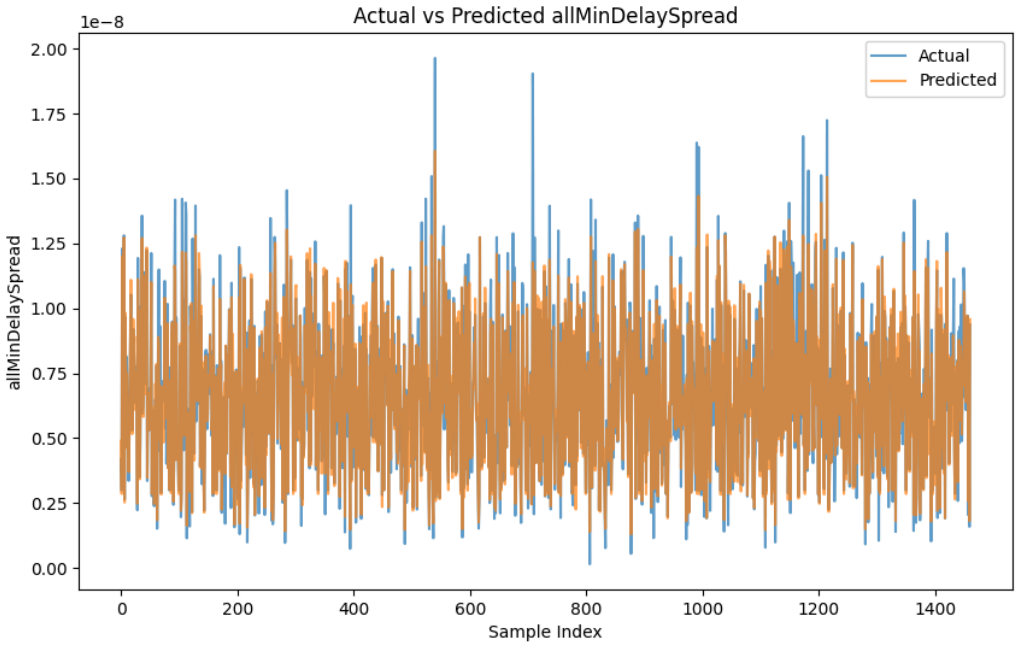


Figure 5

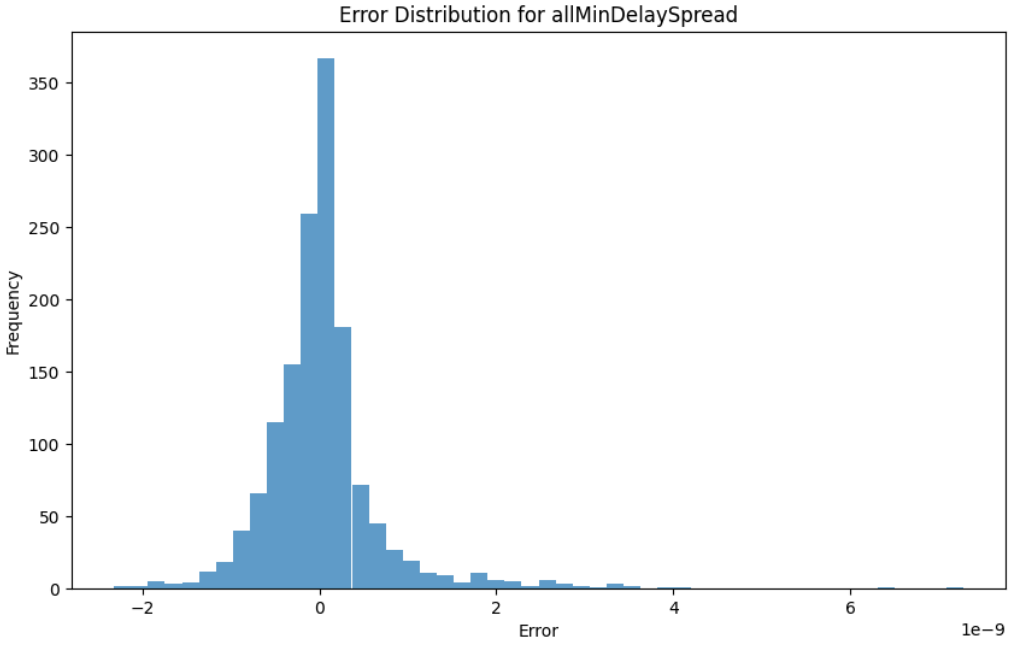


Figure 6

```
Errors for allBERs:
  Mean Absolute Error: 0.029418332953674 84
  Mean Squared Error: 0.0046655592120467082
```

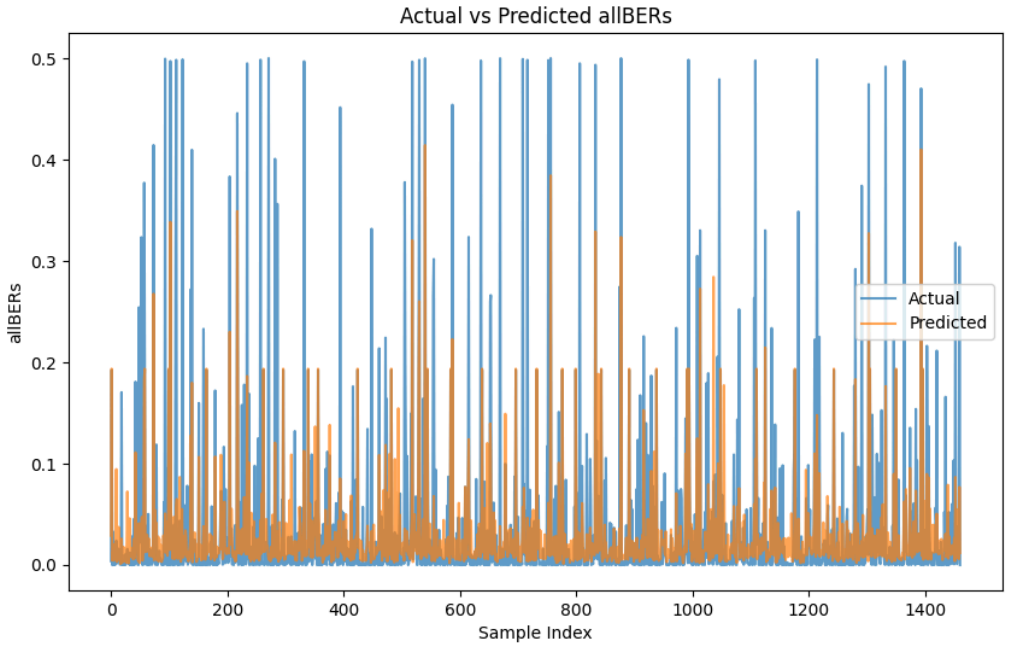


Figure 7

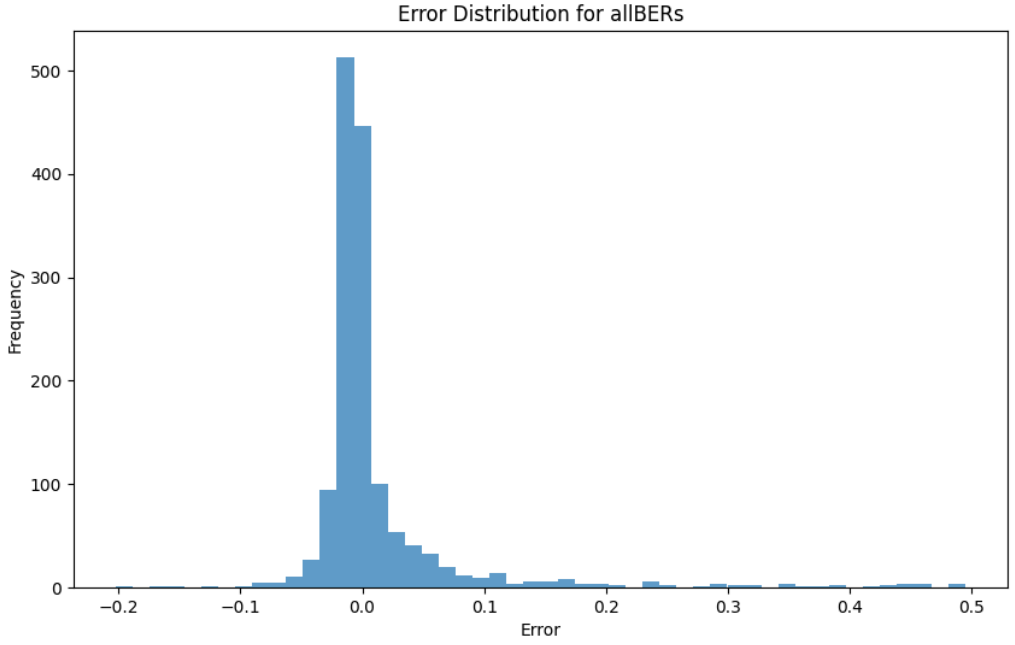


Figure 8

From the histogram, we can observe that most errors are concentrated around zero, indicating that the predicted RMS DelaySpread values, (Min-Max)-DelaySpread values & BER values, are generally close to the actual values.

Additionally, the line plot below compares the actual vs. predicted metrics values for the sample indices.

In the plot, the blue line represents the actual metrics values, and the orange line represents the predicted metrics values. The proximity of the two lines indicates the accuracy of the predictions. While there are some deviations, the general trend of the predictions aligns well with the actual data.

The low Mean Absolute Error and Mean Squared Error values for delay spreads and relatively low for ber indicate high prediction accuracy.

4.1.2.Predicted Results for Power Delay Profile of Path Loss vs Propagation Delay

Using AutoGluon, we predicted Path Loss and Propagation Delay based on simulation data.

