Identification of Flying Drones in Mobile Networks using Machine Learning

Identifiering av flygande drönare i mobila nätverk med hjälp av maskininlärning

Elias Alesand

Supervisor : Giovanni Interdonato
Examiner : Danyo Danev

External supervisors : Sakib Bin Redhwan, Roman Zhohov
Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera åktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida

http://www.ep.liu.se/

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years starting from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page:

http://www.ep.liu.se/

© Elias Alesand
Abstract

Drone usage is increasing, both in recreational use and in the industry. With it comes a number of problems to tackle. Primarily, there are certain areas in which flying drones pose a security threat, e.g., around airports or other no-fly zones. Other problems can appear when there are drones in mobile networks which can cause interference. Such interference comes from the fact that radio transmissions emitted from drones can travel more freely than those from regular UEs (User Equipment) on the ground since there are few obstructions in the air. Additionally, the data traffic sent from drones is often high volume in the form of video streams. The goal of this thesis is to identify so-called "rogue drones" connected to an LTE network. Rogue drones are flying drones that appear to be regular UEs in the network. Drone identification is a binary classification problem where UEs in a network are classified as either a drone or a regular UE and this thesis proposes machine learning methods that can be used to solve it. Classifications are based on radio measurements and statistics reported by UEs in the network. The data for the work in this thesis is gathered through simulations of a heterogenous LTE network in an urban scenario.

The primary idea of this thesis is to use a type of cascading classifier, meaning that classifications are made in a series of stages with increasingly complex models where only a subset of examples are passed forward to subsequent stages. The motivation for such a structure is to minimize the computational requirements at the entity making the classifications while still being complex enough to achieve high accuracy. The models explored in this thesis are two-stage cascading classifiers using decision trees and ensemble learning techniques.

It is found that close to 60% of the UEs in the dataset can be classified without errors in the first of the two stages. The rest is forwarded to a more complex model which requires more data from the UEs and can achieve up to 98% accuracy.
# Contents

Abstract ........................................ iii  
Contents ......................................... iv  
List of Figures ................................... vi  
List of Tables ..................................... vii  

1 Introduction .................................. 1  
  1.1 Motivation .................................... 2  
  1.2 Aim .......................................... 2  
  1.3 Research questions .......................... 2  
  1.4 Delimitations ................................. 3  
  1.5 Thesis outline ............................... 3  

2 Related Work .................................. 4  
  2.1 Background ................................. 4  
  2.2 Drone detection .............................. 4  
  2.3 Device identification ....................... 5  

3 Theory ........................................ 7  
  3.1 Mobility in LTE .............................. 7  
  3.2 Drones in LTE networks .................... 9  
  3.3 Drone identification ....................... 10  
  3.4 Binary classification ....................... 10  
  3.5 Feature selection ............................ 11  
  3.6 Decision trees .............................. 13  
  3.7 Ensemble learning ........................... 13  
  3.8 Cascading classifiers ....................... 14  
  3.9 Model Validation ............................ 14  

4 Method ......................................... 16  
  4.1 Data collection .............................. 16  
  4.2 Data analysis ............................... 17  
  4.3 Model construction ......................... 20  
  4.4 Model validation ............................ 23  

5 Results ........................................ 25  
  5.1 Threshold selection ......................... 25  
  5.2 Validation results ........................... 27  
  5.3 Comparison with single-stage model ......... 31  
  5.4 One model per cell type ..................... 31  
  5.5 One model per cell ........................... 33
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>LTE cell coverage</td>
<td>7</td>
</tr>
<tr>
<td>3.2</td>
<td>Visual representation of precision and recall</td>
<td>11</td>
</tr>
<tr>
<td>3.3</td>
<td>PCA example with two-dimensional data</td>
<td>12</td>
</tr>
<tr>
<td>3.4</td>
<td>Decision boundaries and corresponding binary tree in a two-dimensional feature space</td>
<td>13</td>
</tr>
<tr>
<td>3.5</td>
<td>Data split in k-fold cross-validation with $k = 4$</td>
<td>15</td>
</tr>
<tr>
<td>4.1</td>
<td>Workflow of machine learning in the networking domain</td>
<td>16</td>
</tr>
<tr>
<td>4.2</td>
<td>Best cell RSRP over time for a drone and a regular UE</td>
<td>18</td>
</tr>
<tr>
<td>4.3</td>
<td>Gini-importance of the different features</td>
<td>19</td>
</tr>
<tr>
<td>4.4</td>
<td>Model performance as a function of the number of principal components averaged over different models</td>
<td>20</td>
</tr>
<tr>
<td>4.5</td>
<td>Classification ratio as a function of max depth</td>
<td>21</td>
</tr>
<tr>
<td>4.6</td>
<td>Classification ratio as a function of the number of trees</td>
<td>22</td>
</tr>
<tr>
<td>4.7</td>
<td>Drone recall as a function of maximum tree depth</td>
<td>23</td>
</tr>
<tr>
<td>4.8</td>
<td>Drone recall as a function of the number of trees</td>
<td>23</td>
</tr>
<tr>
<td>5.1</td>
<td>Threshold selection for the three models in stage one</td>
<td>26</td>
</tr>
<tr>
<td>5.2</td>
<td>Threshold selection for the three models in the second stage and the single-stage model</td>
<td>27</td>
</tr>
<tr>
<td>5.3</td>
<td>Validation results for stage one</td>
<td>28</td>
</tr>
<tr>
<td>5.4</td>
<td>Cumulative distribution function of RSRP for classified and unclassified UEs in the stage one BST model</td>
<td>28</td>
</tr>
<tr>
<td>5.5</td>
<td>Validation results for stage two</td>
<td>29</td>
</tr>
<tr>
<td>5.6</td>
<td>Validation results for stage two over time</td>
<td>30</td>
</tr>
<tr>
<td>5.7</td>
<td>OLD model results</td>
<td>31</td>
</tr>
<tr>
<td>5.8</td>
<td>Results when combining the two stages</td>
<td>31</td>
</tr>
<tr>
<td>5.9</td>
<td>One model per cell type results for stage one</td>
<td>32</td>
</tr>
<tr>
<td>5.10</td>
<td>One model per cell type results for stage two</td>
<td>32</td>
</tr>
<tr>
<td>5.11</td>
<td>One model per cell results for stage two</td>
<td>33</td>
</tr>
</tbody>
</table>
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Mobility-related events in LTE</td>
<td>8</td>
</tr>
<tr>
<td>4.1</td>
<td>Simulation setup</td>
<td>17</td>
</tr>
<tr>
<td>4.2</td>
<td>A subset of features gathered through simulation</td>
<td>17</td>
</tr>
<tr>
<td>4.3</td>
<td>All features used as input for the machine learning models</td>
<td>19</td>
</tr>
<tr>
<td>4.4</td>
<td>Chosen parameters for each model for stage one</td>
<td>22</td>
</tr>
<tr>
<td>4.5</td>
<td>Chosen parameters for each model for stage two</td>
<td>23</td>
</tr>
<tr>
<td>5.1</td>
<td>Thresholds of the first stage</td>
<td>26</td>
</tr>
<tr>
<td>5.2</td>
<td>Threshold of the second stage</td>
<td>27</td>
</tr>
<tr>
<td>5.3</td>
<td>Validation results of the first stage</td>
<td>28</td>
</tr>
<tr>
<td>5.4</td>
<td>Validation results of the second stage</td>
<td>29</td>
</tr>
<tr>
<td>5.5</td>
<td>Average times to classify all 977 UEs</td>
<td>30</td>
</tr>
<tr>
<td>5.6</td>
<td>Results of single-stage model and two-stage model</td>
<td>31</td>
</tr>
<tr>
<td>5.7</td>
<td>One model per cell type results for stage one</td>
<td>32</td>
</tr>
<tr>
<td>5.8</td>
<td>One model per cell type results for stage two</td>
<td>32</td>
</tr>
<tr>
<td>5.9</td>
<td>One model per cell results for stage one</td>
<td>33</td>
</tr>
<tr>
<td>5.10</td>
<td>One model per cell results for stage two</td>
<td>33</td>
</tr>
</tbody>
</table>
Nomenclature

3GPP  3rd Generation Partnership Project
BST   Boosting
CNN   Convolutional Neural Network
DL    Downlink
DSL   Digital Subscriber Line
DT    Decision Tree
eNB   eNodeB
FN    False Negative
FP    False Positive
FPR   False Positive Rate
IoT   Internet Of Things
LTE   Long Term Evolution
MME   Mobility Management Entity
PCA   Principal Component Analysis
RF    Random Forest
RSRP  Reference Signals Received Power
RSRQ  Reference Signals Received Quality
RSSI  Reference Signals Strength Indicator
SDR   Software-defined Radios
SVM   Support Vector Machine
TP    True Positive
TTL   Time To Live
UE    User Equipment
UL    Uplink
An increase of flying drones has been observed during the past years [1]. However, many drones suffer from a limited range as a result of their chosen ways of communication, e.g., IEEE 802.11 (Wi-Fi™) which relies on a point-to-point connection between the user and the drone. This problem can be addressed by communicating over a mobile network, such as LTE, instead of the previously used proprietary protocols or range limited communication channels and thereby increase the range significantly. The range is increased since the controller commands is routed through the internet instead of being transmitted directly from the user to the drone. However, controlling a drone over an LTE network can cause some problems. Since most users in an LTE network are on the ground, such as cars or people with mobile devices, LTE deployment is optimized for terrestrial users by often having the eNodeB(eNB) antennas tilted towards the ground to serve terrestrial user equipment(UE) [2]. A side-effect of down-tilting the antennas is that UEs above them, such as aerial drones, often will experience worse connectivity than UEs on the ground. This causes drones to require more resources from the eNB, thus reducing the overall performance of the network.

