Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2022

2D Orientation Estimation Using Machine Learning With Multiple 5G Base Stations

Nikil Johny Kunnappallil and Jianxin Qu

LINKÖPING UNIVERSITY

Master of Science Thesis in Electrical Engineering

2D Orientation Estimation Using Machine Learning With Multiple 5G Base Stations

Nikil Johny Kunnappallil and Jianxin Qu

LiTH-ISY-EX–22/5527—SE

Supervisor: **Isaac Skog, Chuan Huang**
ISY, Linköpings universitet
Deep Shrestha and Yuxin Zhao
Ericsson Research, Linköping

Examiner: **Fredrik Gustafsson**
ISY, Linköpings universitet

Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Abstract

Localization of mobile devices has implications on a multitude of use cases such as estimating the location of the user originating an emergency call, localization of devices to enable autonomous operation required by industrial Internet of Things (IoT) use cases, etc. In futuristic use cases such as Augmented Reality (AR), Virtual Reality (VR), Extended Reality (XR), autonomous navigation of Unmanned Aerial Vehicles (UAVs), we will require the capability of estimating orientation in addition to position of such devices for efficient and effective provisioning of these services to the end-users.

One way to handle the problem of finding the orientation of devices is to rely on the measurements from different sensors like the magnetometer, accelerometer and gyroscope but the limitation of this method is the dependency on these sensors, and thus cannot be used for some devices which does not have these sensors. Hence these limitations can be overcome by using data-driven approaches like Machine Learning (ML) algorithms on received signal features, where a training dataset with orientation measurements are used to train the ML model that can transform the received signal measurements to orientation estimates.

The data for the work is generated by using simulator that can simulate the environment with multiple base stations and receivers. The measurements or features that are generated from the simulator are the Received Signal Received Power (RSRP), Time of Arrival (ToA), Line of Sight (LoS) condition, etc. In-order to find the relationship between the received signal features and orientation, two nonlinear ML algorithms namely K Nearest Neighbors (KNN) and Random Forest (RF) are used. The received measurements were investigated and RSRP was identified as the feature for the ML models.

The ML algorithms are able to estimate the orientation of the User Equipment (UE) by using KNN and RF, where different features likes RSRP and the information about LoS and Non Line of Sight (NLoS). These features were used alone and also combined to evaluate the performance. The results also shows how interference of radio signals affects the performance of the model. Adding to that, different combination of received signal features were also used to compare the performance of the model. Further tests were also done on the trained model to identify how well it can estimate orientation when a new UE with new position is introduced.

Acknowledgments

This thesis concludes our master's studies and these two years studying at Linköping University. The journey in Sweden started with a lot of differences as we were both from different cultures and backgrounds.

Thanks to the Ericsson Research Linköping, LINLAB for allowing us to work on the interesting thesis topic. Special thanks to the supervisors at Ericsson, Deep Shrestha, and Yuxin Zhao, it would have been impossible to complete the thesis without their input throughout the thesis, answering all the questions that we had and giving us directions.

We would also like to thank our supervisors Isaac Skog and Chuan Huang, and our examiner Fredrik Gustafsson, for providing guidance and new ideas. The inputs from Deep, Yuxin, Chuan, and Isaac helped us to create the thesis report.

We would also like to use this opportunity to thank all the teachers, friends, and colleagues we had during the whole two-year master's program. All of them taught us a lot in both academics and social life, and together made beautiful and memorable moments in life. They were always with us when we were in need and helped us to grow.

Family is the reason why we are here, their support through these times especially when you are far away from home was valuable and helped us to complete our studies on time. We are grateful for their unconditional love and care.

Nikil would like to thank Abin, Agnes, Akshay and Amal for support in different ways during my studies and thesis project so that I was able to complete the journey of my master's thesis on time. Finally, thanking God for everything.

Jianxin would like to thank Ahmad, for all his love and support. And the encouragement and company from family and friends.

Linköping, June 2022
Nikil Johny Kunnappallil and Jianxin Qu

Contents

List of Abbreviations

Abbreviation	Full Form
ANN	Artificial Neural network
AoA	Angle of Arrival
AoD	Angle of Departure
AR	Augmented Reality
BS	Base Station
CDF	Cumulative Distribution Function
CNN	Convolutional Neural Network
3D	3 Dimension
6D	6 Dimension
DL	Deep Learning
DNN	Deep Neural Network
EXIP	Extended Invariance Principle
GPS	Global Positioning System
IoT	Internet of Things
IMU	Inner Measurement Units
KNN	K Nearest Neighbors
LiFi	Light-Fidelity
LoS	Line of Sight
MARG	Magnetic-Angular Rate-Gravity
MIMO	Multiple-Input Multiple-Output
ML	Machine Learning
MLP	Multilayer Perceptron
NaN	Not-a-Number
NED	North-East-Down
NLoS	Non Line of Sight
NR	New Radio
OFDM	Orthogonal Frequency-Division Multiplexing
PRS	Positioning Reference Signal
RF	Random Forest
RMSE	Root Mean Squared Error
RSS	Received Signal Strength
RSRP	Received Signal Received Power
ToA	Time of Arrival
TRP	Transmission and Reception Point
UAV	Unmanned Aerial Vehicle
UE	User Equipment
VLP	Visible Light-Based
VR	Virtual Reality
WLAN	Wireless Local Area Network
XR	Extended Reality

1

Introduction

1.1 Background

A User Equipment (UE) can be any device that is used by an end user for communication such as, a mobile device, laptop, etc. A hardware that is installed on a device that is used to transmit and receive signals during communication is called the antenna panel. Orientation of a UE can be defined as the angle between the antenna panel and a reference plane as shown in Figure 1.1. Orientation of the UE helps to identify the direction to which the UE is heading with respect to the reference plane. Orientation can be defined in 3 dimensions in terms of yaw, pitch and roll as represented in Figure 1.1. This work focuses on estimating the orientation in terms of yaw.

Currently, accurate localization of mobile devices has implications for a multitude of use cases such as estimating the location of the user originating an emergency call (even when the user is in a Global Positioning System (GPS) denied zone), localization of devices to enable autonomous operation (both indoor and outdoor) required by industrial IoT use cases. It is anticipated that the futuristic use cases such as AR, VR, XR, autonomous navigation of UAVs, will require capability beyond the positioning of the devices for efficient and effective provisioning of these services to the end-users. The extended information that is vital for these use cases, among many others, is the orientation estimation of the UE to determine the heading of the device.

For orientation estimation, sensors are typically used to obtain the orientation of the user equipment. For example, smartphones use inner measurement units (IMUs) and magnetic-angular rate-gravity (MARG) units to capture user's motion to estimate orientation. Generally speaking, three types of sensors are

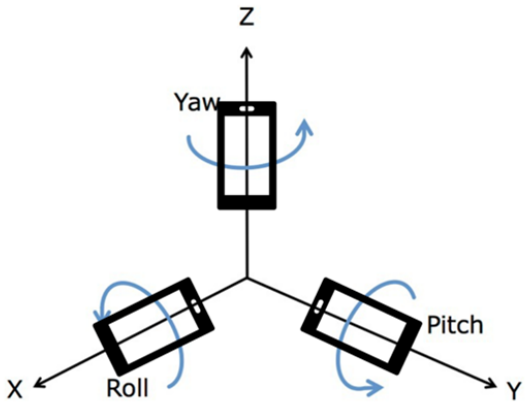


Figure 1.1: *UE Orientation in 3D.*

commonly used for orientation estimation: accelerometers, gyroscopes, and magnetometers [1]. Accelerometers measure the direction of acceleration in 3D space. It consumes little power but has low precision. Gyroscopes measure angular velocity of motion along one or several axes and are an ideal technology to complement the capabilities of accelerometers. In fact, by combining the two sensors, accelerometers and gyroscopes, system designers can track and capture complete motion in 3D space, providing end users with a more realistic user experience, accurate navigation systems, and other capabilities. Magnetometers measure the local magnetic field. It is a navigation instrument that can identify the global coordinate system which is North-East-Down (NED) reference frame [2]. Based on the measurements of sensors, many different algorithms are designed to calculate orientation. Grace Wahba in 1965 published the first solution in history for definition of the main rotational problems related to spacecraft attitude [3]. Since then, the algorithms are gradually upgraded to compute more accurate results based on multiple information sources. Gebre-Egziabher in 2004 combined all three types sensors to determine attitude [4].

However, all sensors have measurement error. A gyroscope measures the relative changes in orientation, which drifts slowly over its age and temperature. An accelerometer measurements are polluted by uncorrelated dynamic accelerations such as the gravity acceleration. A magnetometer measurements are disturbed by various magnetic disturbances such as the local magnetic anomalies. Therefore, the main challenge of fusion algorithms is to eliminate errors to get accurate estimates [5]. The solution of most fusion algorithms is to use the predictor–corrector structure, i.e. gyroscope measurements are applied as predictions, accelerometer and magnetometer measurements are applied to correct predictions. Kalman filter is one of the most popular fusion methods [6], its principle is

to use the Kalman gain to modify the state prediction value to make it approach the real value. The limitation factor is the assumption that both the process and the measurement noise are based on Gaussian distribution, although there are some extended solutions based on adaptive filters, the expensive computational cost does not make them a better alternatives for small scale applications. Another fusion method is the complementary filtering [7], which designs filters to filter out disturbances. This method requires less computational cost, but the parameter design of the filters is complicated, especially when applied to time-variant systems.

Considering that these sensors cost less than one dollar, consume little power, respond rapidly, and the results are generally accurate, it is always a good idea to consider using sensors to do orientation estimation. But if the application is limited, e.g., the sensors are not available in some scenarios, is there any other way to obtain the UE orientation? Considering that most UEs install antennas for wireless communication, radio signal based methods are popular research topics for high-accuracy orientation solutions, especially for numerous fifth-generation mobile network enabled applications. In this project, the effect of UE orientation on the received signal feature will be studied. The insights from the study will then be used to develop a learning method for UE orientation estimation.

The environment that is explored in this study is an indoor-office scenario with stochastic wireless transmission channels. The scenario uses a downlink transmission with transmitters and receivers. From the signals received at the receiver, the orientation of the device is estimated with LoS and NLoS propagation paths. LoS refers to the path propagation of radio signals where there are no obstacles between the transmitter and the receiver. As the name infers NLoS propagation paths occur when the path between transmitter and receiver is obscured (partially or completely) by obstacles. The effect of NLoS and LoS on orientation will also be investigated in this paper.

The term ML was popularised by Arthur Samuel in 1959 and he created the first self-learning program for playing checkers [8]. ML is the process of extracting useful information from the mathematical model of the data. The mathematical model describes the relationship between the different variables in the observed data [9] and is written as a computer program. The computer program learns from the available sample data or training data, by capturing the important information and automatically adjusting the settings, parameters in precise mathematical form so as to agree with the data. Once the computer program or algorithm learns from the training data, this machine learning model can be used to make predictions or decisions without needing explicit programming.

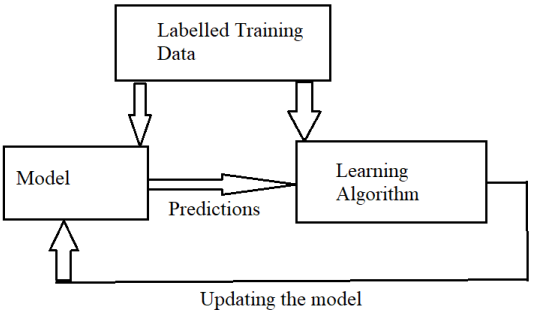


Figure 1.2: *Describes how a model is trained using supervised machine learning.*

For ML, the data is a very important factor. Labelled training data contains multiple instances of an input variable $\boldsymbol{x}$ accompanied with a labelled output variable $\boldsymbol{y}$. When the model is trained by these inputs and outputs that are labelled by experts, it is called as supervised learning as shown in Figure 1.2. Another category of ML algorithms is unsupervised ML algorithms that can be used to cluster and analyze unlabelled datasets, where the output variable $\boldsymbol{y}$ is not present.

The training data in supervised learning contains input variable $\boldsymbol{x}$ that is related to output variable $\boldsymbol{y}$, and the aim is to adjust the mathematical model during training of the model, in such a way, when a new unseen test data $\boldsymbol{x_*}$ is given, it will be able to predict $\hat{\boldsymbol{y}}$ that is close to $\boldsymbol{y_*}$. In other words, supervised learning can be described as learning from examples. Thus training data which is the input-output data points $(\boldsymbol{x_i}, \boldsymbol{y_i})$ out of the total $\boldsymbol{n}$ number of them can be represented as $\boldsymbol{T} = (\boldsymbol{x_i}, \boldsymbol{y_i})$ for $\boldsymbol{i} = \boldsymbol{1, ..., n}$. Thus the aim of supervised ML is to obtain as much information as possible from $\boldsymbol{T}$.

Supervised Learning algorithms are classified into two general types of problems called as classification and regression problems based on the type of the output variable $\boldsymbol{y}$. The task of ML in classification problem is to accurately predict the category of each data point, i.e., the output variable is a categorical value. Unlike classification, in regression the output to be predicted is a numerical value. The input to the model can be a vector $\boldsymbol{X} = [\boldsymbol{x_1 x_2 ... x_p}]^T$ which can be $\boldsymbol{p}$ dimensional, where each element $\boldsymbol{x_1, x_2 ... x_p}$ contains information that can be relevant to the problem, and these elements can contain both categorical and numerical values. In our thesis we have a regression problem to solve since we need to predict the orientation of the UE as our output $\hat{\boldsymbol{y}}$ which is a continuous numerical value and so the focus will be on regression.

Mathematical modelling of input-output relationships from the training data is very important as it helps to reason about and understand the relationship

between the input and the output [9]. Mathematical modelling also helps to generalize the relationship between inputs and outputs, which in turn helps to make predictions $\hat{y}$ for previously unseen test data. Thus, ML algorithms are techniques for learning a target function f that best maps the input variable x to an output variable y which can be represented as:

$$y = f(x) \tag{1.1}$$

Once the function f has been learnt from the available data, then this function can be used to make predictions $\hat{y}$ for a new unseen datapoint x_* as:

$$\hat{y} = f(x_*) \tag{1.2}$$

Different ML algorithms make different assumptions about the underlying functions, thus we have to try different algorithms to find out the best suiting algorithm for the problem at hand. And this thesis aims to try different algorithms to find the best target function (f) that can predict the orientation of the UE.

In order to answer the question of finding the best ML model, we need some tools to evaluate the different models and hence, this will help us to improve the model performance. In supervised machine learning, generalization error or out-of-sample error is the measure of the ability of an algorithm to accurately predict the outcome of previously unseen data [10]. In order to understand the idea, an error function $E(\hat{y}, y)$ is defined, which has the predicted value $\hat{y}$ and measured data point y as the inputs. The error function compares the prediction and the measured data point or true value and help us to analyze the performance of an already learned model. There are different types of error functions that are decided based on the properties of the predictions. The common choice of the error function for regression problems is the squared error which is taking the square of the difference between the predicted value and the truth value as:

$$E(\hat{y}, y) = (\hat{y} - y)^2 \tag{1.3}$$

On an endless stream of unseen data, generated from distribution over data $p(x, y)$ the average squared error can be used to denote mathematically describe the performance of a model. As the model depends on the training data T, we can write the prediction as $\hat{y}(x; T)$, i.e., if we use a different training data to learn the same model, it would result in a completely different model. Thus the new expected data error which is the average over possible data points (x_*, y_*) is

$$E_{new} = \mathbb{E}_*[E(\hat{y}(x_*; T), y_*)], \tag{1.4}$$

Where $\mathbb{E}_*$ is the expectation over all possible data points with respect to the distribution (x_*, y_*). Thus, E_{new} is the new data error and describes how well the

model generalizes from the training data to a new situation. Similarly we can also introduce the training error as

$$E_{train} = \frac{1}{n} \sum_{i=1}^{n} E(\hat{y}(x_i; T), y_i)]. \tag{1.5}$$

E_{train} does not give any information about how well the model performs on unseen data, but it describes how the model performs on the seen training data. In essence, the goal of any supervised ML algorithm is to have the least value for E_{new}. E_{new} helps to choose the hyper parameters, compare different models and also helps us to understand whether the model performance is satisfying relative to other models. It should be noted that there can be situation where we a have small E_{train} but still large E_{new} when given unseen data and this problem is called as overfitting which will be elaborated later.