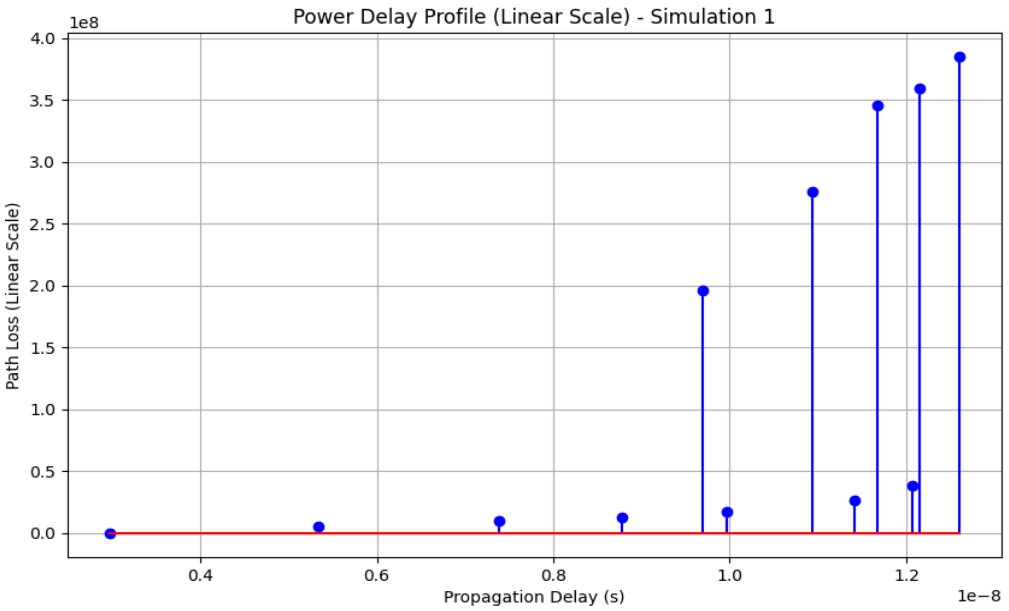


Figure 9

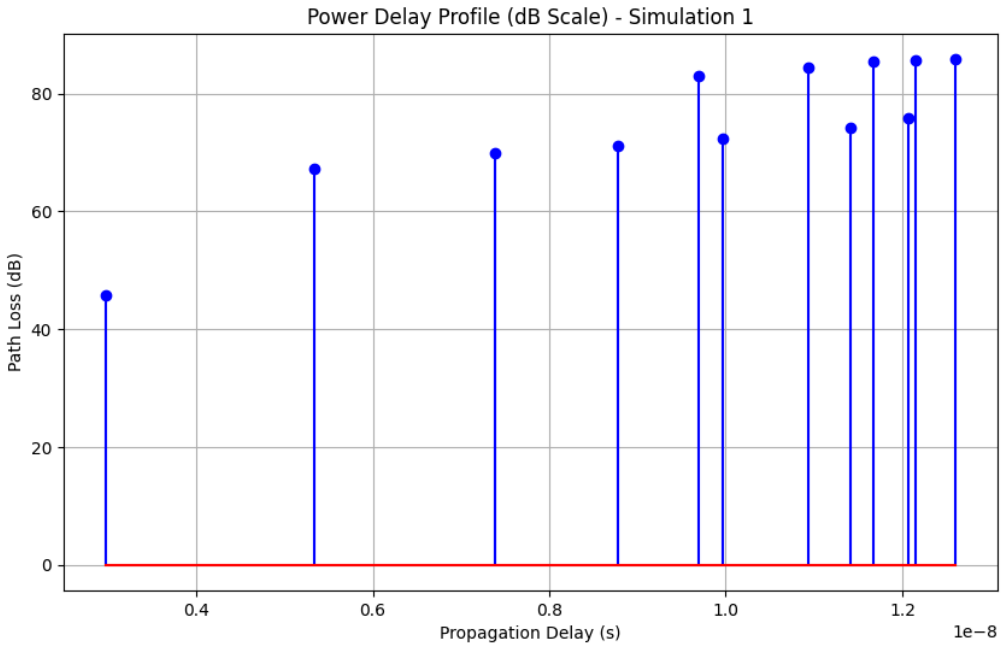


Figure 10

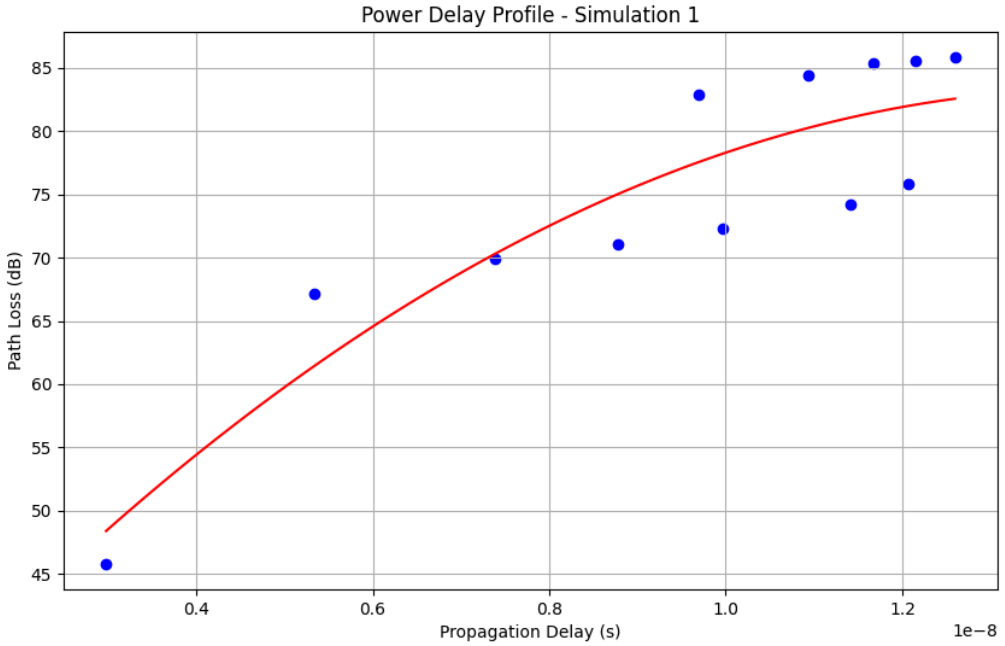


Figure 11

Figure 9: Power Delay Profile (Linear Scale)

This plot shows the Path Loss in a linear scale for better visualisation of power variations.This scale makes it easier to see the relative power of different rays, especially when there are significant differences in path losses.

Figure 10: Power Delay Profile (dB Scale)

This plot shows the Path Loss in decibels (dB) versus the Propagation Delay in seconds.The stems represent individual rays, showing their respective delays and path losses. This helps in visualising the delay spread and the intensity of the received signal components.

Figure 11: Power Delay Profile with Polynomial Fit

This plot includes a polynomial fit to the data, providing a curve of best fit.The red curve represents a second-degree polynomial fit, highlighting the trend of the path loss over the propagation delay. This can be useful for understanding the general behaviour of the signal decay and delay spread.

Here you can see the form needed for the data to make the prediction

```
Tx_x,Tx_y,Tx_z,Rx_x,Rx_y,Rx_z,Ray ID,SimulationID
-0.69,-0.33,0.675,-0.9,0.9,0.775,1,1
-0.69,-0.33,0.675,-0.9,0.9,0.775,2,1
-0.69,-0.33,0.675,-0.9,0.9,0.775,3,1
-0.69,-0.33,0.675,-0.9,0.9,0.775,4,1
-0.69,-0.33,0.675,-0.9,0.9,0.775,5,1
-0.69,-0.33,0.675,-0.9,0.9,0.775,6,1
-0.69,-0.33,0.675,-0.9,0.9,0.775,7,1
-0.69,-0.33,0.675,-0.9,0.9,0.775,8,1
-0.69,-0.33,0.675,-0.9,0.9,0.775,9,1
-0.69,-0.33,0.675,-0.9,0.9,0.775,10,1
-0.69,-0.33,0.675,-0.9,0.9,0.775,11,1
-0.69,-0.33,0.675,-0.9,0.9,0.775,12,1
```

It is necessary to have the position in each Ray ID for the prediction to work and predict the appropriate path loss and prop delay based on the Ray ID number. The Simulation ID provides us with the capability to have multiple PDP predictions made for different positions .

4.2.2 Error Analysis for Power Delay Profile of Path Loss vs Propagation Delay

```
Errors for Path Loss:
  Mean Absolute Error: 3.552975922490833
  Mean Squared Error: 26.462217921769874
```

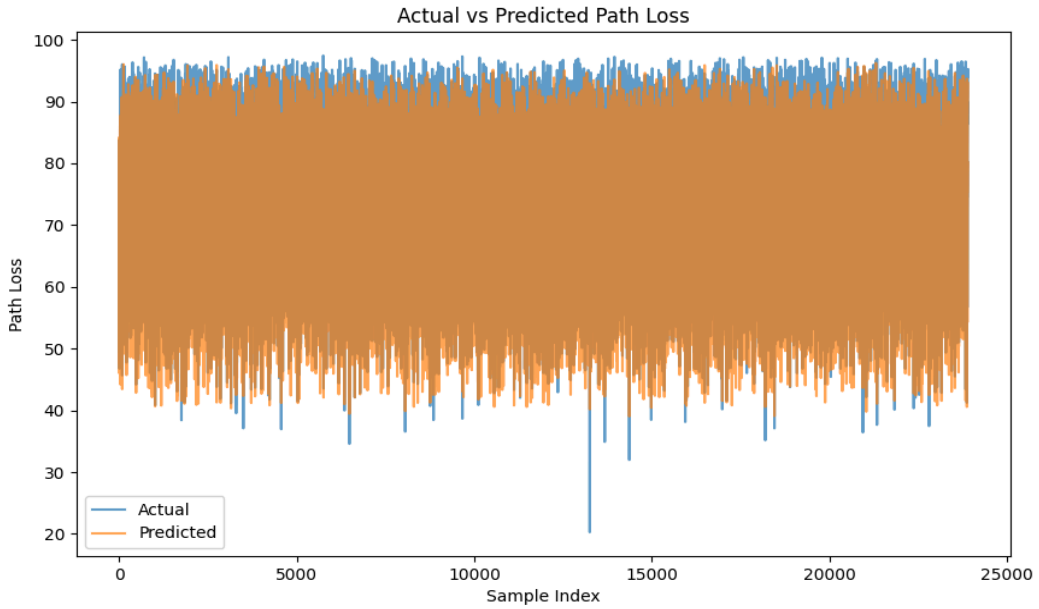


Figure 12

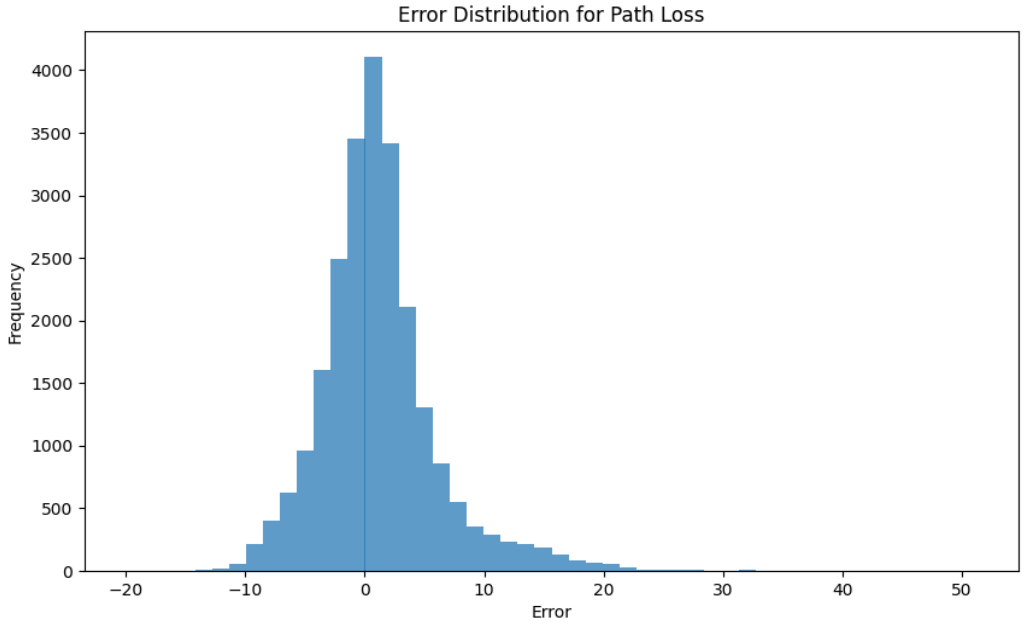


Figure 13

```
Errors for Propagation Delay:
  Mean Absolute Error: 2.8817546520570395e-09
  Mean Squared Error: 1.4111731144997448e-17
```