Another problem is that the signals transmitted from aerial drones will likely move long distances through the air due to there being few obstructions, thus imposing interference on the LTE network over large areas and thereby affecting other UEs in the network [3]. A typical traffic-demanding application is video streaming. Transmitting live video stream from a drone causes heavy load in the uplink (UL), that is from the UE to the eNB. Identifying flying drones would be trivial if they identified themselves as such when connected to the network. But when this is not the case there needs to be another way to identify them without such information. In the context of this thesis, drones connected to the network appearing as a regular UE, such as a mobile device mounted on a drone, are called rogue drones. Previous research [3] has showed that the problem of detecting rogue drones can be solved using machine learning techniques with high accuracy for detecting drones above a certain altitude. However, the performance is worse for drones flying at lower altitudes, therefore there might be a need for other types of models to be able to classify drones at all altitudes which is the aim of this thesis.

1LTE: Long-Term Evolution.
2Hardware communicating with users in a LTE network.
3In this context, UEs are devices served by a mobile network, e.g., cell phones and drones.
1.1 Motivation

In this thesis, a simulated dataset of radio measurements, such as signal strength and signal quality, from UEs in an LTE network is used in order to create a machine learning model which can distinguish rogue drones from regular UEs. The input features to the model consist of radio measurements communicated between UEs and eNBs. What has to be taken into account is that detecting drones can not be done for free. Performing classifications requires computing power and depending on the model there are memory requirements. Additionally, the models might require additional data from UEs which would not be sent otherwise, which would congest the network. The approach explored in this thesis is to use a two-stage classification where the first stage uses a low complex model with minimal memory requirements, and a second stage is used to classify more ambiguous cases with possibility to be more complex and has higher memory requirements.

A number of different models are explored in this thesis. Two different ensemble learning techniques using decision trees are evaluated for the first and second stage. These are compared to a less complex model in the form of a regular decision tree classifier for both stages. Finally, the different two-stage models are compared to a one-stage model proposed by Rydén et al. [3].

1.1 Motivation

By being able to utilize existing LTE networks, drones would be able to achieve a much greater range compared to some other options (e.g., Wi-Fi™). But with the introduction of interference on the network, it poses a risk of affecting non-drone related traffic in a negative way. There are also non network management related scenarios where drones are problematic, for example disruption of airports [4] or flight in other no-drone zones [5]. By creating a classifier that correctly identifies drones, while having a low amount of false positive classifications of non-drone related entities, one can utilize this knowledge to optimize the network for both traffic patterns. This can thereby minimize the interference posed towards non-drones, while being able to optimize the conditions for the drones.

1.2 Aim

The goal of this thesis is to examine how machine learning models can be used for identification of drones in an LTE network with good accuracy and low ratio of false positives. Furthermore, the models should be constructed with computational and memory requirements in mind since they should work in real time deployed on some entity in the network. However, for difficult cases, like drones flying at low altitudes or regular UEs in tall buildings, there might be a need for a more complex model in order to make better classifications.

1.3 Research questions

In order to solve the problem presented in the previous section, this thesis will focus on answering the following questions:

1. Which radio measurements can be used to identify drones in an LTE network?
2. How can machine learning methods be utilized to identify drones in an LTE network?
3. How does classifying the data in two stages affect the computational and memory requirements?
4. Can a model be generalized to be used for multiple cells in a network while maintaining high accuracy?
To answer these questions, the aforementioned dataset from a simulated LTE network involving both terrestrial and aerial UEs is used to extract features that can be used to identify which of the UEs are drones. Machine learning models are trained on said features such that predictions can be made on new data.

1.4 **Delimitations**

The main delimitation for this thesis is that the data used to train the models will be based on simulations and not real-world data. The quality of the simulated data will thereby affect the models ability to identify drone-related traffic in a real-world scenario. The choices that are to be made after an UE is identified as a drone is out of the scope of this thesis, the focus is specifically on the identification. However, some exploration of what can be done is discussed in section 6.3.

1.5 **Thesis outline**

Chapter 2 presents related works that have been done previously, some directly related to drones but also other types of device identification. Chapter 3 contains necessary explanations related to mobility in LTE networks, drones in LTE networks, and relevant machine learning concepts. Chapter 4 describes the methodology of the work done in this thesis. Chapter 5 presents the results of the different machine learning models. Chapter 6 contains discussions around the result and methodology. Chapter 7 concludes the thesis with the most important findings of the study.
2 Related Work

This chapter presents the previous work within the field that is relevant to this thesis. Some background about where and why this thesis is written is presented. Studies about drones in LTE networks are discussed, along with the reasons why identifying drones is a relevant field of study. Examples of how to approach the problem are also described. Following that, studies about identification of other types of devices are presented in order to provide a wider perspective of what has been done in similar domains.

2.1 Background

This thesis is carried out at Ericsson Research in Linköping. Ericsson has done research regarding drones in mobile networks since it needs to understand how to address them in future systems. Future generations of mobile communication systems will have to handle a broader set of scenarios than the current mobile networks. Some examples of such scenarios are ultra-low power sensors, intelligent traffic systems and flying drones. To deal with these new scenarios, which increase the complexity of the networks, Ericsson want to deploy intelligent methods that can analyze data from the mobile network. These methods should minimize the work needed to be done by humans, thus reducing the cost of operating the network and hopefully increasing network performance and reliability.

The aim of this thesis is to extend the previous work done by Rydén et al. [3] by exploring a different approach to drone identification. In [3], the drone identification problem is approached by training logistic regression and decision tree models. The scenario consists of an urban environment with regular UEs and flying drones with an altitude ranging from 15 meters to 300 meters. One relevant finding is that the altitude the drones are flying at indicates how easy it is to correctly identify them. Drones flying above 60 meters could be identified with very high accuracy, while drones flying at 15 meters were very difficult to identify. This shows that there might be a need for other methods to accurately classify low-altitude drones which is why other types of models is used in this thesis.

2.2 Drone detection

Lin et al. [2] explain the possibilities and challenges of flying drones in LTE networks. They point out that the two main challenges of using drones in LTE networks are coverage and in-
Coverage refers to the fact that mobile networks today are optimized for terrestrial UEs and therefore eNB antennas are generally tilted downwards in order to reduce interference that is caused upon nearby cells. This can cause drones to be served by the sidelobes of the eNB antenna, meaning that they will experience less signal strength than UEs being served by the mainlobe on the ground would. The problem of interference mainly appears when drones are causing heavy traffic in the uplink from the drone to the eNB, for example by streaming video data. Other UEs in the network would be affected by this interference and it would have a negative effect on the general user experience. Consequently, it is relevant to be able to detect flying drones. In [2], it is observed that signals from flying drones can reach eNBs far away since there is less blockage in the air than on the ground. If a UE can be seen by eNBs far away the probability of it being a drone should be reasonably high.

As mentioned in section 2.1, Rydén et al. have done a study on the topic of identifying rogue drones using logistic regression and decision trees [3]. They use data from a simulation where the scenario is an urban environment with both regular UEs and flying drones. The features that are considered are the received signal strength indicator (RSSI), standard deviation of the eight strongest reference signals received powers (RSRP), difference between the strongest and second strongest RSRP, and lastly the serving cell RSRP. The motivation for the RSRP gap is similar to the observation by Lin et al. [2] that a drone at a high altitude would be seen by multiple eNBs. The evaluation of the models is made at zero false positive ratio, meaning that no ground UEs are labeled as drones in the test data. They find that decision trees performs better and can achieve 100 percent accuracy at detecting drones flying at a height above 60 meters. The accuracy gets worse the lower the drones are flying as only five percent of drones are correctly classified when flying at a height of 15 meters. They do however point out that the interference is not as severe at lower altitudes. Their research highlights the fact that detecting drones with these machine learning techniques is possible in certain scenarios, however there might be a need for other types of models to be able to classify drones flying at lower altitudes.

Another approach that has been used for drone detection is to make decisions based on sound. In [6], a support vector machine (SVM) is trained to distinguish the sounds of drones from other sound sources in outdoors environments. The different sounds that are considered are drones, birds, planes, and thunderstorms. The best performing SVM achieves 96.7% accuracy for drone detection. Such a system requires microphones to be deployed in the area of interest.

2.3 Device identification

Meidan et al. [7] present machine learning algorithms for identifying IoT devices connected to wireless networks. First, they use a binary classifier to identify if a device is an IoT device (TV, printer, smartwatch etc.) or a non-IoT device (PC, smartphone). In the second stage each IoT device is labeled as a specific IoT device class. The classifiers use HTTP packet properties and network traffic features such as average Time to Live (TTL) and ratio between incoming and outgoing bytes to predict device types. The first classifier is able to distinguish IoT devices from non-IoT devices with close to perfect performance. The second classifier is able to label IoT devices as the correct class in roughly 99 percent of cases.