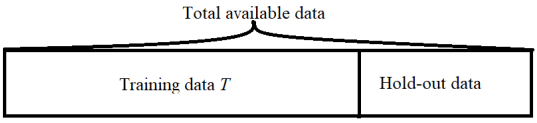


Figure 1.3: *Describes hold-out dataset approach where the available data is split into two sets and the model is trained on the training data and* $E_{hold-out}$ *can be estimated using the hold-out dataset.*

Since we do not know the data distribution $p(x, y)$ in practical cases, we are unable to compute E_{new}. Hence, we use a technique to set aside some hold-out data (x_j, y_j) where j in the range of 1 to $n_{holdout}$. This hold-out data is not in T that is used for training as shown in Figure 1.3. The hold-out error is represented as

$$E_{hold-out} = \frac{1}{n_{holdout}} \sum_{j=1}^{n_{holdout}} E(\hat{y}(x_j; T), y_j). \tag{1.6}$$

$E_{hold-out}$ is an unbiased estimate of E_{new} since all the data points are chosen from $p(x, y)$. The size of the hold-out dataset $n_{holdout}$ should be big enough so that there is less variance in $E_{hold-out}$, but this will decrease the amount of training data. Therefore, the holdout data set size $n_{holdout}$ should be chosen appropriately.

As already described, the main goal of supervised ML is to reduce the E_{new} and in order to understand the concept more clearly, two new terms are introduced which are $\overline{E}_{new}$ and $\overline{E}_{train}$. Consider that the model is trained on different training sets of same size to obtain $(E_{new})_k$, where k ranges from 1 to m. Similarly we also calculate $(E_{train})_k$, where k ranges from 1 to m. Thus $\overline{E}_{new}$ and

$\overline{E}_{train}$ can simply be the average of unseen data error and training error that is represented as,

$$\overline{E}_{new} = \frac{(E_{new})_k}{k}, \quad \overline{E}_{train} = \frac{(E_{train})_k}{k}. \tag{1.7}$$

In general, a method usually performs better on the trained data and worse on the new, unseen data, i.e. , $\overline{E}_{train} < \overline{E}_{new}$ [9]. Generalization of a method is very important as it explains its ability to perform well on unseen data after being trained. Difference between $\overline{E}_{new}$ and $\overline{E}_{train}$ is called as the generalization gap,

$$Generalization\ gap = \overline{E}_{new} - \overline{E}_{train}. \tag{1.8}$$

The problem and the method at hand determines the generalization gap, if the method adapts to the training data more, the larger the generalization gap. Model complexity is another term which describes the ability of the method to adapt to the patterns in the training data. In general it can be said that, a model with high complexity can learn complicated input-output relationships, whereas a model with low complexity has limitations in what relationships it can explain. Typically, it can be observed that the $\overline{E}_{new}$ attains a minimum value for some intermediate value of model complexity and for $\overline{E}_{train}$ the value decreases as the model complexity increases as seen Figure 1.4. Overfitting happens to a model when the value of $\overline{E}_{new}$ is higher than with a less complex model. On the other hand when the model complexity is very low, it is termed as underfitting. Hence, the aim is to find a balance between underfitting and overfitting, i.e, we have to find the spot where $\overline{E}_{new}$ attains the lowest possible value as described by the green line in the Figure 1.4. In our problem of orientation estimation, we also aim to find this balance between overfitting and underfitting.

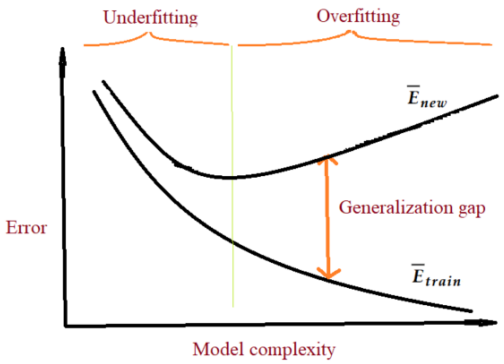


Figure 1.4: *Describes how the values of $\overline{E}_{new}$ and $\overline{E}_{train}$ changes with the model complexity and the difference between these values are called generalization gap. If the model is too complex then $\overline{E}_{new}$ increases again and it is called as overfit and when the model is less complex it is called as underfitting. Hence, the model should have a balanced complexity which is shown by the green line.*

Bias-variance decomposition is a way of examining the expected generalization error $\overline{E}_{new}$ by using the bias, variance and a third unavoidable quantity called as irreducible noise (σ). The term variance can be described as how the prediction changes when trained on different training sets. On the other hand, bias means the difference between the predictions and the truth values. The irreducible error σ^2 is the variance of the independent noise, i.e., random error that is independent of all the other variables. From [9] it can be shown that $\overline{E}_{new}$ can be decomposed as follows:

$$\overline{E}_{new} = Bias^2(\hat{y}(x_*; T)) + Variance(\hat{y}(x_*; T)) + \sigma^2 \tag{1.9}$$

For the bias term to be small, the prediction $\hat{y}$ should be close to the true value y. When the variance term is small it means that the model is not sensitive to the data points that happened to be in the training data set. A high model complexity generally entails that the model performs well on the training data, thus having low bias. Also, when the model is very complex, the variance in prediction is large across different training datasets due to its dependence on the training data. On similar lines, it can be argued that a model with lesser complexity has a high bias and a low variance. Thus the task of achieving a small E_{new} by having the optimal model complexity level is called the bias-variance tradeoff. These concepts are important to find an optimal model that can generalize beyond the training data.

The aim of this paper is to first identify the input variables $x_1, x_2, ..., x_p$ and associated output variable y from a simulated environment. Feature selection techniques discussed in 3.2 are then used to select only the relevant features from the input vector X where $X = [x_1, x_2, ..., x_p]$. Once the important features are

identified, different ML methods are explored to learn an optimal function f which can accurately estimate the orientation $\hat{y}$.

1.2 Problem Formulation

1.2.1 Problem Statement

Based on the introduction and aim, the following are the research questions that will be addressed in the thesis.

1. Study the impacts of UE orientation on received signal features ($\boldsymbol{X} = [x_1 x_2 ... x_p]^T$) such as ToA, RSRP, LoS condition, Identify which are the features that are important for the ML model to estimate the orientation $\hat{y}$ of a UE by utilizing the data generated from the simulated environment?
2. How do the radio frequency interference and the information about LoS and NLoS affect the performance of the ML model?
3. Which ML model has the best performance, given features available?

To investigate the above questions, data was generated by using the Ericsson simulator. This data was used for studying the different features associated with orientation. The data is used for both training and evaluating the ML algorithms in different LoS condition. There is a limitation that different frequency and antenna patterns will not be considered for this study. Even though the orientation is 3D can be in terms of roll, yaw, and pitch, the investigation done in this paper is limited to one dimension due to the limited time frame available for the study.

1.3 Related Work

5G networks are widely used today, and their high data rate communication capabilities make their potential in positioning applications greatly tapped. Afterward, the potential of 5G in orientation estimation needs to be discovered. There have been some recent studies applying 5G mmWave systems for UE orientation estimation. As was demonstrated in [11], in the scenario of communicating with a BS, the UE orientation in 2D can be derived by measuring the angle on the antenna array at the UE side. In [12], more features from channel estimation like AoA at the UE side, AoD at the BS side and ToA are used for high-accuracy estimation of the UE orientation. Another study [13] for visible light system was done by using Received Signal Strength (RSS), ToA and AoA are used as the main features for the deep learning techniques to estimate the UE orientation. In [14], it is derived that the UE orientation in 3D needs more input resources to be sufficient to get estimation, such as multiple available BSs, the known position of reflectors and scatterers, or the known environment.

In this paper, we consider using ML methods to analyze the input features and estimate a UE orientation. In [15], it uses KNN algorithm to estimate the UE

position through the received signal strength indicator (RSSI) sequences received from multiple WiFi access points (APs). The paper considers the human body as an obstacle, and different UE orientations affect the received RSSI, so the model is trained by collecting RSSI sequences of different UE orientations in multiple reference points (RPs), thereby estimating the unknown position of the UE. In addition, the KNN and Random Forest algorithms background theories can be found in [9].

The common part of the above papers is that they focus on estimating the position of the UE, and the UE orientation estimation as an aid to enhance the accuracy of the position estimation. In this paper, we will focus on the estimation of the UE orientation without considering the position estimation.

1.4 Why ML?

ML is having success across many domains in both industry and the research communities, especially in wireless communication, with a focus on networking, resource management, and localization [16]. Hence ML algorithm was chosen to apply to this problem of estimating the UE orientation. Another reason why ML is a potential candidate is that the existing approaches confirms that received signal features have a nonlinear relationship with orientation and ML algorithms are good at finding the nonlinear relationships.

In our task to estimate the orientation of the UE, we need an algorithm that can exploit the information contained in the features that are available to us from the simulator. ML algorithms are popular for identifying complex relationships [16], and this property can be utilized to identify the relationship between the different features and the orientation degrees.

Existing systems make use of the sensors in the mobile device to estimate the orientation of the device. But if we create an ML model, the dependency on the mobile device and its sensors can be reduced, and thus create a model that can be used for all devices.

This chapter introduced the problem and gave an idea about how to approach the task. The next chapter explains the theoretical background that is essential to solving the problem like the features to be analyzed and the ML algorithms.

2

Theoretical Background

2.1 Machine Learning

This section introduces the various machine learning approaches explored in the thesis study.

2.1.1 KNN

KNN is a non-parametric supervised learning algorithm that can be used for both regression and classification problems. The concept of KNN rule was introduced by Fix and Hodges for pattern classification [17] in 1951. KNN is known for its simplicity and effectiveness as it is based on estimating target data point by finding the similarities in the underlying training data. In order to find the similarities, KNN algorithm calculates the distance between the data points, and the smallest distances are identified as the nearest neighbors. The initial letter K defines how many nearest neighbors should be examined to determine the value of target data point [18]. In regression tasks, the output value is the average of all the K nearest neighbors. In classification tasks, the output is a class membership which is determined by choosing the class which is the most common among its neighbors.

As introduced in the Chap. 1 the training data (x_i, y_i) where i in the range of 1 to n and x_i is the input and y_i the corresponding output. KNN also works on the general idea of all supervised machine learning algorithms that if the test point x_* is close to the training data point x_i, then it means that the prediction $\hat{y}(x_*)$ should also be close to y_i [9].

According to KNN, when new data is encountered, two operations are performed. The first step is to analyze which are the nearest neighbors by using

different methods like Euclidean distance, correlation matrix, etc. The Euclidean distance calculates the distance between the test input and all the training inputs, as shown in the Eq. (2.1), where $i = 1,...,n$. The second step is to find x_j which has the shortest distance to x_* and use y_j as the output, i.e, $\hat{y}(x_*) = y_j$.

$$\|x_i - x_*\|^2 = \sqrt{(x_{i1} - x_{*1})^2 + (x_{i2} - x_{*2})^2}. \tag{2.1}$$

This method is called one nearest neighbor method as its prediction depends only upon one data point from the training data. This is simple and not very complicated and the disadvantage is that the output will be erratic and sensitive to noisy training data. In order to overcome this problem we can consider neighbors, i.e., K number of nearest neighbors. Thus we define a set $N_* = \{\, i$: x_i is one of the training data points that is close to the data point $x_* \,\}$ and the information from the K outputs y_j for $j \in N_*$ is combined to make the final predictions. For regression problems we take the average of all the y_j and for classification we take the majority vote. We can summarize the method for KNN as shown,

---

Data: Training Data (x_i, y_i) for i = 1 to n and test input x_*

Result: Prediction output $\hat{y}$

1. Distances $\|x_i - x_*\|^2$ are computed for for all the data points in the training data from i = 1,...,n

2. Let $N_* = \{\, i$: x_i is one of the training data points that is close to the data point $x_* \,\}$

3. Final output $\hat{y}$ is calculated,

$\hat{y} = Average(y_j : y \in N_*) - Regression$

$= Majority(y_j : y \in N_*) - Classification$

There are different ways of calculating nearest neighbors using Euclidean distance, two weight schemes can be considered. The first one is 'uniform' weights, where all the points in the neighborhood are weighted equally. The second weight function is 'distance' weights, where each point is given weight by the inverse of its distances, which in turn will make the closest neighbor has a greater influence than neighbors who are farther away. It can be represented as Eq. (2.2), where W is the weight and d is the distance to the neighbor

$$W = 1/d. \tag{2.2}$$

Another method to calculate the neighbors is using the correlation distance. Correlation coefficient is the strength and direction of a relationship between two vectors and correlation distance is calculated by one minus correlation coefficient, which is expressed as follows

$$d_{st} = 1 - \frac{(\mathbf{x}_s - \overline{\mathbf{x}_s})(\mathbf{y}_t - \overline{\mathbf{y}_t})'}{(\sqrt{(\mathbf{x}_s - \overline{\mathbf{x}_s})(\mathbf{x}_s - \overline{\mathbf{x}_s})'})(\sqrt{(\mathbf{y}_t - \overline{\mathbf{y}_t})(\mathbf{y}_t - \overline{\mathbf{y}_t})'})}, \tag{2.3}$$

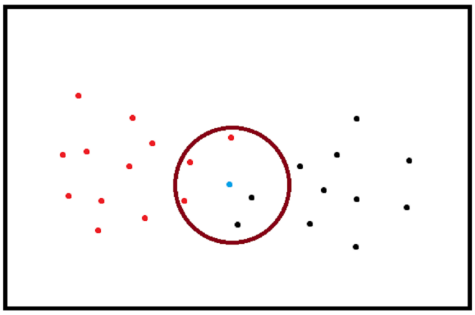


Figure 2.1: *A simple problem using KNN where the number of neighbors set to five. There are two different classes of red and black. The new point blue is classified as red based on the majority vote as we have three red points and two black points among the five nearest neighbors.*

where the two vectors $\mathbf{x}_s$ and $\mathbf{y}_t$ are row vectors , and $\overline{\mathbf{x}_s}$ and $\overline{\mathbf{y}_t}$ are the mean vectors of $\mathbf{x}_s$ and $\mathbf{y}_t$ respectively.

A visualisation of how KNN works can be seen in Figure 2.1, where the number of neighbors is set to five and hence we can see that there are five points in the circle. We can see that there are two different classes which are red and black. In the case of classification, the new point blue is classified as red since the majority of its neighbors belong to red among all the five nearest neighbors. In the case of regression if we consider five nearest neighbors, the output from the regressor will be the average of all the five neighbors.

Determining the optimal value of K for the problem is very important and since K is not a self learned by KNN, it is a design choice that needs to be made and thus referred to as hyperparameter. Finding optimum K requires some iterative investigation. The choice of hyperparameter K has a big impact on the predictions made the KNN algorithm [9]. As already discussed, real world problems usually have a certain amount of randomness in the data hence when K=1 the model gives a shaky behaviour. This is because the model adapts closely to the training data and the predictions will depend not only on the interesting patterns in the problem, but also on the random effects that is present in the training data, thus we can say that overfitting happens. Thus with KNN we can mitigate overfitting by increasing K but this cannot be true for all the cases as this depends on the data and the problem that is studied. However, if we increase the value of K beyond a limit, the averaging effect will clear out all the interesting patterns in the data.

Thus the KNN algorithm can be run for different values of K and can be used to generate predictions with that trained model and then calculate the error us-

ing any performance metric. By comparing the results of different models we can identify the optimum value of K, usually by generating a plot having K values vs. the error values. Input normalization is usually done on the inputs in KNN, as when calculating the euclidean distance there will be unwanted bias due to the difference in magnitudes of inputs. Thus the input variables are re-scaled by using some normalization procedure. One method to normalize the inputs is by using the mean and standard deviation in the training data:

$$x_{i[j]}^{new} = \frac{x_{ij} - \overline{x_j}}{\sigma_j}, \forall j = 1, ..., p, i = 1, ..., n, \tag{2.4}$$

Where p is the number of input variables and n is the total number of training data points. And $\overline{x_j}$ and σ_j are the mean and standard deviation of each input variable respectively.

One main advantage of the KNN technique compared to other methods is that it is robust to noise in the data and is also effective for large training data [18]. However, since KNN uses the entire training data, the computation can be very expensive for large datasets, as the time complexity of this algorithm is exponential which is a disadvantage for KNN [19]. KNN doesn't learn from the training data, instead memorizes it, and then uses that data to predict the target. In that sense, KNN is also called lazy learning algorithm.

2.1.2 Random Forest

In this section, the concept and theory of RF is explained by introducing the concept of decision trees and bagging.

Decision Trees

Decision trees are called as rule-based models since the rules that are used to define the model can be organized into a graph structure called as binary trees [9]. The input space is divided into multiple disjoint regions, and these regions have a constant value that is used for prediction $\hat{y}(x_*)$. Sequential decision making process corresponding to the traversal of a binary tree, is the process of selecting a specific model, when given an input x [19].

