Enhancing Wireless Technologies with Machine Learning

by

Donovan J. Tames

A Thesis
Submitted to the Faculty
of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Master of Science
in
Electrical & Computer Engineering
by

May 6, 2023

APPROVED:

Professor Alexander M. Wyglinski, Research Advisor

Professor Oren Mangoubi, Committee Member

Professor Kaveh Pahlavan, Committee Member

Abstract

This thesis explores ML applied to wireless technologies in two different contexts: 5G cellular networks and radar systems. In the case of 5G, classification ML models are used to identify fundamental scheduling algorithms that a simulated network is using based on a several types of data (UE performance metrics and spectrogram data). As for radar systems, a monostatic radar simulation was built, opening opportunities for future cognitive radar experiments involving adaptive NLFM waveforms via optimization. This thesis exemplifies the importance of ML applied to wireless emissions and sets up future research in the two domains considered.

Acknowledgements

I would like to express my deepest gratitude to all those who have supported and assisted me throughout my studies. First and foremost, I would like to thank my research advisor, Prof. Alex Wyglinski for his guidance, support, and constructive feedback. I would also like to thank Prof. Oren Mangoubi and Prof. Kaveh Pahlavan for partaking in my research committee. I am also grateful for my colleagues in the lab who have given me endless feedback, encouragement, and emotional support. I am particularly thankful for my fellow students, Adriyel Nieves and Mitchell Jacobs, for their valuable contributions to the projects that this thesis consists of. I would also like to acknowledge MIT Lincoln Laboratory Group 33 and Group 65 for giving me the opportunity to continue my education by sponsoring this research. Lastly, I would like to thank the support of my family, who have always been there for me. Thank you all for your contributions, support, and belief in me.

Contents

List of Figures vi

List of Tables vii

1 Revolutionizing Wireless Emission Technology via Machine Learning 1
- 1.1 The Motivation for Using Machine Learning in Wireless . . . 1
- 1.2 5G Cellular Networks . . . 2
- 1.3 Cognitive Radar Systems . . . 4
- 1.4 Thesis Contribution . . . 5
- 1.5 Thesis Organization . . . 6

2 Machine Learning Fundamentals 7
- 2.1 The Premise of Machine Learning . . . 7
- 2.2 Building Accurate Models . . . 11
- 2.3 Data Preprocessing . . . 13
  - 2.3.1 Dimension Reduction . . . 14
  - 2.3.2 Feature Engineering . . . 15
  - 2.3.3 Model Validation . . . 15
- 2.4 Classification Models . . . 16
- 2.5 Optimization . . . 18
- 2.6 Machine Learning in Wireless: Case Studies . . . 18
  - 2.6.1 5G Networks: Predicting Network Traffic . . . 19
  - 2.6.2 Radar Systems: Operating in Challenging Environments . . . 21
- 2.7 Chapter Summary . . . 27

3 Classifying 5G Scheduling Algorithms 28
- 3.1 Technical Challenges . . . 28
- 3.2 Proposed Approach . . . 29
- 3.3 The 5G Network Simulation . . . 31
  - 3.3.1 Simulation Input Parameters . . . 31

3.3.2 Simulation Output Parameters . . . 33
3.3.3 Data Generation and Format . . . 34
3.4 Data Preprocessing . . . 36
3.5 Model Performance . . . 37
3.6 Utilizing Spectrogram Data . . . 40
3.7 Chapter Summary . . . 43

4 Waveform Optimization for Monostatic Radar Systems 45
4.1 Monostatic Radar Simulation . . . 45
4.2 Defined Cost Functions . . . 50
4.3 Chapter Summary . . . 53

5 Conclusions 54
5.1 Research Outcomes . . . 54
5.2 Future Work . . . 55

Bibliography 57

List of Figures

2.1 Unsupervised versus supervised ML 9
2.2 ML model example 10
2.3 Overfit and underfit ML model 13
2.4 Bias-variance trade-off 14
2.5 Bistatic versus monostatic radar configuration 21
2.6 Radar pulses 22
2.7 Radar pulse compression 25
2.8 LFM pulses 26

3.1 Model performance versus timesteps per observation 35
3.2 Pairplot 38
3.3 Highlighted pairplot 40

4.1 Radar simulation LFM pulse 48
4.2 Radar simulation NLFM pulse 49
4.3 Radar simulation NLFM source code 50
4.4 Simulation return signal 51
4.5 Radar simulation range accuracy source code 51
4.6 Simulation return signal (close) 52
4.7 Radar simulation sidelobe height source code 52

List of Tables

2.1 Selection of ML models used in Chapter 3 and descriptive details about each. 17

3.1 Mapping of several UE-to-base-station distances and the corresponding initial CQI measure. 32
3.2 The randomly generated UE distance sets for the experiment. Each set consists of four distances: the first distance corresponding to UE1, the second for UE2, the third for UE3, and the fourth for UE4. 33
3.3 Parameters used to iterate 30 total simulation iterations from which datasets were derived. 33
3.4 Datasets with differing number of input features based on the number of timesteps per observation. 35
3.5 The performance of each ML model, measured by accuracy, precision, recall, and F1 score. 39
3.6 Parameters used to iterate each simulation from which new datasets were derived. 39
3.7 The performance of the ML models when tested on an untrained dataset. . 41
3.8 Parameters used to iterate each simulation from which spectrogram data was derived. 42
3.9 The performance of the top performing ML models using spectrogram data. 42
3.10 The number of m symbols and n REs averaged across to generate lower resolution datasets. 43
3.11 The accuracy of the ML models when tested on an untrained dataset. . . . 43

4.1 A summary of the default value used for several parameters that are used to produce and later shown example result from the radar simulation. . . . 47

Chapter 1

Revolutionizing Wireless Emission Technology via Machine Learning

1.1 The Motivation for Using Machine Learning in Wireless

Wireless technologies have become integral to modern society, providing several different applications including from Internet-of-Things (IoT) devices [1], autonomous vehicles [2], and intelligent tracking and monitoring [3]. As technologies continue to evolve rapidly, the number of practical applications that use these technologies is exponentially growing [4]. These emerging use cases are often built on conventional wireless implementations, such as wireless cellular networks and radar systems [5]. Despite the benefits, contemporary wireless technologies face several challenges. These range from rudimentary challenges, such as simple signal propagation phenomena, such as interference and attenuation [6, 7] to more complex issues, such as limited bandwidth with increasing numbers of cellular users [7] or system failure due to adversarial jamming [8]. Researchers have been actively pursuing solutions years of experimentation and development.

Over the past several years, cognitive computing tools, such as Machine Learning (ML) have been increasingly used in the wireless sector [9, 10], launching the next generation of advancements in the industry [11].

This growth is expected as today's society has entered a new phase of big data with

increasing numbers of available data [12]. From this data, ML algorithms are capable of addressing frequently-studied challenges in wireless such as intelligent resource management and adaptive modulation [10]. The use of ML has become an important technology in wireless while the community continues to generates of vast amounts of data to address new challenges that continue to arise.

This thesis explores how ML can enhance different use cases in wireless transmission systems. In particular, the two applications of interest are 5G networks and radar systems. In the case of 5G networks, this research employs classical ML methods to classify different 5G scheduling algorithms. This could potentially lead to an improved understanding of spectrum usage, potentially advancing development towards sought-after research spaces in cellular communications, such as efficient spectrum utilization and opportunistic scheduling. For radar systems, this thesis investigates waveform parameters and how optimization algorithms could be used to identify better-performing radar waveforms for certain environments. This can enhance radar performance by enabling more accurate and precise target detection. By examining the possibilities of ML in both a 5G and a radar use case, this thesis seeks to advance two of the more profound and conventional wireless applications with practical implications.

1.2 5G Cellular Networks

One active research domain in cellular networks explores coexistence between two or more different networks. In a network, a primary network owns a spectrum license where other networks are not authorized to operate. The primary network may not always use the entirety of this spectrum to its full capacity, leaving opportunities for a secondary network to take advantage of these licensed frequencies. However, identifying unused spectrum in real-time presents difficult challenges. This is a particularly interesting problem in 5G networks as the new 5G standard allows high flexibility in resources assignment [13].

Several studies have explored different methods of predicting this unused spectrum. For instance, one approach predicts future traffic loads based on classified application-layer use cases, such as video streaming, voice calling, or web browsing [14]. Another approach at-

tempts to predict significant spikes in network usage with packet data [15]. Other methods perform traffic clustering with traffic volume data to forecast network utilization [16]. In one case, deep reinforcement learning models were used to adaptively allocate resources using past traffic patterns and channel conditions [14]. Other works have studied how deep neural networks (DNN), recurrent neural networks (RNN), and convolutional neural networks (CNN) can use energy measurements of past resource blocks (RBs) to determine the probability of user presence [13]. An alternative option is to use a long-short term memory (LSTM) model, which can be performed by using data on the number of users and the amount of bandwidth each user will occupy [17]. Furthermore, convolutional LSTM models predict when a primary user on a network is idle to find opportunities for a secondary user to communicate on the network [18]. All of these approaches attempt to forecast future network utilization similarly by analyzing past network traffic patterns. Although these solutions have their own benefits, there are drawbacks each of them carries. These include the following:

1. **An incomplete or narrow perspective**

   Some of the methods focus on identifying specific use cases or behaviors that may not capture the full range of network traffic or events that can impact spectrum utilization. For example, classifying application-layer use cases does not provide a complete picture of how the network resources are utilized. Therefore, these state-of-the-art methods are not feasible because they are not comprehensive approaches to analyzing network traffic.

2. **High model complexity and heavy data dependency**

   Many of the methods use deep learning (DL) models, such as neural networks and LSTMs. These highly complex models require significant computational resources which may not be practical in all cases. Furthermore, these models rely heavily large datasets, which can be difficult to collect, store, and process, especially in predicting future network utilization. As a result, the current solutions are potentially not practical, given that they are expensive and may require high maintenance.

While each of these solutions have made significant strides towards predicting traffic utilization, the described drawbacks limit their effectiveness. A more comprehensive approach that considers the listed problems is necessary to get a complete understanding of network behavior and identify opportunities for secondary networks to operate.

1.3 Cognitive Radar Systems

A common obstacle for radars is operating in challenging environments in which the signal is prone to hindrance that limits the system's operation. This hindrance can be a result of natural interference, caused by attenuation, reflections, and other signal-distorting elements when interfacing with certain environments. Hindrance can also be a result of an adversary intentionally obfuscating the radar's signal. In either case, these hindrances degrade the radar signal and can limit the system's ability to detect targets and measure range, speed, and other properties. In some instances, the hindrance can be severe enough to completely disrupt the radar operation and render the system ineffective.

As a result, several studies have attempted to design robust radar systems that can operate in these environments. One paper explores using frequency hopping techniques and predicting jamming models to avoid interference [19]. Another paper uses adaptive beamforming methods to achieve low sidelobes which reduces noise jamming effects [20]. Other papers take the approach of frequency selection for active jamming devices by analyzing which frequency bands to avoid [21]. Despite the advances for each solution, there also exists several pitfalls, including the following:

1. **Handling dynamic interference**

   The described solutions may not be effective in environments with constantly changing interference, as the radar system's techniques may not be able to keep up. For example, repeat adversaries can adapt jamming strategies to specific frequency hopping or adaptive beamforming patterns used.

2. **Knowledge of the environment**

Some approaches rely on accurate knowledge of the environment or present jamming signals to counteract the effects. For instance, adaptive beamforming techniques need knowledge of a jammer's tendencies to create nulls in the direction of the jamming signals. If the hindrance is not well understood, these techniques may be ineffective.

As previous methods have helped radar systems avoid hindrance for different environments, there remains room for improvement. A solution account for the described pitfalls is needed to further advance avoidance and anti-jamming techniques.

1.4 Thesis Contribution

This thesis seeks to address the aforementioned challenges of ML applied to two distinct contexts in wireless. In the case of 5G cellular networks, the proposed solution considers the following approach:

1. **Classification of 5G network scheduling algorithms**

   Classifying a network's scheduling algorithm can provide a comprehensive and generalizable understanding of the network's behavior. Knowledge of the scheduling behaviors is likely more applicable than current methods that identify categories of network usage. Identifying the schedule type could provide insights as to how a network allocates its resources.

2. **Use of classical machine learning algorithms**

   Using traditional machine learning algorithms will reduce model complexity and computational resources used. Non-DL models are often more accessible, easier to implement, and do not require as much training as DL models. Furthermore, they often do not require as much data, therefore alleviating some of the challenges of handling overly large datasets.

As for radar systems, this thesis proposes the following as a solution to address the aforementioned pitfalls with current methods:

1. **Developing adaptive waveforms using frequency-defining parameters**:

   Using waveforms that can adjust frequency parameters to avoid areas of interference will provide new advancements in radar systems. Unlike other solutions that rely on predetermined patterns, adaptive waveforms have a wider range of flexibility.

2. **Applying optimization algorithms to minimize defined cost functions**:

   By applying cognition techniques, radar systems can learn and identify the hindrance of a given environment and adjust accordingly. This requires no previous knowledge of interference patterns or jamming behaviors. This also addresses concerns of dynamic environments that are constantly changing, as the radar system would have the capacity to change as well.

1.5 Thesis Organization

The remainder of this thesis is organized into the following chapters. Chapter 2 provides an introduction to the basics of ML, an overview of the key techniques utilized in this work, and prefaces two case studies for its application to wireless emissions: 5G cellular networks and cognitive radar systems. Chapter 3 presents the first application – classifying fundamental 5G scheduling algorithms. This chapter overviews data collection procedures, feature engineering techniques, and analysis of the results. Chapter 4 presents the second ML-applied use case – radar waveform optimization. This chapter discusses a simulation used, the cost functions considered, and the waveform parameters. Chapter 5 summarizes each experiment, highlights the contributions of this study, and discusses avenues for future improvement.

Chapter 2

Machine Learning Fundamentals

Prior to understanding how ML can be applied to wireless, it is first necessary to understand the intricacies of how ML works. This chapter will overview what ML is, explain how ML models are developed, highlight how data is prepared for a model, and detail the ML algorithms seen in this thesis' experiments. Additionally, this chapter will provide background on the wireless emission applications this thesis experiments with, 5G networks and radar systems, and how ML can benefit each.