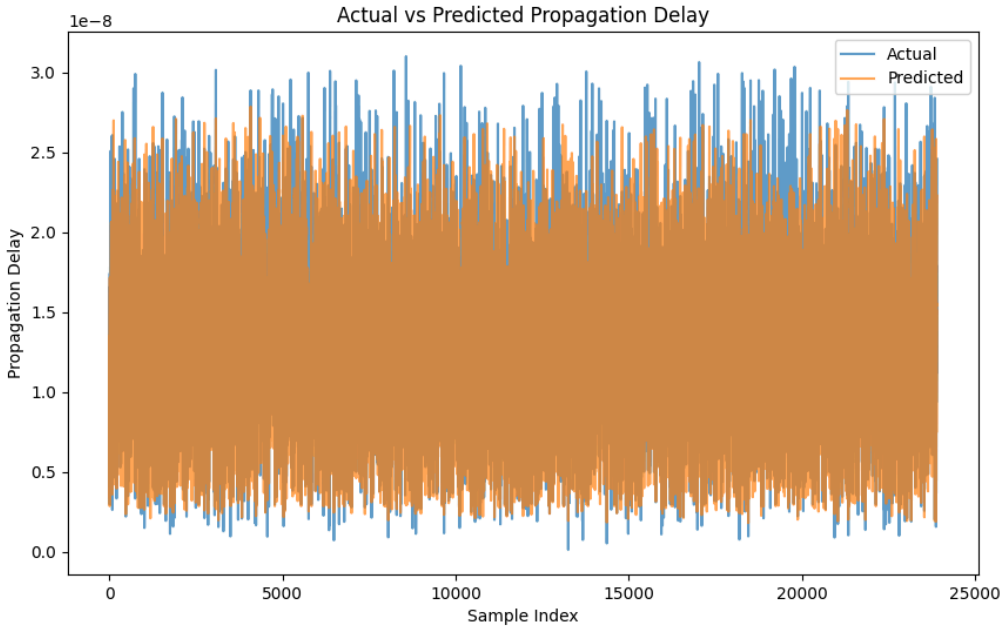


Figure 14

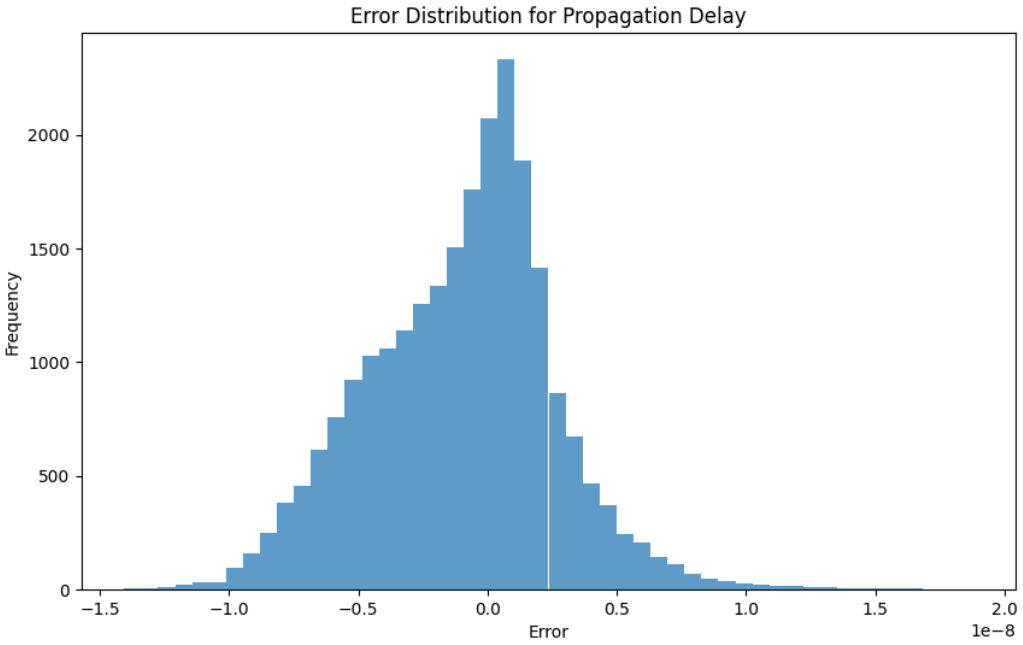


Figure 15

Path Loss Errors:

The MAE and MSE indicate that while the average error in prediction is relatively low, there may be some larger deviations affecting the MSE.

Propagation Delay Errors:

Both MAE and MSE values are extremely low, suggesting high precision in the predicted propagation delays.

Figure 12: Actual vs Predicted Path Loss

In the plot, the blue line represents the actual path loss values, and the orange line represents the predicted path loss values. The proximity of the two lines indicates the accuracy of the predictions. While there are some deviations, the general trend of the predictions aligns well with the actual data.

Figure 13:Error Distribution for Path Loss

This plot shows how the prediction errors for Path Loss are distributed. The x-axis represents the error values (difference between actual and predicted Path Loss), and the y-axis represents the frequency of these errors. This visualisation helps in understanding the spread and concentration of errors.

Figure 14:Actual vs Predicted Propagation Delay

In the plot, the blue line represents the actual propagation delay values, and the orange line represents the predicted path loss values. The proximity of the two lines indicates the accuracy of the predictions. While there are some deviations, the general trend of the predictions aligns well with the actual data.

Figure 15:Error Distribution for Propagation Delay

Similar to the Path Loss error distribution, this plot depicts the error distribution for Propagation Delay. The x-axis represents the error values in seconds, and the y-axis represents their frequency. This plot helps in identifying any significant deviations in the predicted Propagation Delay.

4.1.3.Predicted Results for Power Delay Profile of Amplitude vs Propagation Delay

Using AutoGluon, we predicted Amplitude and Propagation Delay based on simulation data processing to change the path loss to amplitude.

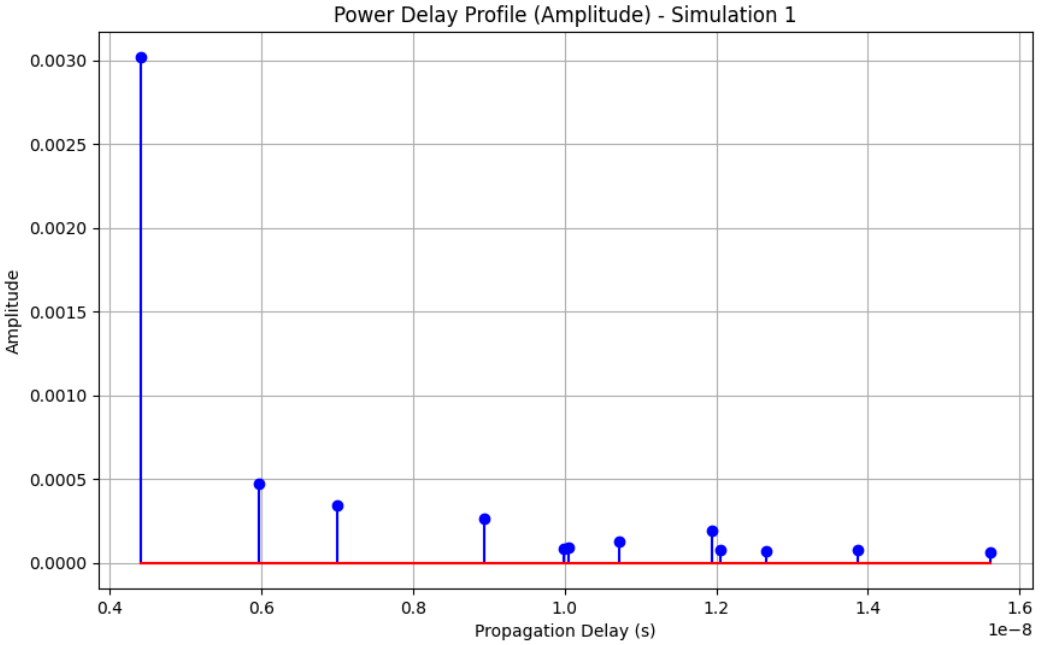


Figure 16

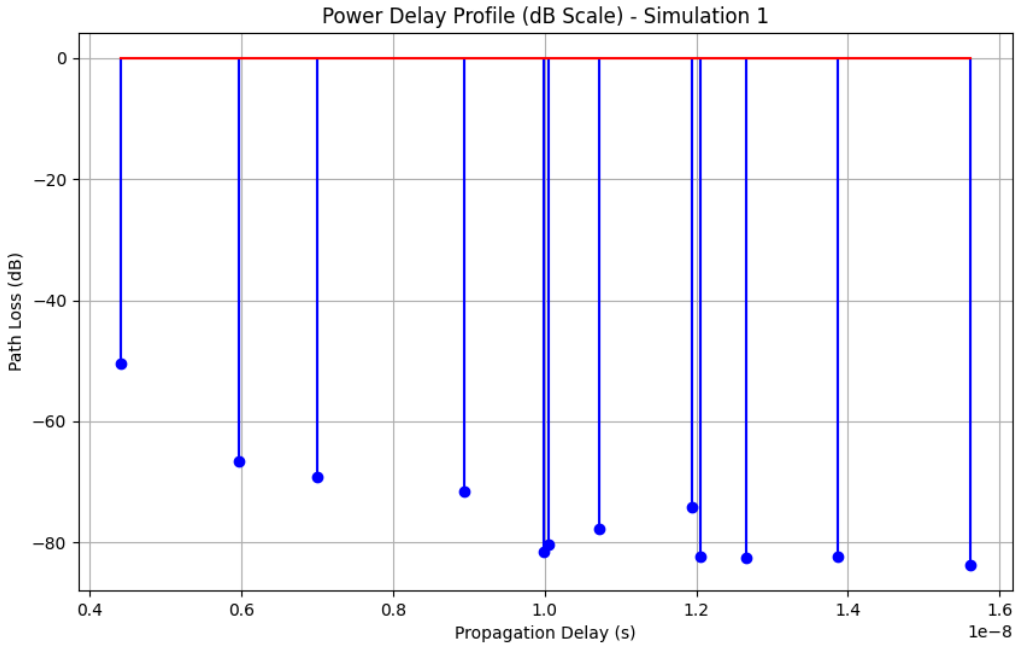


Figure 17

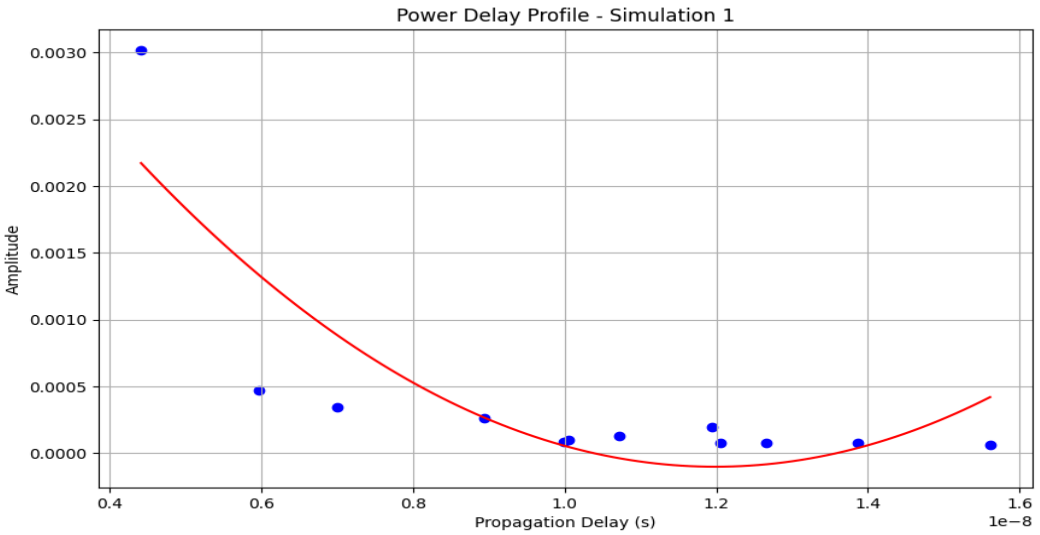


Figure 18

Figure 16: Power Delay Profile (Linear Scale)

This plot shows the Amplitude in linear scale for better visualisation of power variations.This scale makes it easier to see the relative power of different rays, especially when there are significant differences in path losses.

Figure 17: Power Delay Profile (dB Scale)

This plot shows the Amplitude in decibels (dB) versus the Propagation Delay in seconds.The stems represent individual rays, showing their respective delays and Amplitudes. This helps in visualising the delay spread and the intensity of the received signal components.

Figure 18: Power Delay Profile with Polynomial Fit

This plot includes a polynomial fit to the data, providing a curve of best fit.The red curve represents a second-degree polynomial fit, highlighting the trend of the Amplitude over the propagation delay. This can be useful for understanding the general behaviour of the signal decay and delay spread.

- The form needed for the data to make the prediction is the same as above given at the end of 1.2.