From Figure 2.2, a recursive partitioning of the input space along with corresponding tree structure can be seen. The whole input space is first divided into two regions based on whether X1 > Q1, where Q1 is a parameter. This process creates two subregions, and they can be further divided based on whether X2 > Q2. Thus, this type of recursive subdivision can be described by the traversal of a binary tree as shown in Figure 2.3. As seen in the tree structure, for any new input variable x, the region into which it belongs to is determined by following the path from the top of the tree to the root node according to the different decision rules at each node. Each region means that a seperate model to predict the target

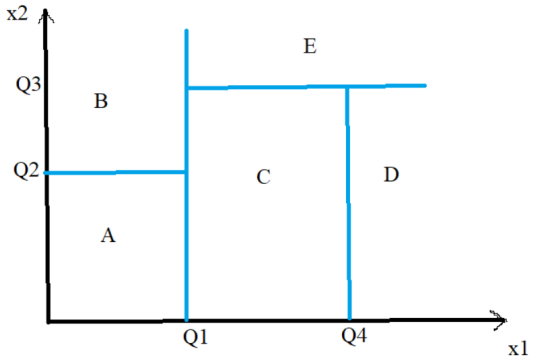


Figure 2.2: *An illustration of the two dimensional input space that is divided into five regions by boundaries.*

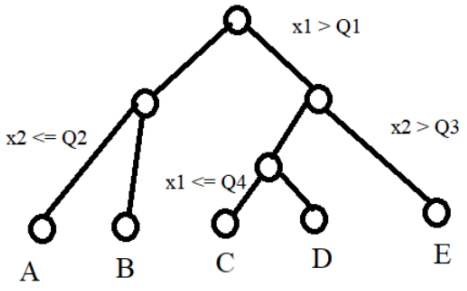


Figure 2.3: *An illustration of the binary tree corresponding to the partitioning of input space as in Figure 2.2.*

variable. A,B,C,D,E in Figure 2.3 are called as the leaf nodes and the internal splits are called as internal nodes.

Decision tree predictions $\hat{y}(x_*)$ for a regression tree is a piecewise constant function of the input x_*, which is

$$\hat{y}(x_*) = \sum_{l=1}^{L} \hat{y}_l \mathbb{I}\{x_* \in R_l\}, \tag{2.5}$$

where L is the total number of leaf nodes in the tree, R_l is the lth region and $\hat{y}_l$ is the constant prediction from the lth region. There is an indicator function $\mathbb{I}$, which means if $x \in R_l$ then $\mathbb{I}\{x_* \in R_l\} = 1$ and otherwise zero.

Finding suitable parameter for the parameters defined in Eq. (2.5), corre-

sponds to learning the tree from the data. We can compute the constants $\hat{y}_l$ from $l = 1, ..., L$ as the average of the training data points which falls into those region:

$$\hat{y}_l = Average\{y_i : x_i \in R_l\}. \tag{2.6}$$

Next we have to find the regions R_l and the idea is to identify regions so that the tree fits into the training data. But the process of finding the tree that can optimally partition the input space to fit the training data is computationally expensive, because of the combinatorial explosion that happens when trying to find the number of ways an input space can be partitioned.

Recursive binary splitting is a greedy heuristic algorithm that is used for solving this problem. The tree is constructed from the tree top to bottom based on the splitting rules one after the other. When determining how to split the root node, we only think about the root node and not the complete tree, and hence this algorithm is called greedy. Once the decision about first split of the input space is made, this decision is kept fixed and we continue in a similar way for the resulting two half-spaces. Consider that we have p input variables $x_1, ..., x_p$ and corresponding cutpoint c that divides the input space,

$$R_1(j, c) = Average\{x|x_j < c\} \quad and \quad R_2(j, c) = Average\{x|x_j >= c\}. \tag{2.7}$$

The predictions associated with the two regions depends upon the index j and also on the splitting cutpoint c and they can be represented as,

$$\hat{y}_1(j, c) = Average\{y_i : x_i \in R_1(j, c)\} \quad and \quad \hat{y}_2(j, c) = Average\{y_i : x_i \in R_2(j, c)\}. \tag{2.8}$$

Once predictions are estimated, we need to find the prediction error which is the difference between the constant prediction associated with that region and y_i. This done for all the training data points and the sum of squared errors can be represented as,

$$\sum_{i:x_i \in R_1(j,c)} (y_i - \hat{y}_1(j, c))^2 + \sum_{i:x_i \in R_2(j,c)} (y_i - \hat{y}_2(j, c))^2. \tag{2.9}$$

Closeness of a prediction and the training data is calculated by using the squared error as they are one of the common loss functions as introduced in Section 1. Thus, to find the optimal split in the tree we find the minimum squared error (2.9). Hence, moving through different possible values of j, we find the pair (j,s) for which the expression of squared error can be minimized. Once optimal split for root node as per the minimum squared error has been fixed, this process is continued for both the left and right branches. Thus each branch is split over all training data points by minimizing the squared errors and this process is continued until there is only single data point in each region.

But the disadvantage of having a tree that is fully grown is that the predictions will exactly match the training data points in each of the regions, which

leads to the problem of overfitting as described in Section 1.1 to the noisy training data. Stopping criterion are generally used to control the growth of the tree at an early stage, like adding restrictions to the number of training data points associated with each leaf node or by having a maximum limit on the tree depth. We can summarize the method for decision tree using recursive binary splitting as,

Data: Training Data $\boldsymbol{T = (x_i, y_i)}$ for i = 1 to n

Result: predictions $\boldsymbol{\hat{y}_1, ..., \hat{y}_L}$ and corresponding regions $R_1, ..., R_L$ of the decision tree

Algorithm 1 Learning a decision tree by recursive binary splitting

1: Let the whole input space be denoted by $\boldsymbol{R}$.
2: Compute the input space($\boldsymbol{R_1, ..., R_L}$) = Split($\boldsymbol{T = (R, T)}$).
3: Compute the predictions $\hat{\boldsymbol{y}}_l$ for $\boldsymbol{l = 1, ..., L}$ as
$$\hat{y}_l = \boldsymbol{Average(y_i : x_i \in R_l)} - Regression$$
$$= \boldsymbol{Majority(y_i : x_i \in R_l)} - Classification$$
4: Function Split$\boldsymbol{(R, T)}$:
5: **if** stopping criterion not fulfilled **then**
6: For all input variables $\boldsymbol{j = 1, ..., p}$, parse through all possible splits $\boldsymbol{x_j < s}$.
7: Select the pair of $\boldsymbol{j, s}$ that minimizes (2.9).
8: Split the regions based on (2.7) into subregions $\boldsymbol{R_1}$ and $\boldsymbol{R_2}$.
9: Split data $\boldsymbol{T}$ into $\boldsymbol{T_1}$ and $\boldsymbol{T_2}$.
10: return Split $\boldsymbol{(R_1, T_1)}$, Split$\boldsymbol{(R_2, T_2)}$
11: **else**
12: return $\boldsymbol{R}$
13: **end if**

Data: Training data $\boldsymbol{T = \{x_i, y_i\}_{i=1}^n}$, test data point $\boldsymbol{x_*}$, Regions of decision tree $\boldsymbol{R_1, ..., R_L}$

Result: predictions for test data point $\hat{\boldsymbol{y}}(\boldsymbol{x_*})$

Algorithm 2 Prediction from the decision tree

1: Go through possible options and find out the region $\boldsymbol{R_l}$ where $\boldsymbol{x_*}$ belongs.
2: Predictions are returned as $\hat{\boldsymbol{y}}(\boldsymbol{x_*}) = \hat{\boldsymbol{y}}_l$

Choosing the depth of a tree is an important factor which influences the final predictions. By depth it means the maximum distance between the root node and any of the leaf nodes. When a tree is fully grown then there is a chance of overfitting as the trees will be very much adapted to the training data points. Hence, shallow trees are used to mitigate this risk. As any other ML models, the optimal size of the tree depends upon the problem and it is a trade-off between flexibility and rigidity [9].

Bagging

Ensemble learning method is a technique where we combine multiple instances of some basic ML models to make more accurate predictions than the individual model, and the resulting methods are called as ensemble methods. In other words, "Wisdom of crowds" is the main idea, where different base models are trained in slightly different ways so that all of them can contribute to learning the input-output relationship, and finally using these individual predictions from all these base models as the average or majority vote to obtain final prediction [9].

Bagging or bootstrap aggregating is a type of ensemble method. Bootstrap is the process of creating slightly different training data for the model, by randomly sample overlapping subsets of training data. Thus, an ensemble of similar, but not identical models are created and this helps to reduce the variance compared to using a single model, which in turn reduces the risk of overfitting. As introduced in the section 1, bias-variance tradeoff is an important concept of ML and complicated input-output relationships can be represented by using flexible models. But the disadvantage of using highly flexible models is the risk of overfitting or high model variance. Thus in essence, these high variance models can be used as base models in bagging and reduce the variance of the base models without increasing its bias.

In Bootstrap method, datasets of size $\boldsymbol{n}$ is generated from one base dataset of size $\boldsymbol{n}$. In statistics, bootstrap is used to quantify the uncertainties associated with estiamtors such as confidence intervals. We have our training data $\boldsymbol{T} = \{\boldsymbol{x_i}, \boldsymbol{y_i}\}_{\boldsymbol{i=1}}^{\boldsymbol{n}}$, and the assumption is that the training data $\boldsymbol{T}$ at hand is a good representation of the real world data generating process, and so if a new dataset is created by collecting data, it will be similar to the training data $\boldsymbol{T}$. Using this assumption we argue that generating new dataset is similar to randomly selecting datapoints that is already contained in $\boldsymbol{T}$. In other words, instead of collecting new data from the population we try to sample from the available training data which is assumed to have a good representation of the population. One important fact is that we do sampling with replacement, hence the bootstrapped dataset might contain multiple copies of the same datapoint from original training data $\boldsymbol{T}$ and also will in turn miss some other datapoints. Bootstrap method can be represented as:

$\boldsymbol{Data}$: Training data $\boldsymbol{T} = \{\boldsymbol{x_i}, \boldsymbol{y_i}\}_{\boldsymbol{i=1}}^{\boldsymbol{n}}$
$\boldsymbol{Result}$: Bootstrapped dataset $\tilde{\boldsymbol{T}} = \{\tilde{\boldsymbol{x_i}}, \tilde{\boldsymbol{y_i}}\}_{\boldsymbol{i=1}}^{\boldsymbol{n}}$

Algorithm 3 Bootstrap

1: **for** $\boldsymbol{i = 1, ..., n}$ **do**
2: Sample $\boldsymbol{m}$ uniformly on set of integers $\boldsymbol{\{1, ..., n\}}$
3: Set $\tilde{\boldsymbol{x_i}} = \boldsymbol{x_m}$ and $\tilde{\boldsymbol{y_i}} = \boldsymbol{y_m}$
4: **end for**

Random but similarly distributed bootstrap datasets $\tilde{\boldsymbol{T}}^{(1)}, ..., \tilde{\boldsymbol{T}}^{(P)}$ are created

by repeating the bootstrap algorithm 3 for P times. Later these P training datasets are used to train an ensemble of P base models and the average predictions can be represented as:

$$\hat{y}_{bag}(x_*) = \frac{1}{P}\sum_{p=1}^{P}\tilde{y}^{(p)}(x_*) \quad or \quad g_{bag}(x_*) = \frac{1}{P}\sum_{p=1}^{P}\tilde{g}^{(p)}(x_*), \tag{2.10}$$

Where $\hat{y}_{bag}(x_*)$ and $g_{bag}(x_*)$ are the final predictions for regression and classification tasks respectively. $\tilde{y}^{(1)}(x_*), ..., \tilde{y}^{(p)}(x_*)$ and $\tilde{g}^{(1)}(x_*), ..., \tilde{g}^{(p)}(x_*)$ are the individual predictions from the ensemble members. Thus we can summarize bagging as:

Data: Training data $T = \{x_i, y_i\}_{i=1}^{n}$
Result: Base models P

Algorithm 4 Base model learning

1: **for** $i = 1, ..., P$ **do**
2: Algorithm 3 is run to obtain bootstrapped training dataset $\tilde{T}^{(p)}$
3: Learn base model from $\tilde{T}^{(p)}$
4: **end for**
5: Compute $\hat{y}_{bag}(x_*)$ or $g_{bag}(x_*)$ by averaging (2.10)

Data: Base models P and test input x_*
Result: Final prediction $\hat{y}_{bag}(x_*)$ or $g_{bag}(x_*)$

Algorithm 5 prediction with base models

1: **for** $p = 1, ..., P$ **do**
2: Use base model p to predict $\tilde{y}^{(p)}(x_*)$ or $\tilde{g}^{(p)}(x_*)$.
3: **end for**
4: Compute $\hat{y}_{bag}(x_*)$ or $g_{bag}(x_*)$ by averaging (2.10)

Thus the main idea of variance reduction in (2.10) can be formalized as, let $Z_1, ..., Z_P$ be a group of identically distributed random variables with variance $Var[Z_p] = \sigma^2$ and mean value $\mathbb{E}[Z_p] = \mu$ for $p = 1, ..., P$. ρ is assumed to be the average correlation between any two pairs. Thus, the mean and variance can be represented as:

$$\mathbb{E} = \left[\frac{1}{P}\sum_{p=1}^{P}Z_p\right] = \mu, \quad Var = \left[\frac{1}{P}\sum_{p=1}^{P}Z_p\right] = \frac{1-\rho}{P}\sigma^2 + \rho\sigma^2. \tag{2.11}$$

The Eq. (2.11) tells us that variance is reduced by averaging, if the correlation $\rho < 1$ and the mean does not change when the averaging a number of identically distributed random variables. In order to understand how bagging affects the

variance, $\hat{y}_p(x_*)$ are considered as random variables and since all the base models and predictions are based on the same training data T, we can say that $\hat{y}_p(x_*)$ are identically distributed but correlated. Hence to conclude, averaging the identically distributed predictions as per (2.10), each with low bias, bias remains low and the variance is reduced as shown in Eq. (2.11).

Construction of bagging only reduces variance and does not make the resulting model more flexible by the additional ensemble members. Also, the addition of ensemble members does not help to obtain a better fit to the training data. But when the individual ensemble member overfits, the averaging across the ensemble member will have a smoothing effect and creates a better generalization. The limiting behaviour of $P \to \infty$ and the identically distributed ensemble members makes the bagging model as (2.12) where expectation is based on the bootstrapping algorithms randomness.

$$\hat{y}_{bag}(x_*) = \frac{1}{P}\sum_{p=1}^{P} \tilde{y}^{(p)}(x_*) \quad \xrightarrow{P \to \infty} \quad \mathbb{E}[\tilde{y}^{(p)}(x_*)|T], \tag{2.12}$$

where when P increases we expect bagging model to converge but in practice the choice of P is mainly guided by the computational complexity as increasing P with no error reduction is of no use.

As we are using bootstrapping, only 63 percent of training points in T will be present in the bootstrapped training data $\tilde{T}$ [9]. This means that one third of the ensemble members will not have seen that datapoint during training, and these ensemble members are called out of bag ensemble. Thus if we compute error for these out of bag ensemble, we get a error $E_{OOB}^{(i)}$. And if we calculate the average E_{OOB} for all data points x_i, y_i, we get a good estimate of E_{new} and this is called as out of bag error estimate.

Random Forest

RF is a non-parametric supervised learning algorithm that uses an ensemble learning method. Ensemble learning as described earlier, is a method where multiple machine learning algorithms are trained on the same problem and the outputs are combined as the final prediction from the model. This approach will help to have more accurate predictions rather than individual predictions. In the context of RF, boosting aggregation or bagging is a technique to combine weak learners (i.e.,decision trees) and produce a more accurate strong learner [20]. RF can be used for regression and classification tasks.

Bagging reduces variance by averaging over ensemble of models,i.e, reducing the correlation between individual ensemble members dependence on ρ, the average correlation [9]. Furthermore, we can use a simple method to reduce the correlation even more and this is called as RF, where the ensemble base models

are either classification or regression trees. In simple words, additional randomness is added when a tree is constructed so that the correlation between base models is reduced.