2.1 The Premise of Machine Learning

ML is a subset of artificial intelligence (AI) that uses algorithms to learn from data and improve through trial-and-error [22] to make predictions about some provided input data. ML algorithms learn by example by analyzing the relationships between given input data and its associated output data. Fundamentally, math is the language of ML, as linear algebra and probability are both heavily used by ML algorithms. Linear algebra is used to represent and operate on data passed to an algorithm, which is often organized into matrices. Probability is used as many algorithms model uncertainty with a distribution [22] to make probabilistic predictions. Two of the most powerful aspects of ML is its ability to function purely from data. Rather than traditional programs that need precise instructions, ML has the capacity to operate without explicitly being programmed. Instead, ML is

fed vast amounts of data allowing the algorithm to learn trends and recognize patterns to make predictions. Moreover, ML models improve with each additional data sample, demonstrating the ability to adapt. This allows prediction refinement since models identify more complex relationships between input and output data.

ML is often classified between *supervised*, requiring labeled data to train, or *unsupervised*, capable of training on unlabeled datasets. Supervised ML learns by building a model to estimate an output based on given examples inputs and their associated outputs. On the other hand, unsupervised ML solely takes input data with no associated output data to guide the model, and is responsible for identifying inherent structures in the data. Figure 2.1 shows a breakdown of these two families of ML. The experiments discussed in the following chapters only involve supervised ML methods, therefore the remainder of this thesis will solely focus on ML with labeled datasets. The goal in ML is to generate a model that accurately relates input data to output data. A strong model understands the relation between inputs and outputs enough to accurately predict the output for future sets of input data that have not yet been passed to the model.

To illustrate an example, consider a model that estimates the amount of data a cellular base station has serviced over the last hour. The model will use various elements of the situation, such as distance to the city and time of day. Distance to the city is relevant since population typically increases as distance to the city decreases. Therefore, this will likely mean more users and more data that the base station will need to service. Time of day is important as well because most users are active and working during the day time, and resting at night. These elements are referred to as *features* – the data that the model uses to make predictions [24]. Features are interchangeably referred to as *input variables*, *independent variables*, *predictors*, and *input features*. On the other side, is the *output variable* (also known as the *dependent variable*), what the ML model is trying to predict or estimate [24]. In this example, the output variable is the amount of data serviced by the base station. Via multiple examples, model quickly learns how each feature effects the output – base stations located closer to cities and operation closer to mid-day, independently, will generally result in higher amounts of data serviced for a base station. Combining the features together, the model should provide a more accurate prediction

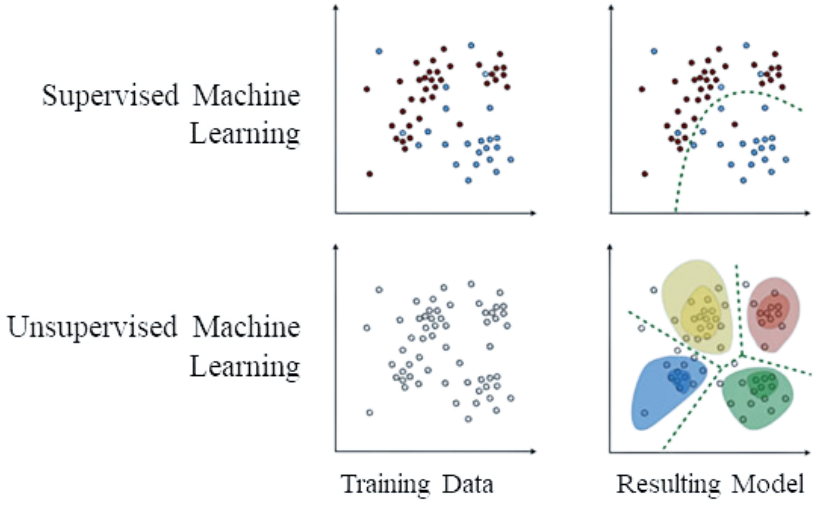

Figure 2.1: The two families of ML: supervised and unsupervised. Supervised ML models are given labeled data and train with ground truth knowledge of the output. Unsupervised ML are given unlabeled data and are responsible for finding patterns on their own. Figure from source [23], modified. Original version distributed under the terms of the Creative Commons Attribution 4.0 International License.

than using only one.

The model described is an example of a *regression* model, since it predicts a *continuous* output variable (data that can take on any value within an interval [24]). In a separate example, consider a model that predicts wireless antenna types. This model uses features such as antenna size and antenna shape. This is an instance of a *classification* model – a model does not predict a numerical value like a regression model, but rather discrete, finite values known as *categorical* data [24]. An application of classification models are be discussed in greater detail in Chapter 3.

For any model, the input variables are usually represented by X with each features distinguished by a subscript X_n. For example, from the base station example model, distance to the city could be identified as be X_1 and the time of day, X_2. The output, often denoted by Y, holds the relationship in Equation (2.1) with each feature $X1, ..., X_p$ for p total input features. This equation represents the general form for any ML model,

where f is an unknown function of X and ϵ is some random Gaussian error [24]:

$$Y = f(X) + \epsilon. \tag{2.1}$$

Since ϵ has approximately a mean error of zero [24], the output, Y, can be estimated with Equation (2.2), where $\hat{Y}$ is a prediction for Y, and $\hat{f}$ is an estimate for the function f [24]:

$$\hat{Y} = \hat{f}(X). \tag{2.2}$$

A possible representation of Equation (2.2) is visualized in Figure 2.2 for a model that predicts the amount of data processed by a base station in the last hour. This model takes

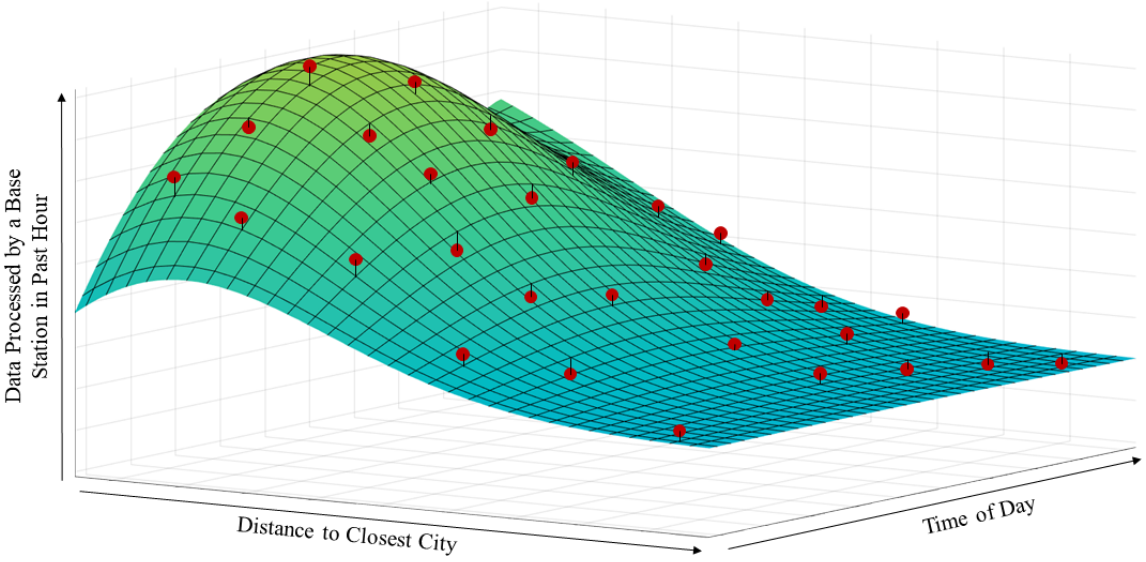

Figure 2.2: A visual demonstration of how a ML algorithm generates a model. The model uses input features, distance to closest city and time of day, to predict the output, data processed by a base station in the past hour. Each observation, denoted by a red point, holds a value for each feature and its associated output, aiding the algorithm to create the blue hyperplane based on trends in the data. Here, a shorter distance to cities and times of the day closer to mid-day generally contribute to higher sums of data processed. The black markers represent the difference between each respective sample's true output versus what the model would predict given that observation.

two input features – distance of the base station to the closest city and the time of day,

represented by X_1 and X_2, respectively.

Each red point on the plot is used to train the model, creating a best fit plane that represents the output of the ML model, Y. For any N number of input features an ML algorithm generates an $N+1$ dimensional hyperplane. For analytical purposes, X can be organized into a two dimensional matrix, such as:

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}. \tag{2.3}$$

Here, x_{ij} represents the value of the jth feature for the ith *sample*, where $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, p$. That is, each column represents the input features of the ML model, and the rows represent the *observations*. In this thesis, an observation refers to a set of input features that describes an output. For the example depicted in Figure 2.2, an array to represent X from Equation (2.3) would consist of two columns for the two input features, distance to closest city and time of day, and each row would represent one of the red data points plotted within the figure, with respect to the plane created by the bottom two axes.

2.2 Building Accurate Models

As stated, the goal in generating an ML model is to accurately predict the response for future predictor data. That is, a well-performing ML model achieves as low an error possible for the *test data* – observations that the ML algorithm did not use to train or fit the model. Any observation that was considered when training the model is referred to as *training data*. In general, achieving low training error is not as important as achieving low test error. This is because training data is comprised of known entities, whereas the test data consists of unseen examples. A model with a low training error does mean that it has learned the patterns and relationships within the training data well, but does not necessarily guarantee that the model will perform well on new data.

In a practical example, consider a model that predicts the behavior of a wireless jammer. While it is certainly possible to generate a model that achieves low error for the jammer's past behavior, it is much more valuable to be able to accurately predict the jammer's future behavior. In the context of Equation (2.2), suppose the training data for the model depicted in Figure 2.2 consists of observations $\{(x_1, y_1), (x_2, y_2), \ldots, (x_{29}, y_{29})\}$. Although $\hat{f}(x_1), \hat{f}(x_2), \ldots, \hat{f}(x_{29})$ may approximately equal $y_1, y_2, \ldots, y_{29}$, that does not ensure that $\hat{f}(x_{30})$ approximately equals y_{30}, where (x_{30}, y_{30}) is an observation of the test data. That is, when tested on new observations the error is can be large. This model is often called to be *overfit* – the model simply memorizes the data rather than learning its patterns. It is desirable to generate a model that can *generalize* to unseen situations. Overfitting occurs when a model is too complex for the number of training observations it is given. That is, a model either has too many input features, capturing small variations in the data, or too few observations where it does not have enough information to capture the underlying patterns.

While it is essential to avoid an overfit model, it is also important to avoid an *underfit* model. Figure 2.3 provides a basic example of what an overfit, underfit, and optimally fit model looks like. Underfitting occurs as a result of a model that is not complex enough and has too few input features. An underfit model does not fit the training data well and is too general for the test data, resulting in both a high training and test error. Therefore, a robust model needs balance in its complexity to ensure it does not overfit or underfit, minimizing the test error.

In ML, test error is comprised of three components: *bias*, *variance*, and *irreducible error*. This is summarized in Equation (2.4), where the first term corresponds to bias; the second term, variance; and the third, irreducible error [24]:

$$E[(Y - \hat{Y})^2] = (f - E[\hat{f}])^2 + E[(\hat{f} - E[\hat{f}])^2] + E[\epsilon^2] \tag{2.4}$$

Bias refers to the amount in which a model's prediction differs from the true value. High bias can occur if a model is too simple and unable to capture the complexity of the relationship in the data – similar to an underfit model. Variance, on the other hand, refers to a model's sensitivity to small fluctuations in the training data. High variance can occur

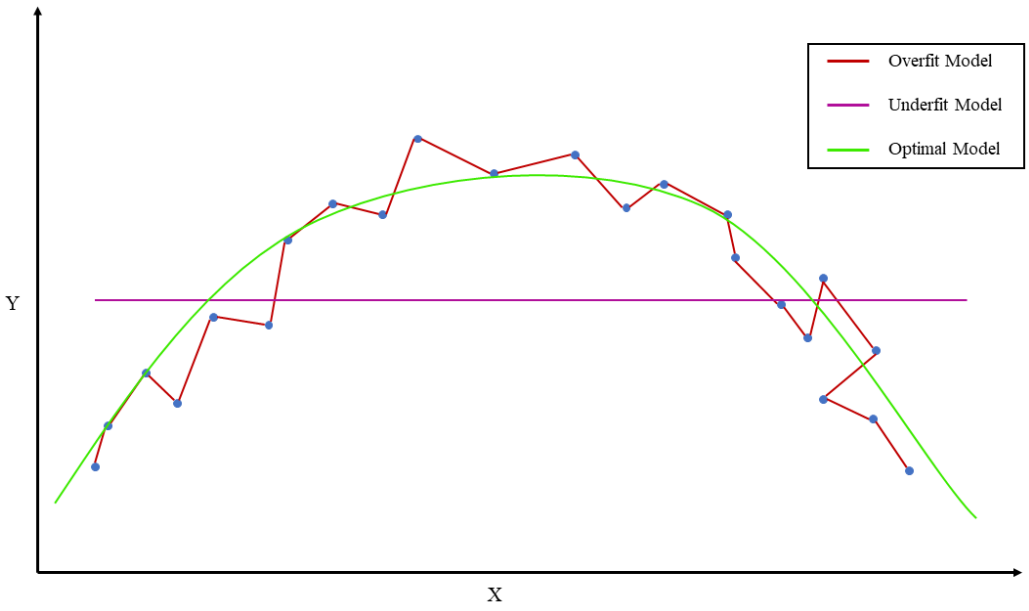

Figure 2.3: An example of how an overfit model, underfit model, and well-fitting model compare to one another. The overfit model matches the data too well, likely leading to poor results on any new data. The underfit model doesn't match the data well enough, missing the overall pattern. The well-fitting model matches the data enough to capture the trend, but not too much that it memorizes the data.

if a model is too complex and fails to generalize to new data – just as an overfit model. Finally, irreducible error is error that from inherent randomness and noise in the data, and is always present regardless of model complexity, hence the ϵ term in Equation (2.2). A model should have low bias and low variance to ensure minimal test error. This balance in model complexity is often called the *bias-variance trade-off* and is demonstrated in Figure 2.4.

2.3 Data Preprocessing

Prior to training a model, there are several steps taken to prepare the data to be used by an ML algorithm. Some preprocessing steps include dimension reduction techniques, feature engineering, and validation methods.