4.2.3 Error Analysis for Power Delay Profile of Amplitude vs Propagation Delay

```
Errors for Amplitude:
  Mean Absolute Error: 0.00012401906803141463
  Mean Squared Error: 3.8981638958956046e-07
```

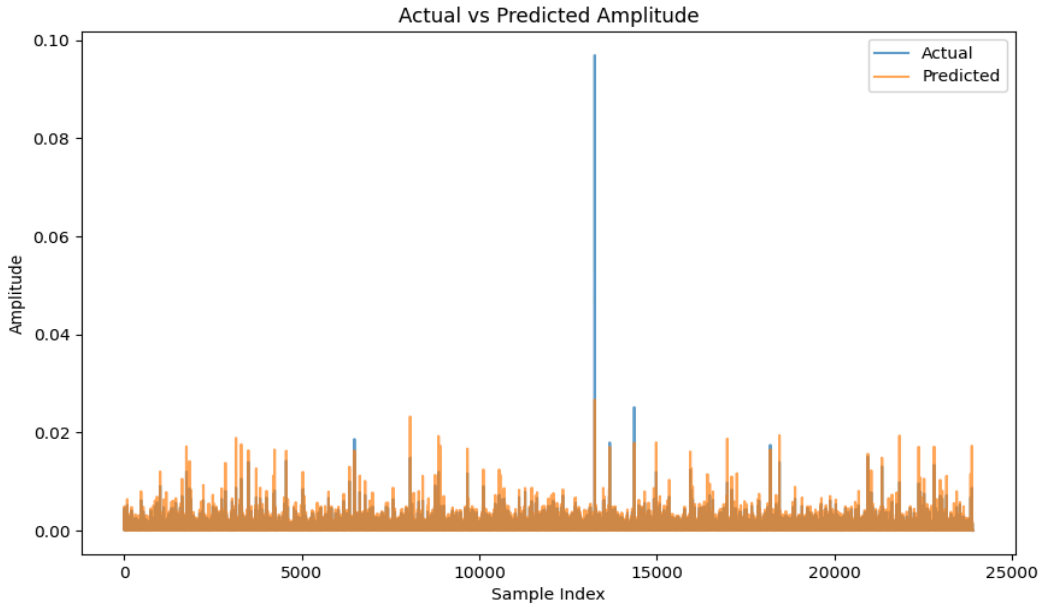


Figure 19

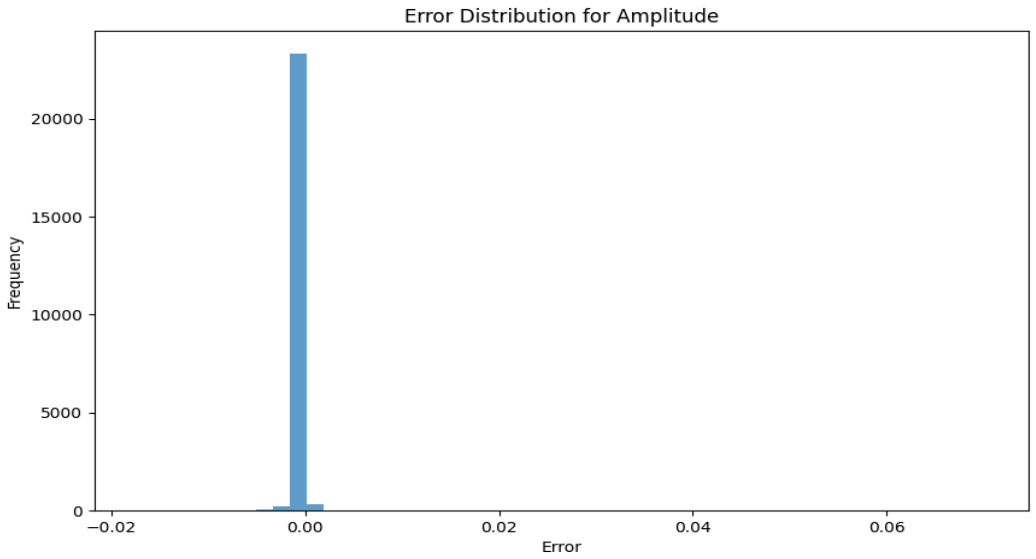


Figure 20

```
Errors for Propagation Delay:
  Mean Absolute Error: 3.177076245403479e-09
  Mean Squared Error: 1.7632334218518292e-17
```

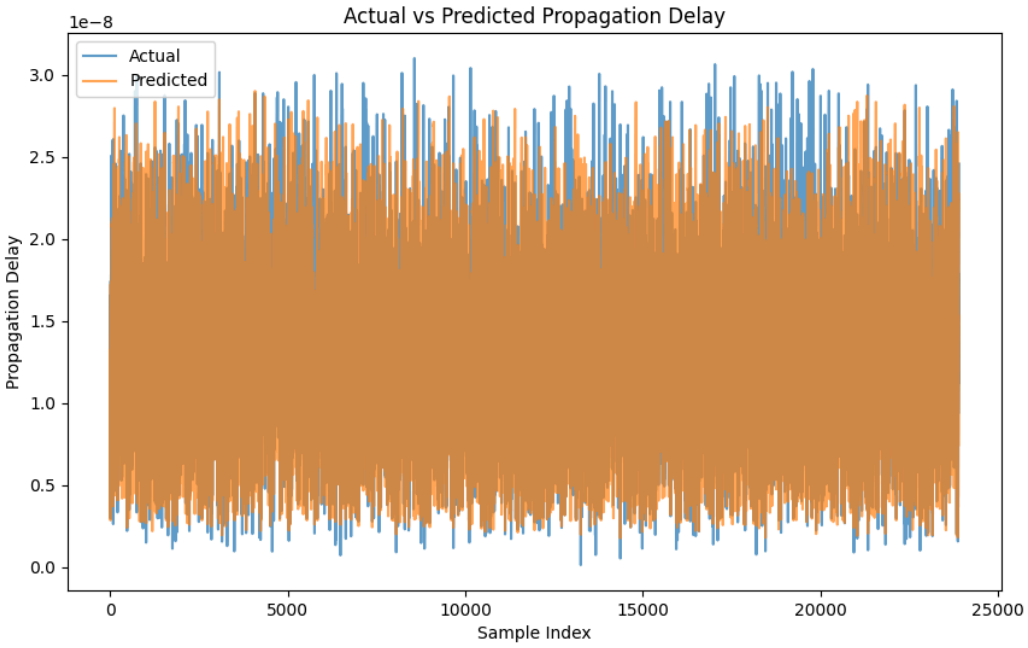


Figure 20

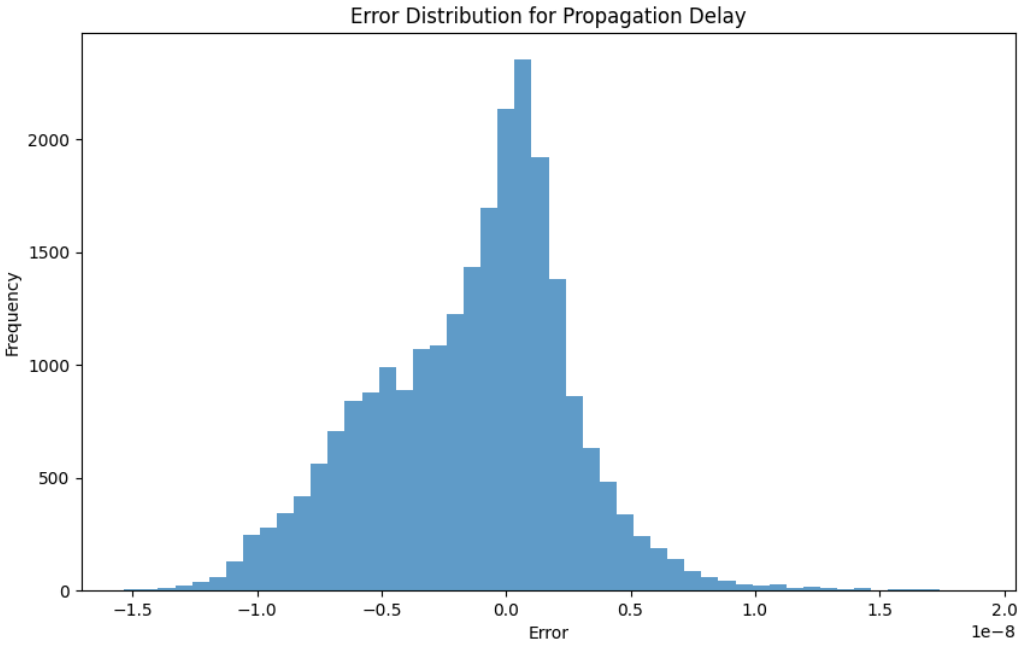


Figure 21

Amplitude Errors:

The low MAE and MSE values indicate that the model is highly accurate in predicting the amplitude of the signal.

Propagation Delay Errors:

The extremely low MAE and MSE values suggest that the model provides precise and consistent predictions of propagation delay.

Figure 18: Actual vs Predicted amplitude Loss

In the plot, the blue line represents the actual amplitude values, and the orange line represents the predicted amplitude values. The proximity of the two lines indicates the accuracy of the predictions. While there are some deviations, the general trend of the predictions aligns well with the actual data.

Figure 19:Error Distribution for amplitude Loss

This plot shows how the prediction errors for Amplitude are distributed. The x-axis represents the error values (difference between actual and predicted Amplitude), and the y-axis represents the frequency of these errors. This visualisation helps in understanding the spread and concentration of errors.

Figure 20:Actual vs Predicted Propagation Delay

Nothing changed from the previous error distribution of propagation delay as the only values changed are the ones from the processing of the path loss to amplitude.

Similar as figure 14.

Figure 21:Error Distribution for Propagation Delay

Similar as figure 15.

4.1.4.Predicted Results for Ray Number

Given a specific position of rx and tx the model predicts the number of rays that would be produced.There was no specific reason to write a script predicting just the ray number, so the model is used instantly in the prediction of Power Delay Profile of Amplitude vs Propagation Delay with predicted number ray.

4.2.4 Error Analysis for Number of Rays

```
Errors for Num_Rays:
  Mean Absolute Error: 1.1373197134352497
  Mean Squared Error: 2.509557889745679
```