A previous study, by Riyaz et al. [8], describes a way of identifying radio devices transmitting IEEE 802.11ac signals, by utilizing deep learning. Their approach for identifying the radio units is based on hardware impairments, that are apparent on the physical layer during transfers. The impairments can then be used for creating fingerprints for each of the radios. In order to conduct their tests they utilize Software-defined radios (SDR) for gath-
2.3. Device identification

ering a dataset in the form of raw RF I/Q samples. The dataset is then utilized as training data for a convolutional neural network (CNN), which then is used for identifying radios by analyzing unclassified I/Q samples. When evaluating the CNN’s performance, it is compared to a support vector machine and a logistic regression based solution. The CNN shows significant improvements over the other two implementations in terms of accuracy. The support vector machine and logistic regression based solutions both have a similar level of accuracy. They also show a higher decrease in accuracy, compared to the CNN, when the number of devices increases. The authors also observe that the accuracy of the classification decreases as the distance between the device and receiver increases.

While Riyaz et al. [8] utilized hardware impairments in order to identify devices in an 802.11 environment, other features could also be utilized for the same purpose. Radhakrishnan, Uluagac and Beyah [9] present a method that utilizes the time interval between successive packets in order to create a feature vector for the neural network. Their approach is thereby independent of the underlying transfer protocol and package content. The method can be utilized for both active and passive identification of physical devices. The system is able to either identify the physical device (e.g. a Kindle), its type (e.g. eReader) or neither, which results in a fully or partially unknown device. While the system does show promising results in identifying both devices and their types, it is concluded that the scalability is the solution’s main limitation. For future work regarding the system, it is planned to include other connection types, including DSL and LTE.

IoT SENTINEL, as proposed by Miettinen et al. [10], is an automated security system for IoT-devices. The system relies on device type classification in order to enforce security constraints within a local network. Data from the whole network stack is utilized (e.g. link and transport layer protocols) for creating feature vectors. Their approach thereby has larger constraints regarding the number of network layers required compared to the previously presented studies [8], [9], which did not depend on such information. Selected features do not include payload data, since the system is supposed to support encrypted traffic. The models that are utilized for fingerprint classification is based on the random forest algorithm. The system is proven to have a small overhead during classifications, where the random forest-based classification takes less than a millisecond to perform. The accuracy of the classification system is high in general, but the system is less accurate in classifying devices with similar characteristics. An important observation is that over half of the classified devices needs a tie break because of similarities between types, which causes the need of a reliable tie breaking solution.

---

3RF I/Q: Radio frequency in-phase and quadrature components.
4DSL: Digital Subscriber Line.
This chapter introduces the necessary concepts required to understand the content of this thesis. Firstly, some basic concepts about LTE are introduced, followed by the characteristics of drones in LTE networks. Following that are details about the machine learning techniques used.

3.1 Mobility in LTE

LTE access networks consists of entities called eNodeBs (eNB) strategically placed geographically such that large areas are covered. The specific locations of eNBs and the distance between them is dependent on the properties of the landscape and how many UEs are expected to require service. For example, urban areas with high population densities tend to be served by multiple eNBs placed closer to one another than in rural areas. Such placement is used to be able to handle all the traffic that is demanded by a lot of people at once while keeping the cost down in areas where fewer people are expected to access the network. The area covered by an eNB is referred to as a cell [1].

The coverage area of an LTE network is depicted in [5.1] In reality the coverage of a cell is not regular, thus overlaps and areas without service do occur. Cells are categorized into
several different types with varying coverage sizes. In the context of this thesis there are two main types of cells: *macro* and *micro* cells. Macro cells are the largest type of cells and micro cells are smaller sized cells and can be located inside of the coverage area of a macro cell for improved quality of service [12].

An important feature in mobile networks such as LTE is, as the name suggests, mobility. UEs should be free to move inside the network without losing connection. In conventional cellular networks, UEs are only connected to one eNB at any point in time and they can only utilize it meaningfully as long as they are close enough for the connection to be stable. Therefore, there are procedures in place to allow UEs to move between cells and establish connections with the best eNB at any point in time according to some metric. Such hops between cells are called handovers. To let the UEs know where handovers are possible, eNBs continuously transmit what are called reference signals in order to let UEs know which cells are available. These reference signals are also used by UEs to measure the quality of service from nearby eNBs [13].

To determine when UEs should perform a handover, there are mobility-related events which are triggered at the UE when certain conditions are met. Such events and their corresponding conditions are shown in Table 3.1.

<table>
<thead>
<tr>
<th>Event</th>
<th>Triggering condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Measurement from serving eNB becomes better than the specified threshold</td>
</tr>
<tr>
<td>A2</td>
<td>Measurement from serving eNB becomes worse than the specified threshold</td>
</tr>
<tr>
<td>A3</td>
<td>Measurement from neighboring eNB becomes offset better than serving eNB</td>
</tr>
<tr>
<td>A4</td>
<td>Measurement from neighboring eNB becomes better than the specified threshold</td>
</tr>
<tr>
<td>A5</td>
<td>Measurement from serving eNB becomes worse than the specified threshold and measurement from neighboring eNB becomes better than the specified threshold</td>
</tr>
<tr>
<td>A6</td>
<td>Measurement from neighboring eNB becomes offset better than neighboring eNB</td>
</tr>
<tr>
<td>B1</td>
<td>When the measurements from neighboring eNB deploying distinct RAT(^2) from that of the serving eNB (known as inter-RAT neighbor) becomes better than the specified threshold</td>
</tr>
<tr>
<td>B2</td>
<td>When the measurements from serving cell becomes worse than threshold1 and the measurements from inter-RAT neighbor becomes better than threshold2</td>
</tr>
</tbody>
</table>

Table 3.1: Mobility-related events in LTE [13]

When a UE moves away from its serving cell the measurements will likely get worse and worse until an event triggers at the UE, most likely event A2 or A3. Event A2 triggers when the measurements of the serving cell fall below some threshold decided by the eNB. Event A3 triggers when measurements from a nearby eNB is better than the serving eNB plus some offset. When such an event triggers, the UE sends a measurement report to the serving cell. Measurement reports contain one of, or both of the following metrics for the serving cell and neighboring cells [13]:

1. *Reference Signal Received Power (RSRP)*: A measurement of the received power from a specific eNB. It does not consider interference or noise components.

2. *Reference Signal Received Quality (RSRQ)*: A ratio of the received power (RSRP) to the Received Signal Strength Indicator (RSSI) for some specific eNB. The RSSI is a measurement of the total received power including the interference from all sources, including serving and non-serving cells.

The eNB compares the measurements from the different cells and chooses an appropriate one, called the target cell. The serving cell then requests the target cell to prepare for a han-

\(^1\)Later releases of LTE support technologies allowing UEs to be connected to multiple cells.  
\(^2\)RAT: Radio Access Technology.
3.2. Drones in LTE networks

Aerial drones are generally limited in range due to the requirement of keeping a line-of-sight connection with the user controlling it. As soon as there are any obstructions like trees or buildings there is a risk of losing connection. Controlling drones over mobile networks has the clear advantage of increasing the range to virtually infinite as long as both UE and the user controlling it are in range of a eNB and the delay is not too long for it to be controlled securely. This is because data is routed through the internet rather than directly between the user and the drone.

The 3rd Generation Partnership Project (3GPP) studied the effects that aerial drones have on the networks they are connected to and how the network affects the drones [14]. They came to the conclusion that there are several scenarios where aerial drones are problematic. These examples generally apply to drones flying at an altitude such that they are in line-of-sight of multiple eNBs. Drones flying at low altitude does not cause interference to the extent of higher altitude drones but they can still be important to identify due to safety or political reasons.

DL Interference on drones

Aerial drones have a high probability of being in line-of-sight of multiple eNBs causing them to experience downlink (DL) interference from multiple sources. This is because drones are likely to be reached by transmissions from nearby cells which disturb the conditions around the drones. Such interference causes drones to generally have worse signal-to-noise ratio compared to terrestrial UEs which are not reached by as many eNBs. Thus, drones generally require more resources allocated from the eNB to be able to be delivered the expected throughput which in turn degrades the overall experience for other UEs in the network since resources are skewed toward drones [14].

UL interference induced by drones

Similar to how drones are affected by interference from nearby eNBs, the reverse situation is also true. Because drones are likely in line-of-sight of multiple eNBs they will cause some interference in those cells. Such interference degrades the throughput of other UEs in the network due to a higher resource utilization needed to provide the expected throughput [14].

Down-tilted eNB antennas

Since the majority of UEs in an LTE network are terrestrial UEs, eNB antennas are often slightly tilted towards the ground to provide as high signal strength as possible to as many UEs as possible, while separating the cells from each other and decreasing inter-cell interference [15]. A side-effect of this is that UEs positioned above eNBs are likely to be served by the side lobes of the antennas, meaning that the signal strength can be significantly lower. This can cause the aerial drone to be served by a eNB further away than the one that is closest geographically. Additionally, the pathloss is not necessarily the same in the DL and UL due to different side lobe orientations [14].
3.3 Drone identification

Some ideas have been proposed that can be used to identify drones [14]. There can be a UE-based solution where aerial UEs send some information to the network which states that they are airborne UEs. Another option is to introduce some kind of certification or license such that drones need explicit permission to be allowed access to the network. However, neither of these solutions is capable of solving the problem of rogue drones. For example, an LTE capable terrestrial UE like a mobile device mounted on a drone would not be expected to identify itself as a drone but the effect it would impose on the network would be similar if not identical to that of a regular drone [16]. To detect rogue drones there needs to be a solution in place which does not rely on UEs being transparent with their identity.