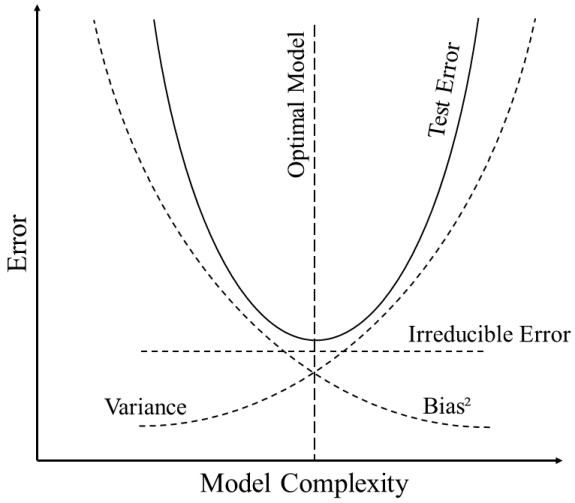

Figure 2.4: The bias-variance trade-off. The plot shows test error is effected by the complexity of a model. With low complexity, variance is low, however bias is too high, leading to high test error. With high complexity, the bias is low, but the variance becomes too high, resulting in high test error again. The best model is balances complexity to minimize both bias and variance for as low a test error possible.

2.3.1 Dimension Reduction

Often times, a model is has too many input features which causes overfitting and therefore poor generalization. One approach to obtain a better model involves *dimension reduction* techniques to limit the number of features to those that are most relevant. In many cases, a high-dimensional dataset contains features that are either narrowly relevant or redundant. Removing these features reduces the complexity of the model, therefore improving the model's ability to generalize. Performing dimension reduction sometimes requires analysis of the input features, specifically their *correlation* – a measure of the relationship between two variables [24]. In a model, it is ideal to have input features that vary independently from one another, so each one contributes unique information to the model.

An effective way to analyze the correlation between multiple input features uses a *pairplot*. A pairplot is a matrix of scatterplots in which each feature in a dataset is plotted against every other feature to provide a visual representation of the correlations between

the variables. This makes the relationships between a model's parameters apparent and understandable. The diagonal plots display the distribution of a single variable.

2.3.2 Feature Engineering

In ML, *feature engineering* refers to alterations that develop raw data into useful features for a model. An example of this is scaling data using *normalization* or *standardization*. Scaling data ensures that all features have the same distribution, mitigating any bias towards features with larger values. This is a critical step prior to applying data to any distance-based ML algorithm, such as k-Nearest Neighbors (KNN) or support vector machine (SVM), which will be discussed later. Normalization confines data to a particular range, usually between zero and one using:

$$X_{normalize} = \frac{X - X_{min}}{X_{max} - X_{min}}, \tag{2.5}$$

where X is the original data, $X_{normalize}$ is the transformed data, and X_{min} and X_{max} are the least and greatest samples in the dataset. Standardization, on the other hand, transforms data such that the mean becomes zero and the standard deviation becomes one [24]. Equation (2.6) defines the process of standardization mathematically:

$$X_{standardize} = \frac{X - \mu}{\sigma}, \tag{2.6}$$

where X is the original data, $X_{standardize}$ is the transformed data, and μ and σ are the mean and standard deviation of X, respectively. Features with vastly different scales can negatively impact ML algorithms, as they may invoke unintentional weight to features with larger values. Scaling the features using normalization and/or standardization counters these problems, ensuring that all features contribute equally to the model.

2.3.3 Model Validation

Although low test error typically indicates a well-performing model, it does not necessarily mean that model generalized and will perform well on all data. Sometimes a model can produce a low test error by chance of luck as a result of the particular dataset it was

tested on. This can occur especially when a test dataset is small or when a model isn't sufficiently tested. A common ML technique known as *validation* addresses this problem by applying several additional datasets during the training phase. These dataset are referred to as the *validation data*, and are used to evaluate the performance of the model and fine-tune its hyperparameters. This ensures the model generalizes to new data.

One of the challenges with with validation data is collecting additional data when the data is limited. A popular method known as *cross validation* (CV) resolves this by resampling the data into multiple subsets, emulating the effects of new datasets. For example, the k-fold CV splits the set of observations into k folds of approximately equal size [24].

2.4 Classification Models

As discussed in Section 2.1, one major type of supervised ML are classification models [24]. The methods used in Chapter 3 relies extensively on classification models to identify scheduling behaviors of 5G networks. The goal in classification is to model the probability that the outcome of an experiment belongs to a certain class given the observation; this is described mathematically as:

$$Pr(Y = k|X = x), \tag{2.7}$$

where Pr indicates the probability measure, X is a random vector of the input data, Y is a random variable of the class, and x is the input value of the random vector X, and k is the class value of the random variable Y. Classification models fall in two different approaches: discriminant function models and probabilistic generative models [25]. Discriminant function models learn the probability equation directly. These models focus on creating decision boundaries to separate the different classes in the data, which can be either linear or non-linear. Logistic regression, SVM classification, and tree-based classifiers fall under this category. Probabilistic generative models learn the probability distribution using Bayes' theorem to calculate Equation (2.7). The Naïve Bayes' Classifier (NBC) falls under this category [25]. Table 2.1 summarizes the key characteristics of each classifier model used in this thesis.

Table 2.1: Selection of ML models used in Chapter 3 and descriptive details about each.

ML Model	**Generative/ Discriminative**	**Linear/Non-Linear Decision Boundary**	**Advantages**	**Disadvantages**	**Feature Engineering**
Logistic Regression	Discriminative	Linear	Simple	Does not perform well with high number of non-linear variables	Normalization, Cross-validation
K-Nearest Neighbor	Discriminative	Non-Linear	Easy to control, Flexibile	Poor generalization	Normalization, Standardization, Cross-validation
Naïve Bayes Classifier	Generative	Linear	Handles large range in scaled data	Assumption of independent features may be false	Standardization, Cross-validation
Decision Tree & Random Forest	Discriminative	Non-Linear	Interpretable	Prone to overfitting	Normalization, Cross-validation
Support Vector Machine	Discriminative	Non-Linear	Robust to data fluctuation and high dimensional datasets	Performs poor with large datasets	Cross-validation

2.5 Optimization

One technique that many ML models, such as neural networks, use to train is optimization [26]. Optimization refers to the process of finding the best possible solution to a problem by minimizing or maximizing a given cost function. Gradient descent is a common example of an optimization algorithm [27]. Gradient descent works by calculating the gradient of the cost function with respect to given parameters and updating the parameters in the direction of the negative gradient. Mathematically, consider a cost function R that evaluates a metric to describe error. A set of parameters, represented by a vector θ, are used to determine the cost. The gradient of the cost function with respect to the parameters indicates the error given the current value of each parameter:

$$\nabla R(\theta^m) = \frac{\partial R(\theta)}{\partial \theta} \mid_{\theta=\theta^m} . \tag{2.8}$$

Since the gradient identifies the direction in the θ-space which R increases the most, the negative gradient identifies where R decreases the most. Decreasing the cost function is desired as it minimizes the error. This process is summarized as [24]:

$$\theta^{m+1} \leftarrow \theta^m - \rho \nabla R(\theta^m), \tag{2.9}$$

where θ^{m+1} is a small step in the direction of negative gradient and p controls the size of that step [24].

2.6 Machine Learning in Wireless: Case Studies

As previously discussed, ML has been used to enhance several wireless technology use cases. The two wireless applications of interest this thesis concerns are cellular 5G networks and radar systems. To provide context for the following chapters that exemplify ML applied to these applications, it is first important to understand the fundamentals of each technology and identify their associated challenges.

2.6.1 5G Networks: Predicting Network Traffic

Research in network traffic patterns has grown since the launch of 5G, as the new standard allows greater flexibility for wireless service providers (WSPs) to define their own networking solutions. The 5G standard uses orthogonal frequency division multiple access (OFDMA), which has enabled this flexibility for WSPs [28]. OFDMA uses subcarrier spacing to define the operating frequency of orthogonal signals that do not interfere with each other, giving WSPs control over resource allocation of users in frequency and time [29]. WSPs develop their own scheduling methods to optimize resource allocation in support of enhanced mobile broadband, massive machine type communication, and ultra-low latency reliable communication use cases [30].

Predicting network traffic is one challenge in wireless that ML has been used to address. This thesis presents analyzing scheduling behaviors as a novel method to understand network traffic. In particular, ML is used to classify different types of scheduling algorithms employed in a simulated 5G network.

At a high level, the downlink communication between the base station and a user for a stand-alone 5G network architecture works as follows. A user equipment (UE) (*i.e.*, mobile device, sensor node) can connect to base stations referred to as gNodeB (gNB) nodes using a handshaking procedure based on the hybrid automatic repeat request (Hybrid-ARQ). The Hybrid-ARQ defines a bit that indicates a successfully decoded information block on the uplink and downlink and retransmits the information if the block was not successfully decoded [31]. The gNB determines a schedule that prioritizes UEs based on Quality of Service (QoS), throughput requirements, the number of UEs on the network, and the number of retransmissions a UE has made [32].

The scheduling algorithm that the gNB uses is integral towards several network parameters that are indicative of patterns in its traffic: the throughput (TP), goodput (GP), and buffer status (BUF) that devices achieve on the network. TP, GP, and BUF are calculated based on the schedule resources for each UE. TP is directly proportional to the number of resource blocks (RBs) used by the UE. Therefore, the number of scheduled resources correlates to a higher throughput. Similarly, GP is also proportional to the number of RBs,

but only accounts for the data that has been successfully transmitted on the network. Therefore, a higher number of scheduled RBs correlates with a higher GP for data that has been successfully transmitted on the first attempt, indicating a higher CQI. Finally, BUF is correlated to the traffic on the network and the priority of the device required to use the network. A UE's BUF will be larger if it is not a high priority and has lower CQI, resulting in a high latency for that specific device. Each of these metrics are dependent on the scheduling algorithm a network is using.

There are several types of scheduling algorithms that networks can use to allocate resources and schedule users. This thesis considers three fundamental types that demonstrate different scheduling styles – Round Robin (RR), Best Channel Quality Indicator (BCQI), and Proportional Fair (PF).

A RR algorithm cycles through the users within the network one-by-one based on a predefined order. This is advantageous because all users get maximum fairness and it is simple to implement [33, 34]. However, RR does not account for channel quality, therefore the throughput, latency, and spectral efficiency suffer performance.

A BCQI algorithm schedules the next user on the network based on the channel quality of the users [33, 35]. Channel quality indicator (CQI) is calculated based on the 5G New Radio (NR) Medium Access Control standard that is representative of a gNB's ability to connect to a user [32, 33]. Several factors that affect channel quality include the environment, weather, and transmit power of the device. BCQI reduces the number of retransmissions and increases efficiency since it schedules the user with the best CQI first, thus eliminating the likelihood of needing to retransmit. The downside of BCQI is that it is prone to starving users on the network that may not have as good a connection.

PF considers both the CQI of users on the network and the average data rate of each user. The algorithm assigns resources so that the data rate remains equal from other users on the network [33,36,37]. The PF algorithm is the best of both worlds for optimizing users of the best CQI while sharing the resources fairly. Fairness comes at a disadvantage since it is not as spectrally efficient and introduces latency that may ignore service requirements for mission critical applications.

2.6.2 Radar Systems: Operating in Challenging Environments

Radars face significant challenges when operating in interference and jamming-infested environments. Cognitive radars, driven by ML techniques, are often used to help overcome these challenges [19]. This thesis presents adaptive waveforms as a method to mitigating the effects that interference and jammers have on a radar's performance. Specifically, a radar simulation is built to enable experiments involving waveform optimization with defined cost functions. Before describing the simulation, it is first essential to understand radar fundamentals.

Radar is one of the oldest forms of wireless technology that is still widely used today and remains essential in modern systems [38]. Radars, which stands for radio detection and ranging, are used to collect information about a target via signal echoes. Specifically, they operate by transmitting radio-frequency (RF) electromagnetic waves at a target of interest, and receiving the same signal reflected back off of the target [39]. Some of the information radars are capable of obtaining includes range, velocity, angle, size, and shape [40].

In its simplest form, a radar consists of a transmitter, receiver, and antenna(s). Figure 2.5 illustrates two fundamental configurations of radar systems: monostatic and bistatic [39]. Monostatic radars consist of a single antenna that both the transmitter and receiver

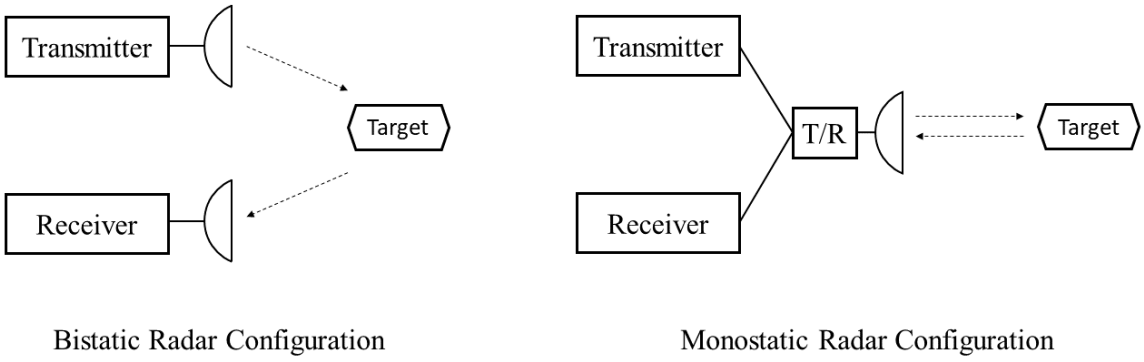

Figure 2.5: A bistatic radar configruation (left) versus a monostatic radar configuration (right). Bistatic radar systems consist of two separate antennas, one for each the transmitter and the receiver. Monostatic radar systems, on the other hand, consist of one antenna that serves both. A duplexer is used to switch between which one the antenna is actively serving.

use, or an antenna for each set in the same location. Bistatic radar consists of two antennas with separation between them [39]. This thesis focuses on the monostatic configuration for simplicity [39].

In radar systems, various types of waveforms are used depending on the application. The highlest level taxonomy of these waveforms are classified between continuous wave (CW) or pulsed [39]. In CW radars, both the transmitter and receiver continuously operate. CW typically adopts the a bistatic configuration [39], given that monostatic systems usually consist of one antenna and therefore can only have one of either the transmitter or receiver active at a time.