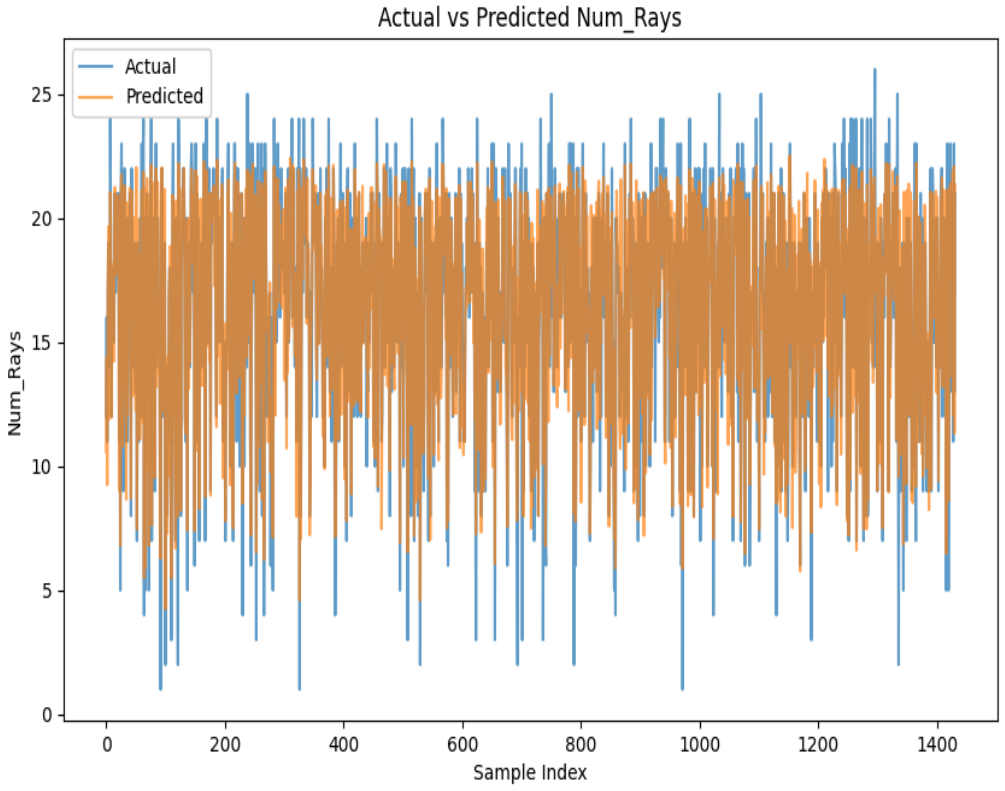


Figure 22

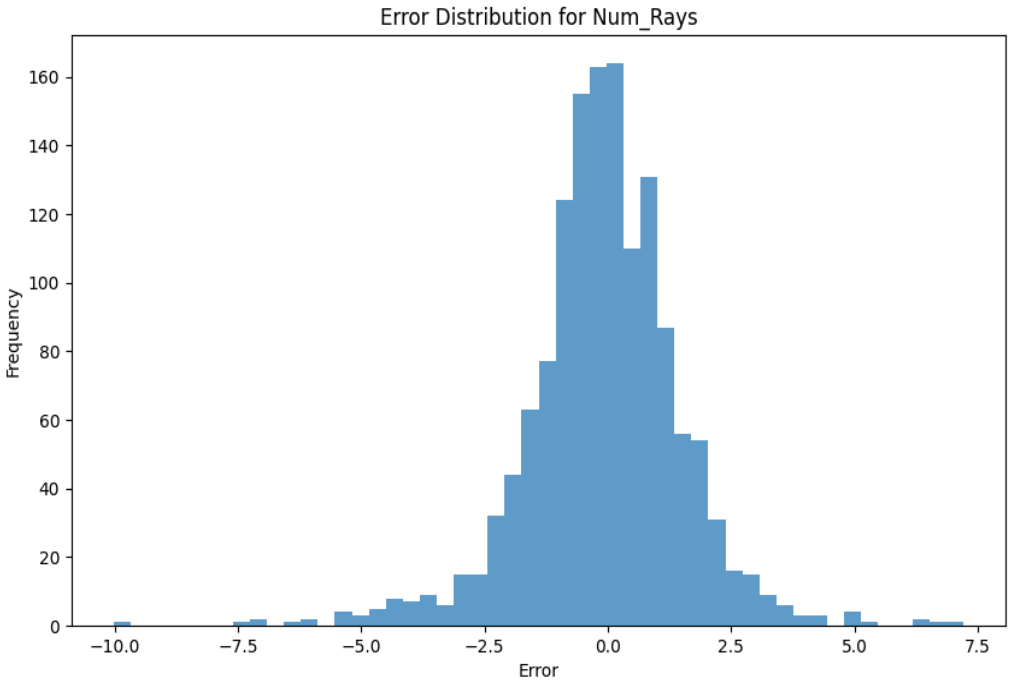


Figure 23

Number of Rays Errors:

The MAE of approximately 1.1371 indicates that, on average, the model's predictions for the number of rays deviate from the actual values by about 1.14 rays. This level of error suggests a reasonable level of accuracy, considering the complexity of predicting discrete values like the number of rays in a propagation environment.

While an MAE of around 1.14 rays may seem slightly high, it is important to contextualise this within the typical range and variability of the number of rays in the simulation. The total number of rays varies widely, an MAE of 1.14 represents a relatively small percentage error.

An MSE of 2.51 indicates that while there are some deviations from the actual number of rays, these deviations are not excessively large, but the presence of a few larger errors could be influencing this metric.

Figure 22: Actual vs Predicted Number of Rays

This line plot compares the actual values of Number of Rays with the predicted values over a series of samples.The blue line represents the actual values, while the orange line represents the predicted values.The predicted values track the actual values closely, indicating that the model generally performs well.There is variability in Number of Rays across the samples, but the model's predictions follow this variability reasonably well.

Figure23:Error Distribution for Number of Rays

This histogram shows the distribution of the errors (actual - predicted) in the model's predictions of Number of Rays.The errors are centred around 0, suggesting that the model's predictions are unbiased on average.The distribution of errors appears to be approximately normal, with most errors falling between -2 and 2.There are a few outliers with errors beyond ±7.5,±5, but these are relatively rare.

4.1.5.Predicted Results for Power Delay Profile of Amplitude vs Propagation Delay with predicted number ray

Using AutoGluon, we predicted the number of rays and then used it to predict based on that, the Amplitude and Propagation Delay. The difference here is in the data provided to the model to make the prediction of the pdp. In the final model with the predicted num rays the only provided data is the location of the receiver and transmitter. As shown below:

```
Tx_x,Tx_y,Tx_z,Rx_x,Rx_y,Rx_z
-0.69,-0.33,0.675,-0.9,0.9,0.775
```

The current predictor provides the data produced in printed form in the terminal as well. Because of the use of the previous amplitude and propagation delay models, this predictor needed to expand the data provided in the file (and in the block above) in order to have a form of data supported by the previous models. Also as in the previous predictors multiple predictions are supported by providing different positions. The printed data produced are provided below:

```
Predicted Number of Rays for each data point:
   Tx_x  Tx_y   Tx_z  Rx_x  Rx_y   Rx_z  Predicted Number of Rays
0 -0.69 -0.33  0.675  -0.9   0.9  0.775                        11
Expanded data based on predicted number of rays:
   Tx_x  Tx_y   Tx_z  Rx_x  Rx_y   Rx_z  Predicted Number of Rays  Ray ID
0 -0.69 -0.33  0.675  -0.9   0.9  0.775                      11.0   1.0
0 -0.69 -0.33  0.675  -0.9   0.9  0.775                      11.0   2.0
0 -0.69 -0.33  0.675  -0.9   0.9  0.775                      11.0   3.0
0 -0.69 -0.33  0.675  -0.9   0.9  0.775                      11.0   4.0
0 -0.69 -0.33  0.675  -0.9   0.9  0.775                      11.0   5.0
0 -0.69 -0.33  0.675  -0.9   0.9  0.775                      11.0   6.0
0 -0.69 -0.33  0.675  -0.9   0.9  0.775                      11.0   7.0
0 -0.69 -0.33  0.675  -0.9   0.9  0.775                      11.0   8.0
0 -0.69 -0.33  0.675  -0.9   0.9  0.775                      11.0   9.0
0 -0.69 -0.33  0.675  -0.9   0.9  0.775                      11.0   10.0
0 -0.69 -0.33  0.675  -0.9   0.9  0.775                      11.0   11.0
Predicted Amplitude and Propagation Delay for each ray:
   Ray ID  Predicted Amplitude  Predicted Propagation Delay
0     1.0             0.003645                3.579598e-09
0     2.0             0.000501                5.148923e-09
0     3.0             0.000369                6.626560e-09
0     4.0             0.000284                8.042598e-09
0     5.0             0.000241                1.013441e-08
0     6.0             0.000186                1.235002e-08
0     7.0             0.000164                1.324394e-08
0     8.0             0.000101                8.408587e-09
0     9.0             0.000086                9.653400e-09
0     10.0            0.000077                1.062892e-08
0     11.0            0.000075                1.217900e-08
```