A third option, which is the one explored in this thesis, is a network-based solution where the responsibility of identifying drones is put on the network. Such solutions can analyze the mobility patterns and handover characteristics of UEs in order to make an informed decision about whether they are drones. Therefore, network-based solutions are able to detect rogue drones. Since the network only makes predictions about whether a UE is a drone or not they can’t be treated as absolute truths. Some drones are likely to be predicted as terrestrial UEs and vice versa. One viewpoint adopted in this thesis is that the goal should be to have a low false positive ratio (FPR), i.e., few ground UEs should be incorrectly classified as a drone since, depending on the way the identified drones are treated, false positives could lead to regular ground UEs having unpleasant user experience [8].

The different approaches to identify drones are not necessarily exclusive. For example, they can co-exist such that self-identified drones can be treated as known labels during the training phase of a network-based solution.

3.4 Binary classification

Binary classification is the problem of classifying data into one of two classes. The work done in this thesis, where UEs are classified as either drone or non-drone (referred to as normal or regular UE), is an example of this.

Evaluation metrics

The straightforward approach to evaluate a classification system might be to calculate the accuracy, i.e., the fraction of predictions that were correct. However, it can be a misleading metric in a scenario where the classes are not equally important to classify correctly, or in case the classes are imbalanced. It is often more informative to look at the ratios of positives or negatives being correct or incorrect.

To help interpret these values there are two metrics called precision and recall. They are defined as follows:

\[
\text{precision} = \frac{TP}{TP + FP},
\]

\[
\text{recall} = \frac{TP}{TP + FN},
\]

where \(TP, FP,\) and \(FN\) are the number of true positives, false positives, and false negatives [17].
A visual representation of precision and recall is found in Figure 3.2. In this thesis, precision is interpreted as the ratio of correctly classified drones among all classified drones. Recall is interpreted as the ratio of correctly classified drones among all drones in the data.

**Importance of false positives and false negatives**

If false positives are considered to be a worse outcome than false negatives, or vice versa, it can be desirable to weight the model to be more likely to predict one class over the other. This can be done by setting a threshold, called the discrimination threshold, such that all predictions are made based on if they are above or below that threshold. This means that the model should be able to output a numerical value, such as a probability between 0 and 1 of an example belonging to a specific class, and all outputs with a probability above the threshold will be classified as that class. A threshold of 0.5 puts no bias towards either false positives or false negatives. By using such an approach a model can be designed to fit a specification of a target ratio of false positives or negatives [18].

Weighing a model towards false positives or false negatives will likely have a negative effect on the overall accuracy, since in such a scenario it is deemed more important to avoid specific types of misclassifications as opposed to all misclassifications as a whole.

### 3.5 Feature selection

Feature selection in this context is the process of choosing a specific subset of available data to be used as features in a machine learning model. There are many benefits that can be gained when doing some sort of feature selection. The models become more simple by reducing the number of features, thus making it easier for humans to understand the model. Additionally, using fewer features generally means faster training of the model and it can also help to reduce overfitting. Using many features can lead to additional data collection which degrades network performance. Thus, an optimal set of chosen features can also save network resources. Effective feature selection is done by removing features that are not going to be
useful predictors for the problem. Another goal is to remove redundant features, meaning that if multiple features are heavily correlated they should not all be needed in the selected dataset.

Feature selection typically requires domain knowledge in order to have an idea of which features are relevant to the problem at hand. Additionally, there are some techniques that can help with feature selection. A simple approach is to remove every feature with a variance below a certain threshold. Such a solution makes sure that all selected features have some degree of variability which is important since features with very low variance are unlikely to be useful when making predictions. However such an approach does not remove redundant features.

**Principal component analysis**

Principal component analysis (PCA) is a technique commonly used for dimensionality reduction and feature extraction. The goal of PCA is to transform features that are possibly correlated into high-variance uncorrelated features, called principal components. The total number of principal components that can be extracted for some data with dimensionality $D$ containing $N$ data points is $\min(N, D)$, meaning that as long as there are more data points than the number of dimensions there exists one principal component per dimension [18].

The first principal component is chosen as a linear combination of the input features such that when data is projected onto it, the variance of the projected data is maximized. The second principal component is also chosen such that the variance of projected data is maximized but it has to be orthogonal to the previously chosen principal components to make them uncorrelated. Subsequent principal components are chosen in the same fashion until $D$ principal components have been extracted. No more can be found beyond that point because of the orthogonality constraint [18].

An example of PCA on a dataset with two dimensions is shown in Figure 3.3. The longer green line shows the first principal component and it is evident that it is parallel with the direction of highest variance of the data. For the second principal component, shown as the shorter green line, there is only one possible choice due to the orthogonality constraint.

In the case of high dimensional data, PCA can be used for dimensionality reduction by only using some subset of the principal components, preferably the components which correspond to the highest variance.
3.6 Decision trees

The decision tree algorithm is a type of machine learning model, commonly used for classification problems. The basic principle of decision trees is to partition the input-space into cuboid-like shapes (depending on the dimension of the input space) with decision boundaries that are parallel to the axes of the input-space. Each enclosed area formed by the partitions corresponds to a specific class. Making a classification, given some input, can be viewed as a traversal of a binary tree ending up with a class corresponding to one of the enclosed areas formed by the decision boundaries. An advantage with decision trees, in contrast to something like artificial neural networks, is that when a model has been trained it is easy for a human to interpret how a new example is classified by analyzing the tree structure [18].

![Decision Boundaries and Binary Tree](image)

Figure 3.4: Decision boundaries and corresponding binary tree in a two-dimensional feature space

An example of decision tree boundaries and the corresponding binary tree is illustrated in [3.4]. The decision tree is built during the training phase of the model. It involves choosing which region is to be split, which feature is responsible for the split, and deciding the threshold parameter $\theta_i$. The class for each region has to be decided as well. Finding the optimal structure for a decision tree is very computationally heavy. Therefore, a greedy search approach is often used. Starting with the first node, which represents the entire feature space, a search is made over all input features and values of $\theta_i$ to find the ones resulting in the lowest sum-of-squares error. This is repeated until a satisfactory number of splits are made. The choice of when to stop the training is not trivial. Theoretically, training could keep going until there is one leaf node per training example although this would result in an overfitted model in most cases. If very few splits are made the model might instead be too simple to model the data with high accuracy. A common method for early stopping the training of a decision tree is to set the minimum amount of training examples needed in each leaf, meaning that each region in the feature space has to contain at least some set number of examples. Another way to stop the training early is to set a maximum depth of the tree. In this way, the tree will only grow to a specific depth and then stop.

3.7 Ensemble learning

According to Caruana et al. [19], ensemble learning with decision trees is generally one of the best performing machine learning models when it comes to binary classification. Ensemble learning involves methods of combining multiple classifiers into one classifier through
3.8. Cascading classifiers

A cascading classifier is a series of independently trained, increasingly complex classifiers. It was first presented by Viola and Jones [22] for use in a face detection algorithm. The idea is to send positive results from one classifier through to the next more complex classifier, and any negative results are immediately classified as negative and not sent forward. This process is repeated once for each classifier. A discrimination threshold is decided for each classifier such that data points falling to one side of it are classified as negative and the rest are sent to the next classifier. Such an approach is likely to be quicker than using one complex classifier since parts of the data will be filtered out by the initial simpler classifiers.

In this thesis, a two-stage cascading classifier is used. In addition to reducing classifying times, it also reduces the amount of data needed and thus reducing the load on the network since the second classifier uses a lot more data than the first. In the cascading classifier presented by Viola and Jones, only negatives were filtered out in each stage. In this thesis however, an altered method is used. Both negatives and positives are filtered out by using two thresholds instead of one. The data points falling between the thresholds are sent to the second classifier and the rest are classified in stage one. Such an approach is chosen to further reduce the data points sent to stage two. A risk with such an approach is that false positives are possible in the first classifier.

3.9 Model Validation

When evaluating the performance of some machine learning model it is important to use different data from the training data. This is because the model is likely to overfit to the training data, meaning that the model will perform well on that specific data but might not generalize well to other unseen examples. Therefore, it is common to split the data into one
training set and one test set designated to evaluate a model after training [18]. However, using just a single test set might not be the best approach since there is a possibility of that set being particularly easy or difficult just by random chance.

**K-fold cross-validation**

One common model validation technique is *k-fold cross-validation*. It is done by splitting the data into *k* disjoint sets (folds), and then training a model *k* times where each fold acts as the test data exactly once and the remaining data is used for training.

![Figure 3.5: Data split in k-fold cross-validation with k = 4](image)

An example of cross-validation with *k* = 4 is illustrated in Figure 3.5. Such a solution creates a more robust validation than just having a single test set since the model is trained and tested multiple times on different datasets without the need for additional data. One drawback of k-fold cross-validation is that *k* models is trained and evaluated which is time consuming if the training itself is computationally heavy [18].
4 Method

The work done in this thesis is inspired by the workflow for machine learning in the network domain outlined by Wang et al. [23], shown in Figure 4.1.

![Workflow of machine learning in the networking domain](image)

Figure 4.1: Workflow of machine learning in the networking domain [23]

The first step, problem formulation, consists of identifying which type of problem is being investigated. Drone identification is a binary classification problem and, at this stage, we consider methods that have shown to work well with binary classification problems. Steps two through five are an iterative process going from data collection to model validation. This process is described in detail in this chapter. Step six involves deploying the complete model and is outside the scope of this thesis, although a lot of choices throughout the data analysis and model construction are made with possible deployment in mind. Machine learning models are made in Python 3 using scikit-learn [24].