In contrast, pulse radars transmit RF waves in short bursts of finite duration, normally at regular intervals [39]. Figure 2.6 explains the basic operation of pulse radar systems. The duration at which the the pulses are transmitted is often referred to as the *pulse*

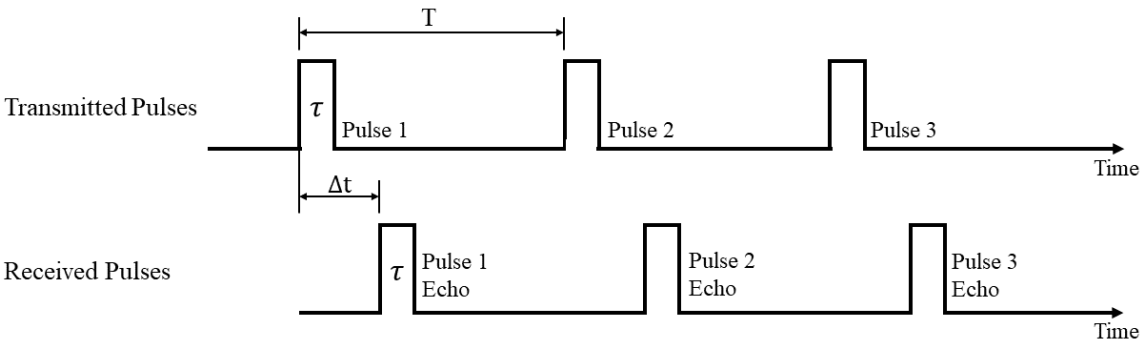

Figure 2.6: The general operation of a pulse radar system. On top shows periodically transmitted pulses, and below shows the received echos.

width, denoted by τ [39]. The variable Δt represents the elapsed time between the start of a transmission to the initial time the echo returns. The time from the start of one pulse to the start of the next is known as the *pulse repetition interval* (PRI) and is sometimes denoted by T, as in Figure 2.6. Similarly, the number of PRIs per second is called the *pulse repetition frequency* (PRF) and is measured in pulses per second (hertz). The relationship between PRI and PRF is provided by [39]:

$$PRF = \frac{1}{PRI} = \frac{1}{T}. \tag{2.10}$$

The receiver is active any time the radar is not transmitting. This transmit period is another important concept known as *duty cycle* – the amount of time in one cycle that the radar is actively transmitting. This is mathematically described as [39]:

$$d_t = \frac{\tau}{PRI} = \tau \times PRF, \tag{2.11}$$

where d_t is the duty cycle. Since emitted RF waves travel through the air at the speed of light, we can calculate the distance to a target, represented as:

$$r = \frac{ct}{2}, \tag{2.12}$$

where r is the line of sight range between the radar and the target in meters, c is the speed of light which is 299,792,458 meters per second, and t is the time elapsed between signal transmission and the received echo in seconds [39]. The factor of two in the denominator comes from the fact that the signal must to the target and back to the receiver.

One common problem in radar systems is known as *range ambiguity*. This occurs when a radar is unable to accurately determine the range to a target. For example, refer to Figure 2.6. We can observe that ranges corresponding to a Δt within T is an unambiguous range. However, ranges that correspond to Δt greater than T are ambiguous, as the return pulse could be interpreted as either the return from transmit pulse 1 or transmit pulse 2 [41]. Range ambiguity is avoided if the PRI is long enough such that the echo(es) from a certain transmit return prior to the next transmit [39].

Another concept in radar systems, *range resolution*, describes the radar's ability to distinguish between multiple targets close in range [41]. Range resolution is important for instances in which multiple targets are close enough in proximity that the reflected signals overlap in time and appear as a single echo. Pulse width (the duration of a transmitted signal) plays the biggest role in determining the minimum distance between two targets for the system to be able to identify them as separate entities. To demonstrate this, consider two targets at ranges r_1 and r_2, respectively. The difference between these ranges is calculated as:

$$r_2 - r_1 = \frac{c(\Delta t_2 - \Delta t_1)}{2}, \tag{2.13}$$

where Δt_1 and Δt_2 is the time elapsed from transmission to return for target 1 at r_1 and target 2 at r_2, respectively. The minimum that Δt_2-Δt_1 can be such that the radar will be able to identify the two separate targets is directly related to pulse width, τ. Two separate return pulses will result assuming the targets are at least a distance of $c\tau/2$ apart [41]. This can be rewritten using the pulse bandwidth, β, or the inverse of τ [41]:

$$r_2 - r_1 \geq \frac{c\tau}{2} = \frac{c}{2\beta}. \tag{2.14}$$

As a result, minimizing $r_2 - r_1$ and therefore the pulse width enhances a radar's range resolution. However, shorter pulses have less energy [39] and therefore decreases the power of the transmit. Additionally, the pulse bandwidth will become too large if the pulse width is minimized, as the two are inverses of one another. An important technique known as *pulse compression* maintains both a reasonable pulse width and high range resolution.

Pulse compression is a powerful method for improving a radar's range resolution, while keeping a satisfactory pulse width. Most pulse compression techniques make use of a matched filter at the receiver [41]. The matched filter convolves the return pulse with a reference signal, which is the time-reversed complex conjugate of the transmitted pulse; the following expression explains this operation:

$$y(t) = x(t) * h(t), \tag{2.15}$$

where $y(t)$ is the resulting pulse compressed signal, $x(t)$ is the return pulse, $h(t)$ is the reference signal (output of the matched filter), and the $*$ operator denotes convolution [41]. This operation amplifies the components that match the reference signal and attenuate those that do not. An example of what this looks like in practice is depicted in Figure 2.7. Here, two the return pulse from different targets overlap in the time domain, making it challenging to distinguish the targets. However, the resulting pulse compressed signals clearly reveal them.

With pulse compression, radars achieve high range resolution using long pulses (high τ); longer pulses result in higher average power, therefore improving the signal's signal-to-noise ratio (SNR). More specifically, the SNR improves by a factor of the *time-bandwidth product*, or $\tau\beta$ [39]. The relationship between the SNR prior to pulse compression versus

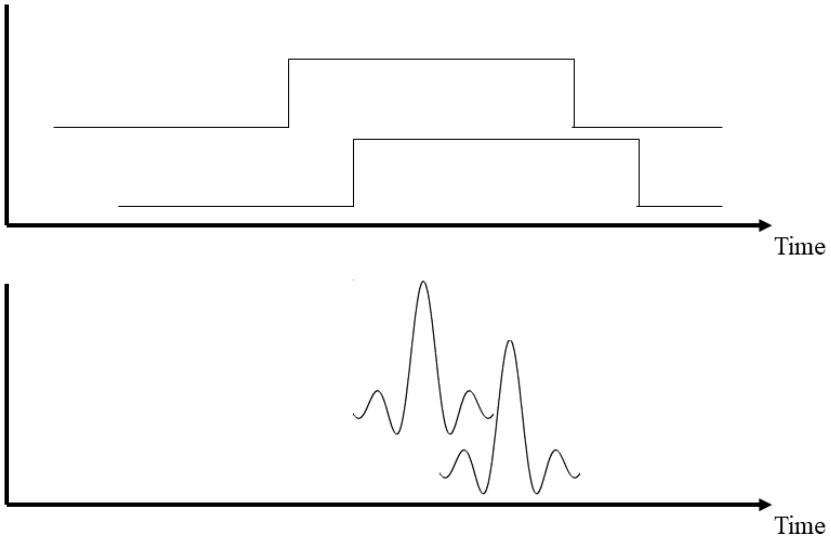

Figure 2.7: Two received pulses before pulse compression (top) versus after pulse compression (bottom).

after is defined as:

$$SNR_{pc} = SNR_u \tau \beta, \tag{2.16}$$

where SNR_{pc} is the signal-to-noise ratio (SNR) after pulse compression, and SNR_u is the SNR without pulse compression [39].

The level of impact pulse compression has on a return pulse can depend on which modulation scheme is used. Specifically, modulation schemes are selected to maximize the pulse compression ratio [41]. The pulse compression ratio refers to the difference between the power of a pulse compressed return signal's main lobe and power of its sidelobes – it is also called *sidelobe height*.

As with many wireless systems, in radar a message signal $x_m(t)$ is used to modulate a carrier signal $x_c(t)$ prior to transmission. The carrier signal is defined mathematically as:

$$x_c(t) = a(t)cos(2\pi f_c t + \phi(t)), \tag{2.17}$$

where $a(t)$ is the envelop of the carrier signal, f_c is the carrier frequency, t is time, and $\phi(t)$ is the phase modulation [41]. The carrier signal can be modulated in a variety of ways,

modifying its amplitude, phase, or frequency [41]. This thesis mostly concerns frequency modulation (FM), particularly two foundational types: *linear frequency modulation* (LFM) and *non-linear frequency modulation* (NLFM) [42]. In frequency modulation, the frequency of the carrier signal varies in accordance with the message signal. In the case of LFM, frequency sweeps linearly for the duration of the pulse width as shown in Figure 2.8 for both an increasingly LFM (*up-chirp*) and decreasing LFM (*down-chrip*) [41].

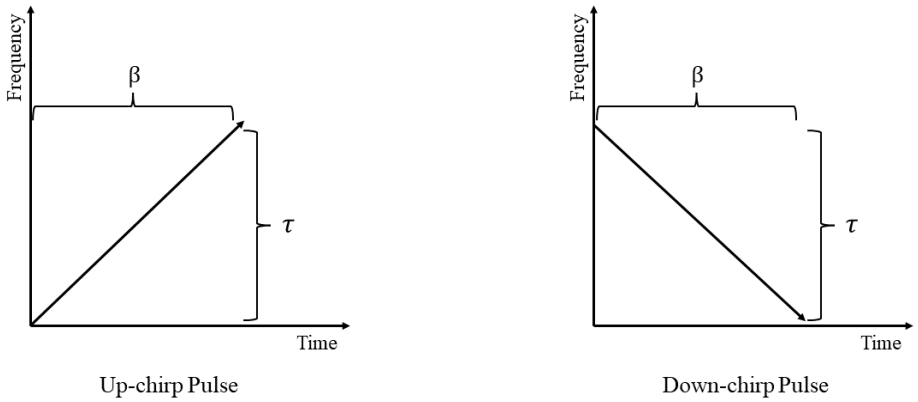

Figure 2.8: An increasing and decreasing LFM pulse plotted in time-frequency. The characteristics of each pulse (pulse width and pulse bandwidth) are marked to visually demonstrate their role in defining the pulse.

In general, the complex envelope of an increasing LFM waveform can be formulated as a complex exponential expression:

$$\tilde{x}(t) = a(t)e^{j\pi(\beta/\tau)t^2}, \tag{2.18}$$

where $\tilde{x}(t)$ is the complex envelope of the increasing LFM waveform, $a(t)$ is envelope that defines the amplitude modulation of the pulse, β is the pulse bandwidth, τ is the pulse width, t is time [43]. The frequency is defined by $\frac{\beta}{\tau}t$ – a linear function of time with a slope of the pulse bandwidth divided by the pulse width [43]. The main advantage of LFM waveforms is that they are easy to generate and provide excellent range resolution [44]. Therefore, they are one of the most commonly used waveforms in radars for maximizing the benefits of pulse compression [45].

Similar to LFM waveforms, NLFM waveforms are frequency modulated. However,

NLFM waveforms' frequencies do not sweep linearly for the duration of the pulse width. Rather, NLFM waveforms' frequencies are defined by some non-linear function, for example, a quadratic or logatithmic function [46]. For a quadratic NLFM waveform, frequency is defined by the following:

$$f(t) = f_0 + \beta t^2, \tag{2.19}$$

where $f(t)$ is the frequency at t time and f_0 is the initial frequency at time $t = 0$ in the pulse duration [46]. The pulse bandwidth, β, is equal to $(f_1 - f_0)/t^2$ where f_1 is the frequency at a reference time [46].

In some instances, a NLFM waveform is better than a LFM waveform, offering improved the sidelobe performance and range resolution [47, 48]. This is particularly true when monitoring targets that reflect weak signals or targets in a cluttered environment [48]. However, NLFM waveforms can be challenging to generate [47] and it can be difficult determining which NLFM waveform will provide optimal performance. This thesis seeks to explore how an optimal NLFM waveform for radar systems can be identified.

2.7 Chapter Summary

This chapter explained the fundamentals of ML, highlighted how 5G cellular networks schedule, and described the basic concepts behind radar systems. In ML, models are generated to make predictions based on some provided data. One type of model this thesis uses, classification, predicts discrete categories that observations belong to. Chapter 3 uses the following ML algorithms to develop several models: logistic regression, KNN, NBC, DT, random forest, SVM. These models are used to classify scheduling algorithms (RR, BCQI, PF) used by 5G networks. The models use UE performance metric data, such as TP, GP, and BUF. A different technique many ML models use is optimization. Chapter 4 establishes the framework of a radar simulation for waveform optimization. The simulation is built using integral pulse-defining parameters, such as pulse width, pulse bandwidth, modulation type, etc. Two fundamental modulation types in radar includes LFM and NLFM.

Chapter 3

Classifying 5G Scheduling Algorithms

This chapter highlights how ML can contribute to 5G network implementations. The experimented described in this chapter overviews a novel approach to predict future traffic of a 5G network: analyzing 5G scheduling algorithms [1]. This experiment uses several ML classification models (multinomial logistic regression, KNN, NBC, DT, random forest, and SVM) to classify the scheduling algorithm (RR, BCQI, PF) used by a simulated 5G network.

3.1 Technical Challenges

There exist several obstacles in understanding network traffic. To implement a novel approach, three relevant technical challenges were identified. These are listed and explained as follows:

1. **Collecting 5G network data in a repeatable and controlled environment**

[1] The work presented in this chapter was completed in collaboration with fellow graduate student Mr. Adriyel Nieves. My contributions to the presented research includes concept inception; experiment framework discovery, such as the simulation and model input/output; ML model implementation; and spectrogram data experimentation. Mr. Nieves efforts focused on data collection; data preprocessing, formatting, and preparation; model performance evaluation; and integrating the simulation's physical layer.

Real-world network traffic records sourced from WSPs are used as datasets for many models [14, 16, 17, 49]. Although this approach has worked for many studies, much of the metadata for the datasets is unknown (*i.e.*, schedule algorithm, subcarrier spacing, number of users, etc.). An alternative data collection method would be to passively record 5G traffic over-the-air from a nearby base station [18], but similarly, with no ground-truth knowledge of the metadata this approach was deemed infeasible.

2. **Determining a method capable of interpreting 5G network behavior to predict traffic**

   Previous studies explore various methods used to understand network traffic. These methods involve tracking spikes in usage, determining users in an idle state, adaptively allocating resources, etc. Each method is similar in the way that they all attempt to directly forecast utilization [13–18, 49]. Discovering *new* methods to analyze network behaviors that previous studies have not attempted is key.

