Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Brage Wichstrøm

Predicting Sheep Predation Events Through the Use of Machine Learning Models Trained on Tracking Necklace Data

Master's thesis in Informatics
Supervisor: Svein-Olaf Hvasshovd
July 2023

NTNU
Norwegian University of
Science and Technology

Brage Wichstrøm

Predicting Sheep Predation Events Through the Use of Machine Learning Models Trained on Tracking Necklace Data

Master's thesis in Informatics
Supervisor: Svein-Olaf Hvasshovd
July 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

Each summer, millions of sheep are let out on pasture in Norway to graze, and thousands will be killed due to predation, incurring millions of kroner in economic loss. Lately, there has been a trend of equipping the sheep with tracking necklaces and reporting the sheep's position on pasture through satellite systems to increase farmers' insight into the location and movement of their sheep. Recently, some research has been conducted into using this data to detect anomalies in sheep behavior due to predators.

In this master's thesis, I have investigated a novel method of combining this necklace data with reports of sheep predation in the tracked sheep grazing area to train a set of machine learning models to make predictions if a sheep is close to a predation attack. I also compare the results from two implementations of the same base algorithm to see if the more advanced implementation provides better predictive accuracy.

Evaluation of the models' performance reveals a nontrivial ability to predict whether a sheep is close to a predation event. The XGBoost model achieved the highest F1-score (0.734), and the Random Forest model achieved the highest recall (0.905), a key predation detection metric for this dataset. Thus, it is shown that for this problem, both models can be used, with the model choice being driven by the metrics wished to be optimized for and the type of tuning to be done. Nonetheless, further research is warranted with more precise sheep and predator data to extrapolate these findings beyond the scope of this thesis.

Sammendrag

Hver sommer blir millioner av sauer sluppet på beite, og av disse blir tusener drept av rovdyr som leder til millioner av kroner i økonomiske tap. I det siste har det oppstått en trend med å utstyre sauene med halskjeder som sporer de mens de er ute på beite ved hjelp av satellitter for å øke bøndenes innsikt inn i sausenes lokasjon og bevegelser. I det siste har det også blitt utført forskning på denne dataen for å detektere avvik i sauenes adferd.

I denne masteroppgaven har jeg undersøkt en metode for å detektere avvik i sauens adferd ved hjelp av lokasjonsdata samlet fra sau på beite sammen med innrapporteringer av sau som har blitt tatt av rovdyr, dette for å gjøre prediksjoner om en sau er nært et rovdyrangrep. Jeg sammenligner også to modeller basert på samme type algoritme for å se om en mer avansert implementasjon av denne gir bedre prediktiv nøyaktighet.

En evaluering av modellenes prestasjon viser en ikke triviell evne til å prediktere om en sau er nær et rovdyrangrep. En XGBoost modell oppnådde den høyeste F1-poengsummen (0.734) og Random Forest modellen oppnådde høyest tilbakekall (0.905), som er en nøkkelstatistikk for dette datasettet. Av dette ble det vist at begge modellene passer til å løse dette problemet, og det spesifikke modellvalget vil bli drevet av hvilken verdi det ønskes å optimalisere for. Det er dog behov for videre forskning på temaet hvor det tas i bruk mer nøyaktig data om både sauens bevegelser og rovdyrangrep før man kan trekke konklusjonene i denne oppgaven til situasjoner utenfor det som er beskrevet her.

Acknowledgements

First of all, I want to thank my supervisor Svein-Olaf Hvasshovd for his guidance and advice while working on this thesis. I also wish to thank the team at NIBIO, especially Lise Grøve, for providing the sheep tracking data used for this thesis.

Finally, I would like to thank my family and especially my father for the unending support and guidance provided to me while working on this thesis and in general during my studies.

Brage Wichstrøm
Trondheim, July 2023

Table of Contents

List of Figures

Acronyms

GNSS: Global Navigation Satellite System
GPS: Global Positioning System
EDA: Exploratory data analysis
KNN: k-nearest neighbors
DBSCAN: Density-based spatial clustering of applications with noise
XGBoost: eXtreme Gradient Boosting
API: Application Programming Interface
MET: The Norwegian Meteorological Institute

1 Introduction

1.1 Motivation

Each spring in Norway, around 2 million sheep are let out to graze in pastures over the summer. While on pasture, they will range freely on a stretch of land for a longer period of time, eating, living, and sleeping, all in that area with limited to no supervision by humans. This type of pastoral farming is one of the oldest forms of farming and has been practiced for thousands of years in Norway. [1] After the summer months, the sheep are gathered from pasture, and several of them are slaughtered for meat, which is, in turn, sold. Of the around 2 million sheep let out to pasture, 1,1 million were butchered in 2022, providing 23,400 tons of edible meat[2]. However, one of the major challenges in sheep farming is the loss of sheep while on pasture. As sheep are expensive to raise, and farmers do not get a return on investment unless they make it to slaughtering, any sheep lost before this will incur a loss for the farmer.

One of the major causes of sheep being lost is predators, which prey upon the livestock while they are out on pasture, with some studies suggesting predation may be behind up to 60% of sheep loss. However, the number will vary from year to year [3]. To offset some of the cost associated with loss due to predation, the government has a program to compensate farmers for sheep lost to predation, which in 2022 paid 45 million NOK in compensation to farmers, and compensation paid totaling almost 1 billion since 2006. This compensation is also paid for losses above what is seen as normal due to predation, with there being 26.619 applications for compensation but only 16.106 being paid out; the rest were categorized as expected losses under normal circumstances due to grazing and not covered by the incentive [4]. From this, it is clear that predation of sheep while on pasture has a large economic impact on farmers and the government.

Farmers have started equipping them with necklace trackers that transfer the sheep's location periodically to better track the livestock while on pasture. This tracking allows the farmers to easily see where their flock is located and aids in searching for them when they are gathered from pasture in the fall.

Lately, there has also been an interest in seeing if it is possible to use these trackers to detect anomalies in the behaviors of the sheep. It is believed that if it is possible to detect if a sheep is in trouble through the use of the data from these tracking necklaces, then some of the losses of sheep while on pasture could be detected before happening and thus avoided, which in turn will benefit both the farmers and government.

1.2 Project Description

As predation poses a great challenge, this thesis will investigate an approach to this problem through the combination of the already existing practice of using necklace trackers with a variety of advanced machine learning algorithms to make predictions regarding sheep predation.

The project of this thesis will revolve around the use of location data gathered from the tracking necklaces of sheep while on pasture to detect instances of predation. The Norwegian Institute of Bioeconomy has made available a dataset containing historical location data from a set a herd of sheep belonging to a farmer in Trønderlag, Norway, from 2015 to 2021. This tracker was spread out among a select few sheep worn for the grazing season for that year, periodically reporting the location of that sheep. Another dataset was sourced from Rovdata, who developed the Norwegian Large Predator Monitoring Program. This dataset contained reported instances of sheep predation for the same general area in which the sheep were on pasture.

This thesis will combine these two datasets to create a sheep location dataset labeled with whether the sheep was close to predation when their location was reported. This dataset will further be provided with other external sources of information, such as altitude and temperature, to give a more comprehensive insight into the circumstances of each data point. The dataset will then be used to train supervised machine-learning models. As sheep have a typical set of reactions to predators, which will deviate from normal behavior patterns, it is believed that these models will be able to detect these anomalies in the sheep's behavior when close to a predator attack and thus be able to classify if a sheep at a point is close to a predator attack.

There has also been a recent rise in the popularity of more advanced versions of classical machine learning algorithms, such as gradient-boosting tree algorithms. As such, I will also be using two versions of the same machine learning algorithm family to evaluate if there is any worthwhile improvement between the more basic and advanced implementations of such models.

Suppose it is possible to create a model capable of identifying abnormal sheep behavior due to predation attacks. In that case, this can, in turn, be used to develop an early warning system that can help prevent predation, bearing the prospect of substantial benefits to all interested parties, both the government and farmers.

1.3 Research Questions

From the general project description above, these two research questions were formulated.

Research Question 1:

Can sheep's movement patterns, derived from tracking necklace data, be utilized to predict if a sheep is in the proximity of a predator attack effectively?

Research Question 2:

Compared to conventional tree algorithms, is there a significant improvement in the prediction accuracy of sheep predation events when using gradient-boosting tree algorithms?

2 Theory

2.1 Sheep

Sheep (Ovis aries) are herbivorous mammals that are present in most parts of the world. In Norway, sheep have been a vital part of society for a long time, providing food and wool to the population. Each year it is estimated that 2 million sheep are let out on pasture to graze all over Norway, being let out in early spring and retrieved in late summer or fall. Although grazing is a good way to feed the sheep using the natural resources that grow in the landscape, it also poses some dangers to the sheep through the environment, terrain, and other wildlife that may prey upon them. [1]

2.1.1 Herd behavior and communication

The behavior of sheep on pasture will vary somewhat depending on the breed, but most all breeds graze in a herd. The reasons for forming flocks may be attributed to multiple factors, mainly resource utilization and reducing predation risk. As sheep stand a significant risk of being attacked by predators, they naturally form flocks, which has been shown to increase the survivability of each individual. [5] In a flock, sheep may share the task of scanning for predators and grazing between them, giving each sheep more time to forage, decreasing the time needed for foraging and the risk of undetected predator attacks.

One of the main ways sheep communicate is through vision. As sheep possess good vision with excellent depth and movement perception, they can communicate using movement and postures. When grazing, sheep will strive to maintain other individuals of their flock in sight, allowing for communication between the individuals; when one sheep assumes an alert posture, the other sheep in the flock will stop grazing and turn towards the direction of the alert. Movement, as mentioned, is also a form of communication used by sheep to discern the intentions of other sheep. One example of this behavior is when one sheep senses danger and starts to run away. The others in the flock will follow suit and flee, allowing for communication of the danger to the entire group. [6]

Home range Each flock of sheep has an area where they tend to rest, graze, and spend most of their time while on pasture, known as the home range. The sheep will naturally restrict their movement to this range, and they will not tend to defend themselves from outsiders and predators. [7] These home ranges will have varying sizes and characteristics depending on multiple factors such as the breed of sheep, resource availability, terrain, and size of the flock. As one of the main reasons for being on pasture is to graze, the

availability of nutrition to graze on is one of the main driving factors of home range size. The sheep will increase their home range size if suitable vegetation is scarce to cover their nutritional requirements. The size of a herd amplifies this, as larger groups will need more nutrition, and this results in a larger home range. [8] Multiple flocks of sheep can also have overlapping home ranges depending on the quality of vegetation and the number of herds in a given area. Sheep also tend to stay and graze in the same home ranges year after year, which, while comforting for the sheep, may pose problems for the farmers when attempting to relocate the flock to different pastures. There have been instances where sheep have been moved to a pasture only to migrate back to the old pasture they see as home. [9]

Diurnal Routines Sheep have a general dinural pattern when it comes to grazing. There are commonly two peaks during the day, one earlier in the morning, not long after dawn, and one later in the afternoon. [8] Although the times near dawn and late afternoon are the most active, it has been shown that they also continually graze in smaller bouts during the day. [10] During this time, they seek areas of high-quality vegetation to forage on. The rest of the time during the day is either spent resting or ruminating. When evening comes, the sheep retreat to an area to sleep. This area is usually further up in the terrain, where they feel more protected and less at risk of predators. [9] This is the most common behavior of sheep foraging, but they may deviate from this behavior depending on environmental factors. The temperature has been shown to play a role in this. If temperatures are unusually high, they may rather choose to rest than graze; if temperatures are low, they may choose to graze during a rest period to increase energy. [8]