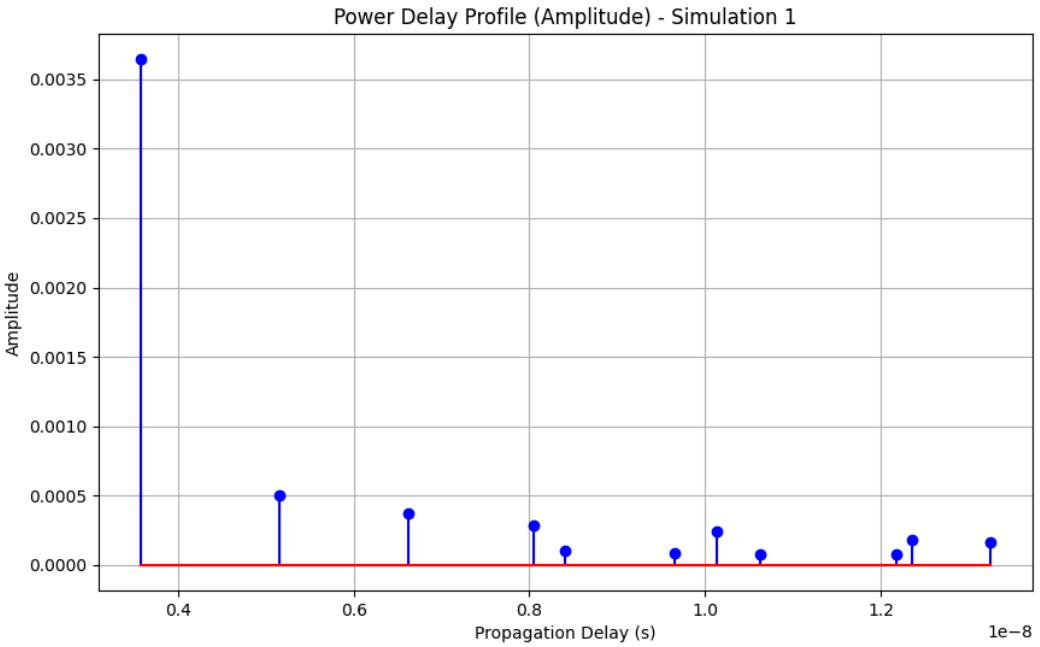


Figure 24

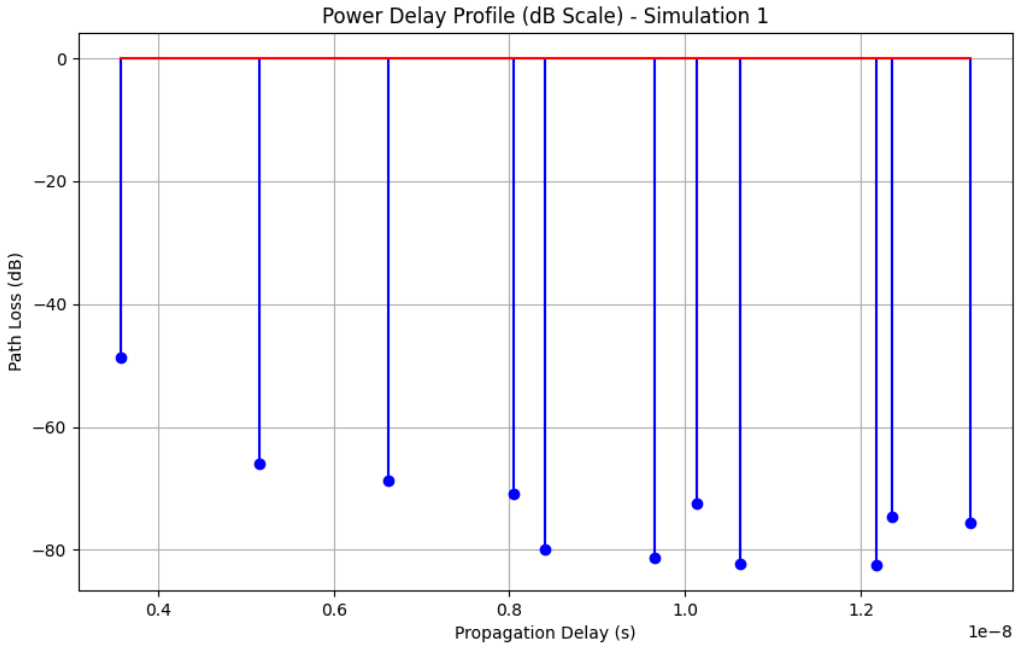


Figure 25

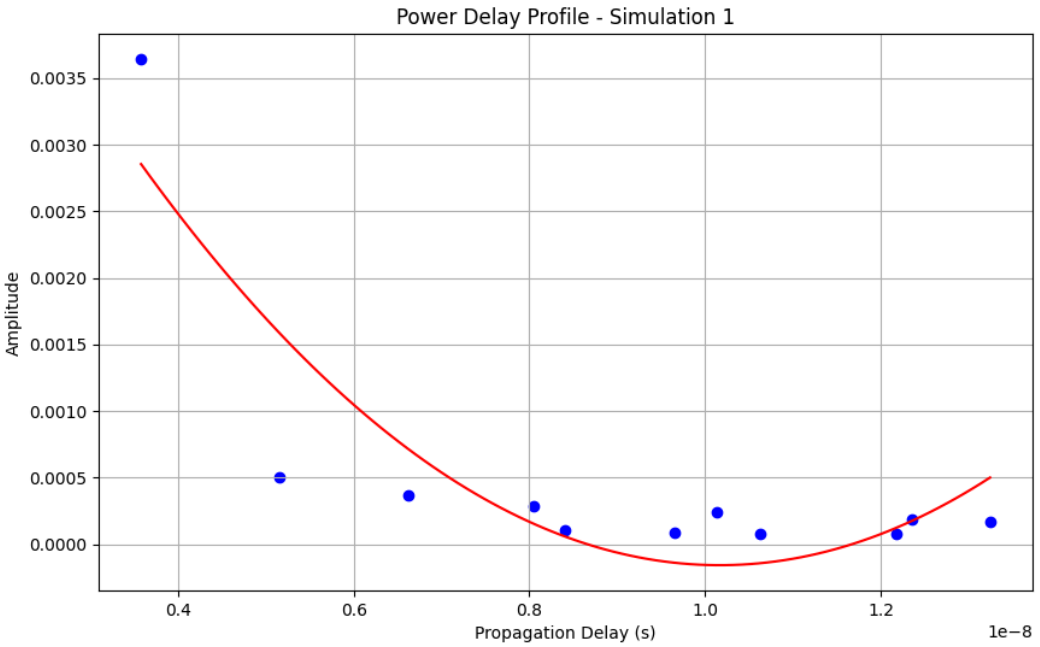


Figure 26

Figure 24: Power Delay Profile (Linear Scale)

This plot shows the Amplitude in linear scale over propagation delay with the predicted number of rays

Figure 25: Power Delay Profile (dB Scale)

This plot shows the Amplitude in decibels (dB) versus the Propagation Delay in seconds.The stems represent the predicted rays, showing their respective delays and Amplitudes.

Figure 26: Power Delay Profile with Polynomial Fit

This plot includes a polynomial fit to the data, providing a curve of best fit.The red curve represents a second-degree polynomial fit, highlighting the trend of the Amplitude over the propagation delay.

Comparing the results of this prediction with the previous one made in section 1.3, we can see that the results are very similar with only one ray missing as the median error for the ray number predicted in section 2.4 .

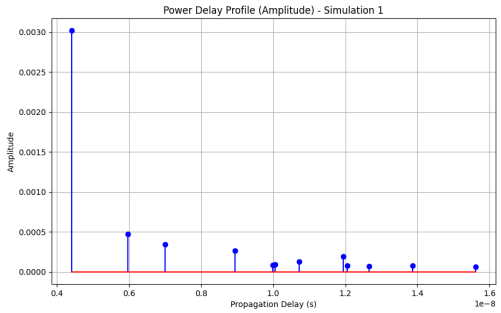
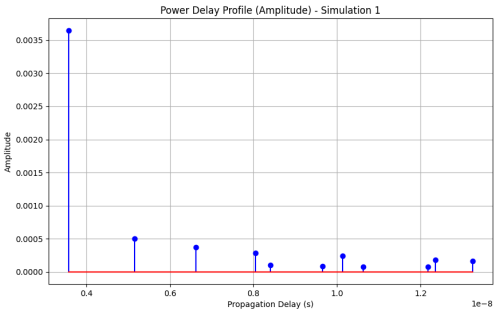
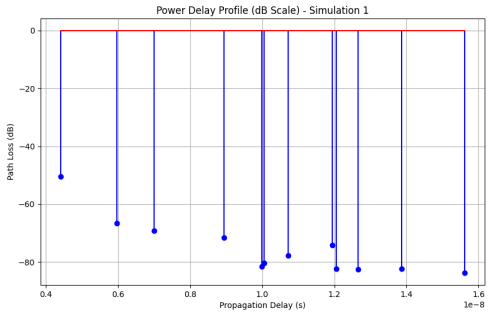
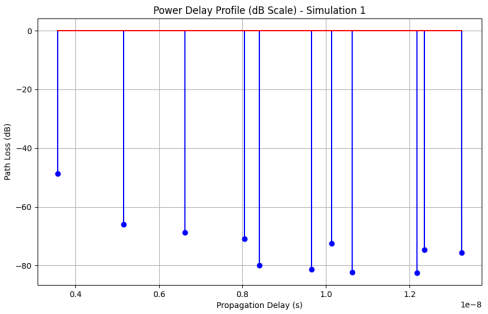
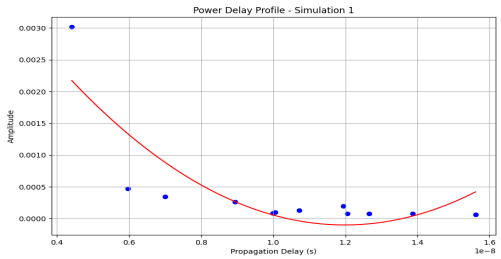
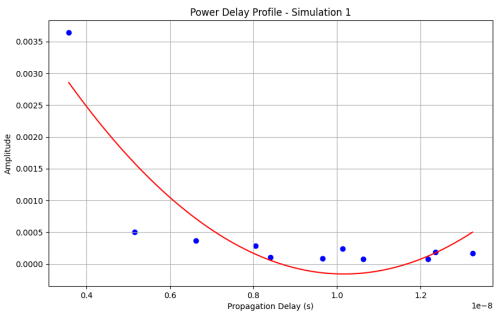
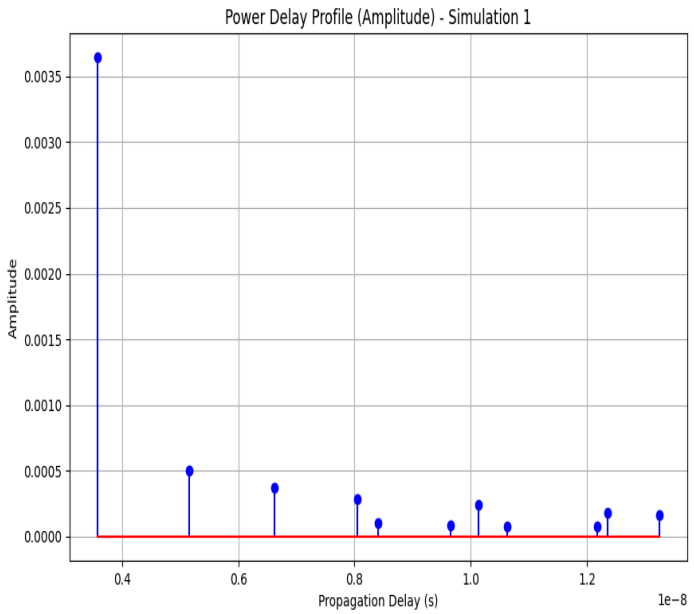
Without Predicting Number of Ray	With Predicting Number of Rays
	
	
	

Figure 27

The comparison of our prediction model with the actual simulation results of the Power Delay Profile are provided below:



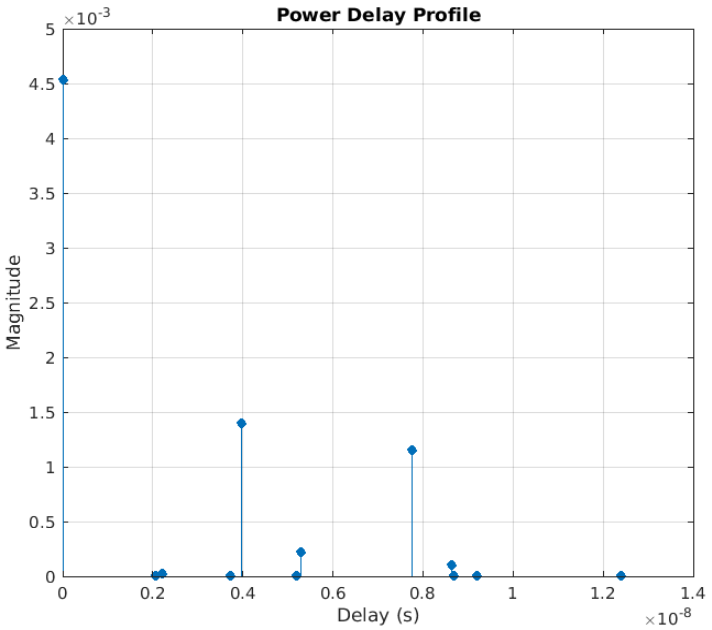


Figure 28

In this comparison example with the actual simulation we can juxtapose the predicted results of the simulation with actual results. These are provided as examples the correctness of the predictions are discussed in greater detail in the error sections.

4.2.5 Error Analysis for Power Delay Profile of Amplitude vs Propagation Delay

```
Errors for Amplitude:
  Mean Absolute Error: 0.00012401906803141463
  Mean Squared Error: 3.8981638958956046e-07
```

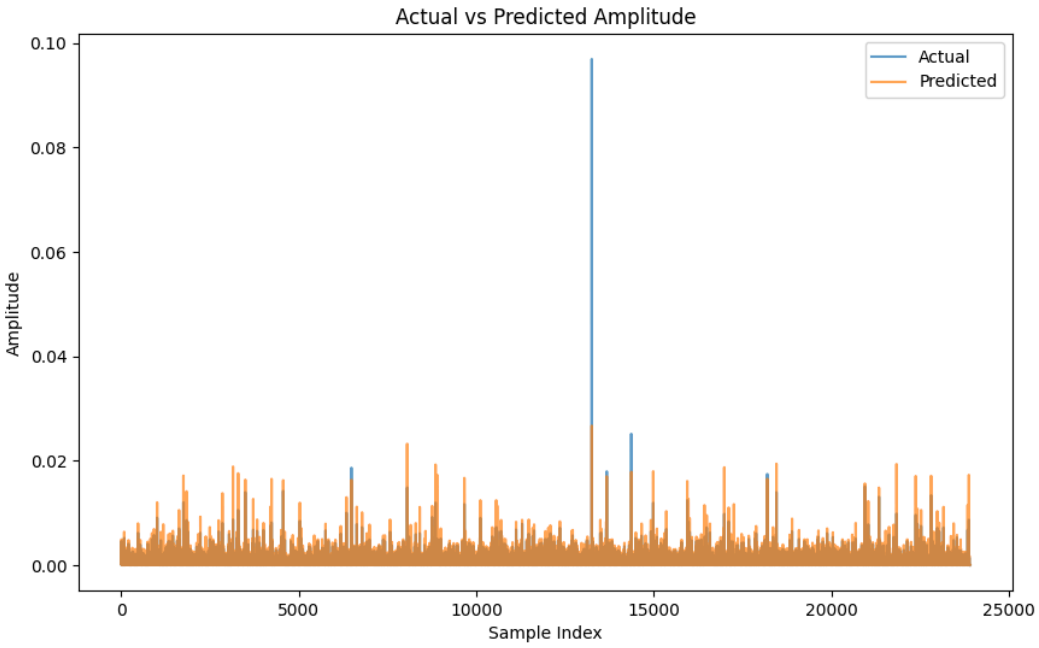


Figure 29

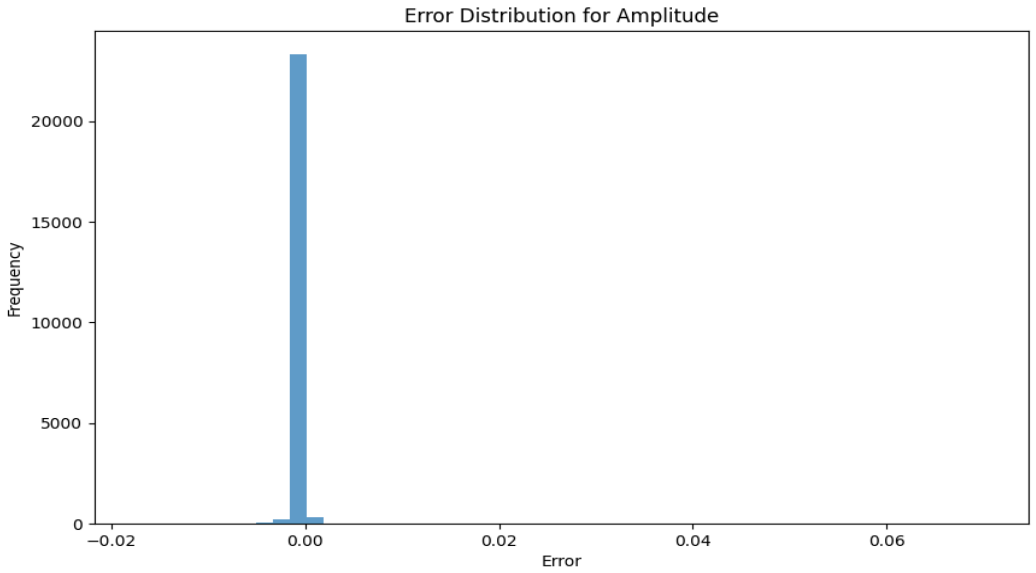


Figure 30

```
Errors for Propagation Delay:
  Mean Absolute Error: 3.177076245403479e-09
  Mean Squared Error: 1.7632334218518292e-17
```