3. **Identifying features that characterize 5G network behavior**

   Choosing which metrics to use as input to the ML models is critical. Previous studies used a variety of different input features. Several papers used packet information to predict network traffic volume [14, 15, 17]. Others use historical traffic utilization to predict peak trends of the network [16, 49]. Furthermore, I/Q data and energy levels have been used to predict the presence UEs [13, 18]. Using the right amount of data types to balance high accuracy and the use computational resources is another consideration.

3.2 Proposed Approach

ML has served well in predicting network traffic, as the open literature explores the previously tried methods. Several types of ML models have been used to predict network utilization [13–18, 49]. Most of these papers use some method of deep learning, whether it was a convolutional neural network, recurrent neural network, or LSTM model [13, 17, 18]. However, these models are computationally expensive and require large amounts of data,

this making deep learning techniques an unviable option. As a result, this experiment uses classical ML methods. The following approach offers a novel method of predicting 5G network traffic, while considering the previously outlined technical challenges:

1. **Constrain the environment to a simulation that mimics the operation of a 5G network**

   A simulated 5G network offers a repeatable and controlled environment to generate a robust dataset. The simulation has parameters that define the physical layer, medium access control layer, and control user attributes such as QoS and channel quality indicator (CQI). One alternative to a pure simulation, emulating a hardware network using open Radio Access Networks (O-RAN), was considered but was deemed inflexible because the number of emulated devices was limited by the available hardware. Therefore, a simulated environment that mimics a 5G network was determined as the best option [13, 15].

2. **Classify the scheduling algorithm that the simulated 5G network is operating on**

   One paper uses a similar method [16], which applies k-nearest neighbor (KNN) to classify each user's network usage type as an approach to predicting resource allocation. This led to exploration of other ML methods that deviated from deep learning and novel approaches to predict resource allocation that do not directly involve forecasting network traffic, as many papers have already attempted. This method involves trying a variety of ML models – logistic regression (multinomial), KNN, NBC (Gaussian), DT, random forest, and SVM – to classify different scheduling algorithms. ML offers both generative and discriminative models that can learn linear and non-linear decision boundaries of a selected dataset which is advantageous because the optimal decision boundary shape is unknown [24, 25].

3. **Use throughput, goodput, and buffer status from the 5G scheduling simulation as model predictor variables and format the data such that it can be analyzed with ML techniques**

The resource grid (RG), a mapping of the resource allocation in the time and frequency domain, has an inherent structure based on its scheduling algorithm. This thesis proposes that time-series throughput (TP), goodput (GP), and buffer status (BUF) measurements efficiently represent important aspects of the RG [14, 15]. The time-series data format was based on how many input features constitute a single observation – more timesteps per observation increases the model complexity. The end goal for this experiment is a selection of traditional ML models that use simulated 5G user performance data to classify different scheduling algorithms.

The proposed solution poses a new approach to predicting network traffic, while considering the aforementioned challenges. The simulation that tied all of these steps together was integral in implementing each of these steps.

3.3 The 5G Network Simulation

Before diving into the implementation of the experiment, it is first important to understand the simulation used, a modified version of the NR FDD Scheduling Performance Evaluation example from MATLAB's 5G Toolbox™ [50]. In the following subsection, the operation of the simulation, its input parameters, and output data is explained in detail.

3.3.1 Simulation Input Parameters

The simulation generates various UE performance metrics on the simulated 5G network based on several input parameters, by modeling scheduling of DL and UL resources. The simulation schedules based on a number of parameters:

1. UE Distance Sets: Each UE's distance from the base station.
2. CQI Update Periodicity: How often the base station requests a new CQI from each UE.
3. ΔCQI: The amount by which the CQI changes every period.
4. Scheduling Algorithm: The algorithm the network uses to schedule UEs.

Each of these parameters act as inputs to the simulation, and influences how the simulation schedules. Several of these parameters are based on CQI, a measure used to represent the quality of a communication channel between a transmitter and receiver; environmental factors that hinder communications, such as a greater distance between nodes will result in a worse CQI. CQI quantified as a number between 0 and 15, where greater values indicate a stronger channel quality. CQI is one of the fundamental variables of the scheduling algorithms studied in this experiment, hence, the three of the simulation parameters are centered around it: UE distance sets, CQI update periodicity, and ΔCQI.

In the simulation, the number of UEs is defined; this experimented uses 4 UEs. The distance between a UE and the base station is used to determine the initial CQI for each UE. The distance for every UE is defined in the simulation, and each distance is associated with a particular CQI measure. Table 3.1 summarizes each distance's corresponding initial CQI. A distance set refers to an array of N distances, where N is the number of UEs – in this case that is 4. The simulation defines a distance set by randomly selecting 4 distances from [1200, 800, 500, 200] meters, one for each of the 4 UEs. The these UE distance sets are used for the first simulation input parameter, and introduces various scenarios of the network users' initial CQI values. For this experiment, the number UE distance sets was chosen as 5, therefore the simulation used 5 different sets of randomly selected distances. The particular randomly generated distance sets used are specified in Table 3.2

Table 3.1: Mapping of several UE-to-base-station distances and the corresponding initial CQI measure.

UE Distance	**CQI Value**
200	15
500	12
800	10
1000	8
1200	7

While the UE distance defines an initial CQI, the CQI for each UE does not remain constant – this is meant to model a dynamic environment, which is similar to how a real-world network operates, where channel conditions are constantly changing. CQI update periodicity defines how often the CQI changes in the simulation. This experiment consisted

Table 3.2: The randomly generated UE distance sets for the experiment. Each set consists of four distances: the first distance corresponding to UE1, the second for UE2, the third for UE3, and the fourth for UE4.

Set Number	**Distance Set (meters)**
1	[1200, 1000, 200, 500]
2	[1000, 200, 500, 1000]
3	[800, 800, 500, 500]
4	[1200, 1000, 800, 500]
5	[1200, 1000, 1200, 1000]

of 2 update periodicities: 0.1 seconds and 0.2 seconds. Finally, the amount by which the CQI changes is defined by the ΔCQI parameter. The value of ΔCQI is either added to or subtracted from the CQI each update periodicity cycle. ΔCQI was chosen to be 2 for the experiment. The last input parameter was the scheduling algorithm itself. This parameter included a selection between RR, BCQI, and PF, and acted as the ground truth for the ML algorithms to train – the output, or Y from Equation (2.1) and (2.2). A breakdown of the simulation parameters is provided in Table 3.3

Table 3.3: Parameters used to iterate 30 total simulation iterations from which datasets were derived.

Simulation Parameter	**Selection**
UE Distance Sets	Sets 1-5 from Table 3.2
CQI Update Periodicity	0.1, 0.2 seconds
ΔCQI	2
Scheduling Algorithm	RR, BQI, PF

3.3.2 Simulation Output Parameters

The output of the simulation, on the other hand, consisted of various time-series metrics for each UE. This data included the TP, GP, and BUF. Based on the scheduling algorithm and channel quality (*e.g.*, CQI) captured by the simulation's other three input parameters, each user on the network will see a different TP, GP, and BUF [33]. TP is defined as the bits-per-second transmitted from each user; GP is the successfully transmitted bits-per-

second without including the retransmitted data; BUF is the data locally stored by the user waiting to be transmitted. High quality channels exhibit high CQI values, where the TP and GP approach the same value since there would be no need for retransmissions and BUF would be empty for the same reason. However, this is not the case for real-world scheduling algorithm implementations.

3.3.3 Data Generation and Format

Given the four different input parameters for the simulation, each unique combination of the parameters was used to generate a robust dataset, that captures various different looks for a 5G network. The dataset for the ML models consisted of an iteration of the simulation for each unique combination of the input parameters, shown in Table 3.3. That is, the simulation was run for every combination across the five different UE distance sets, two different CQI update periodicities, one ΔCQI, and three scheduling algorithms. This amounted to 30 distinct experiments that constitutes the entirety of the ML train/test dataset. Each iteration of the simulation captured five seconds worth of data, generating a sample of the output data every millisecond, for a total of 5,000 timesteps per simulation iteration. Each timestep consists of a total of 12 measurements: three metrics, TP, GP, and BUF, for each of the four UEs.

One consideration of the data format was defining the size of one observation for the ML models. That is, how many timesteps are to be included for each observation. Here, each additional timestep per observation contributes 12 more input features (from the 12 measurements per timestep). Recall the number of input features determines the complexity of a model, and the model should not be too complex or not complex enough to balance the bias-variance trade-off. Similarly, it is important to provide enough timesteps per observation for the models to learn the behavior of the scheduler across a subset of time, but not too many timesteps for the model to overfit. Table 3.4 outlines different numbers of timesteps per observation, and the corresponding number of total input features per observation for each.

It is clear that with more timesteps per observation, the number of input features increases, therefore increasing the model complexity. To determine how many input features

Table 3.4: Datasets with differing number of input features based on the number of timesteps per observation.

Timesteps per Observations	Input Features per Observation
1	12
5	60
10	120
20	240
50	600
100	1200
200	2400

to use, Figure 3.1 demonstrates each model's performance plotted over a varying number of timesteps per observation (1, 5, 10, 20, 50, 100, and 200).

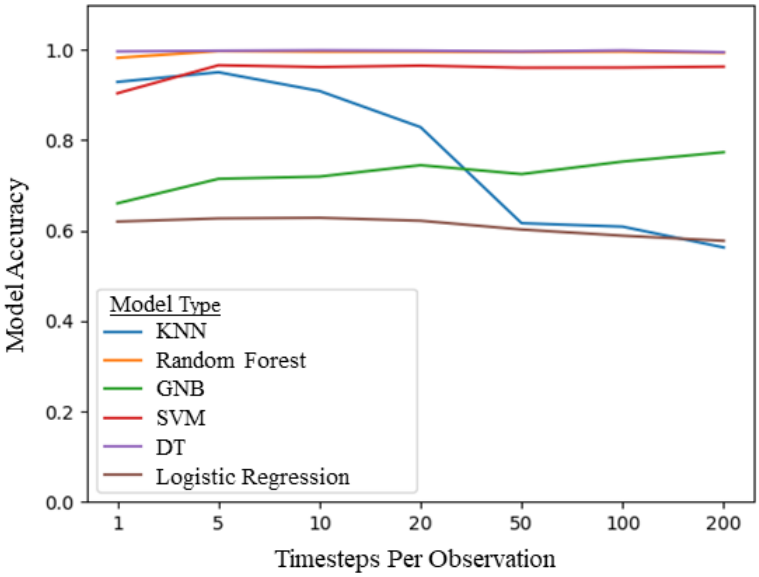

Figure 3.1: Certain machine learning models improve or degrade with performance based on the number of timesteps per observation. Random forest and DT classifiers maintained a high performance regardless of the number of time steps per observation

Based on Figure 3.1 most of the models perform with higher accuracy when they are less

complex, where the number of timesteps per observation is equal about one to five. As the complexity increases, the random forest and DT classifier is still able to achieve near-perfect classification accuracy. The SVM classifier and the NBC exhibit a slight improvement as the complexity of the model increases. Finally, the logistic regression shows a slight decrease as model complexity increases and the KNN has the steepest decrease in accuracy as the model complexity increases [24]. For the experiment, five timesteps per observation was selected – a total of 60 input features for each.

3.4 Data Preprocessing

Prior to training ML models, several analysis and modification techniques were applied the dataset. These preprocessing steps included data reduction, data scaling, validation, in additional to several other.

In classification models, it is good practice to ensure that an approximately equal number of observations belongs to each class. This is often done by reducing the number of observations for classes with significantly more data. Since the dataset was simulated, the dataset is balanced with an equal number of RR, BCQI, and PF observations, and therefore no modifications were required.

Some of the classification types required a normalization and/or standardization step prior to being passed into the model. In particular, normalization was applied to the NBC and logistic regression models using the min-max transform, which scales the values by the difference of the minimum and maximum values within the dataset, as explained mathematically in Equation (2.5). Standardization was applied to the KNN and SVM models using the standard scaler method, as explained in Equation (2.6).

A sufficient cross-validation step was also implemented to ensure that the trained models did not get lucky with the dataset. The cross-validation method selected was K-fold cross-validation with a size of five folds. The precision and recall were used as the scoring mechanisms of the cross-validation step. The dataset was split with a test-train split of 10% and 90%, respectively. The train data was used as the test-validation data in the cross-validation step. The corresponding score was averaged for each of the train models

and the maximum score was selected as the best algorithm.

A small, but noteworthy step was defining any necessary hyperparameters that pertain to a specific model. Some model types required additional parameters to control how the algorithm trained that particular model. The first hyperparameter is k for a KNN model, which was selected as 30 for this experiment. Another hyperparameter is the depth control for DT models, which helps to prevent overfitting. A minimum number of observations per split was used, and was set to 50. The last hyperparameter accounted for was the number of trees within the RF model, which was chosen to be 100.

Analyzing the input features to visualize their correlation was the final data preprocessing step. The relationship between the features can be assessed by plotting in a pairplot, as shown in Figure 3.2. With observation having 60 input features, the pairplot was limited to a 6×6 matrix, enough to capture the relevant relationships. This plot considers the corresponding TP, GP, and BUF for two different UEs of a single timestep (the first six input features). The correlation plot presents a strong positive correlation between TP and GP: an expected result given the GP is calculated using TP measurements. Although the throughput and goodput are highly correlated we decide to keep them to ensure that the model has the degree-of-freedom when learning on other datasets.

3.5 Model Performance

The full performance of each model is displayed in Table 3.5, which includes multiple measures. The results of the models show that the random forests model performs the best with an accuracy greater than 99%. Overall, it appears that the models that performed the best were non-linear models. The least accurate of any non-linear model, KNN, performed at an accuracy level of 95%. Of the two linear models, the best performance achieved was from the NBC with a prediction accuracy of 72% – a 23% drop from the worst performing non-linear model.