RF is similar to decision tree explained in 2.1.2 except that whenever a node is split we do not consider all possible input variables $x_1, ..., x_p$ as splitting variable. Instead of considering all the possible inputs, only a small subset $q <= p$ inputs are only used as splitting variables. In the following split points, another subset of q inputs are used as splitting variables and this process continues. The B ensemble members end up using different subsets for the different trees as the random subset selection is done independently. Larger reduction in variance is achieved in random forest compared to bagging since the B trees are less correlated but the training procedure tends to increase the variance of each individual trees. That is, in Eq. (2.11), random forest decreases ρ but at the same time increases σ^2 compared to bagging. In practise, the average prediction variance seems to be often reduced as the reduction in correlation has dominant effect [9].

As explained in 2.1.2, trees are build using recursive binary splitting, the algorithm make choices that appear to be good in the beginning but can turn out to be suboptimal further down the splitting procedure. In case of bagging, there can be a situation where a dominant variable is chosen for the first split for all ensemble members, which will inturn make all trees identical after the first split. But in this scenario if random forest is used, some ensemble members won't be having access to this dominant variable, as the random subset q is chosen for the first split. This process may not improve the performance of individual trees but can be usefull when splitting further nodes, and thus the average of all these different ensemble members can improve the overall performance.

In the case of regression, RF will be having B decision trees that will be trained on B different randomly sampled subsets of the data and also with randomly chosen features. This property of RF makes it impossible to overfit the data when more trees are added since each tree is trained on a different dataset. The B trained models form an ensemble and the final result for a regression task is the average of all the predictions from the individual trees. An example of RF is shown in Figure 2.4, where there are 200 different trees and the output from the RF will be the average of all these 200 trees.

RF can be used for feature selection by using the property of variable importance. Prediction error increases when the out of bag data for that variable is changed without changing the others, this increase in prediction error tells us the variable importance. Variable importance can be measured using impurity score, the higher the value of impurity score, the more important the feature. When a tree is constructed, impurity score decides which variable is chosen to split a node. The calculation of impurity is based on the reduction in the sum of squared errors whenever a variable is chosen to split. Hence the split maximizes the decrease in impurity score [21][22]. In other words, each split is made in

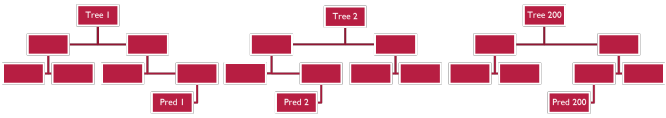


Figure 2.4: *A visualization of RF for a regression task with 200 trees.*

such a way that once the split is done there is more similarity within individual tree branch than between two different branch. Thus, this property of RF can be leveraged to identify the important variables from the whole set of input variable vector $\boldsymbol{X}$.

Another advantage of using RF is that there are only very few parameters to optimize, unlike other machine learning methods. Examples of parameters are the number of trees in the RF and the number of leafs in each random subset at each node. Thus RF is a robust algorithm against the noise in the data which is prone to overfitting, faster in execution and easily parallelizable [20][22][19].

2.2 Features

This part is to study the features of received signals, such as RSRP, ToA, and LoS condition for each radio propagation path.

2.2.1 RSRP

RSRP is an average power received in subcarriers/resource elements comprising an Orthogonal Frequency-Division Multiplexing (OFDM) symbol [23], and it uses reference signals to calculate average received power. Reference signal is a predefined signal occupying specific resource elements within the downlink time–frequency grid. In the simulator, the value of RSRP is linear, the unit is watt (W).

Considering a downlink communication scenario with one UE and one TRP, which is shown in Figure 2.5 as an example. The position $P_{TX} \in \mathbb{R}^2$ and antenna orientation $\theta_{TX} \in [0°, 360°)$ of the TRP are known, while the position $P_{RX} \in \mathbb{R}^2$ and antenna orientation $\theta_{RX} \in [0°, 360°)$ of the UE are unknown. In this thesis, our primary objective is to determine UE orientation only. The TRP transmits reference signals, and the reference signals are received by the UE after having transmitted though multiple propagation paths. There are M_P propagation paths. When the signals transmit though ith path, $i \in 1, ..., M_P$, the transmit power in direction $\varphi_{T,i}$ is $F(\varphi_{T,i})$ dB, where $\varphi_{T,i}$ is the Angle of Departure (AoD) of the ith path. Similarly, the receive power of ith path is $G(\varphi_{R,i})$ dB, where $\varphi_{R,i}$ is the Angle of Arrival (AoA) of the ith path.

The transmission power is P_T dBm. The received RSRP s is given as

$$s = \text{db2pow}(P_T + \sum_{i=1}^{M_P}(p_i + F(\varphi_{T,i}) + G(\varphi_{R,i}))) + n, \tag{2.13}$$

where the function $\text{db2pow}(x) = 10^{\frac{x}{10}}$ is to transform power from logarithmic scale to linear. p_i dB is the path loss of the ith path, and n is the noise.

According to Eq. 2.13, the RSRP value depends on the transmission power, the path loss, AoD and AoA of each propagation path. In this paper, transmission power and the orientation of TRP antenna panel are predefined. Path loss is mainly due to the distance between the transmitter and receiver, the height and position of the antenna (i.e. spatial free space loss), and the indoor office environment (e.g. reflection, diffraction, etc, in the propagation path). AoD and AoA are related to transmit power and receive power in different directions since the propagation path perpendicular to the antenna panel transmits/receives the highest power according to the multiple antenna communication theory [24]. Therefore, the RSRP can be used as a feature for UE orientation estimation since it is affected by the antenna orientation. In this thesis, we will use a RSRP vector $S = [s_1, s_2, ..., s_{36}]$ from 36 TRPs to investigate UE orientation estimation.

2.2.2 ToA

Since the transmission speed c of the signal is constant, the signal receiving time from i^{th} propagation path for $i \in 1, ..., M_P$ is proportional to the path distance d_i. ToA t is the time when a signal is received at UE from the TRP, which can be calculated by

$$t = \min_{i \in 1,...,M_P} \frac{d_i}{c}. \tag{2.14}$$

Since ToA only depends on the signal propagation path which is not related to the orientation of the UE, ToA cannot be used as a feature for estimating the orientation of the UE.

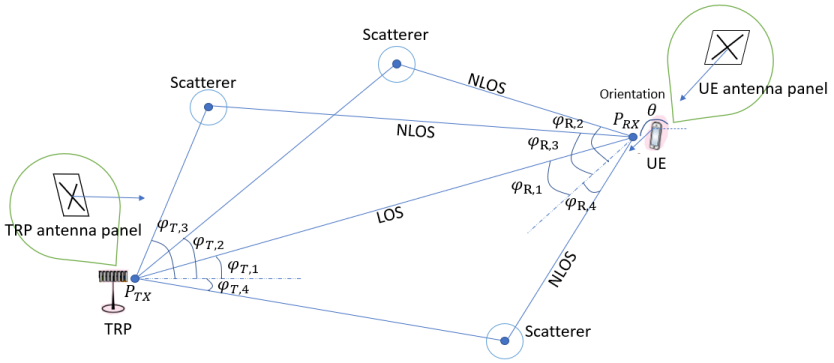


Figure 2.5: *A transmission model example including the LoS path and three NLoS paths between TRP and UE. The TRP has predefined position P_{TX} and antenna panel orientation θ_{TX}, the UE has unknown position P_{RX} and antenna panel orientation θ. $\varphi_{T,i}$ and $\varphi_{R,i}$ are the AoD and AoA of the ith propagation path.*

2.2.3 LoS Condition

When there is no obstruction between the transmitter and the receiver, and the transmitter orientation and the receiver orientation are facing each other, the channel model is LoS, otherwise it is NLoS. Since LoS path is the shortest among all the propagation path, the received power from LoS path is the strongest because of the low attenuation when the channel model is LoS. Therefore, if UE receives the strongest power at ToA, the channel model is LoS, otherwise, the channel model is NLoS.

To find the UE receives the maximum power from which propagation path, the calculation is

$$z = \underset{i \in 1,\ldots,M_P}{\arg\max}(p_i + F(\varphi_{T,i}) + G(\varphi_{R,i}) + n_i). \tag{2.15}$$

The UE receives the maximum power from the z^{th} propagation path. If ToA of z^{th} path is the minimum among the ToA of all the path, we can say that LoS exists in the propagation path, the channel model is LoS. Otherwise, the LoS does not exist in the propagation path, the channel model is NLoS.

According to Eq. 2.15, the LoS condition depends on the the path loss, AoD and AoA of each propagation path. Therefore, the LoS condition can be used as a feature for UE orientation estimation since it is affected by the antenna orientation. In this thesis, we define 'isLoS' as a flag to distinguish the LoS condition. When isLoS is true or 1, the channel model is LoS; when isLoS is false or 0, the channel model is NLoS. We will use a LoS condition vector $L = [isLoS_1, isLoS_2, ..., isLoS_{36}]$ from 36 TRPs to investigate UE orientation estima-

tion.

In this chapter, we discussed the theoretical background of KNN and RF and learned about the features of the received signal. In the next chapter, we will introduce the system model used in this thesis and analyze its wireless transmission model.

3
Method

In this chapter, we will discuss the scenario where the signals transmit, and explain the downlink transmission model. The feature selection section will elaborate on the useful features for the UE orientation estimation. The dataset generated from the selected features will be pre-processed before being fed to ML models. The analysis of angles section will talk about the orientation degree calculations. The performance metrics section will discuss the methods to evaluate the ML models estimation performance. The ML algorithms will be explained in the way that they have been applied in this thesis.

3.1 Scenario

The scenarios used in this thesis are the indoor-office scenario specified in the 3GPP standard [25]. The wireless transmission channels are stochastic whose parameters are defined in the standard. We generated these indoor-office scenarios using the Ericsson simulator. The layout of indoor-office scenario is shown in Figure 3.1, and its parameters are list in Table 3.1. The circles in Figure 3.1 represent the location of Base Stations (BS). There are a total of 12 BSs in the layout. Each BS has 3 sectors facing 90°, 210°, and 330°, which represent the 3 Transmission and Reception Points (TRP). 5G New Radio (NR) Positioning Reference Signal (PRS) is transmitted in the downlink by TRPs, which is the main reference signal supporting downlink-based positioning methods [26]. The carrier frequency is 2GHz and the antenna panel of a TRP consists of two antenna elements. The top and side views of radiation pattern of the antenna panel used at a TRP are shown in Figure 3.2a and 3.2b.

The antenna panel of the UE is also composed of two antenna elements like the antenna panel of the TRP, as shown in Figure 3.3, but does not have the re-

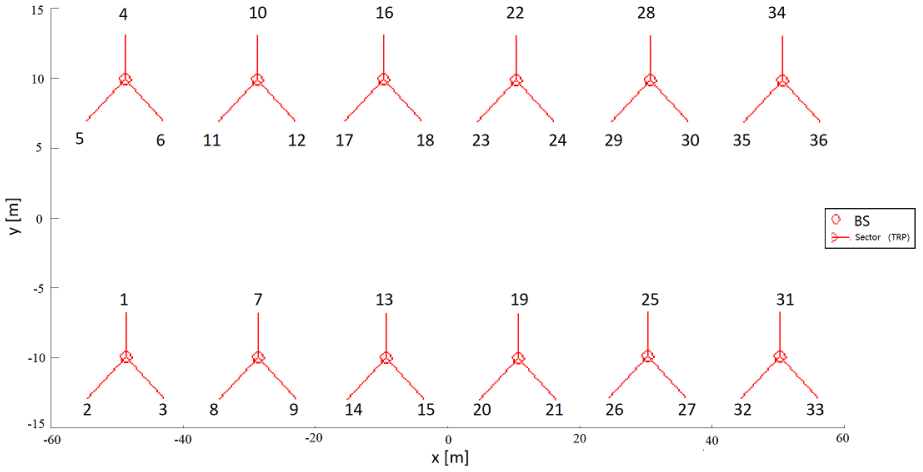


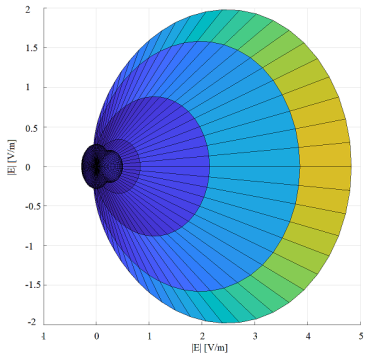
Figure 3.1: *Layout of indoor-office scenario.*

Table 3.1: *Parameters of indoor-office scenario.*

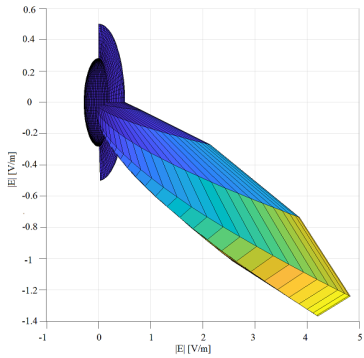
Parameter	**Value**
Room size ($W \times L \times H$)	120 $m \times 30$ $m \times 3$ m
BS antenna height	3 m (ceiling)
UE antenna height	1.5 m
Transmission direction	Downlink
Carrier frequency	2 GHz
Subcarrier spacing	30 kHz
Total transmission power	23 dBm

ceive beamforming. UE receives the maximum power when the input signal is perpendicular to the UE antenna panel. The height of the UE antenna is $1.5m$. The UE can be in any orientation degree in the 2D horizontal plane $\in [0, 360°)$.

In this thesis, we consider different wireless communication scenarios, which are list in Table 3.2. The interference in the table refers to the radio frequency interference caused by two or more radios using the same time and frequency resources[27]. Due to frequency interference, the received signal has large noise when different BSs send the same frequency signals at the same time. For each scenario, we randomly drop 1000 UEs at different positions, and collect each UE's received signals from each TRP in every orientation degree. In this thesis, we will analyze the relationship between the received signals in different orientation degrees at the same position, and compare the orientation estimation performance of employed ML algorithms in different scenarios.



(a) *The top view of antenna panel radiation pattern.*



(b) *The side view of antenna panel radiation pattern.*

Figure 3.2: *Antenna panel radiation pattern. The darker the color, the stronger the radiation. The blue color indicates the strongest radiation.*

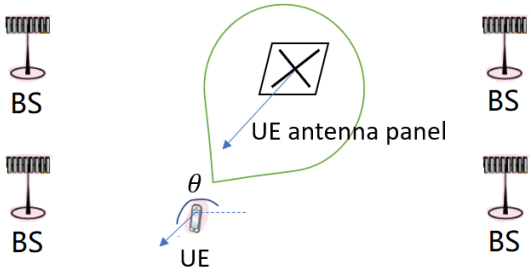


Figure 3.3: *UE antenna panel. The blue arrow indicts the orientation of the UE and θ is the orientation degree.*

3.2 Feature Selection

The feature selection technique is the process of selecting the most relevant features for the ML model. One of the key benefits of using feature selection is that it improves accuracy since the misleading data is removed. Other benefits are it reduces training time and reduces over-fitting. From the received signal, we can get feature vector $\boldsymbol{X}$ which contains features like the RSRP, ToA, and LoS condition. Each feature is defined in Section 2.2. To understand the influence of each feature on orientation, the feature selection property of the RF algorithm as explained in Section 2.1.2 is utilized. The way to calculate the impurity of each feature is also described in Section 2.1.2. The output of the algorithm is illustrated in Figure 3.4, which shows that RSRP is the most important feature to determine the orientation degrees, then is LoS condition, and the least important is ToA.

Table 3.2: Description for different indoor-office environment and wireless communication scenarios.

Scenario	Description
No obstruction	There is no obstruction between a UE and a TRP in the indoor-office. Therefore, whether there is the LoS channel between a UE and a TRP only depends on the antenna orientation of the TRP and the orientation of the UE.
Obstruction	There may be obstruction between a UE and a TRP in the indoor-office, the probability of the existence of obstruction is defined in [25]. Therefore, whether there is a LoS between a UE and a TRP depends not only on the antenna orientation of the TRP and the orientation of the UE, but also on whether there is obstruction in the transmission path.
No interference	TRPs transmit signals in different time to avoid interference.
Interference	TPRs in the same BS transmit signals in the same time and in the same frequency to cause interference. TRPs in the different BSs transmit signals in the different frequencies to avoid interference.
Extreme interference	All the TRPs transmit signals in the same time and in the same frequency to cause interference.

To get a better understanding of feature selection, we compare the feature value in each orientation degree at a certain position to study the impact of the change of the orientation degree of the UE on the received signal feature vectors. As shown in Figure 3.5, RSRP has similar vector values when the orientation degrees are adjacent, and relatively different vector values when the degrees are not adjacent. ToA has similar vector values regardless of degree. isLoS, which is the vector name for LoS condition feature described in 2.2.3, has the same vector values when the orientation degrees are adjacent, and different vector values when the degrees are not adjacent.