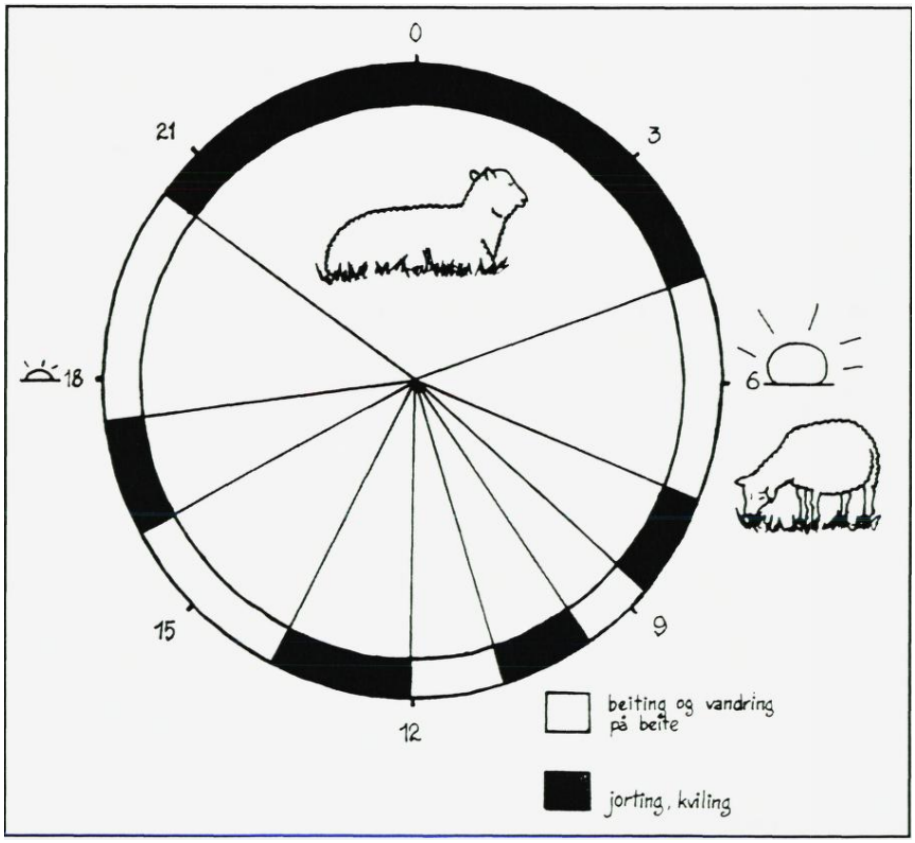


Figure 1: Graphics showing the diurnal rhythm of sheep

Source: [10]

2.1.2 Behavior during predator attack

Sheep behavior during a predator attack will vary from breed to breed and flock to flock, but most sheep use some common techniques. One of these is forming a tight cluster to face a threat to increase the chances of survival. Forming a tight unison cluster to face a predator makes it harder to single out any single sheep, increasing each chance of survival. [11]

If the clustering fails to deter a detected predator, or the sheep elect not to flock, they will attempt to flee. Sheep usually have a distance between them and a possible predator that, if breached, they will elect to flee, known as the fleeing distance. [12] With a top speed of 45km/h, they may cover a large amount of ground quickly; given that there is enough distance between them and the predator at the moment of fleeing, they may be able to outrun it. They may either elect to flee in a group or spread out to flee individually. They will also tend to flee toward areas they perceive as safe; examples of this are areas of high vegetation, high terrain, or places of human settlement. During their escape, they may also implement techniques to make pursuit more difficult, including a sudden change of

direction and speed. [13] The tactics employed may vary from breed to breed and flock to flock, showing that smaller and lighter breeds have a stronger reaction to predators, taking a longer time to return to a normal state and fleeing further with higher deviations in normal behavior. [14] As sheep have a strong memory and ability to learn, the chances of survival and techniques used will vary and increase as they age and gain experience. They may also learn behaviors from other sheep; thus, young lambs may learn how to behave during a predator attack from older sheep in the flock without having survived one.

2.1.3 Tracking sheep

Throughout time there have been multiple ways of tracking and monitoring sheep while on pasture. Shepherds have been common since the early days of husbandry to control sheep and keep predators away. Another way to keep track of sheep used to be for the farmers to periodically travel out to pastures where the sheep are to count them and see that everything is fine, but this is only feasible with smaller and more localized herds. There have also been proposals to cooperate with tourists and cabin owners who frequent nature and have them report if they see any sick or lost sheep. [9]

For a long time, these were some of the only ways of livestock management available, but with technological advancements, new ways of tracking and monitoring sheep have been introduced. One early technological advancement was using radio signalers attached to sheep to triangulate their position; although efficient, the labor and need for specialized equipment proved challenging. Other problems often encountered were selective availability errors, where terrain and environmental factors degraded signals so that accurate pinpointing of senders was a major problem. There has lately been research into how one may alleviate the need for manual labor and the cost of equipment. A study into using relatively cost-efficient UAVs from NTNU has shown promising results. [15] Closely related to this is the use of IoT devices for tracking; as Norway has an increasingly robust IoT infrastructure, the use of IoT devices for tracking and monitoring will become more and more reliable and feasible as a way of tracking sheep while on pasture without the need for specialized equipment or drones. [16]

One of the most used trackers today is Global Navigation Satellite System (GNSS) necklaces, also known as GPS necklaces about the commonly used American GNSS system, GPS. These trackers consist of a GNSS receiver, microcontroller, power source (typically battery but sometimes solar), and often a transmission module. The general flow of the system is that the GNSS components receive their position from the GNSS system periodically and then transmit it to a remote server, where it is stored so the farmer may

access it through a preferred interface. This data may also be integrated into other systems used for monitoring, detecting, and analyzing the sheep while on pasture, one of these possibilities being movement analyses to detect and predict abnormal behavior and predator attacks. Overall these trackers provide many benefits to the farmer; they reduce the need for manual oversight of sheep, which is especially good for large herds or flocks in difficult or spread-out terrain. They make it easier to locate sheep when they are to be brought back from pasture, reducing the workforce need and thus the cost of this process.

The ability to locate sheep also helps farmers quickly find any attacked sheep, either allowing them to be saved or making it easier to prove that they are lost to predation, thus easing the process of receiving government compensation, as outlined below in the loss section. However, GNSS trackers have some drawbacks; first, they need a higher initial investment, with common necklaces currently ranging between 2,000 and 3,000 NOK. There is also the cost of use, as satellites are not free. A farmer will often have to pay per message sent using this system. Thus, covering a sufficiently large enough flock with trackers may be challenging if one cannot access these funds. Some farmers may also perceive the need for technical expertise as a barrier to entry. As some farmers are not knowledgeable or comfortable with new technology and prefer traditional methods, it may be hard for them to change how they herd their sheep and adapt to new technologies and changes, such as GNSS necklaces. [17]

2.1.4 Loss

When out on pasture, it is expected that there will be losses amongst the sheep. Diseases are one of the common reasons for the loss, having been found to represent 20% of losses [3]. The home range's environment may also contribute to the loss, as a more challenging home range will lead to more accidents. For example, sheep getting stuck in bogs, drowning in rivers, or falling off cliffs are all accidental causes contributing to sheep's yearly loss. [9] Disease may also increase this number as a contributing factor. One study from the Norwegian Institute of Biology noted some cases where sheep had died from accidents, such as getting stuck in a bog and not having the strength to free themselves when they usually would have due to being sick. [3] However, the most common factor of all sheep loss while on pasture is due to predation. The same study found that up to 60% of the loss was incurred due to predation, although it noted that there was liekly a high number of dark figures as many sheep are never found or found too late to determine the cause of death.

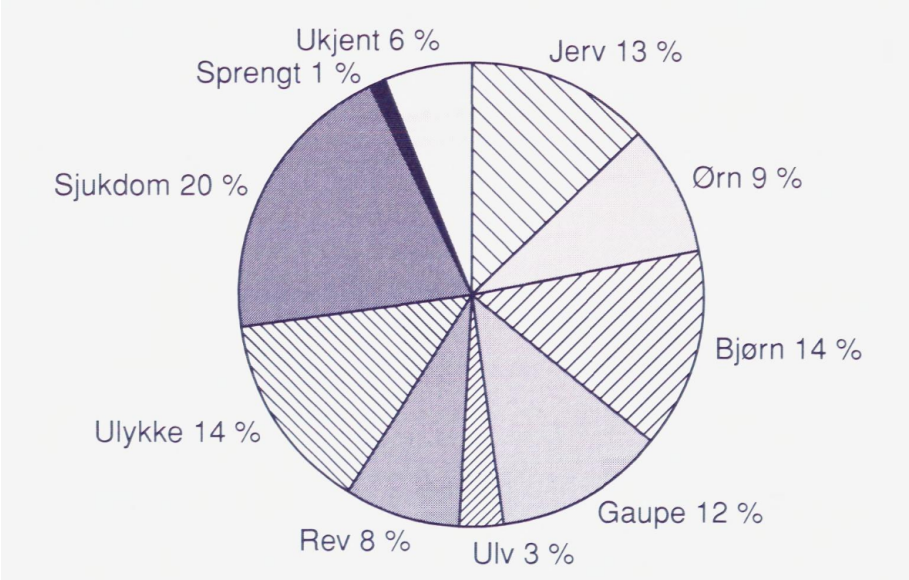


Figure 2: Chart showing the causes for sheep being lost while on pasture in 1988, note that numbers vary each year

Source: [9]

2.1.5 Predators in Norway

Norway has a varied range of predators that pose a threat to sheep on pasture. As mentioned, predation is one of the main threats posed to a flock, and while sheep have several techniques to attempt to avoid predation, they are still at significant risk. In 2022 the Norwegian Environment Agency reported that 16.106 cases of compensation were paid out due to sheep being predated, with a total of 26.619 applications for payment. The main predators present in Norway that pose a risk to sheep are the following:

- **Wolverine:** Of all the predators in Norway, the wolverine is the one that kills the most sheep; according to examinations by the Norwegian Environment Agency, they were behind close to 40% of identified predations in 2022, as can be viewed in the figure 3. We can see that there has been a sharp increase in the share of predations due to wolverines over time, as in the older distribution seen in the chart above 2, the wolverine was not as dominant. It is a smaller animal, between 10kg to 20kg, growing up to 85cm long. They possess a great sense of smell, sight, and hearing; this, paired with their small build, allows them to effectively stalk prey using vegetation as cover to get close and then launch a sudden ambush attack. Wolverines also tend to split up the corpse of prey and cache parts distributed around the terrain; as such, they may hunt multiple sheep and cache parts of the cadavers living off these stores for up to half a year. [18]

- **Lynx:** After wolverines, the lynx is the most common predator of sheep, about 25% of all known causes of predation [4]. With an average weight of 23kg and a length of up to 1,2m, this is another smaller predator in the Norwegian fauna. The lynx is another stalker predator, sneaking close to its prey before pouncing on it to kill. It, however, does not have great endurance; as such, it is rare that one chases down its prey and prefers to end the hunt at a distance of 20 to 30 meters. Unlike the wolverine, it does not cache any food preferring to eat it raw, usually staying by a cadaver until it has completely devoured it. While its main prey is reindeer, it also hunts sheep during the grazing period in the summer. [19]

- **Golden Eagle:** In third place among the most prolific sheep hunters in Norway is the Golden Eagle. However, the number of sheep killed by golden eagles will vary depending on the current population each year. A large bird of prey with an average of 5 kilos and a wingspan of 2 meters, it is the second largest eagle in Norway, the sea eagle being the largest. They are hunters who prey on many animals, including mammals such as sheep. While they may prey on adult sheep from time to time, they prefer to hunt smaller lambs earlier in the grazing season. Their hunting method consists of locking their prey from the air, quickly diving, and attempting to immobilize their target using their talons and beak. They may either feed in place or attempt to carry the caracs to a more secluded place. [20] In Norway, the region of Trønderlag is where most labs are killed each year by golden eagles as this is the place with the highest density of birds and a large sheep and lamb population. [21]