The best-performing models are capable of creating non-linear decision boundaries, which indicates that the datasets are non-linear. Random forest and DT models are capable of isolating decision boundaries using piecewise functions. DT-based models being the most

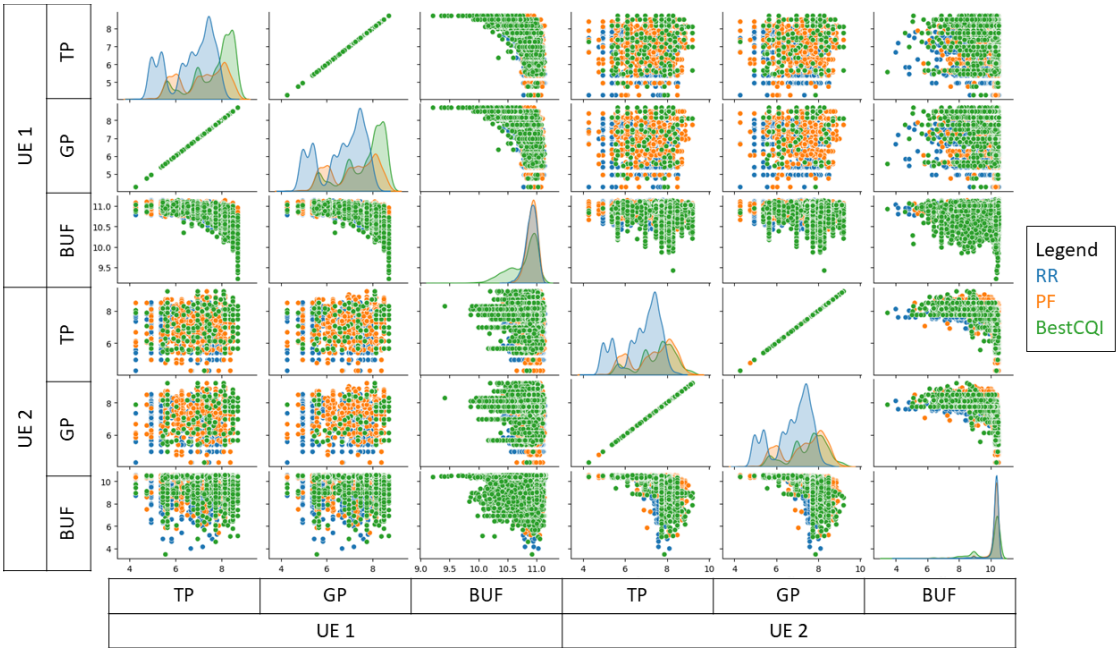

Figure 3.2: The pairplot shows the correlation between a subset of the input features with a logarithmic function applied to each feature. This full dataset includes a total of 60 input features but for the sake of brevity shows the correlation between two UE1 and UE2 for the fifth timestep.

accurate performers indicates a segmented structure in the dataset that they are able to learn well from. Possible signs of this structure in the dataset can be seen in the pairplot (Figure 3.2) and are highlighted in Figure 3.3.

To ensure that the models did not overfit, they were tested using completely new simulated datasets that were not introduced during training. In these simulation iterations, the number of UE distance sets parameter was chosen as two, however, the distances were not randomly selected. Instead, each UE was selected to have equal distance ([800, 800, 800, 800] for one set, and [1200, 1200, 1200, 1200] for the other), therefore providing the same initial CQI value. ΔCQI was chosen to be 0 and 2, allowing one instance in which the CQI remained constant and another in which the CQI changed as normal. The other simulation parameters were kept the same. These new parameters are summarized in Table 3.6. The goal of the new dataset was to test if the models generalized to different situations

Table 3.5: The performance of each ML model, measured by accuracy, precision, recall, and F1 score.

	Accuracy	**Precision**	**Recall**	**F1**
Random Forest	99.7%	99.7%	99.7%	99.7%
DT	98.9%	98.9%	98.9%	98.9%
SVM	96.4%	96.4%	96.4%	96.4%
KNN	95.0%	95.0%	95.0%	95.0%
NBC	72.1%	71.9%	72.2%	71.7%
Logistic Regression	63.0%	63.2%	63.1%	63.0%

Table 3.6: Parameters used to iterate each simulation from which new datasets were derived.

Simulation Parameter	**Selection**
UE Distance Sets	[800, 800, 800, 800] [1200, 1200, 1200, 1200]
CQI Update Periodicity	0.1, 0.2 seconds
ΔCQI	0, 2
Scheduling Algorithm	RR, BQI, PF

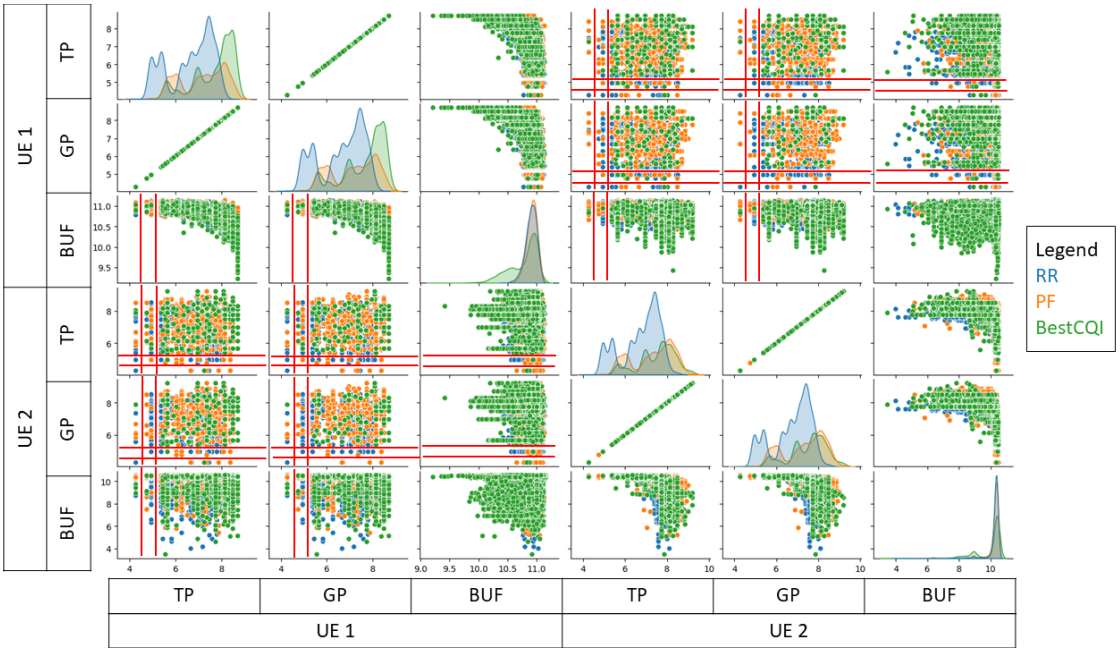

Figure 3.3: The pairplot highlights several noticable indications of a piecewise structure in the dataset, marked by red boundaries. It is likely that this entirety of the dataset follows a similar pattern, leading to high performance in the DT-based models.

that were not captured in the training data. The performance of each model after testing on this new data is shown in Table 3.7.

Both the random forest and DT models maintained a high accuracy. This increases confidence that the model does not overfit the training data, and generalized to new situations – potentially ones relevant to real-world 5G networks. SVM experiences the largest decrease in accuracy of the non-linear models. The linear models performed with a near 50% accuracy, demonstrating that they are not good for classifying the scheduling algorithm and do not generalize well to new data.

3.6 Utilizing Spectrogram Data

Although the ML models were successful in accurately classifying the scheduling algorithm that the simulated network was using, the network traffic parameters used to train

Table 3.7: The performance of the ML models when tested on an untrained dataset.

	Accuracy	**Precision**	**Recall**	**F1**
Random Forest	99.0%	99.0%	99.0%	99.0%
DT	98.2%	98.2%	98.3%	98.2%
SVM	80.1%	80.1%	86.5%	79.1%
KNN	89.3%	89.3%	91.5%	89.5%
NBC	55.2%	55.2%	72.5%	55.2%
Logistic Regression	46.9%	46.9%	62.8%	45.2%

the models, TP, GP, and BUF, are not realizable in practice. Expanding to real-world 5G networks will require data that can be passively collected. TP, GP, and BUF for this experiment were successfully calculated as a result of having full knowledge of each node on the network – the base station and every UE. However, complete knowledge of a network is not attainable in most instances for practical applications. The use of spectrogram data addresses these challenges. A spectrogram can capture network activity by collecting the power of signals communicated by network's devices.

The 5G network simulation was altered to include physical layer integration, a modified version of the NR Cell Performance Evaluation with Physical Layer Integration example from MATLAB's 5G Toolbox^{TM} [51]. These modifications allowed spectrogram data for each UE to be collected from the simulation. Each spectrogram contained an energy measure plotted in a frequency-time grid, where the resolution in frequency was one resource element (RE), and the resolution in time was one symbol.

With the modified simulation, a new set of input parameters was used. This new set of parameters excluded CQI update periodicity and ΔCQI, since CQI was a method of modeling channel conditions at the MAC layer – with a PHY layer implementation, CQI is no longer necessary. Instead, channel conditions are determined by a UE's distance to the base station, and a dynamic environment was modeled using a random seed. The random seed was used in the place of the CQI parameters instead. The new set of parameters consisted of UE distance sets, random seed, and scheduling algorithm. Each of these

parameters and their selection is summarized in Table 3.8 The new dataset consisted of

Table 3.8: Parameters used to iterate each simulation from which spectrogram data was derived.

Simulation Parameter	**Selection**
UE Distance Sets	[500, 1000, 800, 200, 800]
Random Seed	63, 68, 73, 78, 83, 88, 93, 98
Scheduling Algorithm	RR, BQI, PF

24 spectrograms for each UE, which was increased to five, thus yielding a total of 120 spectrograms.

Similar to the data formatting procedure of the previous dataset in defining how many timesteps constituted an observation, it was important to define how many symbols per observation. This was set to four symbols, where a symbol included energy measures for all REs at a particular index in time.

Each model from the previous experiments was trained on this new dataset, given that the data is different and it was unclear if the top-performers would remain for the subsequent experiments. As suspected, the top performing models from the spectrogram data versus the network metrics were not the same. The results for the top two models are outlined in Table 3.9. The continued success of the models caused for further analysis. One

Table 3.9: The performance of the top performing ML models using spectrogram data.

	Accuracy	**Precision**	**Recall**	**F1**
Logistic Regression	87.3%	87.1%	87.3%	87.1%
NBC	73.0%	73.1%	73.2%	72.6%

of the challenges of the spectrogram data used is the amount used to train the models. The resolution of the spectrogram data was reduced to lower the size of the dataset, while still capturing activity over the same number of REs and symbols. Several additional datasets were produced from the original spectrogram dataset, by averaging over m REs and n symbols. A breakdown of the chosen number of m symbols by n REs to average over is

shown in Table 3.10. The results of each of these modified datasets for the models listed in

Table 3.10: The number of m symbols and n REs averaged across to generate lower resolution datasets.

Number of m symbols and n REs averaged across displayed as $m \times n$
2×2
2×4
7×6
14×12
7×12
14×6

Table 3.9 is summarized in Table 3.11. The results show generally as the granularity of the

Table 3.11: The accuracy of the ML models when tested on an untrained dataset.

Model Accuracy							
	Spectrogram Data Averaged Over m Symbols × n REs						
Model Type	**1×1**	**2×2**	**2×4**	**7×6**	**14×12**	**7×12**	**14×6**
Logistic Regression	87.3%	85.3%	81.9%	83.5%	80.7%	86.7%	81.0%
NBC	73.0%	72.6%	64.6%	53.8%	42.2%	54.0%	43.3%

spectrogram data decreases, the model accuracy decreases. This is especially noticable for in NBC. For NBC, its highest accuracy reaches 73% coming from resolution of 1 symbol and 1 RE. However, its lowest accuracy is as low as 42% from resolution of 14 symbols and 12 REs. The file size for the 1×1 dataset was the largest at 2.6 Gigabytes. The file size for the 14×12 dataset was the smallest at only 14.6 Megabytes.

3.7 Chapter Summary

The goal of this experiment was to use several ML models to classify one of three scheduling algorithms (RR, BCQI, or PF) that a simulated network is operating. First,

this was achieved using time-series TP, GP, and BUF data for each UE. To generate plenty of data, several simulation parameters (UE distance sets, CQI update periodcity, ΔCQI, and scheduling algorithm) were used to generate a complete dataset. It was observed that non-linear models performed significantly better than linear models, indicating that the dataset used may be non-linear. Some of the models classified with questionably high accuracy, suggesting that the they may not have been generalizing well. New datasets were generated by altering the simulation parameters again, and testing the models on this un-trained-on data. Some of the models took a significant hit in accuracy, however others performed at nearly the same rate. The best performers were the DT-based models (DT and random forest).

To use data that is more applicable to real-world networks, the type of data in the dataset was changed from MAC layer network metrics to PHY layer spectrograms captures from each UE on the network. The spectrogram data included an energy measure at each RE for every symbol for the duration of the simulation. The results demonstrated success for some of the models' ability to classify the scheduling algorithm. To lessen the size of the data and retain the amount of data represented, the resolution of the spectrogram data was reduced by averaging over subsets the data in both the frequency and time dimenions. The data was reduced in the frequency dimension by averaging over subsets of REs, and in the time dimension by averaging over subsets of symbols. The results for several of these modified datasets were observed and demonstrated a decrease in performance given a lower resolution.

In summary, several models are capable of classifying 5G scheduling behavior based on the time-series network metrics for each UE that the simulation produced, as well as simulated spectrogram data for each UE. It is unclear whether the models will generalize to real world datasets, but this study gives confidence that a model can be trained to classify rudimentary scheduling algorithms, and could potentially be expanded to classify customized 5G scheduling algorithms.

Chapter 4

Waveform Optimization for Monostatic Radar Systems

This thesis focuses on establishing the framework to enable future research for cognitive radar systems. Cognitive radar can be defined as a radar system capable of learning and adapting to achieve improved performance [52]. The application this thesis will discuss involves intelligent selection of waveforms, specifically an optimal NLFM waveform. That is, which NLFM waveform will produce the highest performance possible in a given environment.

A simulation tool was built to set the framework of NLFM waveform optimization for radar systems. The simulation was designed with the focus of adaptability for future expansion to capture various applications via different radar characteristic and differently modeled environments. The simulation consists of a simple program to model the theoretical behavior of a monostatic radar.

4.1 Monostatic Radar Simulation

At its core, the simulation is comprised of two functions that generate the transmit pulse train and simulate the return signal. This simulation makes practical use of methods implemented in [53]. These functions lay the initial groundwork for a basic monostatic

radar implementation. Several parameters, such as pulse width and pulse bandwidth, described in detail in Chapter 2, are used to define the characteristics of the radar system. These are combined with other related elements as the input parameters to the transmit and return functions. A full list of the each function's parameters are specified in Table 4.1.