The correlation coefficient matrix is then used to verify the findings. In order to do that, we collect the received signal feature vectors in each orientation degree at a certain position, then calculate the correlation coefficient matrix for each feature. For example, for each orientation degree, the RSRP feature value is a 36×1 vector which represents the received signal power from 36 TRPs. There are a total of 360 orientation degrees, so we have 360 RSRP vectors. Then we can calculate the correlation coefficients between all possible RSRP vector pairs, which together form the RSRP correlation coefficient matrix. An example of

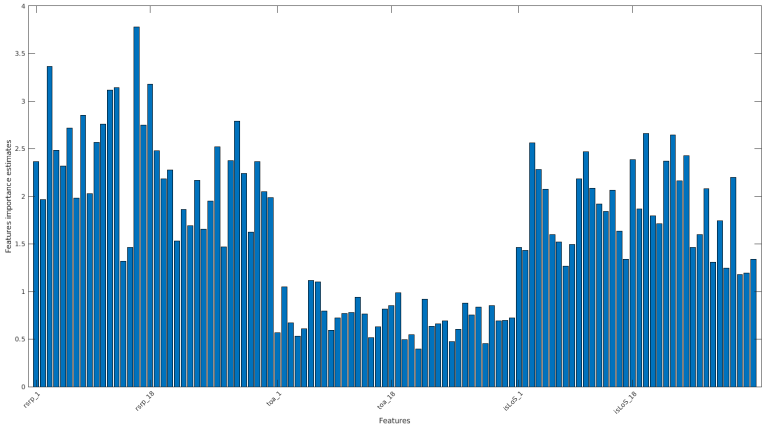


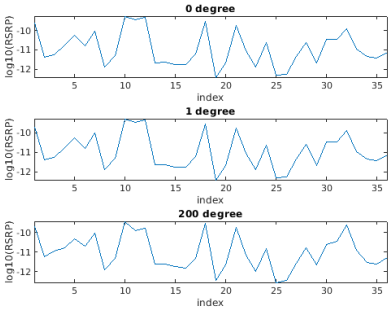
Figure 3.4: *The importance of each feature for orientation estimation. There is a total of 108 features as input features, and the first 36 features belong to the RSRP feature, after 36 features belong to the ToA feature, and the last 36 features belong to the LoS condition feature.*

RSRP correlation coefficient matrix is shown in Figure 3.6a, that UE is in a scenario of no obstruction and no interference. From the figure, we can see that the RSRP correlation coefficients in the diagonal are larger than the rest, which means RSRP vectors have strong relationship when the orientation degrees are adjacent, and RSRP vectors have weak relationship when the orientation degrees are non-adjacent. This shows that by looking for similar RSRP vectors, we can find similar orientation degrees to some extent. The same process goes for ToA and isLoS vectors. The correlation coefficient matrix results are presented in Figure 3.6b and Figure 3.6c correspondingly. From the figures, we can infer that RSRP and LoS condition features can be used for orientation estimation, but ToA feature can not be used for orientation estimation since the ToA has similar vector values regardless of the orientation degree.

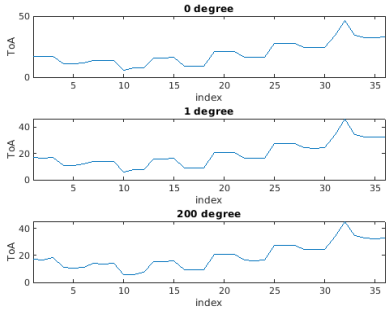
In summary, by using the feature selection property of the RF algorithm and correlation coefficient, we conclude that RSRP and LoS condition features are related to orientation degree. The thesis will investigate using RSRP and LoS condition to estimate the orientation degree of the UE, and compare their performance.

3.3 Pre-processing of Data

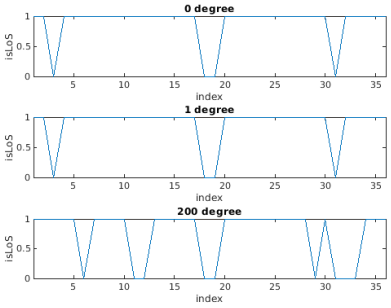
This section describes different ways of pre-processing the data before feeding it into the model.



(a) Vector values of RSRP in the orientation of 0°, 1°, and 200°.

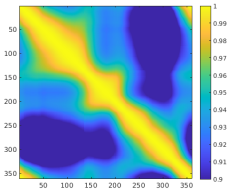


(b) Vector values of ToA in the orientation of 0°, 1°, and 200°.

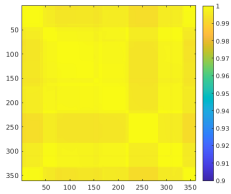


(c) Vector values of isLoS in the orientation of 0°, 1°, and 200°.

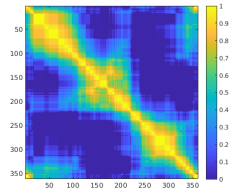
Figure 3.5: *The vector values of received signal features at a certain position.*



(a) The correlation coefficient matrix of RSRP feature.



(b) The correlation coefficient matrix of ToA feature.



(c) The correlation coefficient matrix of LoS condition feature.

Figure 3.6: *The correlation of received signal features.*

3.3.1 Dataset Structure

First we studied the dataset structure of features and target values. From the last section we know that the features to be investigated are RSRP and LoS condition, as we introduced in Section 2.2, the value of RSRP vector generated from the simulator is linear, the value of isLoS vector is binary. For the UE in a certain position and a certain orientation, it receives signals from 36 TRPs. From the received signal, RSRP value and the LoS condition between the UE and the TRP are determined. So for each UE position and each orientation we have a vector with 36 RSRP values and a vector with 36 binary values which denotes LoS condition of the UE with all TRPs. We have a total of 1000 UE positions and each position has 360 orientation degrees, so the dataset contains $1000 \times 360 = 360000$ elements, as shown in Figure 3.7. Each element (x_i, y_i) in the dataset is a data object, and the object properties RSRP, isLoS are inputs to the model x_1, x_2 along with the corresponding output target variable y, the orientation degree. The data object structure is shown in Table 3.3.

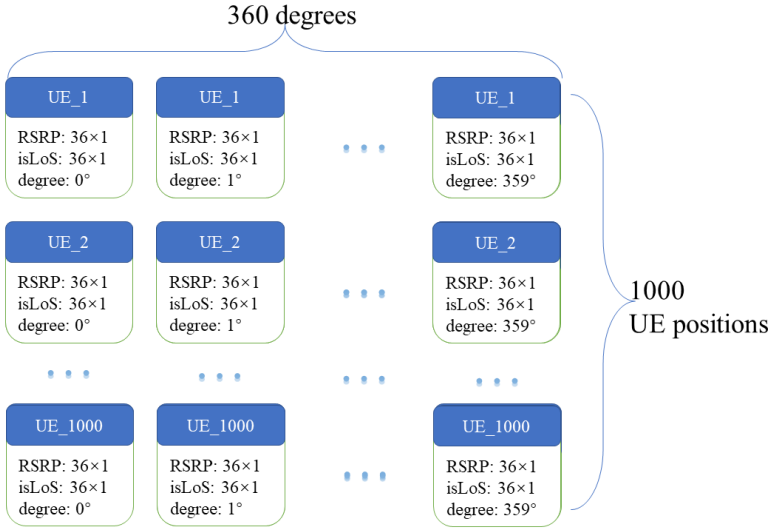


Figure 3.7: *The dataset structure. Each element is a data object.*

Table 3.3: *Data object structure.*

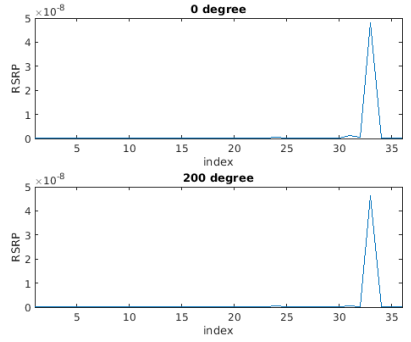
Property	**Size**	**Type**	**Value**
RSRP	36×1	Linear value	$10^{-16} \sim 10^{-6}$
isLoS	36×1	Binary value	0 (NLoS), 1 (LoS)
degree	1×1	Scalar	0, 1, 2, ..., 359

3.3.2 Imputing the Missing Values

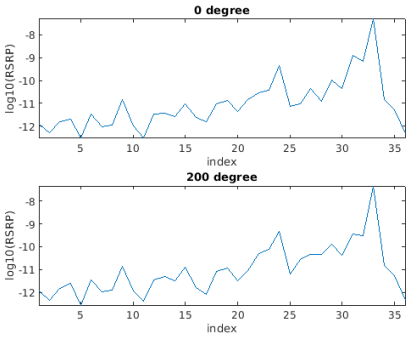
Once the dataset is created it is observed that some RSRP values ($\boldsymbol{x_1}$) are Not-a-Number (NaN) values in some scenarios. The reason is that the simulator only considers the signals which are received within a specific time frame and when it exceeds this limit, the value becomes NaN. Hence the NaN values need to be formatted before feeding into the model and we used imputing. Imputing is the process of replacing missing values. The missing NaN values in RSRP are replaced by the smallest RSRP value in the dataset since the received power tends to be smaller as transmission time passes.

3.3.3 Logarithm Transformation

As discussed in Section 2.1.1 normalization of inputs are important for KNN especially when calculating the neighbors. Since the RSRP vectors ($\boldsymbol{x_1}$) have linear values, when analyzing the difference among different RSRP vectors, the difference between the numbers with small values is too small to be identified compared to the difference between the numbers with large values, so that the algorithm would ignore the difference of the numbers with smaller values when finding similar vectors, resulting in a larger estimation error. Therefore, we apply the logarithmic transformation to the base 10 to the RSRP vectors to alter the scale and make the small element differences visible to the algorithm. Figure 3.8 compares the RSRP vectors of linear value and logarithm value. The two vectors of linear values in Figure 3.8a are almost the same, we can not tell the difference between small values of two vectors. The two vectors of logarithms values in Figure 3.8a have clear difference such as when index are 26 and 28, which helps the algorithm to distinguish the difference between the two vectors.



(a) *The linear values of RSRP vector at different degrees.*



(b) *The logarithm values of RSRP vector at different degrees.*

Figure 3.8: *The comparison of RSRP vectors of linear value and logarithm value.*

3.4 Analysis of Angles

Angle analysis is different from linear analysis. Below we will discuss how to calculate the difference in angles, and how to use a minimum-variance estimator to find the best estimate of angles from a set of estimates.

3.4.1 Difference in Angles

When calculating estimation error, estimation performance, and regression average, we need to calculate the angular difference since the estimated values are 2D degrees. As shown in Figure 3.9a, in 2D, zero degree is specified along the positive x-axis of the coordinate system, and the degrees increase in the counterclockwise direction. The range of degrees is $[0°, 360°)$.

Calculating the difference in angles is different from calculating the difference in numerical values, the difference in angles is not linear. For example, when we have two angles of 358° and 0°, their linear difference is $0° - 358° = -358°$, which should be 2° counterclockwise rotated according to the physical meaning. In addition, the linear average between 358° and 0° is $(358° + 0°)/2 = 179°$, which is not reasonable in physical meaning. So calculating the angular difference in a proper way is necessary when considering the actual physical meaning.

To calculate the angular difference, first is to subtract the two degrees to get the linear difference. Then the angular difference can be calculated by

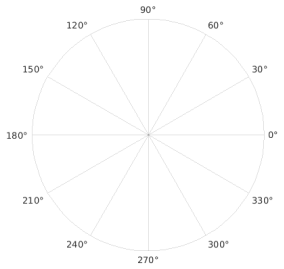
$$\text{diff}_{angle} = ((\text{diff}_{linear} + 180) \mod 360) - 180, \tag{3.1}$$

where $\text{diff}_{linear} = \text{deg}_2 - \text{deg}_1$, and deg_1, deg_2 are two degrees $\in [0, 360)$. The equation graph for correspondence between linear difference and angular difference is shown in Figure 3.9b. Accordingly, the angular difference between 358° and 0° is 2°. The physical meaning is to rotate 2° counterclockwise to transform from 358° to 0°.

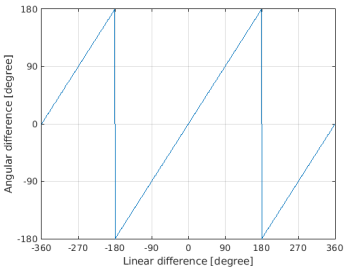
3.4.2 Minimum-variance Estimator of Degrees

Our goal is to get the best estimate for the orientation degree $\hat{y}$. When using machine learning algorithms such as KNN and RF, the output are continuous real-valued degrees, which requires regression techniques to get the best estimate. For example, KNN calculates the K nearest neighbors for the input features and then uses regression to get the best estimate from these K neighbors. RF constructs M decision trees during the training period, and M estimates are generated through these decision trees using the input features. Similarly, RF also needs to use regression to get the best estimate from M estimates.

Generally, regression techniques use the mean as the best estimate. But as analyzed in Section 3.4.1, linear averaging is not suitable to calculate the average of the angles. Considering the difference in angles derived in Section 3.4.1, we



(a) *Schematic diagram of angles in the directions. Degrees increase in the counterclockwise direction.*



(b) *Correspondence between linear difference and angular difference. On the x-axis is the linear difference between two degrees and on the y-axis is the corresponding angular difference.*

Figure 3.9: *Difference in Angle.*

can use the minimum-variance estimator to calculate the best estimate from a set of degree estimates.

The minimum-variance estimator is an estimator to get an estimate with the smallest variance among all estimates. The variance reflects the variation of the data around the average. The smaller the variance value, the closer the data is to the average, and the smaller the dispersion. Therefore, the best estimate is obtained by using a minimum-variance estimator, which is closest to the average of the angles.

The minimum variance estimator is calculated as

$$\min_{\forall d \in D} \frac{1}{N-1} \sum_{n=1}^{N} |d_n - d|^2 \tag{3.2}$$

where $D = (d_1, d_2, ..., d_n)$, d_n is nth estimate of the angle in degrees, and N is the number of estimates.

3.5 Performance Metrics

Performance metrics are performance measures that are used to evaluate the effectiveness of the ML model to solve the problem in consideration. As explained in Section 1 error functions helps us to compare the predicted value $\hat{\boldsymbol{y}}$ and the measured data point $\boldsymbol{y}$. Since the task at hand is a regression task, the most commonly used error functions $\boldsymbol{E}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ for calculating the estimation error of a forecasting model are the Root Mean Squared Error (RMSE). And for describing the estimation error probability distribution Cumulative Distribution Function

(CDF) plot is used. In the thesis, both RMSE and CDF plot are used to evaluate the ML estimation performance.

3.5.1 RMSE

RMSE calculates the square root of the mean of squared differences between the estimated values $\hat{\boldsymbol{y}}$ and the actual values $\boldsymbol{y}$. The mathematical equation of RMSE is

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i)^2}, \tag{3.3}$$

where n is the number of element in the vector $\boldsymbol{y}$, $\boldsymbol{y}_i$ is the i^{th} element of the actual vector and $\hat{\boldsymbol{y}}_i$ is the i^{th} element of the estimated vector.

Since in this thesis the estimates are the orientation degrees, we need to calculate the difference in angles. The calculation method is described in Section 3.4.1.

3.5.2 CDF

CDF plot is used to specify the probability that a variable X will take a value less than or equal to value x. The CDF of X is the function given by

$$F_X(x) = P(X < x). \tag{3.4}$$

In the thesis, CDF plot is used to evaluate the estimated error distribution of each model. At the same value of estimated error, the larger the value of CDF, the better the performance. Therefore, CDF plot together with RMSE provide a more comprehensive evaluation of the model.

3.6 ML Algorithms Implementation

The two different ML algorithms adaptation are explained in this section.

3.6.1 KNN

KNN regressor as introduced in the theoretical Section 2.1.1 works by taking the average of the neighbor's orientation degree. In the problem, the Euclidean distance (2.1) was used initially as the matrix to calculate the neighbors.