- **Brown bear:** Bears are coming close to the eagles in the cause of sheep losses, causing 9.6% of known predations in 2022 compared to the golden eagles at 11.9%, although, as mentioned, this will vary from year to year depending on population sizes. A large animal that may weigh upwards of 300kg, it is a herbivore receiving most of its energy from vegetation such as berries. Although mainly eating vegetation, it will also eat animals if the opportunity presents itself, thus preying on sheep during the summer if possible. The bear will use its superior sense of smell to locate its prey and then stalk it through vegetation and terrain before swiftly striking, using its superior strength to kill and consume the sheep in place. [22]

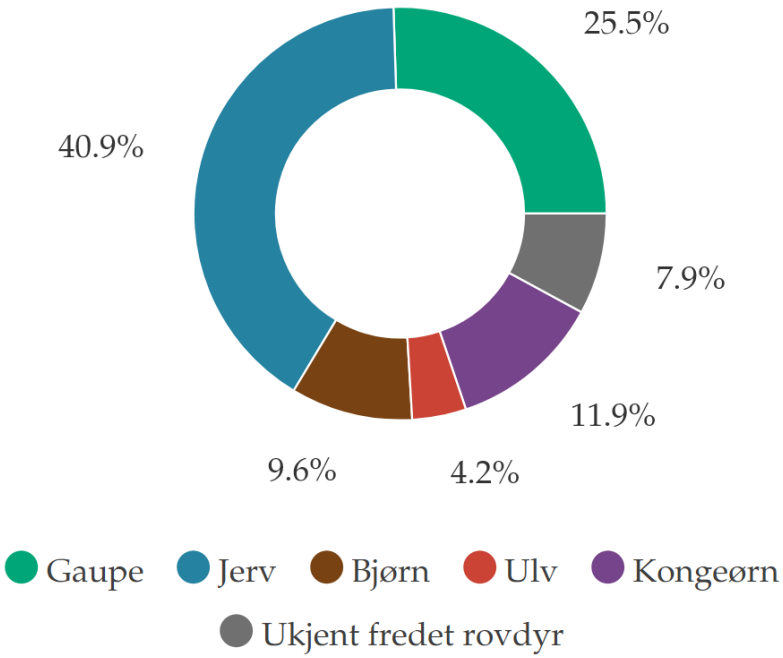


Figure 3: Chart showing the distribution of cause of death by predation in cases where compensation was paid out, in the percentage of how many sheep were killed by a given type of predator in 2022

Source: [4]

2.1.6 Consequences of predators attacks

Sheep farming is an important part of Norway's agriculture and economy. Thus, the loss of sheep is problematic for the nation and the farmers. As previously discussed, one of the main reasons for the loss of sheep is predation, which may be attributed to up to 60% of all sheep lost while on pasture. To mitigate the impact loss of sheep due to predation has on farmers, the government will compensate farmers for sheep that can be proven to have been lost to predators or likely lost to predators, as outlined in the "Regulation on wild game compensation for domestic animals." [23] This can amount to a substation sum; in 2022, the total compensation paid to farmers by the government was 45 million NOK, and from 2006 to 2022, the total amount paid has almost reached 1 billion NOK. [4] Predation also carries other consequences than predation; another notable aspect is the psychological impact on farmers who loos a large number of their sheep and the work put into them. Another trouble with predation is the growing social conflict between those with differing views on wildlife conservation and sheep farming. As more livestock is predated upon the farmers, conservationists and politicians have a harder time reaching an agreement in wildlife conservation politics, increasing social conflict and animosity

between the groups. [24]

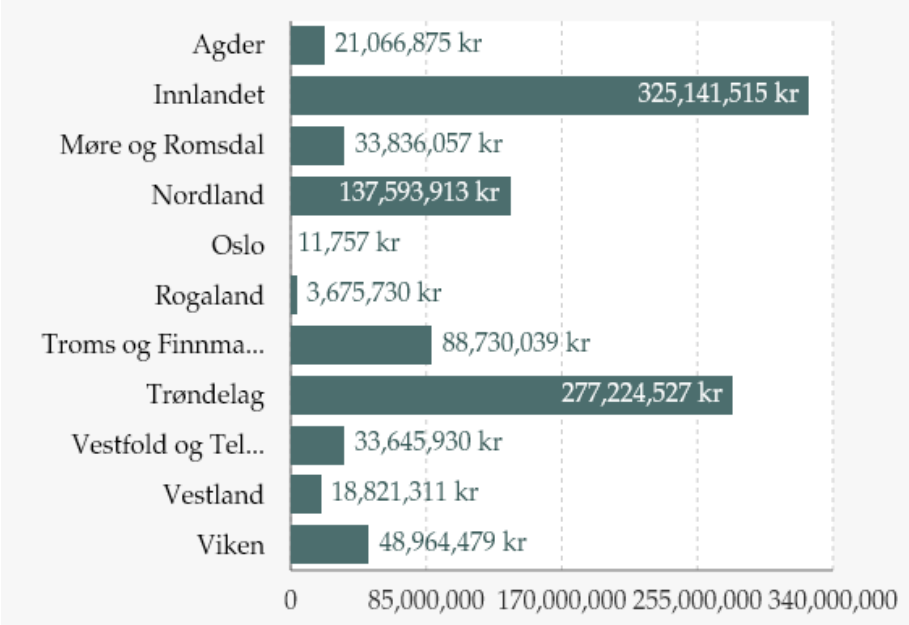


Figure 4: Graph showing the total compensation paid to farmers for sheep predation from the year 2006 up to and including 2022 in Norwegian kroner

Source: [4]

2.2 Global Navigation Satellite System

One of the main ways sheep are tracked is through a Global Navigation Satellite System (GNSS). The first GNSS, GPS, was developed in the 1970s by the United States Department of Defense and later made available for use by the public. Later other GNSS initiatives have also been launched by other nations, such as the European Galileo navigation system, Chinese BeiDou system, and Russian GLONASS, to name a few. It can accurately pinpoint a location on Earth by measuring the distance from a device to a set of satellites with known orbits.[25] It is used by, and crucial to, many industries, such as, in the case of this thesis, tracking sheep when they are out on pasture.

GNSS is broken up into three main segments: the space segment, the user segment, and the control segment.

The space segment is part of the system encompassing satellites orbiting Earth. These satellites have a known orbit and are strategically placed so that at any given point, at least three satellites are visible from any point on Earth to ensure coverage. [26]

The control segment is the part of the system that controls and maintains the satellite constellations. It consists of a network of ground stations that tracks and communicates with

the satellites to ensure the system is working as intended and providing needed accuracy. The location and spread of these ground stations will vary depending on which navigation systems are used.

The final segment is the user segment. A user would interact with this part of the system to get a geospatial position. This is done through an item with a compatible GNSS receiver, be it a standalone device, smartphone, vehicle, or tracking necklace, in the case of the data used in this thesis. The device processes signals sent by at least three satellites to calculate the device's position, velocity, and time. [26]

The GNSS functions through a process known as trilateration, which is a process where the satellites send out a signal with their position and current time. The device then receives this from each. It can calculate the distance from each satellite using the time it took for the signal to travel and its locations. Again, it is used to calculate its place on the planet. [25]

The accuracy of the GNSS varies depending on the system, conditions, and interference, but accuracy, often referred to as "User Range Error," is usually somewhere between one to ten meters, and most often on the lower end of that scale, averaging 0,5 meters in 2019. [25]

2.3 Machine Learning

Machine learning is a subfield of artificial intelligence which enables computers to learn from data and improve through explicit programming. The field focuses on the constriction of algorithms that can learn data patterns and make decisions or predictions based on the learned patterns. Machine learning is often split into three main branches, this being unsupervised learning, supervised learning, and reinforcement learning. Some also consider there to be a fourth branch, known as semi-supervised. [27] Machine learning models may use data consisting of multiple variables or features, which make up the input data the model will have available on unseen data. Some algorithms may also consider the labels of that data; this allows them to learn the connection between the input data and their corresponding outputs.

2.3.1 Unsupervised Learning

Unsupervised learning is a machine learning technique used to identify patterns in data without guidance from labeled data, data containing a labeled target feature. Unsupervised learning can uncover hidden structures and relationships in data, allowing for a large

application area, such as clustering and anomaly detection. One of the often-used clustering techniques is DBSCAN, which uses the density of data points to identify clusters in data. [28]

DBSCAN Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a commonly used and efficient clustering algorithm to identify clusters of different shapes and sizes in noisy datasets. The algorithm consists of two primary parameters, the neighborhood radius (ϵ), which defines a distance threshold for considering neighboring points in the same cluster. The second parameter is the minimum points (minPts) needed to qualify as a cluster. The main idea behind DBSCAN, like most other standard clustering algorithms, is to group data points. DBSCAN does this by constructing clusters from data points based on their density and proximity. This approach makes DBSCAN more robust to noise and outliers than many other clustering algorithms, such as k-mean, as it is not a centroid-based algorithm that attempts to form spherical clusters of the same size. [29]

Another advantage of DBSCAN is that it is not needed to pre-define the number of clusters but rather discovers this itself based on the data. The lack of need for user-defined cluster size may help combat over- and underfitting due to a poor choice in the number of clusters. There are, however, some challenges to consider with DBSCAN, as with most machine learning algorithms, it is sensitive to the parameters chosen. The parameters chosen to be used in the classification of clusters may significantly impact the algorithm's results and accuracy. As such, it is important to select optimal parameters to allow the algorithm to discern the clusters correctly. There is also a challenge with the algorithm's performance when used on higher dimensional data, often referred to as the "Curse of dimensionality," the accuracy of the clustering. [30]

The basic steps of the DBSCAN algorithm can be abstracted to these steps: [31]

1. Identify core points by computing neighbors of each point using ϵ and minPts
2. Create clusters by joining neighboring core points into the same cluster
3. For each non-core point:
    3.1 If possible, add to a neighboring core point as a border point
    3.2 If not possible, assign as a noise point

2.3.2 Supervised Learning

Supervised learning is another branch of machine learning where the models are trained on labeled data to make classifications or predictions. A model is given a set of input data consisting of one or more features and a set of labels, or classifications, of each data point. The model then uses this information as a guide to learn the underlying relationship between the input data and the labels to construct a model capable of making predictions on unseen data. This method deviates from the approach of unsupervised learning, which, as previously discussed, does not consider the label of each data point and only focuses on the input features. While a large span of algorithms fit in the branch encompassing supervised learning, one key algorithm is decision trees. [32]

Decision trees Decision trees are a widely used machine learning technique suitable for classification and regression, offering an efficient way to model complex relationships in data handling continuous and categorical variables. Creating a decision tree model involves recursively partitioning the input into regions representing a class. The tree will also consist of nodes, each testing the input values to decide the "descent" path down the tree, finally arriving at a leaf node which is the model's output. One major challenge with decision trees is that they are prone to overfitting. Allowing a tree to grow too deep may lead it to overfit and thus not be able to generalize when used on unseen data. A common mitigation technique to combat this is early stopping, as mentioned in the section on common errors. Another challenge is their instability; slight changes in the training data can result in a major change in the tree's structure, leading to instability in the model and poor performance.