4.1 Data collection

The data used to train the models to detect drones is generated through simulation. The simulation models a multi-cell LTE network with cells of different sizes. It is simulating an urban area with both macro and micro cells serving three types of UEs: drones flying at random altitudes, outdoor UEs at ground level, and indoor UEs at random altitudes, simulating UEs...
inside buildings. Exactly one minute of activity is simulated. A more detailed view of the simulation setup is in Table 4.1.

<table>
<thead>
<tr>
<th>Deployment</th>
<th>Sites</th>
<th>7 Sites, 21 Macro Cells and 63 Micro Cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>BS power</td>
<td>46 dBm</td>
<td></td>
</tr>
<tr>
<td>Inter Site Distance (ISD), macro cells</td>
<td>500 m</td>
<td></td>
</tr>
<tr>
<td>Carrier Frequency</td>
<td>2 GHz</td>
<td></td>
</tr>
<tr>
<td>Carrier Bandwidth</td>
<td>10 MHz</td>
<td></td>
</tr>
<tr>
<td>Propagation Model</td>
<td>SCM (3GPP TR-36.777 for Drones, 3GPP TR-38.901 for regular UEs)</td>
<td></td>
</tr>
<tr>
<td>Mobility Params</td>
<td>Event</td>
<td>A3</td>
</tr>
<tr>
<td></td>
<td>A3 Offset</td>
<td>2.0 dB</td>
</tr>
<tr>
<td></td>
<td>A3 Hysteresis</td>
<td>1 dB</td>
</tr>
<tr>
<td></td>
<td>TTT</td>
<td>160 ms</td>
</tr>
<tr>
<td>UE properties</td>
<td>Drone UE:</td>
<td>Speed: Random between 0 to 120 km/h, Height: Random between 10 to 300 m</td>
</tr>
<tr>
<td></td>
<td>Outdoor regular UE:</td>
<td>Speed: Random between 0 to 120 km/h, Height: 0m (Ground level)</td>
</tr>
<tr>
<td></td>
<td>Indoor regular UE:</td>
<td>Speed: 5 km/h, Height: Random between 0 to 60 m</td>
</tr>
<tr>
<td>UE classification</td>
<td>UE distribution</td>
<td>977 UEs: 52% drones, 30% indoor regular UEs, 18% outdoor regular UEs</td>
</tr>
</tbody>
</table>

Table 4.1: Simulation setup

The data gathered through the simulation consists of radio measurements such as RSRP, RSRQ, RSSI, and UL/DL throughput for UEs. It contains both periodically reported data and event-based data, as described in Section 3.1. The periodic data is measured every 40 ms. Additionally, the data is labeled with the true UE type (drone or non-drone) such that supervised learning can be utilized. A list of potentially interesting (for the purpose of drone predictions) features gathered during the simulation is in Table 4.2.

<table>
<thead>
<tr>
<th>Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>allCellsRsrp</td>
<td>Signal power of all cells measured by a UE</td>
</tr>
<tr>
<td>allCellsRsrq</td>
<td>Signal quality of all cells measured by a UE</td>
</tr>
<tr>
<td>rssi</td>
<td>Signal strength indicator for the serving cell measured by a UE</td>
</tr>
<tr>
<td>sinr</td>
<td>Signal-to-interference-plus-noise ratio of the channel between a UE and the serving cell</td>
</tr>
<tr>
<td>userDLThroughput</td>
<td>The downlink throughput of a UE</td>
</tr>
<tr>
<td>userULThroughput</td>
<td>The uplink throughput of a UE</td>
</tr>
</tbody>
</table>

Table 4.2: A subset of features gathered through simulation

4.2 Data analysis

When data has been generated, the goal is to find information in it that is suitable to use as predictors in a machine learning model. This section describes the features that are considered and which of these are chosen to be used as input for the different models and why these choices are made.
4.2. Data analysis

Figure 4.2: Best cell RSRP over time for a drone and a regular UE

Available input features

Table 4.2 contains some candidates for features that can be used. However, when the measurements are made over a period of 60 seconds, it is not necessarily sufficient to consider a snapshot of the network but instead how it, and the measurements, change over time.

Figure 4.2 shows the best cell RSRP for one drone and one non-drone over the duration of the simulation. Note that the value at a single point in time might not give enough information whether a UE is a drone or not since the measurements overlap a lot in the range $[-80, -70]$ dBm. However, looking at how the values change over time, it is evident that the best cell RSRP of the non-drone has more variance than the drone does. Using something like the variance, or the difference between the maximum and minimum value, of some metrics of the last N data points might be a good indicator for detecting drones.

A property of aerial drones is that they are likely to have a good reception from multiple cells since there are less obstructions at higher altitudes. A feature that captures this can be useful when predicting drones. Two ways used previously to capture this property are the variance in RSRP of some N best cells from a UE’s perspective, and the gap in RSRP between the best and second best cell [3].
4.2. Data analysis

<table>
<thead>
<tr>
<th>Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>bestCellRsrp</td>
<td>Signal power of the best cell</td>
</tr>
<tr>
<td>rssi</td>
<td>Signal strength indicator of a UEs serving cell</td>
</tr>
<tr>
<td>ulThroughput</td>
<td>Uplink throughput of a UE</td>
</tr>
<tr>
<td>dlThroughput</td>
<td>Downlink throughput of a UE</td>
</tr>
<tr>
<td>rsrpVar/rsrqVar</td>
<td>Variance in signal power/signal quality of the top three cells of a UE</td>
</tr>
<tr>
<td>rsrpGap/rsrqGap</td>
<td>Difference in signal power/signal quality of the two best cells of a UE</td>
</tr>
<tr>
<td>rssiVar</td>
<td>Variance in RSSI from measurements from the last five seconds of measurements</td>
</tr>
<tr>
<td>rssiMaxMinDiff</td>
<td>Difference between the maximum and minimum RSSI value from the last five seconds of measurements</td>
</tr>
<tr>
<td>bestCellRsrpVar</td>
<td>Variance in best cell RSRP from measurements from the last five seconds of measurements</td>
</tr>
<tr>
<td>bestCellRsrpMaxMinDiff</td>
<td>Difference between the maximum and minimum best cell RSRP value from the last five seconds of measurements</td>
</tr>
</tbody>
</table>

Table 4.3: All features considered as input for the machine learning models

The complete list of features used as input to train the machine learning models is in Table 4.3 and includes features from Table 4.2 and features generated through the reasoning described in this section.

Figure 4.3: Gini-importance of the different features

Figure 4.3 shows the gini-importance of each of the features from Table 4.3. The gini-importance is a metric acquired from a trained random forest model and describes the importance of each feature when making classifications. It is calculated by examining how many splits in the decision trees in the forest are based on the specific features [25].

Selected features

It is important to separate the features based on time-series of data from those measured at single points in time since the first step should only be using measurements from a single
point in time independently from what happened beforehand, as described further in Section 4.3. Therefore, the $rssi\text{Var/bestCellRsrpVar}$, $rssi\text{MaxMinDiff/bestCellRsrpMaxMinDiff}$ are not considered for the first step. The features are selected as the top $K$ principal components after performing PCA on the data. $K$ is selected such that the classification ratio$^1$ is maximized for stage one, and the drone recall$^2$ is maximized for stage two. The performance is calculated as an average of a number of arbitrarily chosen ensemble and decision tree models.

![Figure 4.4: Model performance as a function of the number of principal components averaged over different models. Dots denote maximum performance.](image)

Figure 4.4 shows how the classification performance is affected by the number of principal components used for both stages. Four components are chosen for stage one and ten for stage two. Note that there are more choices for $K$ in the second stage due to there existing more input features to choose from.

### 4.3 Model construction

Introducing drone identification capability to eNBs imposes computation and memory requirements, and the data to be used might put additional load on the network. Therefore, it is desirable not to use compute-intensive models and not to require the model to store data in memory. However, computation-heavy models are generally better performing. Additionally, basing classifications on previously observed data, and therefore requiring some memory capacity from the eNB, is likely to increase the predictive performance of the model. To find a balance between the complexity and accuracy of the models, a two-stage cascading classifier is used.

The first stage classifier uses measurements from a single point in time, and therefore only requires a single measurement report per UE and does not require any data to be stored at the eNB aside from the data associated with the model structure. It is used to classify UEs which have a very high probability of belonging to one specific class. For example, a drone flying at 300 meters is likely to have very different characteristics from any regular UEs. And conversely, an indoor UE on the bottom floor is likely to have different characteristics from any drone UEs. Such cases can be classified with high certainty without requiring additional measurements from more than one point in time. The second stage classifier is used to classify more ambiguous cases. This is done by using measurements from consecutive points in time in order to have more information about the behaviour of the UEs.

$^1$The classification ratio is the fraction of UEs correctly classified out of all UEs in the first stage.

$^2$The drone recall is defined as the number of drones correctly classified out of all drones.
4.3. Model construction

Three different machine learning models are evaluated for both stages, they include two ensemble learning models using decision trees (boosting and random forests), and one simple decision tree classifier. Additionally, a single-stage model proposed by Rydén et al. [3] is used to evaluate how the two-stage classifier approach performs in relation to a more conventional approach. The single-stage model is a decision tree with input features \( rssi \), \( servingCellRsrp \), \( rsrpVar \), and \( rsrpGap \).