When calculating the distance using Euclidean distance, two weight functions were considered as described in Section 2.1.1. 'Uniform' weighting scheme was used as the first technique and later 'distance' weighting scheme was implemented on the same data to identify which weighting scheme was suitable for our data. Another technique called correlation (2.3) was also used to calculate

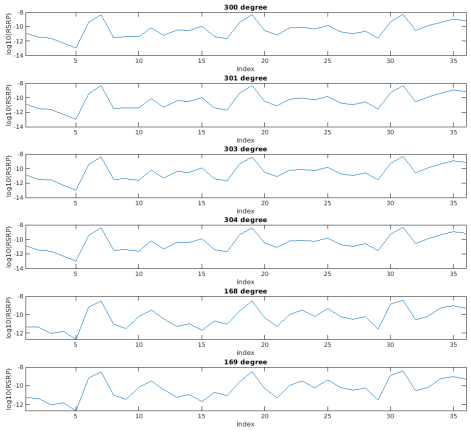


Figure 3.10: *An example of RSRP values of adjacent degrees and non-adjacent degrees. Sometimes they are similar.*

the nearest neighbors so that the best performing case could be identified for the problem at hand. The results of different techniques will be discussed in the result section.

After using the correlation distance as the metric, it was observed that the worst-performing cases were for UEs with orientation angles of around 0 degrees. Furthermore, the neighbors were analyzed and it was observed that the orientation angles which were far apart were having similar RSRP values as shown in Figure 3.10. The true value of orientation of the UE is 300 degrees which is the first plot in the figure. From the result, we can see that the RSRP values of 301 degrees and 168 degrees look similar with an extra peak for the 168 degrees.

However, the error can be reduced if the variance of each neighbor's degree is calculated and then use the degree with minimum variance as the final prediction. The algorithm is shown in 3.4.2. The algorithm first takes the nearest neighbor vector, and then computes the differences between each neighbor to all other neighbors and calculates the variance, Finally, the one with minimum variance is chosen as the output. A more detailed explanation is provided in Section 3.4.2. The increase in performance after this modification will be discussed in the result section.

Value of Optimal K

The value of the optimum number of neighbors for the KNN was done by calculating the RMSE against different K values and choosing the one which has the lowest RMSE value. This plot was generated after the addition of the minimum

variance estimator for the prediction and the plot shows us how the number of neighbors affects the RMSE value in Figure 3.11.

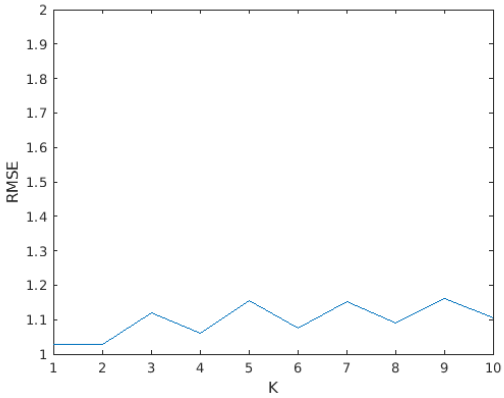


Figure 3.11: *RMSE of different numbers of neighbors.*

From the figure it can be observed that increasing the number of nearest neighbors has less influence on the RMSE value. Thus in our problem at hand choosing the best nearest neighbors has more impact on the final prediction rather than the number of neighbors considered. Hence, we chose 4 as the K value initially because it fits to most of the scenarios that have been studied in this paper.

3.6.2 Random Forest

RF is a non-parametric supervised algorithm as described in section 2.1.2. For the regression problem of estimating the orientation $\hat{y}$, the RF regressor was implemented as explained in the sub-sections to follow.

Number of Trees

In RF, having more number trees generally gives better predictions. However, as discussed in Section 2.1.2 more number of trees also mean that the computational time increases, and after a certain number of trees the improvement in performance is very small. A similar trend was observed in the problem at hand as shown in Figure 3.12 and considering the time and performance, the number of trees was chosen to be 130.

Depth of the Tree

The depth of the trees is an important factor which influence the final prediction as described in Section 2.1.2. They can be controlled and one of them is the minimum number of observations per tree leaf. To decide the optimum number of

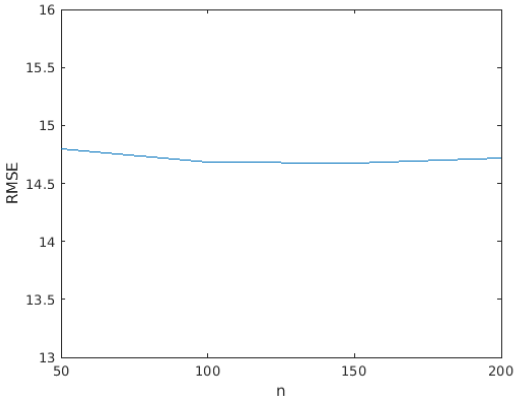


Figure 3.12: *RMSE of different numbers of trees.*

observations per tree leaf, different values were used as shown in Figure 3.13. It is observed that the RMSE value increases as the number of leaf nodes increases. Hence 3 was chosen as the optimum value for minimum number of observations per tree leaf.

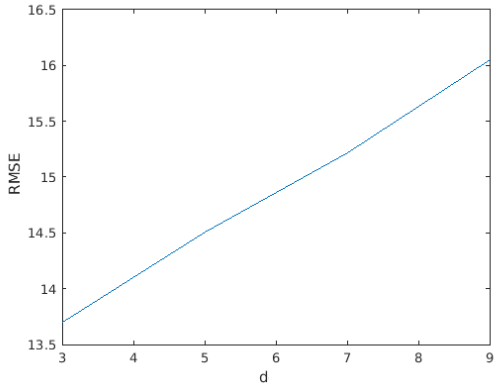


Figure 3.13: *RMSE of different numbers of tree depth.*

In this Chapter, we discussed the scenarios layout, the wireless communication model, and features for ML models as well as their data structures. And the implementation of KNN and RF algorithm in this thesis is introduced specifically. In the next Chapter, we will present the result and compare performance of these methods.

4

Results and Performance Evaluation

This chapter presents the results and performance evaluations of the KNN and RF algorithms. The methods of generating training dataset and test dataset are introduced. The performance of the two ML algorithms in different scenarios are also discussed based on different input features.

4.1 Dataset Splitting

As analyzed in section 3.3.1, we have 360000 data objects$(\boldsymbol{x_i}, \boldsymbol{y_i})$ in the dataset. For evaluation, there are two different ways to separate the dataset into a training dataset $\boldsymbol{T}$ and a test dataset, which are called random or consecutive selection. The reason for dividing the whole dataset is explained in Section 1.

4.1.1 Random Selection

The first way is to randomly select 90% of the data objects from each UE position dataset to train the ML model, and the remaining 10% of the data objects are used for testing, as shown in Figure 4.1. In this way, the UE position of the training dataset includes the UE position of the test dataset, while the orientations of the training dataset at a certain position are different from the orientations of the test dataset at the same position. The purpose is to evaluate the orientation estimation performance of the ML model for the existing UE position. From the results we can evaluate which feature performs the best for the UE orientation estimation.

4.1.2 Consecutive Selection

The second method is to consecutively select all the data objects belonging to random 10 UE positions as the test dataset, and the remaining data objects belonging

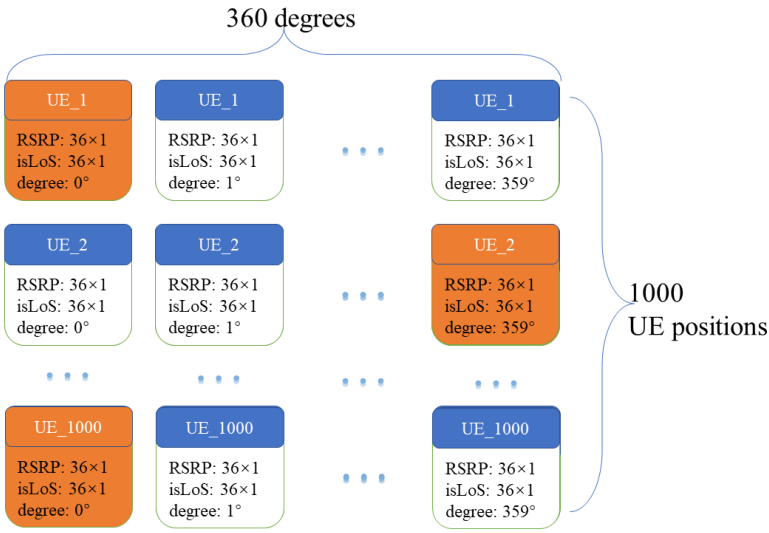


Figure 4.1: *Random selection. 10% of the data objects are randomly selected from each UE position dataset as the test dataset, and the remaining 90% of the data objects are used as the training dataset. The orange data objects are selected as the test dataset.*

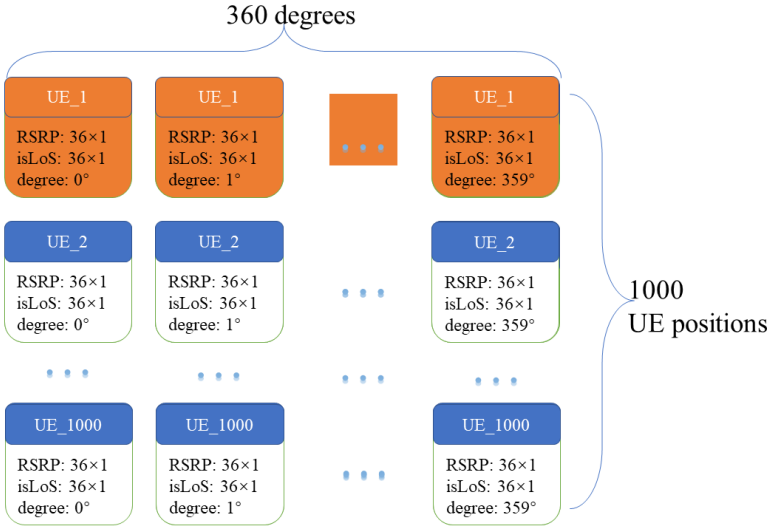


Figure 4.2: *Consecutive selection. All the data objects belonging to random 10 UE positions are selected as the test dataset, and the remaining data objects belonging to 990 UE position as the training dataset. The orange data objects are selected as the test dataset.*

to 990 UE position as the training dataset as shown in Figure 4.2. The purpose is to evaluate the orientation estimation performance of the ML model for the new UE position.

The training dataset obtained by these two ways of dataset splitting are not biased, since the UE positions and the orientations in the training dataset are uniformly distributed.

4.2 KNN

In this section, the results obtained by KNN are described. We will discuss KNN estimation performance for different input features, investigate the obstruction influence in the scenarios, and compare the impact of different levels of signal interference. Furthermore, the KNN orientation estimation performance for the UE at a new position will be evaluated.

4.2.1 Obstruction Influence

In this section, we discuss the obstruction influence by comparing scenarios with and without obstruction, described in Section 3.2. When there is no obstruction in the scenario, whether a UE and a TRP are in LoS only depends on the antenna orientation of the TRP and the UE. Moreover, the RSRP value only depends on the antenna orientation of the TRP and the UE, and the position of the UE in the indoor-office. When there is obstruction in the scenario, both LoS condition and the strength of the received RSRP are influenced by the obstruction. Therefore, we discuss the obstruction influence on UE orientation estimation for different input features to analyze which feature can provide the better performance in the different scenarios.

To exclude the influence of interference, we use scenarios without interference to compare the obstruction influence, and the random dataset selection is applied. The CDF of the orientation estimation errors of KNN for scenarios with and without obstruction is shown in Figure 4.3. From the results, RSRP as the input feature provides better performance than others, and the estimation error increases when obstructions in the scenario increases. Using the isLoS vector alone as the input feature can estimate the UE orientation to some extent, and obstruction improves its performance slightly. The reason is that obstruction makes the environment more complex, which brings more different changes to the isLoS vectors. When there are more differences between feature vectors of two different orientation degrees, the estimation gets more accurate. Combining RSRP and isLoS has worse performance than using RSRP alone. The reason is that the KNN algorithm treats each feature equally important, adding isLoS feature to RSRP will reduce the performance of RSRP alone.

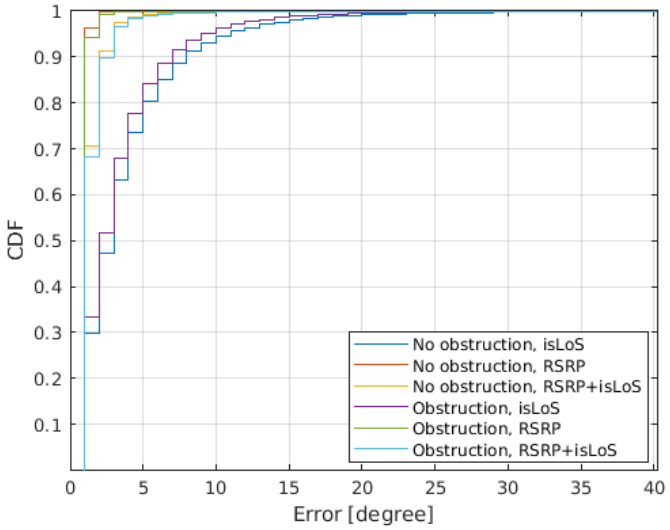


Figure 4.3: *In the no interference scenarios, the CDF of the orientation estimation errors of different features using KNN, with and without obstruction scenarios.*

4.2.2 Interference Level

Ideally, we want the received signal to be free from interference. But in reality, interference always exists. In this section, the estimation performance of the different levels of interference in the received signals, described in Section 3.2, are compared. Similar to the previous section, we will also compare the estimation performance of different features in different scenarios.

Figure 4.3 shows the CDF of the orientation estimation errors with no interference, Figure 4.4 is with interference, and Figure 4.5 is with extreme interference. Considering all the results together, the estimation performance of each feature gets worse when the level of interference increases. Similar to the previous section, RSRP alone as the input feature performs better than other features in all the scenarios. Combining RSRP and isLoS decreases performance a bit. When considering isLoS alone, the estimation performance reduces significantly, and stronger the interference, even worse is the performance. The reason is that the interference affects the detection of LoS condition, which makes isLoS vector not able to describe the actual condition of the UE, which makes the estimation performance decrease greatly.

In summary, the RMSEs of each feature in different scenarios of KNN are shown in Table 4.1. The smaller the RMSE value, the better the estimation performance. Considering all the scenarios, RSRP alone as input feature provides

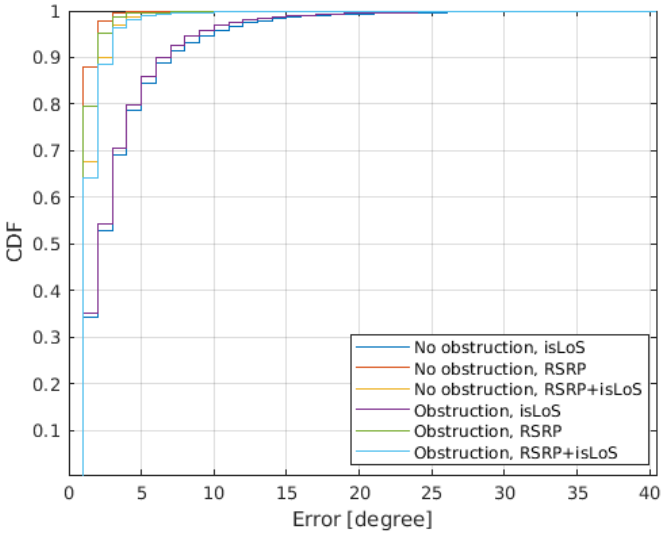


Figure 4.4: *In the interference scenarios, the CDF of the orientation estimation errors of different features using KNN, in the with and without obstruction scenarios.*

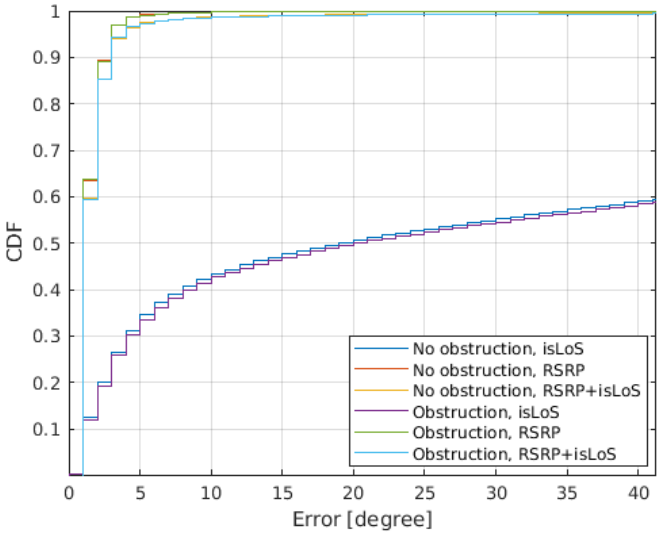


Figure 4.5: *In the extreme interference scenarios, the CDF of the orientation estimation errors of different features using KNN, in the with and without obstruction scenarios.*

Table 4.1: The RMSEs of KNN using different features for orientation estimation in the different scenarios.