Random Forest While powerful decision trees, as mentioned, suffer from some limitations, such as overfitting and sensitivity to changes in data. To mitigate this, a technique known as ensemble learning was developed, with the most well-known method, bagging, proposed by Leo Breiman [33]. Bagging, or bootstrap aggregating, involves creating multiple decision trees, each trained on a random subset of the training data. Predictions of a random forest model are then made by averaging the output of multiple decision trees. This ensemble technique reduces some overfitting challenges that are present with simple decision trees because not all trees have seen all the data, allowing for overfit predictions to be averaged away, reducing the effect noise and random patterns have on the predictions. Further, using an ensemble will increase the stability of the model, as any small changes in data that may majorly impact a decision tree will have less effect on the entire random forest ensemble due to the small changes being averaged out. Another technique

implemented in the random forest is the implementation of feature randomness. Random forest will select a random selection of features to consider at each node; thereby, the trees in the ensemble will have an increased diversity, which will, in turn, increase the robustness of the model and prevent overfitting. Another benefit of random forest is introducing Out-of-Bag error for estimating model performance. As not all data points in the dataset are used in each tree, the sample left out may be used to estimate the model's performance. Out-of-Bag error allows for efficient model evaluation without removing parts of the data for use in a separate evaluation set. [34]

Gradient boosting Gradient boosting is another ensemble learning technique that forms an ensemble of weaker models, such as decision trees, to form a stronger model. The idea behind gradient boosting is to iteratively improve the model by fitting new trees focusing on the errors made by the previous models. Each new tree is trained to predict the difference between actual target values and the predictions made by the previous iteration, known as residuals. By doing this, the new tree focuses on the instances where the previous model performed poorly and attempts to correct this. Research has shown that a gradient-boosted ensemble often outperforms a random forest ensemble. [35] This is due to the iterative process, which increases its ability to learn complex patterns and relationships. While models such as random forest build trees without considering the errors made by previous iterations, this is not the case for gradient boosting machines, often resulting in a model with superior prediction performance.

XGBoost XGBoost is a state-of-the-art advanced implementation of gradient boosting machines. It has gained significant attention in recent years due to its success in multiple machine-learning competitions. [36] XGBoost has implemented multiple techniques to increase its effectiveness, making it one of the superior predictive ensembles and boosting algorithms. One of the main reasons for its effectiveness is that it has implemented multiple regularization techniques to control complexity and reduce overfitting, which, as mentioned earlier, is a big challenge with many models. It also implements techniques for handling sparse data where either data may be missing or only a few features are relevant. It is also computationally efficient compared to many other gradient-boosting models, reducing cost and time to train. [37]

2.3.3 Feature Engineering

Feature engineering is the process of selecting, changing, and creating features to help a machine learning algorithm achieve optimal performance. Using domain knowledge,

mathematical theory, and intuition, one will wrangle the data so that it is easier for a model to detect an underlying pattern and generalize the given data. [38] Feature engineering may be broken down into multiple techniques and categories, but these are some of the main relevant ones for this thesis:

Feature Transformation Feature transformation is the subset of techniques used to modify and reshape the data to represent patterns better. Using mathematical functions, one can remove some aspects of the data that may pose problems to some algorithms. Examples are non-linear data, skewed distributions, or instances where the scale might be skewed. If these characteristics are present in the data, they may hinder the performance of certain algorithms. One may implement certain mathematical techniques, such as a log transformation to reduce skewness, to address this. Here one performs a natural log transformation on a numerical feature to reduce the impact of extreme values. Although a powerful and often necessary technique, it is also important to be aware not to transform away the characteristics of the data; if done erroneously, one may introduce new issues or distort relationships in the data and decrease accuracy and robustness. [39]

Feature Scaling Feature scaling is another often-used technique in feature engineering. Here one scales and normalizes the data to be within a certain range to prevent a subset of features from dominating due to their highly deviating values. As many algorithms, such as k-nearest neighbors (KNN), are very sensitive to the scale of the input, it is often important to handle any scaling issues in data to avoid a loss in accuracy. In the example of KNN, the algorithm's results may differ depending on whether the input was properly scaled. A common scaling technique is standard scaling, also referred to as z-score. Z-score is a normalization technique where the features are centered around the mean of the values with a standard deviation of one. This technique is often used as a robust scaler as it retains the original shape of the distribution and is less sensitive to outliers than other techniques, as found in the article by Singh and Singh, where they tested the effect of scaling on a large variety of datasets and found that z-score was one of the best performers. [40]

Categorical encoding Categorical feature encoding is a critical part of feature engineering for most machine learning algorithms. As the algorithms are mathematical, they can only handle numerical input but not non-numerical variables such as strings. The challenge with this is that a lot of real-world data is not represented numerically but rather in a categorical way. These variables may also be unordered, such as color, or ordered, such as weekdays or ratings. For a machine learning model to use this data, it must be

transformed into a meaningful numerical representation. Correct numerical encoding is important, and it must be done for most algorithms to use the data and fit the model. However, it also allows a model to interpret the relationship between data, which is especially important for any ordered values; this again increases the accuracy of a given model. Notable techniques include one-hot encoding, where one feature is expanded into multiple binary features, representing the presence or absence of each. Another common technique is ordinal encoding, where ordered variables are converted into numerical values based on their order. Such a conversion allows the algorithm to use the categorical values and extract the ordered relationship between them. However, the danger with this is that one may introduce unwanted distance that negatively affects distance-based algorithms. [41]

Feature selection Feature selection is part of feature engineering, where one aims to identify the most important features in a dataset that will be of use to attain the best predictions of the model. By reducing the number of features and removing the ones that are unimportant, one may mitigate issues such as noise and overfitting and reduce the complexity of the dataset, which will again reduce the cost of fitting a model and boost its ability to generalize. [42] Choosing which if any, features to remove is an essential and often challenging step. It requires an understanding of both the problem, the dataset, and the models one will use. A common technique used for future selection is correlation-based feature selection. Here one calculates the correlation between a given feature and the target variable, and selecting features that correlate with the target value but has a low inter-feature correlation is common. Another way of selecting features is recursive feature elimination, an iterative process where one trains a model, ranks each feature based on its importance, and then removes the weakest-performing feature. The model is then retrained on the new feature set. This process is repeated until the desired result is achieved, be it a certain computational complexity or accuracy of the model. [43] Choosing which methods to use can, as mentioned, be challenging and will often vary depending on the dataset one possesses and which models are chosen.

Feature construction Feature construction is part of feature engineering that encompasses the creation of new features. The steps and techniques discussed until now have mainly focused on transforming or removing/selecting features. On the other hand, this step introduces new features into the dataset, either derived from existing features or by incorporating external sources. Feature construction enhances the representation of the data and improves the model's performance by capturing complex patterns and interactions between features and relationships. One way to approach feature construction is

through manual construction. Here one uses domain knowledge and intuition to transform and combine already existing features, often with the help of domain experts. This technique is an effective approach to feature construction as one leverages the expertise and understanding of the problem and the data one might possess. However, if one possesses a very large or complex dataset, this method may become challenging due to human limitations. Common manual techniques include:

- **Polynomial feature construction:** A technique where one creates new features by combining existing features with mathematical operations such as addition, subtraction, and division, to name a few. An example is dividing the total distance traveled by the time it took to travel, which will give the average speed.
- **Domain knowledge-based construction:** Here, one creates leverages specific domain knowledge and expertise to create new features. Given certain features, a person with domain knowledge may create new features using their expertise. An example is a person with medical knowledge who may create a blood pressure classification feature such as normal, elevated, and so on by using blood pressure measurements.
- **Temporal feature construction :** Here, one creates features using information from temporal data to capture patterns, trends, and relationships better. Using the time of day to create a feature that may indicate that it is either morning, evening, or afternoon may increase the accuracy of an algorithm and prove a useful feature. Splitting or combining dates, times, and weekdays are other ways of creating new features from temporal data that benefit many algorithms. [42]

Another way of constructing new features is through automated feature construction. These are ways of creating new features algorithmically without the same need for human intervention. The advantages of these methods are that they scale well to large datasets with higher complexity, which, as mentioned, can pose challenges to humans. They may also discover unexpected feature combinations the domain expert might not have known of. One drawback of automated feature construction is that it may be computationally expensive and thus increase the time and cost of feature engineering. One method of automated feature creation is through the use of clustering algorithms. By using clustering machine learning models, one may group similar data points into groups that may be of use as a feature for other algorithms. A pair of MIT researchers found the use of clustering and similar machine learning algorithms for feature synthesis to be a very effective way to increase the accuracy of a prediction model without the need for human involvement. [44]

2.3.4 Common sources of error

Overfitting Overfitting is a common issue with many machine learning models that arises when a model fits the data too closely: this often causes the model to learn the noise and random patterns in that specific dataset and not capture the underlying relationship between the input and target, which allows for generalization; this will result in the model performing very well on the dataset but poorly on unseen datasets. The leading causes of overfitting are training a model for too long on a set of sample data or having a too complex dataset given the number of instances in the trainset. [45] To mitigate overfitting, one may implement multiple strategies. Regularization is a set of techniques that limit a model's complexity, limiting its ability to fit the noise or random patterns present. One of the methods for this that were mentioned when discussing decision tree algorithms is early stopping. The idea behind this is to have a validation set that the model is not trained on but used to check the model's performance. The training will be stopped when the model's performance degrades on the validation set. This helps prevent the model from fitting to the noise and random patterns in the training data. [46]

Unbalanced Dataset A challenge that one might encounter when working with a real-life dataset is an unbalanced dataset. An unbalanced dataset refers to a dataset where the distribution of target features is unequal, and one target is too heavily represented; this can lead to several problems when training machine learning models. One such problem is introducing a bias into the model favoring the majority class simply because it appears more frequently in the dataset. Overfitting may also arise in unbalanced datasets simply because the model may start to pick up on noise or random patterns in the majority class due to the lack of other classes. An unbalanced dataset often reduces sensitivity and specificity, two common metrics for calculating a model's accuracy. The model may run the risk of not being able to generalize and, as a consequence, will not properly classify new unseen data points; this will lead to a high number of false negatives, also known as reduced sensitivity, due to not learning the minority class patterns. This may also cause the model to suffer from reduced specificity, which is the proportion of true negatives again all negatives due to a bias toward the majority class. Lastly, one major pitfall is that some common evaluation methods become misleading with an unbalanced dataset. A common metric, such as accuracy, may be misleading. Even if the model misclassifies a large number of the minority class, since the majority class is dominating, it may give a high accuracy score that may mislead. Therfore, using other robust metrics that handle unbalanced datasets better, such as precision, recall, and F1-score, is important. [47]

Data leakage Another common source of error that specifically encompasses the validation and testing of a model is data leakage. Data leakage occurs when information about the validation/test set is unintentionally used during model training or feature engineering. Data leakage will lead to the model having access to information it should not have, as it will not be present in completely unseen data; this may lead to it performing well on the validation and test dataset but badly in real-world applications. There are many causes of data leakage, but a common error is the inappropriate splitting of data into training and validation/test sets. An example of this is the random splitting of time series data into train and validation sets; this may lead to related datapoint that is in the future for the validation set ending up in the training set, giving the model the ability to glimpse into the future which is not realistic in a real-world setting. Another common error is target leakage, where features that are highly correlated with the target variable but will not be available in real-life predictions or do not represent independent information are included. [48]

Not stratifying Stratified sampling is a technique used to ensure that the distribution of classes in the training and validation/test sets is similar to the overall dataset. Stratifying is particularly important when dealing with an unbalanced dataset, as there is an increased risk that the minority class may be underrepresented or absent in one of the sets. Again, this can lead to a model that cannot learn the difference between the classes if absent in the training set or give an unreliable performance hesitation if underrepresented in the validation/test sets. In general, properly stratifying the splitting of the dataset will lead to better generalization of the model and more reliable performance evaluation. [48]

3 Recent research

As new technology emerges and becomes affordable, research into livestock management and analysis is an ever-evolving field of study. It is hence important to scrutinize recent studies using updated technologies, whereas few lessons can be learned from older reports. Although there have been many studies on sheep in the past, recent research has been on cattle. Although the behavior and threats posed towards sheep and cattle differ, the methods for tracking, managing, gathering, and analyzing data are often similar. Thus, the research being done into cattle management while on pasture is similar enough to be relevant to this thesis on sheep. One study released in 2022 looked at accelerometer and GNSS data from cattle to analyze behavior and anomalous events [49]. The authors decided to use data gathered to train machine learning models to gain insight into the data, using two different branches of machine learning for each dataset. They employed a supervised learning algorithm for the accelerometer data, while with the GNSS data, they applied an unsupervised method. They found that using accelerometer data could help classify the animals' activity. However, a more interesting insight gained from this article with respect to this thesis is that GPS data may be used to train a model to identify different animal groups and sudden dispersion, which may be triggered by predator-induced panic.