Stage 1

The goal of the first stage is to find the UEs with definite and unambiguous radio characteristics, such that they can be classified with a very high certainty of being either a drone or a regular UE, e.g., drones flying at high altitudes or indoor UEs at ground level. The remaining cases are forwarded to the second stage. The predictions are made using data from only one point in time for each UE. The features based on the change of some measurements over time cannot be used in this stage since, for example, the variance of a single measurement is meaningless. As stated in Section 3.7, ensemble learning with decision trees has shown to be a good model to use for binary classification problems. Additionally, decision trees are able to output probability values in addition to the binary classification value which is crucial for the first stage since the first classification is based on the probability of the UEs being drones.

In order to decide which UEs can be adequately classified by the first stage and which should be forwarded to the second classifier, two discrimination thresholds are decided such that UEs with a drone probability falling between the thresholds are sent to the second classifier. UEs with a probability above the highest threshold are classified as drones and UEs with a probability below the lower threshold are classified as regular UEs.

The thresholds are set during training. They are selected such that no regular UE is classified as a drone and no drone is classified as a regular UE in the test data while minimizing the number of UEs that are being sent to the second stage. Note that just because the thresholds are set such that there are no misclassifications in the test set it does not necessarily prevent misclassifications when new data is introduced. The selection of the different thresholds for each model is presented in Section 5.1.

To choose the specific parameters of the models to be compared, the max tree depth will be chosen such that it maximizes the classification ratio, shown in Figure 4.5. Additionally, the number of trees in each of the ensemble models should be decided. Figure 4.6 shows how the number of trees affects the classification ratio. Note that the decision tree model is not present since it consists of one single tree by default. Table 4.4 compiles the chosen parameters for the different models.

![Figure 4.5: Classification ratio as a function of maximum tree depth](image-url)
4.3. Model construction

![Figure 4.6: Classification ratio as a function of the number of trees](image)

Table 4.4: Chosen parameters for each model for stage one

<table>
<thead>
<tr>
<th>Model</th>
<th>Max depth</th>
<th>Number of trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>BST</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>RF</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>DT</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

Stage 2

The goal in the second stage is to classify the cases where the first classifier cannot make a decision with a high probability of being correct. This means that the cases lying between the two thresholds defined in the first stage are forwarded to the second stage. The same three model types are constructed as in stage one. The predictions of this stage are based on measurements over 60 seconds as opposed to the first stage which only uses measurements from one single point in time. Therefore, the features that are based on the change over time are used in this case. For each point in time and for each UE, the models output a probability of being a drone as in the first stage. The probabilities are then averaged for each UE such that the final probability value of a UE being a drone is an average of all the probabilities calculated for previous observations.

As stated in Section 3.3, it is important to minimize the amount of regular UEs being classified as drones. Therefore, a threshold is set such that no regular UEs are classified as drones in the test set. Since this is the final classification, only one threshold is required as opposed to the two thresholds used in stage one. All UEs with a higher probability of being a drone than the threshold are classified as a drone and the UEs with a lower probability are classified as a regular UE. It is expected that the UEs are better separated in the second stage since the models should perform better when more data is available to them.

The max depth and number of trees is chosen differently in the second stage since the models are not the same. The metric used to select these parameters is the drone recall at zero FPR. How this is affected by max depth and the number of trees in the models is shown in Figure 4.7 and Figure 4.8 respectively and Table 4.5 shows the selected parameters.
4.4 Model validation

The two stages have different objectives, and therefore they are evaluated differently. The data is split into two sets (75%/25%): one training/test set used for training, cross validation, and setting the threshold values, and one validation set to validate the performance of the trained models and previously decided thresholds on unseen data. The data is split on a UE-basis, such that the same UE is not present in both datasets. Cross-validation is used to reduce the variability of the results and to detect overfitting.
4.4. Model validation

Stage 1

The purpose of the first stage is to filter out as many UEs as possible such that the amount of UEs forwarded to the more computation-heavy second stage is minimized. Therefore, one metric that measures the performance of the first stage is the ratio of classified UEs in the first stage, i.e., UEs that fall outside the two thresholds. The thresholds are set based on the probabilities acquired from the test data in all cross-validation folds combined. To validate the generalization of the models, evaluations are made on the validation set containing unseen data.

Since the first stage does not use previously observed data, there are no memory requirements other than the information about the model structure itself. The models will however differ in complexity which is evaluated by measuring the time it takes to make predictions.

Stage 2

It is important to have a low amount of false positives, meaning that few ground UEs should be classified as drones, because depending on the choices made by the network the UE might be affected in a negative way. Therefore, one metric of interest is the recall of normal UEs, i.e., the fraction of normal users correctly classified. The drone recall is also measured. Another point of interest is how quickly the performance increases. For example, if there are no major performance increase between 20 seconds and 60 seconds, the predictions could be made already at 20 seconds which would reduce the computation and measurements needed by two thirds. The threshold is based on probabilities from the test data of all folds. Like stage one, model evaluations are made with the validation set.

In the second stage there are two types of values to be stored in memory to make predictions. The first are the measurements of the features based on previous observations. The second are the previously calculated probabilities for a UE, which are needed to produce the final probabilities by averaging over previous probabilities.
This section presents the discrimination threshold selection and model performance of the different models for both stages. The best performing model is compared to a single-stage model. Additionally, in Sections 5.4 and 5.5 models are tested where different cells in the network are trained separately instead of using one general model for all cells.

5.1 Threshold selection

The models use the number of principal components shown in Figure 4.4 and model parameters shown in Table 4.4 and Table 4.5.

Stage 1

The different models output different probabilities for the UEs and therefore different discrimination thresholds are chosen for each model. Figure 5.1 shows the drone probabilities for each UE in test data and how the thresholds are selected for each model. The values of the thresholds are shown in Table 5.1. Note that the UE height on the y-axes is purely for visualization purposes and not used in the classifications.
5.1. Threshold selection

Figure 5.1: Threshold selection for the three models in stage one. UEs in the green area are classified directly in stage one, UEs in the red area are sent to the second classifier.

<table>
<thead>
<tr>
<th>Model</th>
<th>Lower threshold</th>
<th>Upper threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>BST</td>
<td>0.44</td>
<td>0.51</td>
</tr>
<tr>
<td>RF</td>
<td>0.01</td>
<td>0.77</td>
</tr>
<tr>
<td>DT</td>
<td>0.01</td>
<td>0.86</td>
</tr>
</tbody>
</table>

Table 5.1: Thresholds of the first stage

Stage 2

In the second stage, one threshold is selected for each model. These models are trained on the data which is expected to be forwarded by stage one. The single-stage model also uses one threshold and its choice is presented together with the stage two models, denoted as the OLD model. The single-stage model, however, is trained on the same data as the models in stage one since it is supposed to handle all data. Figure 5.2 shows the probabilities for each UE in the test data and how a threshold is selected for each model. The values of the thresholds are shown in Table 5.2.
5.2 Validation results

By evaluating the models on unseen data it is possible to see how well they generalize to new data using the thresholds decided in Section 5.1. For this purpose, a validation dataset is used containing data from 244 UEs which are not present in the training/test dataset. The first stage models use all 244 UEs, and the second stage uses the UEs forwarded by the best performing first-stage model.

Stage 1

The first stage only uses a snapshot of measurements from a UE to make predictions. Such a snapshot can either be a periodic or event-based measurement report. The first measurement for each UE in the data is used.

Figure 5.2: Threshold selection for the three models in the second stage and the single-stage model

Table 5.2: Threshold of the second stage

<table>
<thead>
<tr>
<th>Model</th>
<th>Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>BST</td>
<td>0.54</td>
</tr>
<tr>
<td>RF</td>
<td>0.63</td>
</tr>
<tr>
<td>DT</td>
<td>0.69</td>
</tr>
<tr>
<td>OLD</td>
<td>0.36</td>
</tr>
</tbody>
</table>
5.2. Validation results

Figure 5.3: Validation results for stage one

<table>
<thead>
<tr>
<th></th>
<th>BST</th>
<th>RF</th>
<th>DT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class. drones</td>
<td>67/133 (50.4%)</td>
<td>63/133 (47.4%)</td>
<td>15/133 (11.3%)</td>
</tr>
<tr>
<td>Class. normal</td>
<td>75/111 (67.6%)</td>
<td>62/111 (55.9%)</td>
<td>71/111 (64.0%)</td>
</tr>
<tr>
<td>Class. total</td>
<td>142/244 (58.2%)</td>
<td>125/244 (51.2%)</td>
<td>86/244 (35.2%)</td>
</tr>
<tr>
<td>Drone recall</td>
<td>100.0% (0 errors)</td>
<td>100.0% (0 errors)</td>
<td>100.0% (0 errors)</td>
</tr>
<tr>
<td>Normal recall</td>
<td>100.0% (0 errors)</td>
<td>100.0% (0 errors)</td>
<td>100.0% (0 errors)</td>
</tr>
<tr>
<td>Accuracy</td>
<td>142/142 (100.0%)</td>
<td>125/125 (100.0%)</td>
<td>86/86 (100.0%)</td>
</tr>
</tbody>
</table>

Table 5.3: Validation results of the first stage

Figure 5.3 visually shows the predictions on the validation data. Table 5.3 presents information about the classifying performance of the different models. The classification statistics tell how many UEs were classified correctly, and the recall and accuracy columns show the accuracy of the UEs that are classified.

Figure 5.4: Cumulative distribution function of RSRP for classified and unclassified UEs in the stage one BST model
In order to understand the differences between the UEs that are classified in stage one and those who are sent to the second stage, Figure 5.4 shows the cumulative distribution functions of the best cell RSRP values for both classified and unclassified UEs from the BST model. It shows that the UEs classified in the first stage generally have different values of RSRP, whereas the UEs sent to the second stage show a much more similar distribution.