	Feature	No obstruction	Obstruction
No interference	isLoS	6.0306	4.9440
	RSRP	1.0604	1.1090
	RSRP+isLoS	1.9347	1.9465
Interference	isLoS	5.1407	4.8827
	RSRP	1.2577	1.6073
	RSRP+isLoS	1.9602	2.1509
Extreme interference	isLoS	73.5739	73.8140
	RSRP	2.2359	2.7300
	RSRP+isLoS	8.1861	8.7275

the best estimation performance. isLoS can estimate orientation to some extent. While combine RSRP and isLoS together will reduce performance a bit comparing to the performance of RSRP alone. Perhaps using some other ML algorithms which can consider the two features comprehensively could improve the estimation performance.

4.2.3 Evaluation of Orientation Estimation in New UE Positions

Conclusions can be drawn from the previous two sections, that the best estimation performance of random selection happens under no obstruction and no interference scenario. In this section, we will use this scenario to discuss the orientation estimation performance when the test dataset contains the UE with the new positions.

We use the consecutive selection way to split the dataset. The CDF of the orientation estimation errors of KNN for the new UE positions are shown in Figure 4.6 when K=4. The RMSE of each feature is in Table 4.2. From the results, the CDF of each feature is almost a diagonal line, and the RMSE of each feature estimation is rather poor, which means that using RSRP, isLoS, or combining the two features as the input, when the training dataset does not contain the orientation data of UE at a certain position, KNN has limited ability to predict the UE orientation degree. The result shows that RSRP feature or combining RSRP and isLoS features together as input has the ability to estimate orientation to some extent, but the performance is significantly reduced compared to when the position is being trained. The investigations below explain why KNN has limited ability to predict the UE orientation degree for data with new position.

From Eq. 2.13, the received RSRP vector S from TRPs is mainly affected by UE position P_{RX} and orientation θ_{RX}. Generally, UE receives similar S when both P_{RX} and θ_{RX} are similar, therefore, we will investigate if the neighbors calculated

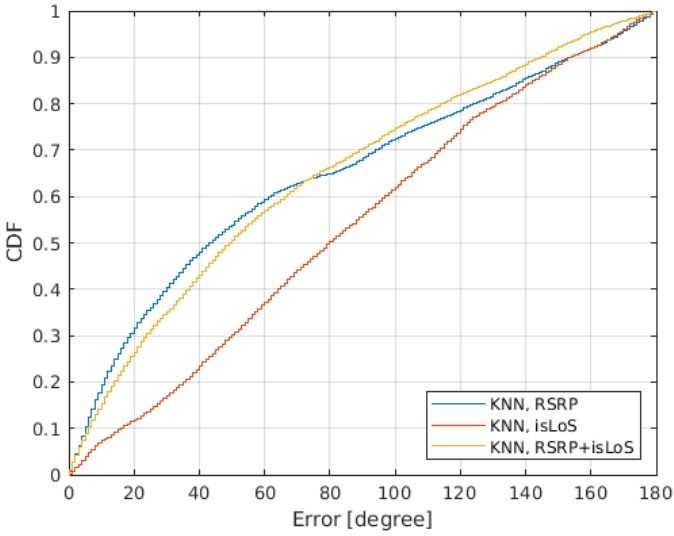
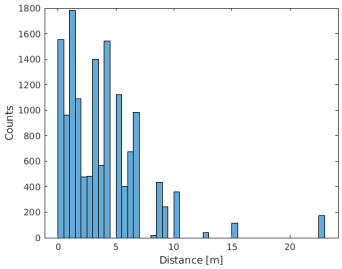


Figure 4.6: *In the no interference scenarios with no obstruction, the CDF of the orientation errors of KNN for different input features.*

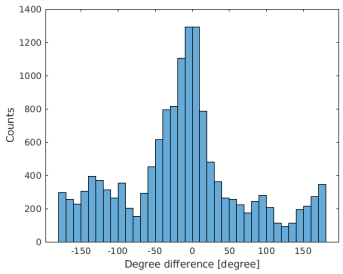
Table 4.2: *The RMSEs of KNN using different features for orientation estimation when test dataset has new UE positions.*

	isLoS	**RSRP**	**RSRP+isLoS**
RMSE	96.7750	84.3009	81.1042

by KNN method have similar position and orientation as the test data. From Figure 4.7a, more than 90% of the neighbors are less than 7 meters away from the test data when K=4 and 3600 test data, which means that most of the neighbors calculated by KNN method have similar positions to the testing data. From Figure 4.7b, around half of neighbors' orientation are similar to the orientation of test data, but the orientation of about half of the neighbors is less than 50 degrees different from the orientation of their test data, indicating that KNN has some abilities to estimate the orientation at the new location. However, neighbors with larger errors (that is, neighbors with an error greater than 50 degrees) also account for about half of the total, and are roughly random distributed. This is because at a certain location, the RMSE received by different orientations does not change much, as shown in the Figure 3.6a, the minimum correlation between each orientation at the same position is around 0.9, so the neighbors calculated by KNN appears randomly at all angles.



(a) Histogram of the 2D distance between the 3600 test data and their neighbors from KNN method when K=4.



(b) Histogram of the orientation difference between the 3600 test data and their neighbors from KNN method when K=4.

Figure 4.7: *Histograms when test data are in the new positions.*

4.3 Random Forest

In this section, the results obtained by RF are described. Similar to KNN, we will discuss the estimation performance of RF for different input features, and investigate the obstruction influence in the environment and different levels of signal interference. Furthermore, the orientation estimation performance of RF for the UE at a new position will be evaluated.

4.3.1 Obstruction Influence

Supposing in the no interference scenario, the randomly selection splitting way is used to compare the obstruction influence on the estimation performance of RF algorithm, and the CDF is shown in Figure 4.8. The results are slightly different from KNN, the best estimation performance is when RSRP is used as the input feature and there are obstruction in the environment. The reason is that instead of comparing the similarity of all the features like KNN, RF makes decision by considering which features to choose and what conditions to use for decision. Figure 4.9 compares the obstruction influence on the received RSRP. The obstruction increases the fluctuation of RSRP, so the RF algorithm can select special features with large fluctuations to do estimation, which makes obstruction scenarios estimation performance better.

Overall, the obstruction has relatively small effect on estimation performance comparing to the type of the input features. When the input feature is RSRP alone or RSRP and isLoS together, the estimated performance is much better than isLoS alone. This result is similar to KNN.

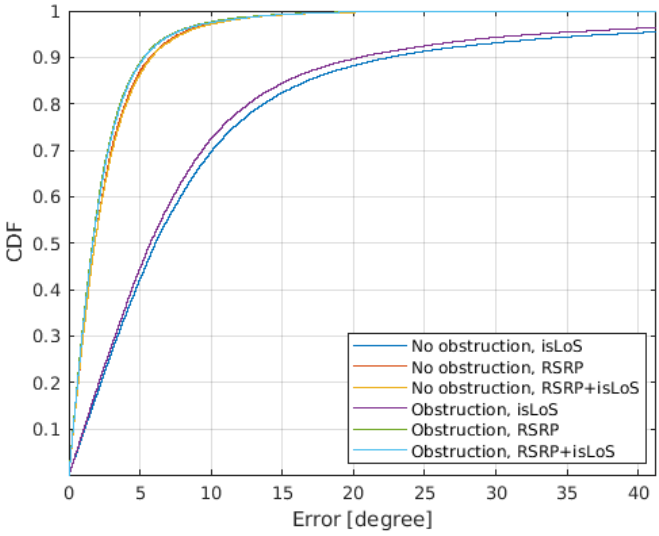


Figure 4.8: *In the no interference scenarios, the CDF of the orientation estimation errors of different features using RF, in the with and without obstruction scenarios.*

4.3.2 Interference Level

In this section, we will investigate the impact on estimation performance of RF when there are different levels of interference in the received signals. The CDF of the orientation estimation errors of RF for no interference are shown in Figure 4.8, for interference are shown in Figure 4.10 , and for extreme interference are shown in Figure 4.11. Similar to KNN, the estimation performance of each feature becomes worse when interference increases. RSRP alone as input feature gives the best estimation performance. Combining RSRP and isLoS together decreases performance a bit. When only considering isLoS, the estimation performance drops significantly, and the stronger the signal interference, the worse the performance.

In summary, the RMSE of RF algorithm estimation in different scenarios are shown in Table 4.3. Considering all the scenarios, RSRP alone as input feature provides the best estimation performance when using the RF algorithm. But combining RSRP and isLoS together reduces performance a bit. Besides, isLoS can estimate orientation to some extent.

4.3.3 Evaluation of Orientation Estimation in New UE Positions

Similarly, in this section we will discuss the RF estimation performance when the testing dataset contains the UE with the new positions. The CDF of the orientation estimation errors of RF for the new UE positions are shown in Figure

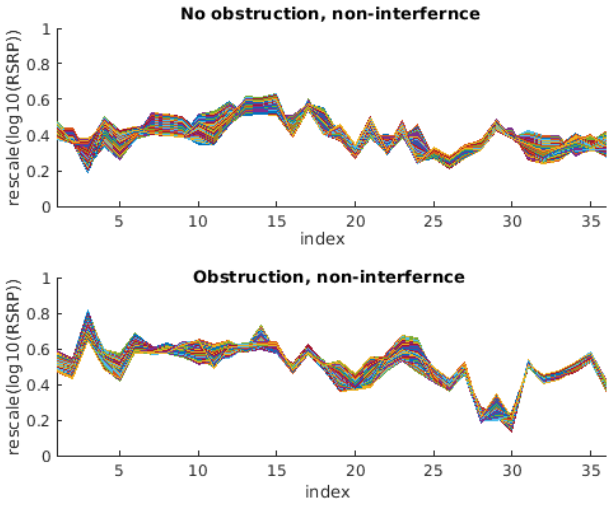


Figure 4.9: *The received RSRP in 360 orientation degrees at a certain position. The first figure is when no obstruction in the environment, the second when obstruction exist. Each different colored line represents the RSRP vector in a certain orientation degree.*

Table 4.3: *RMSE of RF algorithm estimates in different scenarios.*

	Feature	No obstruction	Obstruction
No interference	isLoS	23.7300	20.9849
	RSRP	3.9381	3.6519
	RSRP+isLoS	4.0232	3.7518
Interference	isLoS	21.9658	21.4752
	RSRP	4.0199	3.8881
	RSRP+isLoS	4.0344	4.0098
Extreme interference	isLoS	81.1315	80.9952
	RSRP	5.8205	5.5670
	RSRP+isLoS	5.7378	5.5303

4.12. The RMSE of each feature is in Table 4.4. From the results, the CDF of isLoS feature is almost a diagonal line, which means isLoS only has no ability for orientation estimation. Using RSRP, or a combination of the two as the input feature has some certain ability to estimate UE orientation when the training dataset does not contain the orientation data of a UE at a certain position, and they are better than using KNN method. The reason for the limited performance of using RF to estimate the orientation of the UE at the new position is the same as that of Section 4.2.3, which is because the features are sensitive to position changes,

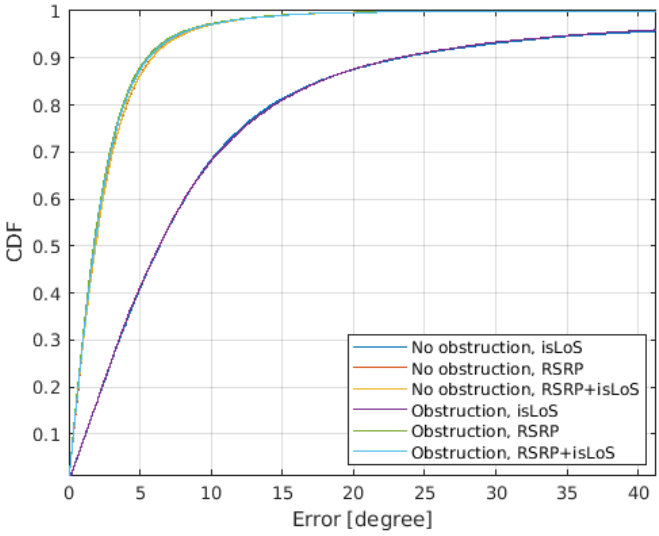


Figure 4.10: *In the interference scenarios, the CDF of the orientation estimation errors of different features using RF, in the with and without obstruction scenarios.*

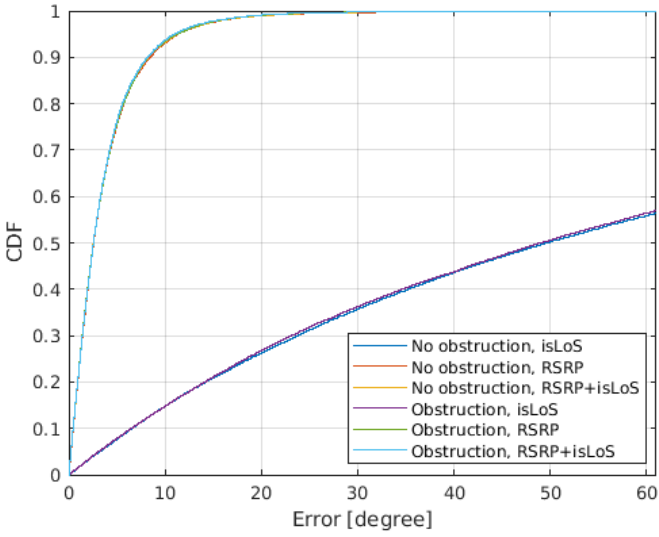


Figure 4.11: *In the extreme interference scenarios, the CDF of the orientation estimation errors of different features using RF, in the with and without obstruction scenarios.*

but not the orientation changes, so the input data cannot be distinguished, causing the not ideal estimation performance. However, using RF method performs better than using KNN method, it is because RF does not use all features, it prioritizes high-importance features to differentiate the input data. As can be seen from the Figure 4.9, for different orientations, several of the features vary greatly, so RF can distinguish the orientation from comparing certain features.

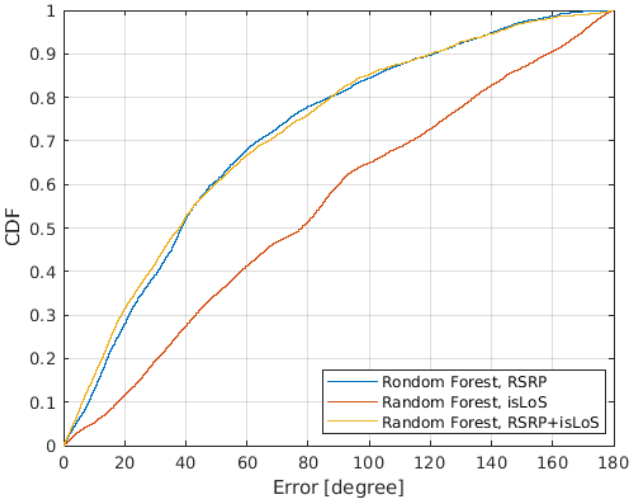


Figure 4.12: *In the no interference scenarios with no obstacle, the CDF of the orientation errors of RF for different input features.*

Table 4.4: *The RMSEs of different features estimation of RF with new UE positions.*

	isLoS	**RSRP**	**RSRP+isLoS**
RMSE	86.9300	66.3523	66.5864

4.4 Random Forest With Modification

In the scenario, where there is no interference and LoS was only considered, the normal RF algorithm without any modification and the RF with modification by implementing the minimum variance estimator was implemented. It is observed from Figure 4.13 that the performance of the RF algorithm improved after we implemented the modification. The reason for the improvement is that instead of taking the average of the estimates from all decision trees, the minimum variance

estimator as defined in Section 3.4.2 estimates the final orientation degree with the smallest variance among all estimates.