The article related to a study on cattle discussed above has a gap that I believe can be researched regarding its use of GNSS data, as it is only used with an unsupervised machine learning method and mainly to identify clusters of cattle. This study also does not look at instances of a predator attack but instead used the cattle's activity as a label for their supervised learning methods.

Another paper from Australia, also released in 2022, conducted an experiment with regards to the impact of a predator's presence, wild dogs, on a herd of sheep on pasture [50]. In this study, each sheep in a herd was fitted with a GNSS tracker reporting the sheep's location every 5 minutes for 52 days. Of those 52 days, 26 of the days, a wild dog was present in the sheep home range, which was later removed. The authors then performed some simple statistical analysis on the spatio-temporal data, such as looking at the average movement at a given time and movement changes in percentage. They used this to discern differences in behavior in the sheep between when a predator was present and when it was absent. This statistical analysis specified the speed of movement, daily activity, and the total distance traveled in a day. This analysis found that the sheep had an overall higher activity level when the predator was present and moved further each day. From this, they concluded that the use of GNSS trackers could prove helpful in the detection of predators in the home range of a herd of sheep by analyzing the temporal-spatial

factors gathered from these sensors. Although they also noted that more research had to be done, especially in the event of an actual attack.

The point of improvement that can be made regarding this article discussed in the paragraph above is that while it uses GNSS data as its primary information for analysis, they do not use any machine learning models and instead rely on more standard statistical analysis. They also, as mentioned, do not look into any instances of actual predator attacks and only look at the general behavior when a predator is in the area, while this may be good for some predators there are, as outlined in the section on predators in Norway, many stalking and ambush-based predators. The sheep may not be able to detect these until shortly before an attack and thus having an attack happen will ensure changes in behavior. While the approaches in the discussed studies are good and valid, I hypothesize that the use of machine learning models on the sheep's GNSS tracking data with labeled attack data may be able to further detect and learn the more intricate relations in sheep behaviors, and from this be able to create more robust methods of predation detection.

Thus, using GNSS in conjunction with a supervised machine learning model with labeled attack data could further these two studies and be an interesting field to look further into.

4 Exploratory Data Analysis and Preproccesing

The dataset I received from The Norwegian Institute of Bioeconomy Research contained information gathered from one farmer's flock of sheep while out on pasture. This data was gathered from 2015 up to and including 2021; the data contained information from 270 trackers spread across those six years. Across this timespan and all trackers, the total number of entries gathered is 481810.

4.1 Rought overview

The first part of working with the data after receiving it was the exploratory data analysis (EDA). EDA is generally the first step in any machine learning task and a vital part as it founds the basis for any decisions made in data cleaning, feature selection, and the actual analysis of the data later on.

The first move was to visually inspect a slice of the data to get an intuition of what I had to work with. I could easily examine the data as needed by converting the data files into a data frame and inspecting a slice. A ribbon slice of the data can be viewed in the table 4.1. As it only contains GNSS data, the rows "position_t," "st_x," and "st_y" are the main point is interest, with the other features mainly being metadata containing information to identify the sheep and owner. The feature "position_t" is the date and time when the data was recorded, "st_x" is the longitude, and "st_y" is the latitude of the sheep as reported by the GNSS necklace. Worthy of note is also that "st_x" and "st_y" have been stored as floats and that "position_t" is stored as a string. This is important as, in the analysis, many machine learning models can only handle numerical data; hence, the "position_t" field must be transformed to conform to this requirement. In this step, the dataset was also checked to see if any rows of the data had missing values; luckily, all data was present, so there would not be any need to either remove invalid rows or impute data.

source_id	owner_id	position_t	st_x	st_y
1519000773	32957	2015/05/07 08:43:24.000	8.467802	55.514840
100828	32957	2018/08/14 16:48:28.000	12.006497	63.447347
2012998	32957	2021/01/11 18:09:54.000	11.717185	63.401630

4.2 Preprocessing

Outliers The next part of the analysis and preprocessing was the handling of erroneous data points and outliers in the data, specifically in the spatial part of the data. As the rest of

the data processing and the analysis will largely be based on these points, it is important to weed out any incorrect data points to get the best results. The first thing that was done was to plot the points onto a graph based on their latitude and longitude values.

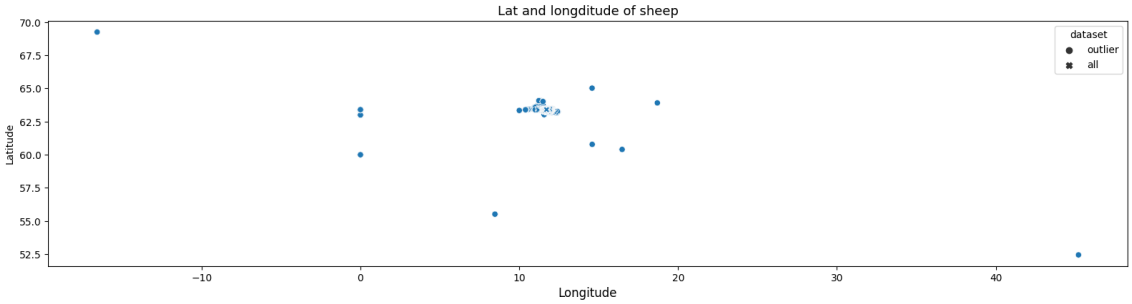


Figure 5: Plot showing the Latitude and Longitude coordinates of all datapoints

As can be seen from Figure 5, there are some major outliers present in the data, some of them being so large that the main bulk of the data points is hard to discern and has combined into mostly one homogenous blob. There were many reasons these outliers may exist in data; most likely, they are caused by "User Range Error" in the GNSS necklaces, or the data may have been corrupted in some way during transfer or storage. While the major outliers are easy to discern due to their large deviations from the rest of the data, as well as the impossibility of a sheep from the Trønderlag region of Norway being present there, such as one of the data points being recorded as in the middle of the Indian Ocean, there is also some outlier that while not obviously an invalid reading at first glance are most likely invalid data points upon further analysis. To better discern the outliers closer to the main bulk of the sheep, I did a rough filtering of any points that could not belong to any of the sheep. This filtering was done by defining a box roughly outside the realistic area and removing any points that fell outside of this, and as such, filtering out any extreme outliers. Then to get more insight into the outliers that were close to the normal range of the data, I plotted the box charts of the Latitude and Longitude, which provided a more easily understandable and clearer way to discern if there were any further suspicious points and what value range they might be in.

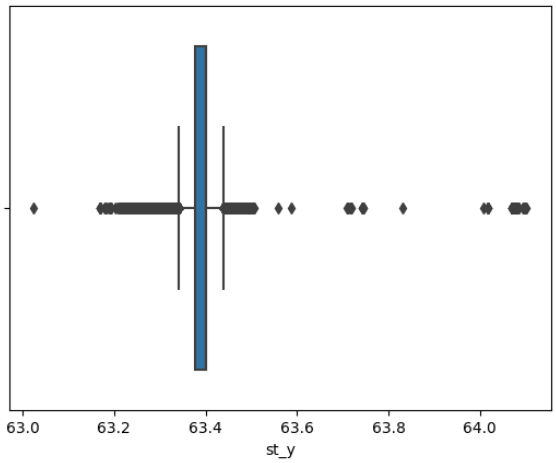


Figure 6: Boxplot showing the distribution of Latitude of recorded points, after initial filtering

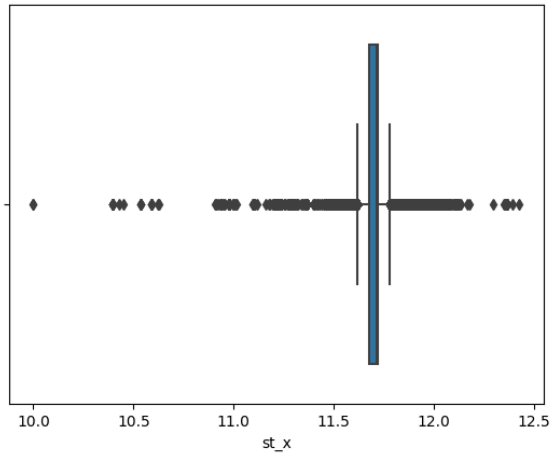


Figure 7: Boxplot showing the distribution of Longitude of recorded points, after initial filtering

As can be seen from each of the box charts after removing some of the most obvious outliers, a few of the points still fall outside the normal range of the data and thus are suspicious. In the boxplot, the two vertical lines on either side of the main blue stripe denote the edge of the interquartile rule, a common way of identifying outliers in a dataset by using the interquartile range multiplied by 1.5. One can see there is a good amount of points that are outside of this range and thus may be classified as outliers, and while using this method of identifying and removing outliers is often used, I chose to use another method with a focus on the Z-score of the data as this method tends to be less aggressive, especially with smaller datasets which this may be considered to be. To select the Z-score to use as cut-off points for the Latitude and, Longitude, I started with one of the most common cut-off points, 3, and worked iteratively from there. The process I used mainly

focused on visual inspection of which points were cut off, along with information such as how many were removed and their location. Through this process, the Z-score cut-offs of 6 for Latitude and 4 for Longitude were chosen; while less aggressive than the usual cut-off points of 3, through the use of visual inspection and z-score distribution, this seemed to be an appropriate limit of removing the largest outliers but not necessarily real anomalies. After completing this step of the preprocessing, the data was now more homogeneous, and there were a few points that largely deviated in their spatial location, although there were some visible outliers left in as they may be of use for later analysis. In the plot in Figure 8, the spatial location of the sheep has again been plotted, but now with the new dataset where the outliers have been removed. As opposed to the original dataset, the shape of the main range of sheep can be discerned.

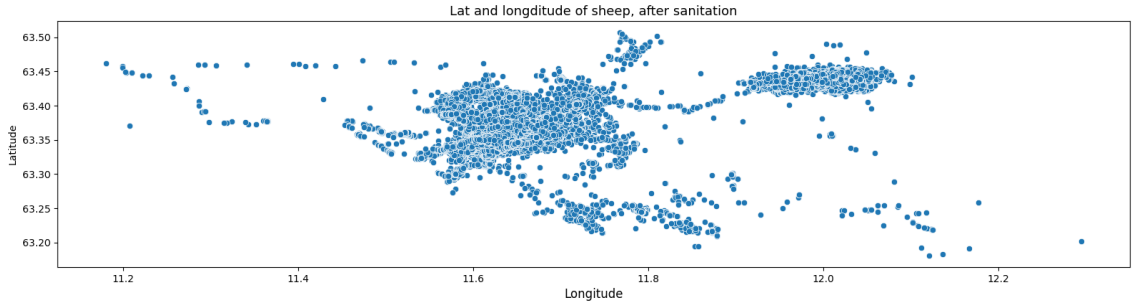


Figure 8: Plot showing the Latitude and Longitude coordinates of all datapoints after outliers have been removed

Data Pruning With the outliers removed, the next step of the preprocessing was further data pruning of undesirable data points for different reasons. The first of which were data points that did not represent the sheep while being out on pasture.