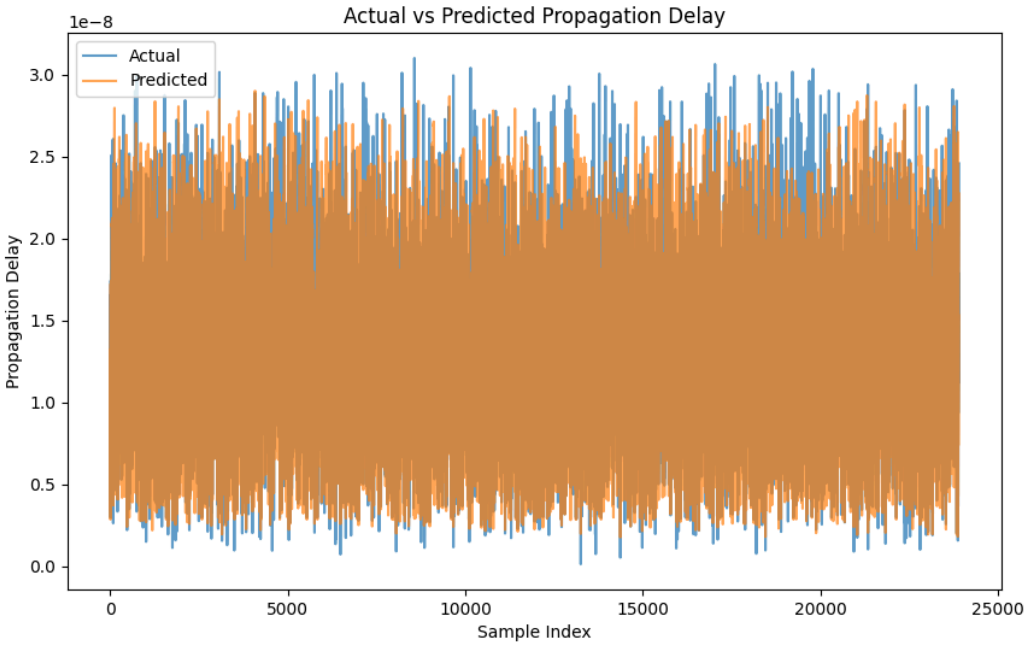


Figure 31

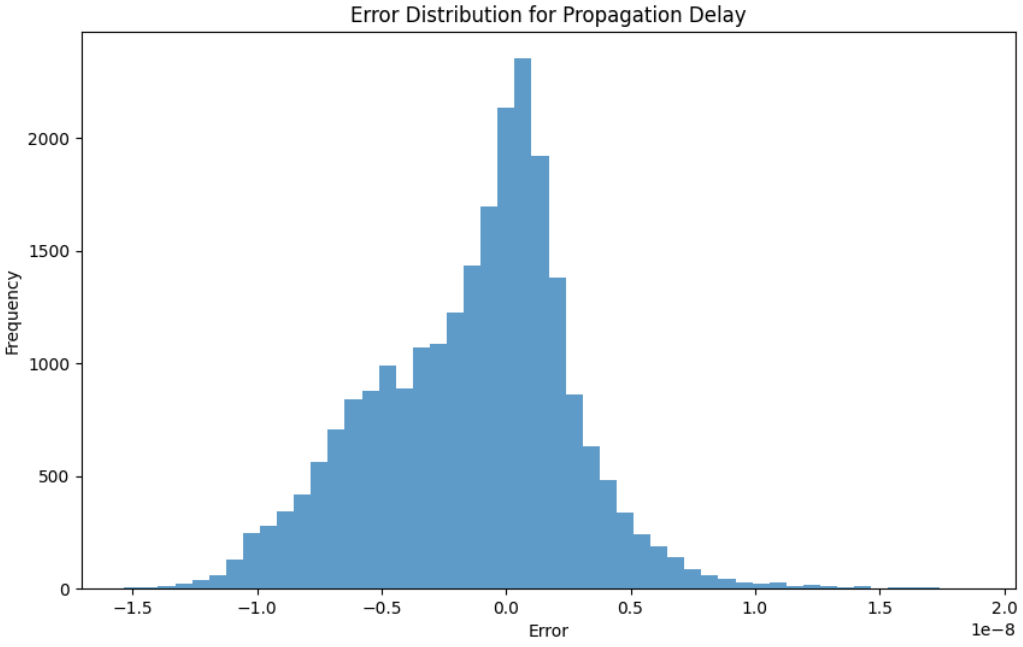


Figure 32

Amplitude Errors:

The errors are very low, with an MAE of about 0.000124 and an MSE of about $3.898 \cdot 10^{-7}$. This suggests that the model is highly accurate in predicting amplitude, with deviations that are very minor.

Propagation Delay Errors:

The errors are even smaller, with an MAE of about $3.177 \cdot 10^{-9}$ and an MSE of about $1.763 \cdot 10^{-17}$. This indicates that the model's predictions for propagation delay are extremely precise, with negligible deviations.

Figure 29: Actual vs Predicted amplitude Loss

In the plot, the blue line represents the actual amplitude values, and the orange line represents the predicted amplitude values. We observe very high overlapping between the lines with minor exceptions.

Figure 30:Error Distribution for amplitude Loss

The distribution is centred around zero, indicating that the errors are balanced and there is no significant bias in the predictions.

Figure 31:Actual vs Predicted Propagation Delay

The results are not affected by the prediction of the number of rays, they are similar with figures 14,20.

Figure 32:Error Distribution for Propagation Delay

Similar to figures 15,21.

4.3 Consideration of Propagating Errors from Predicted Number of Rays

The errors in the number of rays are indirectly considered in the errors for amplitude and propagation delay through the process of prediction:

Error in Ray Number Propagates:

Any inaccuracy in the number of rays will affect the final predictions for amplitude and propagation delay. However, the actual error metrics for amplitude and propagation delay are calculated based on the comparison between the model's predictions and the ground truth data.

The errors from the predicted number of rays are not directly included in the amplitude and propagation delay error calculations. Instead, these errors are propagated through the prediction process, meaning that inaccuracies in the Ray Number will impact the amplitude and propagation delay predictions.

The errors measured here are purely the discrepancies between the predicted values for amplitude and propagation delay against the ground truth values. If the Ray Number predictions were incorrect, this will show up as inaccuracies in the amplitude and propagation delay predictions.

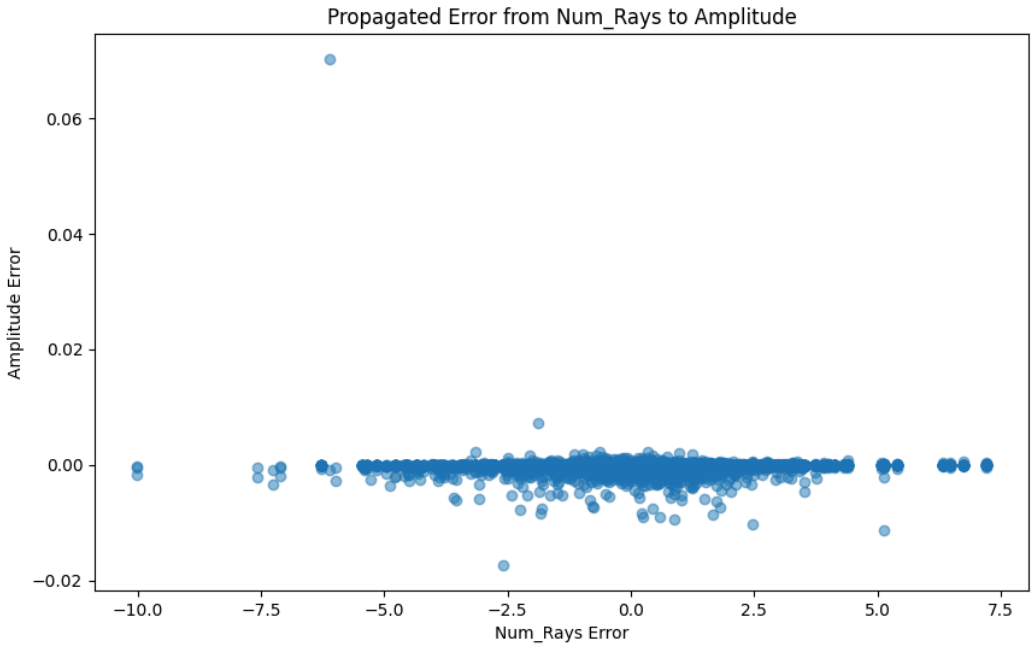


Figure 33

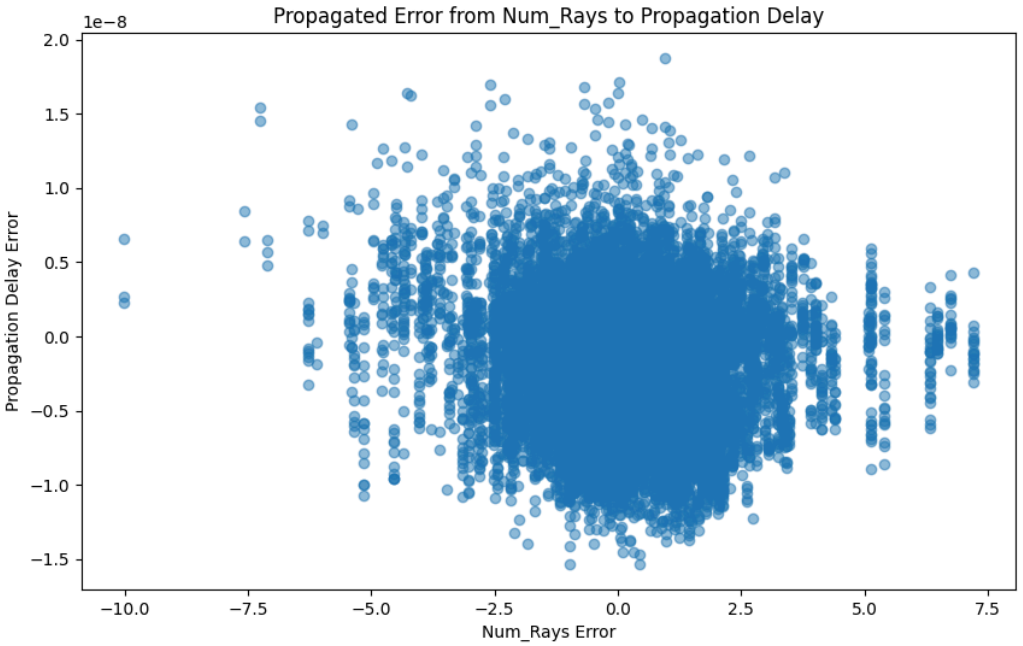


Figure 34

Figure 33. Propagated Error from Num_Rays to Amplitude

X-axis (Num_Rays Error): The prediction error in the number of rays.