Stage 2

The 102 UEs forwarded by the BST model in stage one is used to evaluate the second stage models. All 60 seconds worth of data for each UE is used. Figure 5.5 shows the estimated probability of each UE being a drone after 60 seconds worth of data. Detailed classification statistics is in Table 5.4. Figure 5.6 shows how the performance of the models changes over time as more data is given to the model.

Figure 5.5: Validation results for stage two

<table>
<thead>
<tr>
<th></th>
<th>BST</th>
<th>RF</th>
<th>DT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drone recall</td>
<td>64/66 (97.0%)</td>
<td>64/66 (97.0%)</td>
<td>64/66 (97.0%)</td>
</tr>
<tr>
<td>Normal recall</td>
<td>36/36 (100.0%)</td>
<td>36/36 (100.0%)</td>
<td>36/36 (100.0%)</td>
</tr>
<tr>
<td>Accuracy</td>
<td>100/102 (98.0%)</td>
<td>100/102 (98.0%)</td>
<td>100/102 (98.0%)</td>
</tr>
</tbody>
</table>

Table 5.4: Validation results of the second stage
5.2. Validation results

Figure 5.6: Validation results for stage two over time

Computational requirements

The relative computational load of the models are evaluated by measuring the time it takes to make predictions. Measurements are made on a laptop with an Intel® Core™ i5-8350U CPU @ 1.70GHz and presented in Table 5.5.

<table>
<thead>
<tr>
<th></th>
<th>Time stage 1 (s)</th>
<th>Time stage 2 (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BST</td>
<td>1.96</td>
<td>1690</td>
</tr>
<tr>
<td>RF</td>
<td>1.28</td>
<td>2394</td>
</tr>
<tr>
<td>DT</td>
<td>0.47</td>
<td>752</td>
</tr>
</tbody>
</table>

Table 5.5: Average times to classify all 977 UEs

It is important to note that the values in Table 5.5 are not representative of how the classification times would be if they were deployed in a real network since the hardware is different and the code can likely be optimized. However, they can be used as a comparison between the different models. In terms of memory usage, the models in the first stage do not require any additional memory other than model metadata. For the second stage, the three models presented in this thesis use four metrics which depend on observations over the last five seconds: rssiVar, rssiMaxMinDiff, bestCellRsrpVar, and bestCellRsrpMaxMinDiff. This means that the RSSI and best cell RSRP observations have to be stored. Five seconds of measurements with a 40 milliseconds interval require 125 RSSI and RSRP observations to be stored per UE. Additionally, the previous probabilities of each UE being a drone should be stored in order to calculate a final average. One probability value every 40 millisecond over 60 seconds adds up to 1500 values to be stored. However, it is not necessary to store all these values in memory if averaging is made continuously as new probabilities are calculated.
5.3 Comparison with single-stage model

For the comparison with the single-stage model, the BST model is used for stage one and the DT model is used for stage two. The 244 validation UEs are used and the first stage is given the first measurements in time from every UE to simulate a real scenario. Stage two only classifies the UEs forwarded by stage one. The performance of the two different models is in Figures 5.7 and 5.8 and in Table 5.6. In Figure 5.8, the UEs in the second stage are the same UEs that fall in the red area between the thresholds in stage one.

![Figure 5.7: OLD model results](image1)

![Figure 5.8: Results when combining the two stages](image2)

<table>
<thead>
<tr>
<th></th>
<th>1-stage model</th>
<th>2-stage model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage 1 class. ratio</td>
<td>-</td>
<td>142/244 (58.2%)</td>
</tr>
<tr>
<td>Drone recall</td>
<td>132/133 (99.2%)</td>
<td>131/133 (98.5%)</td>
</tr>
<tr>
<td>Normal recall</td>
<td>108/111 (97.3%)</td>
<td>111/111 (100%)</td>
</tr>
<tr>
<td>Accuracy</td>
<td>240/244 (98.4%)</td>
<td>242/244 (99.2%)</td>
</tr>
<tr>
<td>Time (s)</td>
<td>137.11</td>
<td>79.00</td>
</tr>
</tbody>
</table>

Table 5.6: Results of single-stage model and two-stage model

5.4 One model per cell type

So far, no distinction has been made between the different cells in the network since they all use the same model. Another approach is to use different models for different cells. To test this, one model is trained per cell type, i.e., one model for macro cells and one for micro cells. One BST model is trained for macro cells and one for micro cells in stage one.
model is trained on UEs connected to microcells and the macrocell model is trained on UEs connected to macrocells. The model parameters and threshold values are decided with the same procedure as described in Sections 4.3 and 5.1.

![Figure 5.9: One model per cell type results for stage one](image)

Stage 1

<table>
<thead>
<tr>
<th>Class</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class. drones</td>
<td>67/133 (50.4%)</td>
</tr>
<tr>
<td>Class. normal</td>
<td>84/111 (75.7%)</td>
</tr>
<tr>
<td>Class. total</td>
<td>151/244 (61.9%)</td>
</tr>
<tr>
<td>Drone recall</td>
<td>98.5% (1 error)</td>
</tr>
<tr>
<td>Normal recall</td>
<td>98.8% (1 error)</td>
</tr>
<tr>
<td>Accuracy</td>
<td>151/153 (98.7%)</td>
</tr>
</tbody>
</table>

Table 5.7: One model per cell type results for stage one

The resulting classification results for stage one when using separate models for micro and macro cells is in Figure 5.9 and Table 5.7. For the second stage, a DT model is trained for each cell type.

![Figure 5.10: One model per cell type results for stage two](image)

Stage 2

<table>
<thead>
<tr>
<th>Class</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drone recall</td>
<td>64/66 (97.0%)</td>
</tr>
<tr>
<td>Normal recall</td>
<td>36/36 (100.0%)</td>
</tr>
<tr>
<td>Accuracy</td>
<td>100/102 (98.0%)</td>
</tr>
</tbody>
</table>

Table 5.8: One model per cell type results for stage two
5.5 One model per cell

The classification results for stage two, when using separate models for micro and macro cells, are in Figure 5.10 and Table 5.8. Unlike in stage one, the UEs can now move between micro and macro cells during the classification. Therefore, a common threshold is decided and it is used regardless of if a user stays within the same cell or moves between cells.

To further make models more specialized, one mode per stage is trained for each individual cell. Again, BST models are used for stage one and DT models are used for stage two. The model for each cell is trained on UEs connected to that cell.

### Stage 1

| Class. drones | 35/133 (26.3%) |
| Class. normal | 69/111 (62.2%) |
| Class. total  | 104/244 (42.6%) |
| Drone recall  | 68.6% (16 errors) |
| Normal recall | 98.6% (1 errors) |
| Accuracy      | 104/121 (86.0%) |

Table 5.9: One model per cell results for stage one

The classification results for stage one, when training one model per cell, are in Table 5.9. Like in section 5.4, UEs can move between cells during classification in the second stage. Therefore, a single discrimination threshold is selected for all cells.

### Stage 2

| Drone recall | 60/66 (90.9%) |
| Normal recall| 34/36 (94.4%) |
| Accuracy     | 94/102 (92.2%) |

Table 5.10: One model per cell results for stage two

The classification results for stage two, when training one model per cell, are in Figure 5.11 and Table 5.10.
6 Discussion

This chapter contains discussions about the methodology and results presented in this thesis. Additionally, some suggestions of how this work can be extended into future work is put forward.

6.1 Method

This section includes discussions about the methods used in the work for this thesis. It is structured similarly to Chapter 4.

Data collection

Since this work is based on simulated data it is important to think about the validity of the simulations. The simulations used to gather the data in this thesis are supposed to be as close to the real world as possible with regards to the radio channel modeling. However, the mobility of the UEs is a simplification of the real world. For example, the speed and height of the UEs are constant, which is generally not true for UEs in reality.

The data used to train and evaluate the models is balanced, meaning that there are roughly an equal amounts of drones and non-drones. However, this is rarely the case in a real scenario where drones are rare relative to normal users. Therefore, it would be interesting to see the difference in performance on models using imbalanced data with more normal users than drones. One assumption is that the models would be skewed towards predicting UEs as normal users since most of the training data are normal UEs.

Data analysis

The gini-importance scores in Figure 4.3 show that, as expected, some features are better than others when classifying drones. The throughput features, and \( \text{r} s\text{rpGap/rsrqGap} \) are the least important features. To save resources these features can probably be removed without sacrificing significant performance of the models.

Four out of the top five important features are the time-series based features used for the second stage. This shows that such features are useful. However, the OLD model is not much worse in performance and it does not use those types of features.
Model construction

False positives are possible in stage one. If they are considered unacceptable, one approach is to only use one threshold, as proposed by Viola and Jones [22]. Such an approach would prevent normal UEs from being classified as drones in stage one since all potential drones would be forwarded to stage two. However, since almost half of the drones can be correctly classified already in stage one there is a trade-off to consider between the computation needed and accuracy.

Since the discrimination thresholds are based on the normal UE with the highest probability of being a drone in the test data, and vice versa, they can easily be dependent on outliers. For example, if a drone gets a probability assigned to it which is lower than all other drones, the effect is that the threshold is set such that the model will send more normal UEs to the second stage but it will be better at classifying such outliers and therefore have fewer misclassifications. An example of an outlier affecting the results can be seen in the DT model in Figure 5.2 where the threshold is set to 0.69 but would be closer to 0.5 otherwise. Each threshold decision in stage one is a trade-off between accuracy and classification ratio. In the second stage, the threshold decision is a trade-off between drone recall and normal UE recall. A high threshold makes the model classify few normal UEs as drones, but will misclassify more drones. A low threshold will classify few drones as normal UEs, but will misclassify more normal UEs.