One can decide the general location of sheep into two groups, being on pasture and being on the farm. They are, as discussed, let out onto pasture in spring and gathered in the late summer/autumn to spend the winter on the owner's farm. This thesis is only interested in the sheep while they are out on pasture and not while on the farm; as such, any data points that represent the sheep while on the farm should be pruned away. The reasoning for this is the fact that the data points for the sheep on the farm will not necessarily be relevant to the model. While on the farm, they generally do not move much around and are not influenced by the same factors as while on pasture. Removing these points will help the model focus on the relevant context, sheep on pasture, improving the accuracy of the model. The first step in achieving this was removing the data points on the farm simply by creating a boundary box of coordinates and removing any points that fell into this box. This should remove most of the data points where the sheep have either not been driven out yet or have been gathered from pasture and thus are no longer relevant to the model.

Another choice that was made was to limit the data from July to and including August. This is the general range when sheep have been put out to pasture each year, giving the model a universal range to work while minimizing noise from sheep being either driven out in spring or gathered in autumn, which may introduce noise.

Another choice was to remove any sheep with fewer than 60 entries in the dataset. The reasoning behind this is that all data points will have a sufficient number of entries for each sheep so that they can be tracked over time. Suppose a sheep only has a few entries. In that case, the patterns that may be discerned from this data may not be reliable or representative of the actual movement patterns of that sheep, introducing inaccuracies or noise into the model.

4.3 Analysis

After having completed the preprocessing as discussed in the previous section, the next step was to gain further insight into the data and the attributes.

Spatial and temporal distribution in landscape One of the more interesting features of the data is their actual distribution in the landscape as well as across the time of day. As discussed in the theory section, sheep tend to spend certain times of the day in certain landscapes and then stay in the general area without moving from it without reason. To determine sheep's normal behavior is is vital to gain insight into the movement patterns of the individuals and herds at a given time of day, this to see specifically where the sheep are as, up to now; they have been handled through the use of their coordinates only, with minimal reference to the actual landscape those coordinates represent and so it is hard to interpret where these coordinates actually are in the landscape. As such, the data points were overlayed onto a map representing the home range, placing a dot for datapoint on the map where their corresponding coordinates belong.

Further, to visualize the temporal differences, the dots were color coded to give an idea of the sheep's location at a given time of day. In figure 9, one such mapping can be seen. In this figure, the data is limited to August 2020, as plotting the entire dataset would make it overcrowded and unreadable.

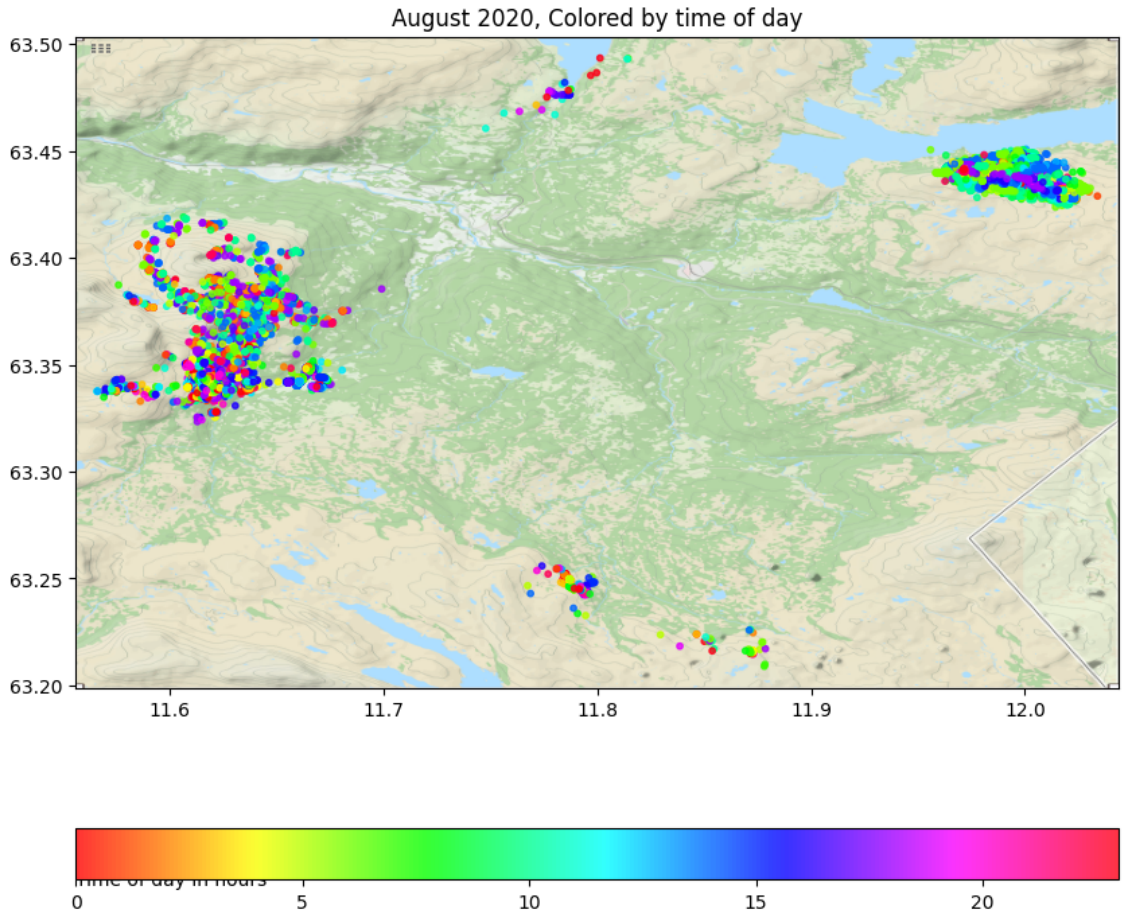


Figure 9: Plot showing the location of each datapoint of the sheep for August 2020, color-coded for the time of day

From this figure, it is possible to gather some intuitions about the data. First of all, there seem to be four main locations where the sheep spend their time, some being closer to a large body of water and some further inland. Another pattern that may be observed from inspecting the groups more closely, along with terrain and time, is that the sheep tend to spend the evening and night in higher terrain and to move to lower terrain during the day, this behavior aligns with the expected behavior outlined in the theory section of this thesis.

To gain further insight into the general movement patterns of the sheep from day to day to create an animation that played through the movement of a selected sheep from day to day. By doing this, some insight was also gained into where the sheep spent time across the entire pasture and if there was any movement to another separate geographical area. This visualization suggested that the sheep did not tend to relocate and rather stayed in a limited geographical area and that the different clusters of sheep as seen in the map are not the same herd that has migrated from season to season but rather multiple herds, all who have generally stayed in their home range.

4.4 Feature generation

Generating herd features using DBSCAN While having visually confirmed that there are distinct groupings of sheep while on pasture and analyzing the movement of a few of these sheep visually to confirm that they stay in their respective home ranges without migrating, it is still of interest to confirm this across all sheep as well as generating a label for which of the herds each of the datapoints belongs. As this would be infeasible to do by hand, the DBSCAN algorithm was implemented to achieve this automatically through cluster analysis. First, the data was split, separating each year, as each year will be distinct herds, and then the latitude and longitude values were extracted for use in the clustering.

For the DBSCAN clustering, the parameters of epsilon and minPts had first to be chosen. For the former parameter, epsilon, the value of 0.045 ended up being chosen. The reason epsilon was set to this value is that as the distances that are used for the DBSCAN algorithm are the latitude and longitude values, and as these are based on real-life distances, the epsilon value should reflect a reasonable distance that a sheep may be away from another while still being regarded as the same herd. For this terrain and environment, an epsilon value of 0.045 will correspond to a real-life distance of 4.989KM and 5.010KM for latitude and longitude, respectively.

The next value that had to be chosen was minPts; for this, the value of 3 was set, as any sheep that is 5KM away from any two other sheep would be suspect. The algorithm was run once each year, with the data used as input only being the data points for that year. By doing this, the problem of two data points being labeled as belonging to the same herd even though the data is from two different years and thus obviously do not belong to the same group in real life. As DBSCAN also reuses cluster labels each year, some post-processing was done to the labels to make them unique for each year. This labeling was done by simply appending the cluster ID, a numerical value starting at 0, to the year that the data point was recorded. Thus, all points belonging to cluster 0 in 2018 would have a cluster ID of 20180, and the same from 2019 would be 20190. While this may be problematic with many algorithms due to the fact that this may introduce problems with both scale and ordinality, as the models used in this thesis will be tree-based, this is not a challenge due to the model's robustness in handling this type of data. These labels were then integrated into the datasets so they may be further used in later analysis.

There was also a cross-check amongst all clusters to verify that no sheep was labeled as belonging to multiple herds; this was to ensure that each cluster represented one distinct herd and that no flocks had been split into multiple clusters. The cross-check, when completed, confirmed that this was the case; all sheep were labeled as belonging to one, and only one, herd. Viewed in this figure 10, the flock for each sheep has been classified

and labeled for the year 2020, with each color corresponding to one unique and distinct flock. This will be used as information in training the model under the feature name "herdID."

It should be noted that if deployed in real life, there would not be any need to run clustering to label the flocks as the farmer deploying the model would already know which sheep belong to which herd.

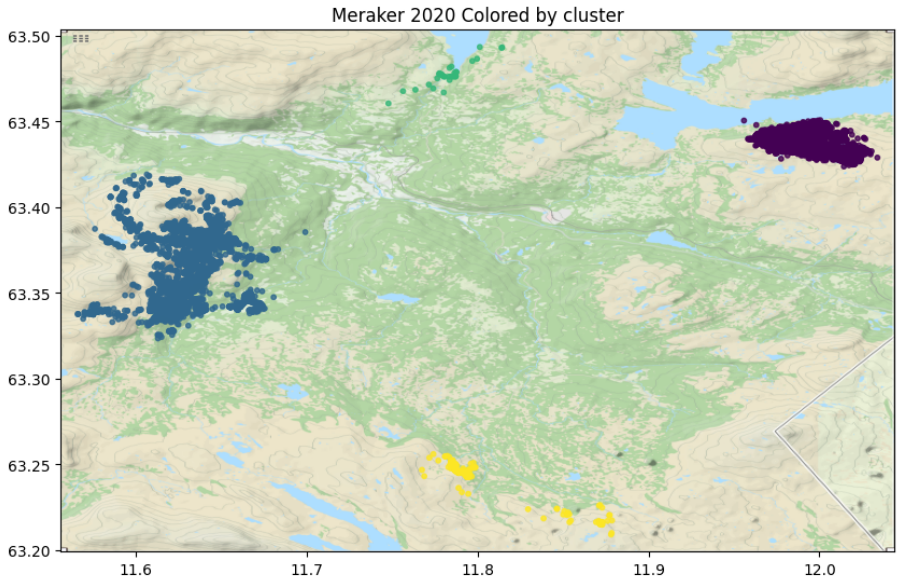


Figure 10: Result of DBSCAN algorithm, each cluster corresponding to a unique flock of sheep for 2020

Generating movement velocity To further aid the model's ability to make accurate predictions, a feature containing the sheep's velocity for each datapoint was generated. As one of the main strategies sheep have for avoiding predation is fleeing from predators, their movement speed might be an important feature for a model. The movement velocity will also provide an insight into the general patterns of the sheep as they should, according to theory, be more active in the morning and afternoon due to migration and grazing. The velocity was generated by comparing the locations of sheep at a data point with that sheep's previous data point and, from this, generating the average velocity in km/h. The distances were calculated using the haversine algorithm with respect to each datapoints latitude and longitude coordinates. At this step, some further thought data pruning was also completed to remove any unuseful or unlikely data points. As sheep can attain a maximum speed of 45km/h when fleeing predators, a value of 55km/h was set as the cutoff point due to a deviation of 10km/h from the theoretical maximum speed being unrealistic, even when accounting for individual differences in sheep.