Y-axis (Amplitude Error): The prediction error in amplitude.

The plot shows a relatively flat distribution of amplitude errors across the range of number of rays errors. This suggests that the error in predicting the number of rays does not significantly affect the error in predicting amplitude. Most amplitude errors are close to zero regardless of the number of rays error.

Figure 34. Propagated Error from Num_Rays to Propagation Delay

X-axis (Num_Rays Error): The prediction error in the number of rays.

Y-axis (Propagation Delay Error): The prediction error in propagation delay.

This plot shows a more dispersed distribution of propagation delay errors across the range of number of rays errors. Unlike the amplitude error plot, this indicates a stronger relationship between the error in predicting the number of rays and the error in predicting propagation delay. The errors are more spread out, suggesting that errors in the number of rays prediction can significantly affect the accuracy of propagation delay predictions.

Interpretation

Amplitude: The prediction error in the number of rays seems to have a minimal impact on the amplitude prediction errors. The errors in amplitude remain relatively low and constant regardless of how large or small the number of rays error is.

Propagation Delay: The prediction error in the number of rays has an impact on the propagation delay prediction errors. The errors in propagation delay are more variable, indicating that accurate prediction of the number of rays is crucial for reliable propagation delay predictions.

Chapter 5: Conclusions

In this thesis, we explored the prediction of delay spread metrics and power delay profile in indoor MIMO-OFDM communication using ray tracing and AutoGluon. Several different metrics have been predicted, above by different methods, here we present only the results that are provided by the simulation. The results of the various prediction methods were compared to determine their accuracy and efficiency. Below, we summarise the key findings and present the results in a tabulated form for clarity.

Prediction Results

We predicted several metrics including RMS Delay Spread, Max Delay Spread, Min Delay Spread, and Bit Error Rate (BER), etc. The accuracy of these predictions was evaluated using Mean Absolute Error (MAE) and Mean Squared Error (MSE).

Metric	**Mean Absolute Error (MAE)**	**Mean Squared Error (MSE)**
RMS Delay Spread	5.241864282718379e-10	6.459919628935503e-19
Max Delay Spread	1.2937923066779813e-09	3.40077314810215e-18
Min Delay Spread	4.1532811431906516e-10	4.839939379547288e-19
BER	0.02941833295367484	0.004655592120467082
Amplitude	0.00012401906803141463	3.8981638958956046e-07
Propagation Delay	3.177076245403479e-09	1.7632334218518292e-17
Path Loss	3.552975922490833	26.462217921769874
Number of Rays	1.1373197134352497	2.509557889745679

Prediction Errors for Different Metrics

From the table, it is evident that the method predicts Min Delay Spread with the highest accuracy, followed by RMS Delay Spread, Max Delay Spread, and then Propagation delay, followed by a bit higher error rates by amplitude, BER, the number of rays and lastly the Path Loss.

Method Comparison

To compare the performance of different methods, we rank them based on their MAE values:

1. Min Delay Spread: 4.1532811431906516e-10
2. RMS Delay Spread: 5.241864282718379e-10
3. Max Delay Spread: 1.2937923066779813e-09
4. Propagation Delay: 3.177076245403479e-09
5. Amplitude: 0.00012401906803141463
6. BER: 0.02941833295367484
7. Number of Rays: 1.1373197134352497
8. Path Loss: 3.552975922490833

AutoGluon Models Machine Learning Algorithm Selection

To see what machine learning algorithm Autogluon chose for each model, we created a leaderboard file which listed the algorithms and characteristics see [19]. The model chosen for every model was WeightedEnsemble_L3 (except the BER model where the WeightedEnsemble_L2 was selected). The WeightedEnsemble_L3 was chosen because it had the lowest mean squared error on the validation set, meaning it provided the most accurate predictions for each individual model. The ensemble ML algorithms are models which combine the predictions of several other models to improve performance. The WeightedEnsemble_L3 ML algorithm, which had a level 3 stack, is a third-level ensemble of other models (BER model had a level 2 stack). Stacking is a technique where models at higher levels use the predictions from lower-level models to make more informed predictions. While the ensemble (WeightedEnsemble_L3) is the most accurate, it is slower to train compared to other models. Ensemble methods, particularly stacked ensembles like WeightedEnsemble_L3, tend to perform best because they combine the strengths of various base models like neural networks (NeuralNetTorch, NeuralNetFastAI),

Gradient Boosting (XGBoost, LightGBM, CatBoost) ,etc. , leading to better overall performance.

Speed Up Analysis

The average time for a thousand simulations was approximately 25.649 seconds for each individual simulation, while AutoGluon provided nearly instantaneous results. This represents a considerable speed-up, making AutoGluon a highly efficient tool for predicting communication metrics in indoor MIMO-OFDM systems.

In conclusion, using AutoGluon demonstrates that it is not only effective in predicting key metrics with high accuracy but also provides a substantial speed-up compared to the simulation method.

Chapter 6: References

[1] MathWorks. "Indoor MIMO-OFDM communication link using ray tracing," MathWorks, 2024. [Online].Available:https://www.mathworks.com/help/comm/ug/indoor-mimo-ofdm-communication-link-using-ray-tracing.html. [Accessed: Sep. 12, 2024].

[2] Wikipedia. “History of telecommunication,” 2024. [Online], .Available:https://en.wikipedia.org/wiki/History_of_telecommunication. [Accessed: Sep. 12, 2024].

[3] Wikipedia. "Wireless power transfer,” 2024. [Online]. Available:https://en.wikipedia.org/wiki/Wireless_power_transfer. [Accessed: Sep. 12, 2024].

[4] Wikipedia. "Nikola Tesla”, 2024. [Online]. Available:https://en.wikipedia.org/wiki/Nikola_Tesla. [Accessed: Sep. 12, 2024].

[5] Wikipedia. "Wardenclyffe Tower design and operational principles," Wardenclyffe Tower, 2023. [Online].Available:https://en.wikipedia.org/wiki/Wardenclyffe_Tower#Design_and_operational_principles. [Accessed: Sep. 12, 2024].

[6] Wikipedia. "Heinrich Hertz," 2024. [Online]. Available: https://en.wikipedia.org/wiki/Heinrich_Hertz. [Accessed: Sep. 12, 2024].

[7] Wikipedia. "Invention of radio," 2024. [Online]. Available:https://en.wikipedia.org/wiki/Invention_of_radio. [Accessed: Sep. 12, 2024].

[8] Wikipedia. "MIMO”, 2024. [Online]. Available:https://en.wikipedia.org/wiki/MIMO. [Accessed: Sep. 12, 2024].

[9] N. Costa and S. Haykin, *Multiple-Input, Multiple-Output Channel Models*. 2010. [Online]. Available: https://doi.org/10.1002/9780470590676. [Accessed: Sep. 12, 2024].

[10] Wikipedia. "Ray tracing (graphics)," 2024. [Online]. Available: https://en.wikipedia.org/wiki/Ray_tracing_(graphics). [Accessed: Sep. 12, 2024].

[11] MathWorks. "Ray tracing for wireless communications," 2023. [Online]. Available: https://www.mathworks.com/help/comm/ug/ray-tracing-for-wireless-communications.html. [Accessed: Sep. 12, 2024].

[12] S. G. Glisic and P. A. Leppänen, *Wireless Communications*. Boston, MA: Springer US, 1997. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4757-2604-6_16. [Accessed: Sep. 12, 2024].

[13] Z. Yun and M. F. Iskander, "Ray tracing for radio propagation modeling: Principles and applications," *IEEE Trans. Antennas Propag.*, vol. 56, no. 11, pp. 3534–3542, Nov. 2008. [Online]. Available: https://ieeexplore.ieee.org/document/7152831. [Accessed: Sep. 12, 2024].

[14] G. Carluccio and M. Albani, ‘‘An efficient ray tracing algorithm for multiple straight wedge diffraction,’’ IEEE Trans. Antennas Propag.,vol. 56, no. 11, pp. 3534–3542, Nov. 2008.

[15] Corucci, A., Usai, P., Monorchio, A., Manara, G. (2014). Wireless Propagation Modeling by Using Ray-Tracing. In: Mittra, R. (eds) Computational Electromagnetics. Springer, New York, NY. [Online]. Available: https://doi.org/10.1007/978-1-4614-4382-7_17

[16] H. Choi, J. Oh, J. Chung, G. C. Alexandropoulos, and J. Choi, “WiThRay: A Versatile Ray-Tracing Simulator for Smart Wireless Environments,” *IEEE Access*, vol. 11, pp. 56822–56845, Jan. 2023, [Online]. Available: https://arxiv.org/pdf/2304.11385. [Accessed: Sep. 12, 2024].

[17] AutoGluon, "AutoGluon documentation," AutoGluon, 2024. [Online]. Available: https://auto.gluon.ai/stable/index.html. [Accessed: Sep. 12, 2024]

[18] AutoGluon, "AutoGluon leaderboard documentation," AutoGluon, 2024. [Online]. Available: https://auto.gluon.ai/stable/api/autogluon.tabular.TabularPredictor.leaderboard.html. [Accessed: Sep. 12, 2024].

[19] Wikipedia, “Machine learning,” 2024. [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning [Accessed: Sep. 15, 2024].

[20] Wikipedia, “Predictive analytics,” 2024. [Online]. Available: https://en.wikipedia.org/wiki/Predictive_analytics [Accessed: Sep. 15, 2024].

[21] Wikipedia, “Regression analysis,” 2024. [Online]. Available: https://en.wikipedia.org/wiki/Regression_analysis . [Accessed: Sep. 15, 2024].

[22] Wikipedia, “Directed acyclic graph,” 2024. [Online]. Available: https://en.wikipedia.org/wiki/Directed_acyclic_graph. [Accessed: Sep. 15, 2024].

[23] Wikipedia, “Multivariate normal distribution,” 2024. [Online]. Available: https://en.wikipedia.org/wiki/Multivariate_normal_distribution. [Accessed: Sep. 15, 2024].

[24] Wikipedia, “Mutation (genetic algorithm),” 2024. [Online]. Available: https://en.wikipedia.org/wiki/Mutation_(genetic_algorithm). [Accessed: Sep. 15, 2024].

[25] Wikipedia, “Crossover (genetic algorithm),” 2024. [Online]. Available: https://en.wikipedia.org/wiki/Crossover_(genetic_algorithm). [Accessed: Sep. 15, 2024].

[26] Wikipedia, “Chromosome (genetic algorithm),” 2024. [Online]. Available: https://en.wikipedia.org/wiki/Chromosome_(genetic_algorithm). [Accessed: Sep. 15, 2024].

[27] AutoGluon, "How It Works," AutoGluon, 2024. [Online]. Available: https://auto.gluon.ai/dev/tutorials/tabular/how-it-works.html. [Accessed: Sep. 15, 2024].

[28] Erickson, Nick, et al. "AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data," 2020. [Online]. Available: https://arxiv.org/abs/2003.06505 [Accessed: Sep. 15, 2024].

[29] IBM, "What Is Bagging?," IBM, 2024. [Online]. Available: https://www.ibm.com/topics/bagging. [Accessed: Sep. 15, 2024].

[30] AWS Amazon, "What is Overfitting?," ASW Amazon, 2024. [Online]. Available: https://aws.amazon.com/what-is/overfitting. [Accessed: Sep. 15, 2024].

[31] R. Caruana, A. Niculescu-Mizil, G. Crew, A. Ksikes, "Ensemble Selection from Libraries of Models,"[Online].Available: https://www.cs.cornell.edu/~alexn/papers/shotgun.icml04.revised.rev2.pdf. [Accessed: Sep. 15, 2024].

[32] AutoGluon, "autogluon.tabular.models," AutoGluon, 2024. [Online]. Available: https://auto.gluon.ai/stable/api/autogluon.tabular.models.html. [Accessed: Sep. 15, 2024].