Transmit Train Function	**Return Pulse Function**
Repetitions	Sampling Rate (Hz)
Sampling Rate	Transmit Train
PRF (Hz)	Targets
Pulse Width, τ (s)	Carrier Frequency (f_c) (Hz)
Pulse Bandwidth, β (Hz)	System Loss (dB)
Envelope Type	Transmit Gain, G_t (dB)
Pulse Type	Receive Gain, G_r (dB)
	SNR (dB)
	Coefficients

Some of these parameters have been previously explained, including PRF, pulse width, and pulse bandwidth. The remainder of the parameters and their effect on the system will be discussed in this chapter. Repetitions indicates how many pulses the radar transmits. Sampling rate simply refers to the number of points per second the signal is constructed using, or the number of points per second the signals is measured. Envelope type is represented in Equation (2.17) and Equation (2.18) by $a(t)$, and controls the amplitude modulation of the pulse [43]; rectangular and Gaussian envelopes are two of the most popular types. Pulse type simply indicates the modulation scheme used for the pulse, and includes a selection between increasing LFM, decreasing LFM, and NLFM. As for the return pulse function, transmit trains comes from the output of the pulse generation function. Targets is defined by a struct in which several instances can be defined. Each includes distance and attenuation values. System loss, transmit gain, and receive gain are all used to determine the total attenuation. SNR helps define the noise in the system. Coefficients will be explained later. Table 4.1 summarizes the values for some of these parameters that are used in any example result generated from the simulation in this chapter.

Table 4.1: A summary of the default value used for several parameters that are used to produce and later shown example result from the radar simulation.

Parameter	**Value Used for Example Results**
Repetitions	128
Sampling Rate	100 MHz.
PRF	100 Hz.
Pulse Width	1.5 ms
Pulse Bandwidth	1 MHz.
Envelope Type	Rectangular
Targets	1 Target at 560094 m
Carrier Frequency	900 MHz.
System Loss	6 dB
Transmit Gain	40 dB
Receive Gain	45 dB
SNR	100 dB

The pulse train generation begins by defining a single pulse and repeating based on the number of repetitions. Producing a single pulse starts with defining time domain samples using the provided sampling rate and PRF. The pulse is then shaped using the given envelope type, which consists of a selection of two types: rectangular or Gaussian. Each envelope type can be expressed with simple expressions, such as a rectangular envelope type [43]:

$$: a(t) = \begin{cases} 1 & 0 \leq t \leq \tau \\ 0 & otherwise \end{cases} \tag{4.1}$$

where $a(t)$ is the envelope, t is time, and τ is the pulse width [43]. Alternatively, a Gaussian envelope type can be defined by [43]:

$$a(t) = e^{-t^2/\tau^2} \quad t \geq 0. \tag{4.2}$$

The pulse is the modulated depending on the pulse type specified. The selection consists of three types – increasing LFM, decreasing LFM, and NLFM. Similar to the envelope type, each can be defined using equations. The increasing LFM has been previous defined in Equation (2.18) and a single pulse of an increasing LFM produced from the simulation can be seen in Figure 4.1. The decreasing LFM pulse type is defined as (4.3):

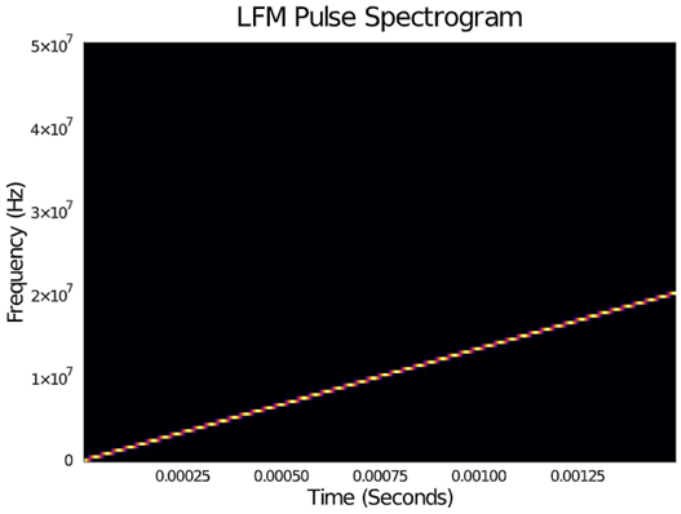

Figure 4.1: An example increasing LFM pulse generated from the monostatic radar simulation.

$$\tilde{x}(t) = a(t)e^{-j\pi\beta/\tau(t^2-2\tau t)}, \tag{4.3}$$

where $\tilde{x}(t)$ is the resulting modulated pulse, $a(t)$ is the envelope, β is the pulse bandwidth, τ is the pulse width, and t is time [43].

The NLFM definition is drawn from the increasing LFM expression specified in Equation (2.18). A simplified version can be expressed as:

$$\tilde{x}(t) = a(t)e^{j\pi ft}. \tag{4.4}$$

Here, the instantaneous frequency is condensed to a single term represented by f. Therefore, the instantaneous frequency can be expressed as:

$$f(t) = \frac{\beta}{\tau}t, \tag{4.5}$$

which is a linear function of time, where the slope equals $\beta\tau$ [43]. To produce a non-linear frequency modulated waveform from Equation (4.4), the frequency specified in Equation (4.5) can be defined as a non-linear function. That is, the frequency can be defined as a non-linear function, such as a polynomial [54] of t, resulting in Equation (4.5) producing a

NLFM waveform. This non-linear function to define the frequency can be expressed as a polynomial:

$$f(t) = a_1 t + a_2 t^2 + a_3 t^3 + \cdots + a_n t^n, \tag{4.6}$$

where $f(t)$ is the frequency with respect to time t to define the NLFM waveform and $a = [a_1, a_2, a_3, \ldots, a_n]$ are arbitrarily defined coefficients that control the shape of the frequency function. For the simulation, the frequency is defined by Equation (4.7):

$$f(t) = (\beta/\tau)^{\frac{1+1}{2}} a_1 t + (\beta/\tau)^{\frac{2+1}{2}} a_2 t^2 + (\beta/\tau)^{\frac{3+1}{2}} a_3 t^3 + \cdots + (\beta/\tau)^{\frac{n+1}{2}} a_n t^n, \tag{4.7}$$

and is controlled by an array input to the transmit function a. The variable a is of arbitrary size, where the number of elements dictates the order polynomial. An example NLFM pulse produced by the simulation is shown in Figure 4.2. The source program

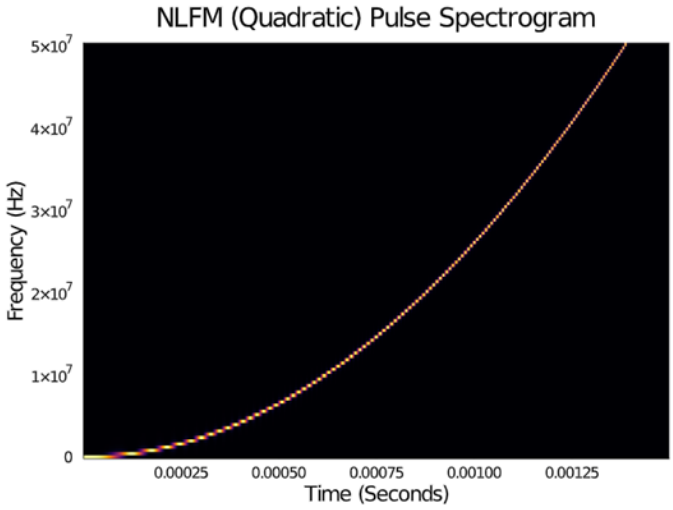

Figure 4.2: An example increasing NLFM pulse generated from the monostatic radar simulation. In the simulation, a is set to $[0, 1]$ to create a quadratic pulse. As a result, the pulse takes the shape of a parabola of frequency $f(t) = t^2$.

that was used to produce this waveform is shown in Figure 4.3. Since the goal of the simulation is to set up NLFM waveform optimization, a can act as the parameters of an optimization algorithm. Stated mathematically, recall Equation (2.9), which explains how optimization works. Vector θ represents the parameters of an optimization algorithm – $a = [a_1, a_2, a_3, \ldots, a_n]$ from Equation (4.7) are designed to represent θ in the simulation.

```
 elseif pulse_type == "NOrderPoly"
 for i in 1:length(coefficients)
     coefficients[i] = coefficients[i]*(β/τ)^((i+1)/2);
 end
pwr = Array((1:length(coefficients)))';
f = (t.^pwr).*coefficients';
f = vec(sum(f, dims=2));
pulse = a.*exp.(im*π.*f.*t);
end
```

Figure 4.3: The source code used to produce NLFM waveforms in the radar simulation.

The resulting output of the pulse train generation function is an array of in-phase and quadrature (I/Q) samples representing the data for the transmit pulse based on the parameters input to the function. The return pulse function operates by modeling effects on the transmitted signal, such as pathloss, attenuation, timing offset, and noise using several function inputs including carrier frequency, system loss, transmit/receive gain, range of the target(s), and SNR. A functional monostatic radar simulation is achieved as a result of the defined transmit and receive functions with the variety of inputs that each uses.

4.2 Defined Cost Functions

To define what an optimization algorithm will optimize in the radar simulation, two cost functions were defined:

1. Range Accuracy: How close the radar's estimated target range is to the target's true distance.

2. Sidelobe Height: The ratio between the pulse compressed return signal's main lobe peak and highest sidelobe peak.

Range accuracy and sidelobe height are two metrics that describe the performance of a radar system [55]. Here, the details of each cost function and how they are calculated are described.

Range accuracy is defined as the difference in meters between the true target range and what the simulation predicts as the range to a target. Recall that a target's range can

be measured using Equation (2.12). In the simulation, t is determined using the defined sampling rate and the number of samples to where the highest peak of the pulse compressed return signal occurs. An example pulse compressed return signal using an increasing LFM generated from the simulation is shown in Figure 4.4 for visual purposes. This plot is what

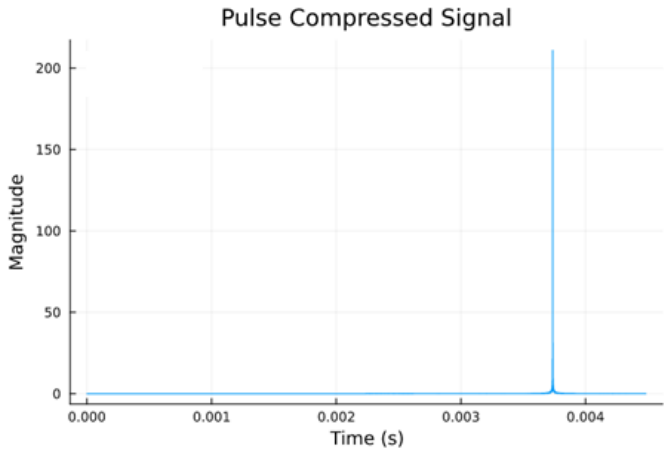

Figure 4.4: An example return signal from the simulation after pulse compression. The time of the peak is used to determine the simulated radar's predicted range to a target.

the following source code in Figure 4.5 uses to determine a target's range, and hence the range accuracy.

```
sigPower = abs.(pulse_compression[begin:first_return_samples_num]);
sigPowerdB = 10*log10.(real.(pulse_compression[begin:first_return_samples_num]).^2
.+ imag.(pulse_compression[begin:first_return_samples_num]).^2);
null, idx = findmax(sigPower);
estimatedRange  = (t[idx]*c)/2;
rangeError = abs(estimatedRange - target_dist);
```

Figure 4.5: The source code used to calculate the range accuracy in the radar simulation.

In radar systems, sidelobe height refers to energy level received by an antenna in directions other than the main lobe direction. The main lobe is the direction the antenna receives the strongest signal and typically corresponds to the target. Sidelobes are unwanted receives that occur at angles away from the main lobe. Achieving low sidelobe height is important because if sidelobes are too high the radar may detect false targets [56]. Sidelobe

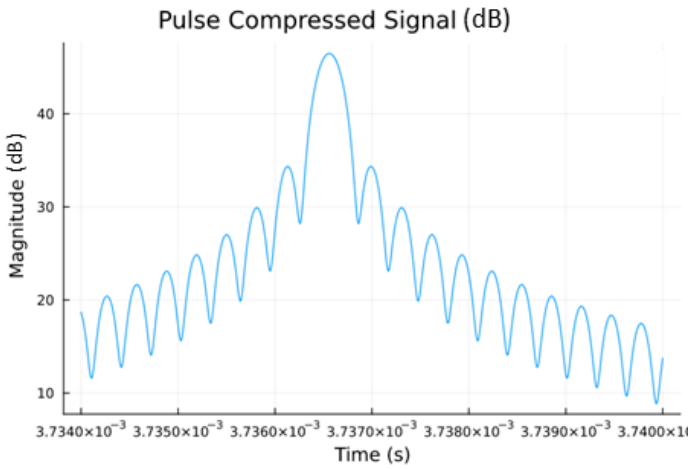

Figure 4.6: An example return signal from the simulation after pulse compression (close up). The difference between the main lobe peak and the peak of the first sidelobe is used to determine the simulated radar's sibelobe height.

height is defined as the difference in decibels between the main lobe peak of the pulse compressed return signal and the highest peak of the first sidelobe. Figure 4.6 provides a close look at an example pulse compressed return signal, where the main lobe and its sidelobes are visible. Here, the main lobe corresponds to a target and the difference between its peak and the peak of the first sidelobe is approximately 13 dB. Figure 4.7 shows the code used to determine the sidelobe height for a return signal.

```
peaks = argmaxima(sigPower);
null, sigPowerMaxPeakIdx = findmax(sigPower);
peaksPeakIdx = first(findall(x->x==sigPowerMaxPeakIdx, peaks));
sigPowerSidelobeIdx = peaks[peaksPeakIdx+1];
sidelobePerformance = sigPowerdB[sigPowerMaxPeakIdx] -
sigPowerdB[sigPowerSidelobeIdx];
```

Figure 4.7: The source code used to calculate the sidelobe height in the radar simulation.

4.3 Chapter Summary

The goal of this experiment was to set up the framework for future algorithms to optimize the coefficients of an NLFM waveform using several cost functions. A functional monostatic radar simulation was built to provide the tools needed for such experimentation. The basic operation of the radar uses two functions which transmit and receive, respectively. A number of parameters for each function are used to alter the behavior of the system, such as pulse width, pulse bandwidth, envelope type, pulse type, and others. The simulation was designed to allow for easy expansion in modeling interference, noise, new waveforms, etc. The pulse type parameter for the transmit function defines whether LFM or NLFM is used. The frequency for the NLFM option is defined by an array corresponding to the coefficients of a polynomial. Additionally, two functions are defined to evaluate the simulated radar's effectiveness. These functions include range accuracy and sidelobe height, which calculate different metrics that describe the radar's performance.