Generating Altitude and calculating the change in altitude Another feature not included in the dataset which may be important to the models is the altitude of the data points. As was discussed in the theory section, sheep tend to spend their time in different altitudes depending on different times of day and activities, seeking higher altitudes during rest periods and then lower when grazing. Seeking higher altitude is also one of the tactics used to evade predators. To generate a more comprehensive picture of the sheep behavior, I generated the altitude features as well as calculated the speed of change in altitude.

The altitude generation was done using a dataset provided by the Norwegian Mapping Authority, a government authority that provides land surveying and cartography services. They offer an open API endpoint where one may submit a latitude and longitude coordinate pair and get the altitude for the given point. This process was done for the entire dataset in batches of 50 coordinates due to the limitations on the number of coordinates checked per call made to the API. However, to speed up the process, this was multithreaded to reduce the time to complete. A similar process to calculate movement velocity was used to calculate the altitude change rate. For each datapoint for each given sheep, the difference in altitude between that point and the previously recorded datapoint was computed; this was then divided by the difference in seconds multiplied by 3600 to get the change rate in meters per hour and added to the dataset alongside the current altitude of that data point.

To validate the theory about altitude change in relation to time, the average altitude for each hour across all sheep was calculated. This data was then used to plot a graph with the y-axis being the average altitude and the x-axis being the hour; this plot can be seen in this figure 11. As the plot shows, there is a notable difference between the daylight hours and the night; this suggests that the behavior of seeking higher ground during the hours when they sleep seems to hold true for the average sheep.

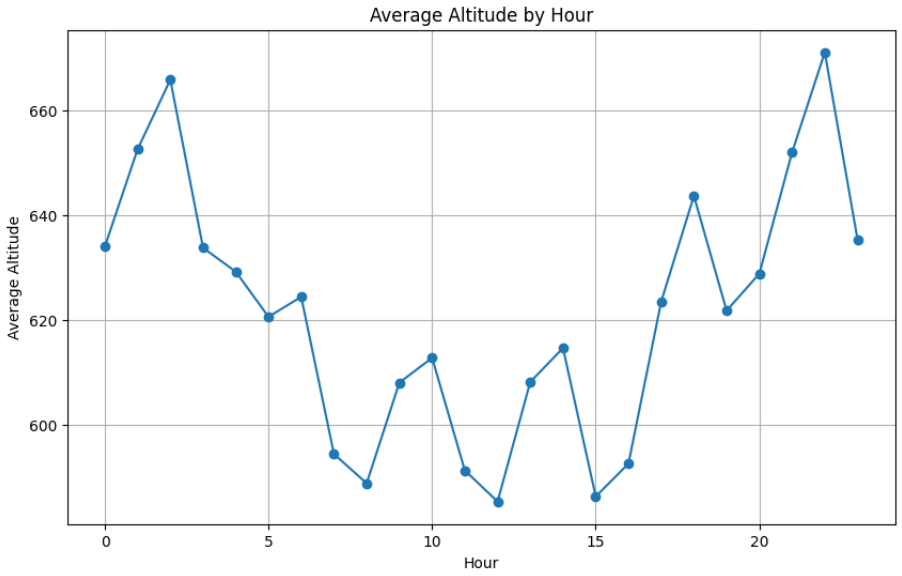


Figure 11: A plot showing the average altitude of all sheep. The y-axis represents altitude, and the x-axis represents hour from 00-23

Generating weather data Another factor that affects the behavior of sheep and may be an element that leads to deviations in normal behavior is temperature, as with too high temperatures, the sheep may choose to stay sedentary and in the shade during the day as opposed to grazing. Unfortunately, the necklace trackers have no temperature tracking built in. Therefore, the temperature data had to be acquired from outside sources, specifically, the Norwegian Meteorological Institute (MET), which has multiple measuring stations spread around Norway which take regular measurements of metrological data such as air temperature. This data includes historical measurements and is made freely available to the public. As such, it is a good source to use with this dataset, particularly since there is a measuring station near the home ranges for the herds. The station used is specifically known as "MERÅKER - VARDETUN" and is located 800 meters west of Meråker. The data collected from this station was used for all the sheep, and while there may be some deviations from the measured value and actual value where the sheep is located due to distance from the station, I made the assumption in this thesis that this is likely to be minimal. The reasoning for this is that air temperature tends not to be so hyperlocalized that there will be sudden large changes over a few kilometers.

The weather data was retrieved using a REST API provided by MET, and the data provided contained information about measurements taken at the top of each hour every day. From this data, the air temperature of each hour as well as the date and time of measurement was extracted and then merged with the sheep dataset, matching the date and hour. Similar to the process of gathering and matching altitude data, there was some challenge with the

time to complete due to the number of data points; as such, this process was multithreaded to reduce the time for completion.

Transforming temporal data One aspect of the data that has to be considered is the temporal aspect, specifically the time of day and its representation. In the dataset, as it was provided, the time of each data point was recorded using the standard 24-hour format. The challenge with this representation is the fact that time is cyclic, but the 24h formatting of time does not represent this. In real life, times of 23:00 and 01:00 the next day are two hours apart due to this cyclic nature, but numerically they are 22 hours apart. This representation is a challenge as machine learning models are based on numerical values and thus will not be able to infer the fact that while being numerically 22 hours apart, they are, in fact, 2 hours apart.

By converting the original 24-hour representation using the cyclic trigonometric functions of sine and cosine, the model is aided in discerning this relationship. Both these functions are needed as each function has two input values that map to the same value after sine transformation; an example is the sine transformation of hour 0 and hour 12, where both map to the value 0, making the two times indistinguishable when viewed only from their sine transformed value. It is, however, possible to mitigate this dual value output problem by implementing the cosine function alongside the sine function. Sine and cosine are both cyclic functions but do not share a mapping structure, and so a number in the original 24-hour format transformed will have a unique cyclic combination of these two transformations so that it is possible to both possible to model the times cyclic and unique nature.

With a combination of the newly transformed temporal features and the velocity that was generated, it is now possible to plot the activity levels of the herd of sheep at each given hour. By plotting the sine value and cos value for each datapoint alongside the average velocity, we generate a 3D plot of the diurnal activity of the sheep; this plot can be seen in figure 12. In this plot, the top, as denoted by a red x, represents the hour of 00, and the bottom, represented by the green X, represents a time of 12H. From this plot, we can see that the activity levels of the sheep tend to be higher in the early morning and late afternoon, as well as generally lowest during the night when sheep sleep. Worthy of note is that there also seem to be some points that deviate highly, and the activity level in an hour was much higher than the normal range.

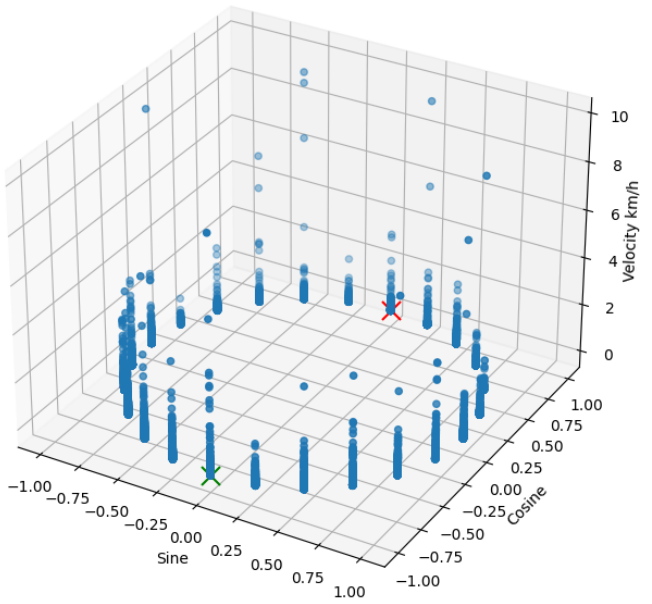


Figure 12: plot showing the activity levels of sheep compared to the time as sine and cosine transformations

Having completed this process, we have the final features that will be used in the training and evaluation of the models. The table 1 lists these features and their descriptions.

4.5 Generating labels from predator data

As some of the models which will be used in this thesis are supervised models, they will require labeled data. The challenge with this is the fact that the dataset is unlabeled, and there is no data provided concerning sheep that have been predated upon. Due to this fact, this data would have to be gathered from an external source; luckily, there is an institution in Norway known as Rovdata that tracks multiple elements related to predators, including sheep that have been subjected to attacks by predators. Through this institution, it was possible to obtain data concerning sheep that have been predated in the area that the herds considered in this thesis frequented. The dataset had multiple features, but the ones that were most useful for this thesis were the columns concerning the possible date range that the attack could have happened, whether the attack date was uncertain, and the location of the attack.

For the task of generating the labels from this information for each data point, the first step was to decide which date range to use as labels. One of the challenges with the predator attack dataset is that some of the possible attack ranges span a large amount of time due

to uncertainty in the time the attack took place, with some ranges spanning up to a month. The problem with labeling all relevant data points in a large uncertain time range of an attack is that this will likely label many data points that are not close to an attack as being close to one. If the time range the attack could have happened spans an entire month, with the attack happening at the end of that month, it is unlikely that the sheep have noticed a predator near the flock at the beginning of the month.

If we were to label all data points in the relevant area for that month as being close to an attack, many of the labels would likely be inaccurate. The poor labeling of these data points will then likely, in turn, affect the accuracy of the model, as it will be a challenge to learn if a point is close to a predator attack from data where most of the data points labeled as being close to predator attacks are not, in actual, close to a predator attack. Moreover, there is another consideration to be made; the fact is that there were only a few reported predator attacks in the relevant time range of 2015 to 2021. Thus if the criteria for filtering out predator attack ranges are too strict, there will likely not be enough data to train a model on; as such, some uncertainty in the time range will have to be accepted. With this consideration, an initial time span of three days was set as the allowed possible attack window, and any possible attacks that had a time range of about three days where the attack could have happened were not used for labeling data.

The next step was to find sheep that were within the geographical area when the attack happened. As only sheep near an attack are relevant for that reported attack, the distance of a given datapoint for a sheep distance from an attack had to be calculated. The way this distance was computed was again using the haversine formula that was utilized when calculating velocity. However, this time the points used for calculation were the sheep attacks extracted from the predation data provided by Rovdata and a datapoint from the sheep location dataset where the datapoints recorded date and time falls in the time span, or attack window, of the sheep attack. From this, the distance of all sheep data points in the possible time window of a predator attack was calculated to find how far the sheep were away from a predator attack when it happened. Although, as mentioned, the distance has to be limited due to the fact that the dataset encompasses multiple unconnected flocks. As such, a criterion was set that a data point for a given sheep in the attack window of a predator attack had to be within 5km of the attack to be seen as relevant. A distance of 5km corresponds roughly to the distance a sheep usually moves in one day, and consequently, is a good trade-off between not filtering out too many sheep that might be in the range of the attack while also not including too many sheep that are not in range of the attack.

By completing the mentioned steps on all the data points in the predator dataset, we created the labels for each of the sheep data points, with the label being a boolean if a datapoint is within an "attack window," an attack window being in the same temporal and

geographical area as a predator attack as mentioned above. After completing this process, the end result was a class distribution of 110,312 / 8,489, with 110,312 being the number of data points outside an attack window and 8,489 being the number of points in that window. As is apparent, the number of data points outside of the attack window greatly exceeds the number in the window; this is due to two factors. First is the fact that the number of sheep that predators killed was fairly low, which translates to few data points meeting the criteria of being in the attack window. The second reason for the low number of positive classes is the strictness of the attack window criteria with respect to time.