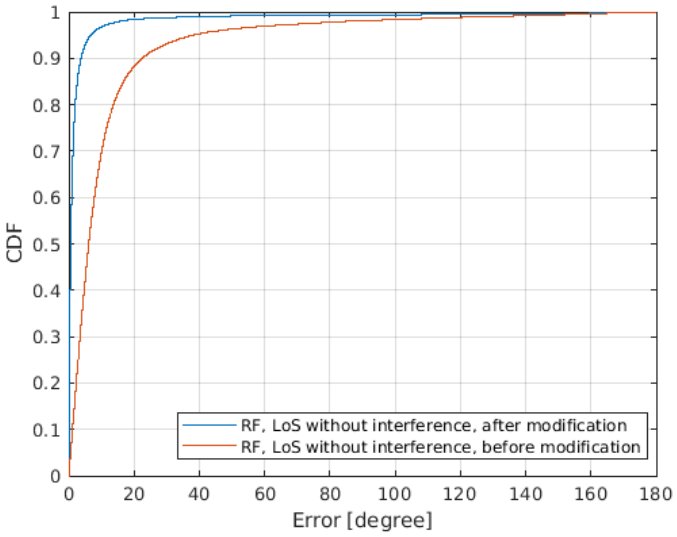


Figure 4.13: *In the no interference scenario with no obstacle, the CDF of the orientation errors of RF before and after implementing the minimum variance estimator.*

4.5 Comparison Between KNN and RF Estimation Performance

In the same scenario, the estimation performance of KNN is better than RF. The main reason is that KNN uses the data with the most similar feature as orientation estimation directly, without doing any regression calculation. RF uses an ensemble learning method for regression, which is a technique that combines predictions from multiple decision trees. It uses linear average for regression calculation, which is not suitable for degree average, as described in Section 3.4.1. So the ensemble learning method introduces some errors in the RF estimation results, which reduces the estimation performance of RF.

Since RF constructs a multitude of decision trees, it takes a certain amount of time during the training period. After the RF model is generated, it takes little time to estimate during the testing period. In contrast, KNN does not need to generate any model, so it does not need to spend any time during the training period. When using the test dataset for estimation, KNN takes a long time since it needs to calculate the distance between data points.

In this chapter, we show the results of this thesis. We evaluated the estimation performance of KNN and RF in different scenarios and with different input Features, and compared the similarities and differences between the two algorithms. In the next chapter, we will do a comprehensive discussion and summary of this thesis.

5

Discussion and Conclusions

This chapter discusses the results that is described in chapter 4, concludes the work by recalling the problem formulation, and suggests the future work.

5.1 Discussion

This section analyses and discusses the results obtained in section 4.

5.1.1 KNN

The KNN algorithm works better than Random Forest after finding the correct neighbor and implementing the minimum variance estimator as shown in Tables 4.1 and 4.3.

From Figure 4.4 it is observed that for most of the scenarios RSRP alone as a feature performs better than the case where a combination of both RSRP and isLoS is used. This is because KNN treats all the features with same importance even though RSRP is more important than isLoS in our case. The addition of isLoS to the training data is not adding any information and only increases the dimensionality of the data. This is further verified in the case when isLoS alone was used as a feature, the performance is lowest for all the interference scenarios. Further it was noticed that, the dependence of orientation angle on LoS is negatively affected when there is extreme interference as seen in Figure 4.5. Another important observation when using isLoS alone as input feature $\boldsymbol{x}$ is that the performance improves when there exist obstructions. This is due to the increase in complexity of the environment which creates more distinct changes in isLoS that helps the model to identify the relationship $\boldsymbol{f}$ between inputs $\boldsymbol{x}$ and outputs $\boldsymbol{y}$

better.

It can also be noticed that the model has the best performance with RSRP as a feature. When RSRP alone is used as a feature, the performance deteriorates for all levels of interference where there is obstruction as shown in Table 4.1.

As described in the Section 2.1.1, one main limitation of KNN is its dependency on the training data $\boldsymbol{T}$, as it does not learn from the training data, but memorizes and uses the closest neighbors' orientation angle to predict the orientation angle $\hat{\boldsymbol{y}}$ of an unseen UE. The dependency of KNN on training data $\boldsymbol{T}$ is also discussed in [28] where another algorithm is used along with KNN to overcome this theoretical limitation. This limitation can also be observed in our results as shown in Table 4.2, when a dataset which contains new UE position is introduced, the model performs poorly. Thus, due to this limitation our model cannot be used for finding the orientation if the data of the UE at that position is not included in the training data $\boldsymbol{T}$ and cannot generalize to untrained/unseen locations. Furthermore, input variables $\boldsymbol{x_0, ..., x_{36}}$ in the data used for training the model have different distributions for two different UEs even with same orientation degree, since the data were recorded in different locations. Hence when a new unseen UE with new position and orientation $\boldsymbol{x_*}$ is given as input to the model it is unable to predict $\boldsymbol{y_*}$ because it has not seen any data that is similar. Capturing these dependencies for different positions and for all orientations can become a complex endeavour. As a starting point the positions of the UEs can be added as an additional input variable.

5.1.2 Random Forest

From Table 4.3 and Table 4.1 it can be seen that RF performs well but not as good as KNN due to the cyclic nature of orientation angles which was solved better in KNN by using minimum variance estimator defined in section 3.4.2, but was not completely solved in case of RF, even though the minimum variance estimator was implemented, as described in Section 4.5.

Similar to KNN, when RSRP is used as a feature, the model performs best comparatively as seen in table 4.3. It can be noted that the model performs better in case there exist scatters. Similar to KNN, when a combination of RSRP and isLoS is used the performance worsened compared to the case when RSRP alone is used as a feature. However, the level of performance degradation is less in the case of RF compared to KNN.

From Figure 4.11 it is observed that regardless of the existence of the obstacles, the performance seems to be similar in experiments where the interference levels are the same. Another interesting observation is that, the performance improves slightly only in the case of extreme interference when RSRP and isLoS are combined as they together give better information about orientation.

Similar to the case of KNN, Section 4.3.3 confirms that the RF model can predict the orientation only if training data $\boldsymbol{T}$ contains information about UE at that specific position. Thus the trained RF model cannot be used for predicting orientation in untrained locations, since the input variables $\boldsymbol{x_0, ..., x_{36}}$ have different distributions in different locations, even if they have the same orientation.

The RF makes predictions that is the average of the previously observed data and it cannot extrapolate. Thus, in our regression problem RF predictions will be bound to the values in the training data, and when the test data differs in distributions it is called as covariate shift which was introduced by Shimodaira [29]. This might be another reason why the RF cannot generalize to unseen/untrained locations as the distributions of relevant variables are different for an unseen UE with a new position.

5.2 Conclusions

Looking back into the problem formulation in chapter 1, where the questions to be investigated were, to study the impacts of UE orientation on received signal features such as ToA, RSRP, and LoS condition, which are the features that are important for the ML model to estimate the orientation of a UE by utilizing the data generated from the simulated environment? How do the interference and the information about LoS condition affect the performance of the ML algorithm? Which ML model has the best performance, given features available? are answered in the following paragraphs.

As described in Section 2.2, different features, such as ToA, RSRP, and LoS condition, were studied. As in Section 3.2 feature selection technique was implemented to identify which are the features contributed to the orientation of the UE from the data. The RSRP was identified as the most relevant feature followed by the LoS condition for the orientation problem and hence the first research question is answered.

The second research question of how the information about LoS condition influences orientation can be found in Section 4 and it was observed that the LoS condition can give some information about the orientation. When it comes to interference, the cases where there is no interference give better results compared to the cases where interferences exist and it can be concluded that interference has negative impact on performance, as seen in Table 4.1 and Table 4.3.

The third research question was to identify which ML model has the best performance, given features available. As per the results in Section 4.1, KNN gave better performance after the minimum-variance estimator was implemented. From Section 4 it is observed that the KNN performs better than the RF once the min-

imum variance estimator has been implemented. Thus the best ML model is identified as per the third research question.

To get a better performance it is very important to take a logarithmic scale, especially for smaller values like RSRP, which helped the model to identify peaks and thus better performance. Another conclusion is to take the cyclic nature of the orientation angles into consideration. Furthermore, the results conclude that RSRP is the best feature that can be used for estimating the orientation of a UE device.

As described in the Chap 1, the training data is very important for any ML problem in order to obtain a model that can predict the target value and we can observe that in Section 4. Thus the conclusion that is that the model can give good performance only in the first case of data splitting where train data is randomly selected as mentioned in Section 4.1, because then the training data also inherits the UE position information. Neither the KNN nor the RF can estimate UE orientation for a UE located in a position which is not covered by training set. As discussed in Section 5.1, the model due to the variance in the input variables in the data is not able to predict the orientation of a new unseen UE with new position, thus the proposed model can only work on trained location.

Much of the time was spent on understanding the features in the data, analyzing the outputs, and why the outputs from the model were not as expected. Once the problem was identified we created a custom solution called minimum variance estimator that helped us to avoid the edge cases that improved the performance of the algorithm as seen in Figure 4.13. This figure proves that the custom method that we implemented works very well for the orientation angles estimation problem.

The results in Section 4 and Section 3.2 prove that for orientation estimation using RSRP values as features is the best for our current scenario, and also KNN works better than RF.

5.3 Future Work

Although the results presented in this thesis are promising, the methods and datasets used for evaluation are very limited. We expect more future work to improve the performance.

5.3.1 Deep Learning And Other Techniques

Deep Learning (DL), which is a class of ML models, has drawn much attention in recent years due to its expressive capacity and convenient optimization capability. Artificial Neural Networks (ANN) are inspired by the sophisticated functionality of human brains where hundreds of billions of interconnected neurons process

information in parallel [30]. Neural networks consist of units that are similar to biological neurons. A feed-forward neural network model also known as Multilayer Perceptron (MLP) is a tool for identifying patterns. Due to the success of deep learning methods in solving complex problems, an MLP was implemented and the performance was similar to a Random Forest without any modification. It was found that the reason why the neural network was not better than the Random Forest was because of the loss function MSE that was used for training the neural network. Hence the future work can be to create a customized loss function for the neural network model that considers the cyclic nature of orientation angles. Another possible approach is to use a regression-enhanced RF by penalized parametric regression technique [31].

5.3.2 More Scenarios

The model that was trained can be evaluated on a different scenario to identify how well this model generalizes to a wider scenario. Another future work is to use the trained model on an outdoor environment and evaluate the performance. If the performance is not as expected then the trained model can be trained again on the new outdoor environment data.

5.3.3 Large/Multi Antenna Panel

In the current receiver configuration that was studied in this paper, there were only two antenna elements. The number of receiver could be increased to achieve better accuracy in the RSRP values by averaging, thus the chance of having a more precise orientation is higher as it considers more antenna elements for calculating the RSRP values.

5.3.4 More Data

In the case that was studied in this paper, 1000 UEs were only considered due to the time it takes for generating the data. A larger data set having more data points will give the ML algorithms a bigger scope of learning, which could boost the performance of the ML algorithm. Hence, a larger data set is promising for obtaining better results.

5.3.5 6 Dimension (6D) Positioning

In our study, we have considered 1D orientation estimation, but in general, orientation is 3D and can be defined by using roll, pitch, and yaw. Thus in the future 3D orientation along with 3D positioning can lead to 6D positioning of the UE.

Appendix

A

Contributions

The thesis project work was done together most of the time especially during the implementation. During the initial days of reviewing previous works, Nikil was responsible for reading works related to machine learning while Jianxin focused on related works which had communication background. The simulation of data for the project from the simulator was carried out together with each of them trying to generate dataset with different set of parameters. Implementation of KNN and RF was done together and with continuos discussions. When the report was written, the communication theory was written by Jianxin and Nikil was reponsible for the introduction chapter. Nikil focused on writing the theoretical chapter of ML whereas Jianxin focused on the related works part. All the other parts was written with equal contribution.

Bibliography

[1] Cynthia L Kendell and Edward D Lemaire. Effect of mobility devices on orientation sensors that contain magnetometers. *CMBES Proceedings*, 31, 2008.

[2] Tatsuya Harada, Hiroto Uchino, Taketoshi Mori, and Tomomasa Sato. Portable orientation estimation device based on accelerometers, magnetometers and gyroscope sensors for sensor network. In *Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, MFI2003.*, pages 191–196. IEEE, 2003.

[3] Grace Wahba. A least squares estimate of satellite attitude. *SIAM review*, 7(3):409–409, 1965.

[4] Demoz Gebre-Egziabher, Roger C Hayward, and J David Powell. Design of multi-sensor attitude determination systems. *IEEE Transactions on aerospace and electronic systems*, 40(2):627–649, 2004.

[5] Josef Justa, Václav Šmídl, and Aleš Hamáček. Fast ahrs filter for accelerometer, magnetometer, and gyroscope combination with separated sensor corrections. *Sensors*, 20(14):3824, 2020.

[6] Angelo M Sabatini. Quaternion-based extended kalman filter for determining orientation by inertial and magnetic sensing. *IEEE transactions on Biomedical Engineering*, 53(7):1346–1356, 2006.

[7] Roberto G Valenti, Ivan Dryanovski, and Jizhong Xiao. Keeping a good attitude: A quaternion-based orientation filter for imus and margs. *Sensors*, 15(8):19302–19330, 2015.

[8] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 44(1.2):206–226, 2000.

[9] Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, and Thomas B Schön. *Machine Learning: A First Course for Engineers and Scientists*. Cambridge University Press, 2022.

[10] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[11] Arash Shahmansoori, Gabriel E Garcia, Giuseppe Destino, Gonzalo Seco-Granados, and Henk Wymeersch. Position and orientation estimation through millimeter-wave mimo in 5g systems. *IEEE Transactions on Wireless Communications*, 17(3):1822–1835, 2017.

[12] Jukka Talvitie, Mike Koivisto, Toni Levanen, Mikko Valkama, Giuseppe Destino, and Henk Wymeersch. High-accuracy joint position and orientation estimation in sparse 5g mmwave channel. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pages 1–7, 2019.

[13] Mohamed Amine Arfaoui, Mohammad Dehghani Soltani, Iman Tavakkolnia, Ali Ghrayeb, Chadi M. Assi, Majid Safari, and Harald Haas. Invoking deep learning for joint estimation of indoor lifi user position and orientation. *IEEE Journal on Selected Areas in Communications*, 39(9):2890–2905, 2021.

[14] Mohammad A Nazari, Gonzalo Seco-Granados, Pontus Johannisson, and Henk Wymeersch. 3d orientation estimation with multiple 5g mmwave base stations. In *ICC 2021-IEEE International Conference on Communications*, pages 1–6. IEEE, 2021.

[15] Jingxue Bi, Yunjia Wang, Xin Li, Hongji Cao, Hongxia Qi, and Yongkang Wang. A novel method of adaptive weighted k-nearest neighbor fingerprint indoor positioning considering user's orientation. *International Journal of Distributed Sensor Networks*, 14(6):1550147718785885, 2018.

[16] Mohamed Amine Arfaoui, Mohammad Dehghani Soltani, Iman Tavakkolnia, Ali Ghrayeb, Chadi M. Assi, Majid Safari, and Harald Haas. Invoking deep learning for joint estimation of indoor lifi user position and orientation. *IEEE Journal on Selected Areas in Communications*, 39(9):2890–2905, 2021.

[17] Kashvi Taunk, Sanjukta De, Srishti Verma, and Aleena Swetapadma. A brief review of nearest neighbor algorithm for learning and classification. In *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, pages 1255–1260, 2019.

[18] Aized Amin Soofi and Arshad Awan. Classification techniques in machine learning: applications and issues. *Journal of Basic and Applied Sciences*, 13:459–465, 2017.

[19] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[20] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[21] Jake Morgan. Classification and regression tree analysis. *Boston: Boston University*, 298, 2014.

[22] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[23] Farhana Afroz, Ramprasad Subramanian, Roshanak Heidary, Kumbesan Sandrasegaran, and Solaiman Ahmed. Sinr, rsrp, rssi and rsrq measurements in long term evolution networks. *International Journal of Wireless & Mobile Networks*, 2015.

[24] Thomas L Marzetta. *Fundamentals of massive MIMO*. Cambridge University Press, 2016.

[25] 3GPP. Technical specification group radio access network; channel model for frequency spectrum above 6 ghz (release 14), 2016.

[26] Ruben Morales Ferre, Gonzalo Seco-Granados, and Elena Simona Lohan. Positioning reference signal design for positioning via 5g. *National Committee for Radiology in Finland: Tampere, Finland*, 2019.

[27] Anthony C. Caputo. 5 - wireless networked video. In Anthony C. Caputo, editor, *Digital Video Surveillance and Security (Second Edition)*, pages 145–204. Butterworth-Heinemann, Boston, second edition edition, 2014.

[28] N Suguna and K Thanushkodi. An improved k-nearest neighbor classification using genetic algorithm. *International Journal of Computer Science Issues*, 7(2):18–21, 2010.

[29] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[30] Sun-Chong Wang. Artificial neural network. In *Interdisciplinary computing in java programming*, pages 81–100. Springer, 2003.

[31] Haozhe Zhang, Dan Nettleton, and Zhengyuan Zhu. Regression-enhanced random forests. *arXiv preprint arXiv:1904.10416*, 2019.