In the second stage, the drone recall does not improve much after 20 seconds of data. One way to lower the computational and memory requirements of the models without sacrificing significant performance is to consider less measurements than the 60 seconds used for the models in this thesis, or by using less frequent measurements over the same amount of time. A dynamic stopping approach could also be implemented, where a UE is classified once the probability is above or below some specific thresholds. In such a solution, a two-stage approach would be redundant since UEs would continuously be filtered out of the model anyways once the probabilities are above or below target values.

Model validation

When evaluating the models there are many metrics to consider in order to properly assess their performance. For example, by looking at just the accuracy in stage one, as seen in Table 5.3, it could seem like the models are equally good since they all have 100%. However, the accuracy is just based on the UEs outside the thresholds which is why the classification ratio has to be taken into account as well. Even then you can not really understand the model's performance on each of the two classes which is why it is a good idea to go one step further and look at the classification ratio of drones and regular UEs separately.

When choosing the model parameters, as presented in Section 4.3, the metrics that are used to select the best models are the classification ratio and the drone recall at zero FPR for stage one and two respectively. This works for stage two since the only types of errors that will occur at zero FPR are drones and therefore, the drone recall will capture all of them. For the first stage however, the classification ratio might not be the best metric to use since it does not say anything about the distribution of the classified UEs. It is possible that in some cases all classified UEs are drones and such a model could have a high classification ratio but it would not be a good model since regular users, which is the largest class in a real world scenario, would seldom be classified in the first stage. Judging from the results shown in Table 5.3 this does not seem to have been an issue since the classification ratios are fairly balanced between drones and regular UEs with the exception being the DT model which classifies many more regular UEs than drones.
6.2 Results

This section is structured similarly to Chapter 5, beginning with discussions about the different models used for the two stages. Following this are discussions about the additional investigations regarding comparisons with the single-stage model and how using different models for different cells affects the performance.

Stage 1

The validation results from stage one is in Table 5.3. The ensemble learning models outperform the simpler DT model with regards to the classification ratio. No model makes any misclassifications using this particular dataset. The BST model outperforms the other models with regards to both drone and normal UE classification. An expectation was that the drone UEs sent to the second stage would be the ones flying at low altitudes. However, as seen in Figure 5.3 many high altitude drones are forwarded in all three models. This is especially notable in the RF model where the drone UEs form a U-shape such that high altitude drones are either very likely or unlikely to be drones with few examples in the middle.

The cumulative distribution function plots in Figure 5.4 clearly show that the UEs that are forwarded to the second stage experience similar conditions regardless of if they are regular users or drones, whereas the UEs classified in stage one have more distinct behaviour.

As expected, the ensemble learning models are computationally heavier than a single decision tree model as seen in Table 5.5.

Stage 2

Looking at the prediction results for the second stage, the performance is very similar for all models. They are all close to their final classification accuracy long before 60 seconds, meaning that less data can be used without sacrificing performance. As expected, the decision tree models are less computation-heavy than the ensemble methods.

By observing the plots in Figure 5.2 it is evident that lower flying drones are more difficult to accurately classify. However, the low flying drones do not have as large of a negative effect on the network as the drones flying at higher altitudes, as described in Section 3.2. Additionally, the ground UEs in tall buildings are more likely to be classified as drones than UEs on the ground. Such UEs are important to classify correctly since they do not have as much of a negative impact as drones, since they are indoors and are surrounded by obstructions.

Comparison with single-stage model

When combining the two stages, the best performing models from each stage were chosen. The two-stage model outperforms the single-stage model both with regards to accuracy and computation requirements, as seen in Figure 5.8. This is because close to 60% of the UEs are classified by the quick first stage model. Since no misclassifications are made in the first stage it shows that such a two-stage approach might be viable. Additionally, the errors made by the single-stage model are false positives, meaning that regular UEs are classified as drones which can be argued to be more damaging than the other way around.

One model per cell type

By observing the plots in Figure 5.9 it is evident that UEs in micro cells are easier to differentiate whereas in macro cells the drone UEs and normal UEs appear to behave more similarly. When combining the two models in the first stage the performance, as seen in Table 5.7 is similar to the general BST model presented in Section 5.2. The separated models achieve a higher classification ratio for normal UEs. However, there are a couple of misclassifications.
Looking at the results for the second stage in Table 5.8, the performance is the same as the second stage models presented in Section 5.2. However, by observing the plots in Figure 5.10, the separated models arguably do a better job at separating the two classes. Additionally, the accuracy is better during the first ten seconds when less data is used.

A thing to keep in mind is that when separating the models like this, each model will have less data to train with. In this case, two models are trained for each stage meaning that the training data per model is halved in comparison to the models in Section 5.2.

**One model per cell**

When training one model per individual cell, the performance is worse for both stages, as seen in Figure 5.11, Tables 5.9 and 5.10. However, it is important to remember that the training data for each model is significantly reduced with such an approach. When training one general model for all cells, 60 seconds worth of data from 733 UEs is used. On the other hand, when training one model per cell, the data per model is reduced by approximately a factor of the number of cells in the network, which is 84. It is likely that the lack of training data per model is the cause of the drop in performance, because in theory and intuitively the performance should be equal or better than the general model provided that they are given the same amount of data.

### 6.3 Future work

One aspect that is not fully considered in this thesis is the choices to make after a drone has been detected. Choices have to be made whether to deny service to the drone, not take any action at all, or to find some middle-ground to accommodate the drone by adapting the network. Drones in no-fly zones should probably be denied any service, whereas low-flying drones flying legally might not require any action. In Section 6.4, some examples are presented to mitigate the interference problems often caused by drones.

The final classifications in the second stage are based on an average of previous probabilities output by the model. However, as UEs move around in the network, different eNBs are responsible for calculating these probabilities assuming that the models are run on the serving eNB of a UE. Therefore, the probabilities has to be transmitted to the entity responsible for the final classification, whether it be an eNB or another central entity such as the MME. This causes additional overhead in the network and is something that has be taken into account should a system like this be deployed.

The predictions are based on periodic measurement reports sent by UEs every 40 ms. In reality all of that data might not always be available without explicitly asking for it.

### 6.4 The work in a wider context

From the network point of view, the main problem caused by drones is the interference. Some solutions to how the interference can be mitigated is presented in [14]. One suggestion to reduce the downlink interference is to equip drones with directional antennas rather than omnidirectional antennas. Using directional antennas would decrease the interference power from many angles. Simulations of such a scenario have shown to improve packet throughput and decrease packet loss at the drone. Efforts have been made to mitigate the uplink interference as well. One approach is to tune network parameters differently for drones and terrestrial users. It has been shown that configuring drones with a different $P_o$ parameter can improve uplink throughput for terrestrial UEs at the cost of worse throughput for drone UEs [14].

---

1. MME: Mobility Management Entity.
2. $P_o$ is a parameter related to the output power levels of transmitters.
Looking back at the problem formulation, there are four research questions to be answered in this thesis.

1. Which radio measurements can be used to identify drones in an LTE network?
2. How can machine learning methods be utilized to identify drones in an LTE network?
3. How does classifying the data in two stages affect the computational and memory requirements?
4. Can a model be generalized to be used for multiple cells in a network while maintaining high accuracy?

Regarding the first question about which radio measurements can be used to identify drones, some conclusions can be made. Out of the different measurements gathered through the simulations, RSSI and RSRP are the best metrics to use to identify drones, as seen in Figure 4.3. Additionally, since the data is a time-series, features based on measurements over periods of time can be extracted to give the models more information. Such types of features generally have a larger impact on the classification than other momentary metrics.

Moving on to the machine learning, three different types of supervised learning models have been used: boosted trees, random forests, and decision trees. For the first stage, a simple decision tree model is outperformed by both the ensemble learning models, showing that the problem requires models that are a bit more complex. However, for the second stage there is virtually no difference in performance between the different models, showing that a simple decision tree is adequate. When comparing this cascading-style two-stage classifier to a single-stage classifier from an earlier study, it shows that it has potential to have better classification accuracy while being less computation-heavy. The reason why the two-stage model can be more effective is that roughly half of the UEs can be classified with a quick classifier using measurements from one single point in time. The second stage uses data from a period of time and therefore requires more resources. That period of time is set to 60 seconds in the work done in this thesis. However, depending on the model, that value can likely be set closer to 20 seconds since the accuracy does not improve much after that point.

Efforts have been made to figure out how general the models can be without sacrificing accuracy. This has been done by trying out two different strategies of separating the training
of the models. One approach is to train one model per cell type which are divided into macro and micro cells. The performance shows to be similar to having one general model for the whole network. It can be argued to be better or worse depending on the metric, since the classification ratio is improved in stage one but there are errors that are not present in the general model. The other approach is to train one model per individual cell. In this case the performance got worse for both stages. However, segmenting the models like this means that the data is also split up on a per-cell basis. Since there are 84 cells in the network, the data per model is reduced by a factor of 84, which might result in too little data to train reliable models. This is a likely cause of the drop in performance. For the specific network simulated in this thesis, one general model works well for all cells. However, it is hard to tell what would happen if the network is expanded and it is something left for future work on this topic.