Next steps would include employing an optimization algorithms, such as gradient descent on the radar simulation. Here, the NLFM coefficients can be used as parameters of the optimization. Doing so will help identify the optimal NLFM waveform for a given environment. Moreover, the cost functions can be used as what the algorithm is optimizing for. Additional cost functions can be defined to capture more elements of the radar's performance. These functions could even be combined into a weighted value to describe all elements with one measure. Modeling the effects of interference and jammers that mimic different environments to test the optimization will help prepare for real-world experiments.

Chapter 5

Conclusions

5.1 Research Outcomes

This thesis detailed ML applied to wireless technologies in two different contexts. These contexts included predicting network traffic for 5G cellular networks, and building a simulation to enable future cognitive radar experiments.

The first example demonstrated a novel method for helping prediction of 5G network traffic using ML. Specifically, classification ML models were used to classify fundamental scheduling algorithms used by a simulated 5G network as a method of analyzing a network's scheduling behaviors. This was done first using UE performance metrics (TP, GP, and BUF for each UE) and the top two models achieved accuracies over 98%. Next, spectrogram data was used to test if the models could repeat the success with realizable data that is easier to collect from real networks. The top two models achieved accuracies over 70%. Next, the resolution of spectrogram data was altered by averaging power levels over several numbers of symbols in time and resource elements in frequency. This was done to reduce the size of the data, which would make it easier to process and transport if a similar set-up were to be used to analyze scheduling behaviors in real-time for a real-world 5G network. The results demonstrated that as the granularity of the data decreases, the accuracy decreases. The highest resolution data used resulted in the highest accuracy, and the lowest resolution used gave the lowest accuracy.

The second example establishes the framework for adaptive radar waveform experimentation in support of advancing cognitive radar systems. In particular, a simulation, which interfaces with the methods implemented in [53], is controlled by several pulse-defining parameters that enable optimization of frequency-controlled NLFM waveforms. The frequency of the NLFM waveform is defined by an array of arbitrary length consisting of coefficients. Each coefficient corresponds to a polynomial term. These coefficients can be used as the parameters of an optimization algorithm in the simulation. Additionally, the simulation measures the performance of the radar using two cost functions: range accuracy and sidelobe height. The range accuracy determines how close the radar's predicted target range is, and the sidleobe height measures the difference between the pulse compressed return signal's main lobe and its highest sidelobe. These functions can be used as the cost for an algorithm to optimize.

5.2 Future Work

Regarding the 5G classification, there remains room for future improvement. First, the experiment should be expanded to an emulated network rather than a simulation. This would involve a testbed with physical hardware, for example, several software-defined radio running OpenAirInterface. This would provide more realistic results that are closer to those observed in real-world 5G networks, and provide insights for the performance of the models using real data. Second, it would also be interesting to classify non-rudimentary scheduling algorithms. Currently, the models only identify three fundamental scheduling types. However, service providers are able to customize their own scheduling algorithms and therefore real networks often operate using proprietary scheduling types. This would likely involve using unsupervised ML methods, as the scheduling types would be unknown. The results from this experiment gives confidence in a possible 5G sensor network in which several distributed sensing devices collect spectrogram data. This data would then be processed to identify areas of opportunity for secondary networks to operate in a primary network's licensed spectrum [53]

There also remains clear next steps with respect to the radar simulation. For instance,

improvement in the modeling of interference would be useful. This could be natural interference as a result of a particular environment the simulated radar is operating in, or an active adversary jamming the radar. Another future step would be applying an algorithm to the simulation to optimize the operating waveform. This would likely involve an optimization algorithm, such as gradient descent (stochastic, batch, etc.), to use the NLFM coefficients as parameters to optimize according to the defined cost functions, range accuracy and sidelobe height. Furthermore, additional cost functions can be defined to capture other important elements of the radar's performance. Combining these costs together by weighting them to form a single metric could serve additional benefits as well.

Each experiment exemplifies the use of ML in different wireless domains. Applying cognitive tools to wireless technologies will become increasingly more important as the relevance of ML continues to increase.

Bibliography

[1] L. Li, H. Xiaoguang, C. Ke, and H. Ketai, "The applications of wifi-based wireless sensor network in internet of things and smart grid," in *2011 6th IEEE Conference on Industrial Electronics and Applications*, 2011, pp. 789–793.

[2] L. Kong, M. K. Khan, F. Wu, G. Chen, and P. Zeng, "Millimeter-wave wireless communications for iot-cloud supported autonomous vehicles: Overview, design, and challenges," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 62–68, 2017.

[3] S. Bartoletti, A. Conti, A. Giorgetti, and M. Z. Win, "Sensor radar networks for indoor tracking," *IEEE Wireless Communications Letters*, vol. 3, no. 2, pp. 157–160, 2014.

[4] C. W. Chen, R. Lagenduk, A. Reibman, and W. Zhu, "Introduction to the special issue on wireless communication," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 357–359, 2002.

[5] F. Dowla, *Handbook of RF and wireless technologies*. Elsevier, 2003.

[6] M. Machedon-Pisu, "The impact of propagation media and radio interference on the performance of wireless sensor networks with micaz motes," in *2014 International Conference on Optimization of Electrical and Electronic Equipment (OPTIM)*, 2014, pp. 866–872.

[7] A. F. Molisch, *Wireless communications*. John Wiley & Sons, 2012.

[8] Y. Arjoune, F. Salahdine, M. S. Islam, E. Ghribi, and N. Kaabouch, "A novel jamming attacks detection approach based on machine learning for wireless communication," in

2020 International Conference on Information Networking (ICOIN), 2020, pp. 459–464.

[9] S. Hu, X. Chen, W. Ni, E. Hossain, and X. Wang, "Distributed machine learning for wireless communication networks: Techniques, architectures, and applications," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1458–1493, 2021.

[10] F.-L. Luo, *Machine learning for future wireless communications*. John Wiley & Sons, 2020.

[11] A. Jagannath, J. Jagannath, and T. Melodia, "Redefining wireless communication for 6g: Signal processing meets deep learning with deep unfolding," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 6, pp. 528–536, 2021.

[12] S. Bi, R. Zhang, Z. Ding, and S. Cui, "Wireless communications in the era of big data," *IEEE Communications Magazine*, vol. 53, no. 10, pp. 190–199, 2015.

[13] M. Wasilewska, H. Bogucka, and A. Kliks, "Spectrum sensing and prediction for 5g radio," in *Big Data Technologies and Applications*. Springer, 2020, pp. 176–194.

[14] R. Li, Z. Zhao, J. Zheng, C. Mei, Y. Cai, and H. Zhang, "The learning and prediction of application-level traffic data in cellular networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3899–3912, 2017.

[15] S. Irina, S. Irina, and M. Anastasiya, "Forecasting 5g network multimedia traffic characteristics," in *2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*. IEEE, 2020, pp. 982–987.

[16] L.-V. Le, D. Sinh, B.-S. P. Lin, and L.-P. Tung, "Applying big data, machine learning, and SDN/NFV to 5G traffic clustering, forecasting, and management," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 2018, pp. 168–176.

[17] Z. Gao, "5g traffic prediction based on deep learning," *Computational Intelligence and Neuroscience*, vol. 2022, 2022.

[18] D. Roy, T. Mukherjee, M. Chatterjee, and E. Pasiliao, "Primary user activity prediction in dsa networks using recurrent structures," in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*. IEEE, 2019, pp. 1–10.

[19] L. Kang, J. Bo, L. Hongwei, and L. Siyuan, "Reinforcement learning based anti-jamming frequency hopping strategies design for cognitive radar," in *2018 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*. IEEE, 2018, pp. 1–5.

[20] Q. Yan, B. Lin, and K. Wang, "Research on adaptive digital beam forming technology," in *2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*. IEEE, 2018, pp. 446–450.

[21] L. Ming and Z. Jingfeng, "Study on anti-jamming frequency selection in radar netting," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*. IEEE, 2016, pp. 1781–1784.

[22] E. Alpaydin, *Machine learning*. Mit Press, 2021.

[23] G. Langs, S. Röhrich, J. Hofmanninger, F. Prayer, J. Pan, C. Herold, and H. Prosch, "Machine learning: from radiomics to discovery and routine," *Der Radiologe*, vol. 58, pp. 1–6, 2018.

[24] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[25] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4, no. 4.

[26] Z. Lin, H. Li, and C. Fang, "Accelerated optimization for machine learning," *Nature Singapore: Springer*, 2020.

[27] A. Mustapha, L. Mohamed, and K. Ali, "An overview of gradient descent algorithm optimization in machine learning: Application in the ophthalmology field," in *Smart Applications and Data Analysis: Third International Conference, SADASC 2020, Marrakesh, Morocco, June 25–26, 2020, Proceedings 3*. Springer, 2020, pp. 349–359.

[28] G. Berardinelli, K. Pajukoski, E. Lahetkangas, R. Wichman, O. Tirkkonen, and P. Mogensen, "On the potential of OFDM enhancements as 5G waveforms," in *2014 IEEE 79th Vehicular Technology Conference (VTC Spring)*. IEEE, 2014, pp. 1–5.

[29] H. Holma and A. Toskala, *LTE for UMTS - OFDMA and SC-FDMA Based Radio Access*. Wiley, 2009.

[30] *5G; Study on new radio access technology*, ETSI 3GPP, 10 2017, version 14.1.0 Release 14.

[31] S. R. Khosravirad, G. Berardinelli, K. I. Pedersen, and F. Frederiksen, "Enhanced harq design for 5g wide area technology," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*. IEEE, 2016, pp. 1–5.

[32] A. Gupta and R. K. Jha, "A survey of 5G network: Architecture and emerging technologies," *IEEE Access*, vol. 3, pp. 1206–1232, 2015.

[33] A. Mamane, M. Fattah, M. El Ghazi, M. El Bekkali, Y. Balboul, and S. Mazer, "Scheduling algorithms for 5g networks and beyond: Classification and survey," *IEEE Access*, 2022.

[34] P. Bhagwat, P. Bhattacharya, A. Krishna, and S. K. Tripathi, "Enhancing throughput over wireless lans using channel state dependent packet scheduling," in *Proceedings of IEEE INFOCOM'96. Conference on Computer Communications*, vol. 3. IEEE, 1996, pp. 1133–1140.

[35] S. Schwarz, C. Mehlführer, and M. Rupp, "Low complexity approximate maximum throughput scheduling for lte," in *2010 Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers*. IEEE, 2010, pp. 1563–1569.

[36] H. Kim and Y. Han, "A proportional fair scheduling for multicarrier transmission systems," *IEEE Communications letters*, vol. 9, no. 3, pp. 210–212, 2005.

[37] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research society*, vol. 49, no. 3, pp. 237–252, 1998.

[38] N. W. Service, "Radar," Silver Spring, Maryland, United States. [Online]. Available: https://www.weather.gov/about/radar

[39] M. A. Richards, J. Scheer, W. A. Holm, and W. L. Melvin, *Principles of modern radar*. SciTech Publishing, Inc., 2010, vol. 1.

[40] M. Skolnik, "An introduction and overview of radar," *Radar Handbook*, vol. 3, pp. 1–1, 2008.

[41] B. R. Mahafza, *Introduction to radar analysis*. CRC press, 2017.

[42] S. D. Blunt, J. Jakabosky, P. McCormick, P. S. Tan, and J. G. Metcalf, "Chapter 1 - holistic radar waveform diversity," in *Academic Press Library in Signal Processing, Volume 7*, R. Chellappa and S. Theodoridis, Eds. Academic Press, 2018, pp. 3–50. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128118870000018

[43] The MathWorks, Inc., "Linear frequency modulated pulse waveforms," Natick, Massachusetts, United States. [Online]. Available: https://www.mathworks.com/help/phased/ug/linear-frequency-modulated-pulse-waveforms.html

[44] S. Parwana and S. Kumar, "Analysis of LFM and NLFM radar waveforms and their performance analysis," *International Research Journal of Engineering and Technology*, vol. 2, pp. 334–339, 2015.

[45] D.-H. Kim, H.-J. Kim, and J.-H. Lim, "Design of optimized coded lfm waveform for spectrum shared radar system," *Sensors*, vol. 21, no. 17, p. 5796, 2021.

[46] The MathWorks, Inc., "Chirp," Natick, Massachusetts, United States. [Online]. Available: https://www.mathworks.com/help/signal/ref/chirp.html

[47] A. W. Doerry, "Generating precision nonlinear fm chirp waveforms," in *Radar Sensor Technology XI*, vol. 6547. SPIE, 2007, pp. 121–132.

[48] S. Liu, Y. Jia, Y. Liu, and X. Zhang, "Research on ultra-wideband nlfm waveform synthesis and grating lobe suppression," *Sensors*, vol. 22, no. 24, p. 9829, 2022.

[49] F. Tang, Y. Zhou, and N. Kato, "Deep reinforcement learning for dynamic uplink/downlink resource allocation in high mobility 5G HetNet," *IEEE Journal on selected areas in communications*, vol. 38, no. 12, pp. 2773–2782, 2020.

[50] The MathWorks, Inc., "Nr fdd scheduling performance evaluation," Natick, Massachusetts, United States. [Online]. Available: https://www.mathworks.com/help/5g/ug/nr-fdd-scheduling-performance-evaluation.html

[51] ——, "Nr cell performance evaluation with physical layer integration," Natick, Massachusetts, United States. [Online]. Available: https://www.mathworks.com/help/5g/ug/nr-cell-performance-evaluation-with-physical-layer-integration.html

[52] J. R. Guerci, R. Guerci, M. Ranagaswamy, J. Bergin, and M. Wicks, "Cofar: Cognitive fully adaptive radar," in *2014 IEEE Radar Conference*. IEEE, 2014, pp. 0984–0989.

[53] M. Jacobs, "Next generation waveform generation and commercial spectrum utilization." Worcester Polytechnic Institute, 2023.

[54] Z. Xu, X. Wang, and Y. Wang, "Nonlinear frequency-modulated waveforms modeling and optimization for radar applications," *Mathematics*, vol. 10, no. 21, p. 3939, 2022.

[55] A. Thakur and D. S. Saini, "Bandwidth optimization and side-lobe levels reduction in pc radar using legendre orthogonal polynomials," *Digital Signal Processing*, vol. 101, p. 102705, 2020.

[56] R. Fry and D. Gray, "Clean deconvolution for sidelobe suppression in random noise radar," in *2008 International Conference on Radar*. IEEE, 2008, pp. 209–212.