Due to the fact that there were many reported incidents of sheep being predated upon where the time of the attack was uncertain, the filtering out of any predation incident where the possible range exceeded three days led to a nontrivial number of these being excluded. As this led to such an unbalanced dataset with only about 10% of data points belonging to the positive "attack window" class and the rest being negative, the choice was made to create multiple datasets with different criteria for this time range of attacks. The first dataset was generated as described above, where the time range of the "attack window" was set to 3 days, with all other elements remaining the same. Then a new labeled dataset was created using a time span of 7 days for the allowed window. This looser definition allowed for more reported incidents of predation to be connected to specific sheep data points, with the trade-off being the uncertainty of the labeling. This was done again with the attack window criteria loosened to 14 days, which again increased the number of sheep predation incidents that can be used for labeling, with the same trade-off as before but with even more uncertainty.

This uncertainty is something that has to be accounted for when analyzing any results and will be a further topic in the discussion section. After having completed this process, there were three labeled datasets that could be used during analysis; all of them contained a total of 118,801 entries. The dataset with the narrowest criteria for an attack window had, as mentioned, 8,489 "True" labels, indicating that they are in an attack window, with the rest of the labels being false, indicating the opposite. The two other datasets, the 7-day and the 14-day, contained 9,668 and 11,380 "True" labels, respectively. Hence we can see that allowing for more uncertainty in the attack window proved to have up to a 34% increase in "True" labels, though, as mentioned, with a possible trade-off for accuracy in the labeling.

4.6 Splitting

After performing the steps above the dataset is almost ready to be trained, the next step is to split the dataset into two groups, test, and train. By sectioning off a part of the data that

the model will not be able to train on, we will be able to evaluate it on unseen data to get a realistic evaluation of its performance.

Data Leakage and real-world simulation The test dataset should be as close to a real-world simulation as possible to properly represent the model's ability in a more realistic setting. One of the important factors for achieving this is to consider when data points are located for each dataset in time. Suppose a data point is included in the training set that is located in the future for a data point in the test set. In that case, the test and train split will not accurately represent the real world, as the model has access to information about the future, which in turn lowers the validity of the testing of the model. Another concern is that data leakage may occur; if the model is allowed to train on information about future predations, it may use information about the future to make its predictions about the past that it would not be able to guess unless it specifically knew about the future predation, which may also decrease the validity of the tests of the models. During the splitting of the datasets, it was ensured that all data points in the test set were in the future for any given data point in the training set to prevent this type of data leakage from occurring.

Stratification Stratification of the dataset, as mentioned in the theory section of this thesis, is an important consideration when splitting the dataset, especially an unbalanced dataset which this may be considered to be. As opposed to a random sampling of the data where rows are assigned to either the training set or test set randomly, stratification ensures that there is equal weight of each label in both datasets. By doing this, it is guaranteed that both datasets will have the presence of both labels. If this was not done, there might be a risk that there would be little or no data points labeled as being within an attack window in the test dataset due to the random chance of the sampling. Thus it would not be able to check the model's performance in predicting attacks using that test set, which lowers the validity of the test. In an effort to avoid the possibility of bad sampling due to this randomness, the dataset containing the labeled sheep data was stratified using the Python library "scikit-learn," which is a well-known and often-used Python library that provides numerous functions for use in data science and machine learning. By using the method known as "train_test_split," the splitting of the dataset into train and test was done automatically. This function has multiple parameters that may be set to modify how this sampling is done, two of which were utilized. The first one is the stratify parameter. When set, this parameter ensures that the split is properly stratified with respect to a provided parameter, in this case, the target value. Secondly, it was provided with a parameter "test_size," which specifies the ratio of the split. This split ratio was set to 80/20, meaning that 80% of the data will be used for training while 20%

will be used for testing. It should also be noted that no validation set was split out, as the models that will be used out-of-bag and k-folding, there is no need for a validation set; thus, more data can be trained on, which is especially valuable in this dataset due to its size.

Feature	Description	Type
Longitude	Logintude of the datapoint corresponding to the location of the given sheep	Numerical
Latitude	latitude of the datapoint corresponding to the location of the given sheep	Numerical
Distance_traveled_m	Distanced traveled since last reading in meters	Numerical
Source_id	Identifier for each sheep	Categorical
Km_h	Average speed of sheep for the distance between current and previous datapoint, in km/h	Numerical
Altitude	Altitude in meters above sea	Numerical
Altitude_change_meter_hour	Average altitude change between current and previous datapoint, in m/h	Numerical
Sin_hour	sine representation of the hour the data was recorded	Numerical
Cos_hour	cosine representation of the hour the data was recorded	Numerical
Temperature	Air Temperature of the area at the current time	Numerical
Time_difference	difference in time between current datapoint and the previous for that sheep, in seconds	Numerical
HerdID	Unique identifier for each herd of sheep	Categorical

Table 1: Features used for training the model

5 Machine Learning

5.1 Optimizing Parameters

As mentioned in the theory section of this thesis, most machine learning models have parameters that need to be optimized for a given dataset, and this dataset does not deviate from that fact. One way to do this is through manual trial and error, and this is possible though very time-consuming and usually not the best approach, especially when testing multiple parameters and models. To minimize the amount of manual work needed to find the best parameters, a hyperparameter optimization framework (HPO) was implemented to do most of this automatically. The framework that was chosen for this task is known as Optuna, which is an open-source framework developed by the company Preferred Networks. This framework provides multiple benefits over manually searching for the optimal parameters. First, it allows for parralization of training, so multiple models can be trained and assessed at a time, saving the time it takes to find the optimal parameters. It is also made to work with multiple machine learning libraries, such as scikit-learn, which will be used to implement the random forest model, and thus well suited to the task.

The general workflow of the framework is that for a given model, one defines what parameters one wishes to optimize, how the search for the best parameter should be conducted, and the boundaries for each parameter. Then the metric that Optuna should seek to optimize, such as the F1 score. The model then attempts to optimize the set parameters by training multiple models and testing the fitness of the models using the specified metric. Finally, when a pre-set number of trials are completed, it will provide the parameters which provided the best performance across the trials.

5.2 Random Forest

The first supervised model that the data was trained on was a random forest model. This model was trained using the "scikit-learn" library in Python, which has a built-in random forest classification model available for use. While there are multiple parameters that can be set, the most important ones that will be mentioned here are the parameters "n_estimators," "min_sample_leaf," "max_depth," and "class_weight." The "n_estimators" parameter decides the number of trees that will be built by the algorithm. A low number will make for a less complex model, which is easier to train but may lead to underfitting. A high number of estimates, in turn, increases complexity, possibly allowing for the model to capture more patterns, but this runs the risk of overfitting.

The second parameter to be set is the "min_sample_leaf," which decides the minimum number of samples of data required to be at a leaf node. This parameter is a way to control overfitting by disallowing the tree to grow too wide and thus have leaves with a small number of samples fitting into the criteria. In turn, setting this parameter too high could lead to an underfit model as it may not be able to capture the nuanced differences between samples due to the limited number allowed to exist in a leaf. Similar to this metric is the "max_depth," which dictates how deep a tree may grow. A tree that is allowed to grow too deep may capture too much of the noise present in the data and become overfit. In contrast, a tree that is not deep enough will, in the same way as with to few leaves, be unable to capture enough nuance and may become underfit.

Finally is the "class_weight" parameter, which will adjust the weighting of each class. By adjusting the weight of the classes, one may tell the model to pay more attention to one class (label) than another if that label is more interesting. This weighting is especially useful in a dataset such as the one in this thesis due to the fact that the data points that are in the attack window are more interesting than the ones that are outside of it. This changing of weights will also aid in the fact that the dataset is unbalanced, as while the classes in the attack window may be in the minority, this can be evened out by assigning them higher weights. There are two ways to go about this; the first is to set the "class_weight" to "balanced," which is a built-in keyword in the scikit-learn library that tells it to adjust the weights of the classes so that their weights are inversely proportional to their frequency in the dataset, the less frequent class being assigned a higher weight. As such, by doing this, one may mitigate the problem of the unbalanced dataset by numerically evening them out.

Another possibility is manually adjusting the weight by telling the model the values of each weighting; by doing this, it is possible to increase further the attention the model pays to one class, in this case, if it is within the attack window so that it will prioritize guessing those classes correctly over guessing the ones outside incorrectly. While doing this, one thing to keep in mind is that if one assigns too high a weight, one might end up with a high number of false positives for sheep being in the attack window.

As noted in the subsection on parameter optimization, the framework Optuna would be implemented to find the best parameters for the model. For the parameters "n_estimators," "max_depth," and "min_sample_leaf," the optimizer was given an integer space where it could freely pick values between two boundaries. The boundaries used are common boundaries that are suggested by the framework to give the optimizer enough space to search but still narrow it down to a realistic range to limit runtime. For the "class_weight," the values tested were, as mentioned, "balanced" for an inverse frequency sampling; but there were also a number of manual values tested. These values would indicate the value the class of being in the attack window had above the class of being outside it. What the

value space given to the optimizer ended up as can be seen in the table 2.

Hyperparameter	Lower Bound/Options	Upper Bound
n_estimators	200	500
max_depth	10	70
min_samples_leaf	1	10
class_weight	{'balanced', 1, 5, 10, 15, 20, 25, 30}	-

Table 2: Hyperparameter tuning ranges for Random Forest model optimization

This process was repeated over the three datasets of different strictness for the attack window time span, finding the best parameters for the random forest model for each dataset. The metric it was set to optimize was the F1-score, as it is one of the more useful metrics in this case due to the unbalanced nature of the datasets. By running the optimizer for 120 iterations over each of the three datasets, it found the values for the parameters that can be seen in table 3. As these were reasonable parameters for the model, they were chosen to be the ones on which the final models would be trained.

Dataset	n_estimators	max_depth	min_samples_leaf	class_weight
3 Days	308	31	7	balanced
7 Days	288	54	7	balanced
14 Days	462	36	4	15

Table 3: Parameters found to be most optimal by Optuna for the random forest model for each dataset

5.3 XGBoost

The second model that will be used is the XGBoost model, and similar to the random forest model, it too has multiple parameters that need tuning to prove the best performance. As it is also a tree algorithm, it has some similar parameters to the Random Forest that work much in the same way; this includes 'n_estimators" and "max_depth," but as it is a gradient boosting algorithm, it also has multiple other parameters that will need tuning. One parameter that is similar to one used by random forest is the "min_child_weight" which aims to limit the complexity of the tree, and thus lower the risk of overfitting, by limiting the number of leaves. The difference between them is that in a random forest, the cut-off is decided by the number of instances in a leaf, while the "min_child_weight" in XGBoost looks at the weights of those instances.

One parameter that is different from Random Forest in XGBoost is the learning rate parameter. This parameter is used to control the step size, or how much the algorithm corrects itself, between each iteration, which in turn translates to how fast the model fits itself to the data. By setting a high learning rate, the model will more aggressively correct itself to avoid errors; this will, in turn, lead to the model fitting the data faster. The trade-off, however, is that a learning rate that is too high may correct itself too aggressively, thus taking too much of the data's noise into account and so become overfit. For this reason, it is important to tune the model to balance a high learning rate that speeds up learning and a low learning rate to avoid overfitting.

There are a number of other parameters that will also have to be tuned that are not used in Random Forests. The parameter "subsample" decides the fraction of the total data to be used in training each tree; this is used to prevent overfitting but runs the risk of underfitting if the value chosen is too low. Another that will be tuned is the 'colsample_bytree", which controls how features are sampled when creating a tree. As not every tree will use every feature in training, this parameter will tell the model what fraction of features to sample when building the trees; this is another method that will help prevent the model from overfitting.

Finally, the "scale_pos_weight" is one parameter that will be especially important to tune in the instance of this dataset due to its unbalanced nature. This parameter is similar to the class weight parameter in the Random Forest model, as it may be used to give one class, in this case, the datapoint being in an attack window, more weight, thus penalizing the model more when it incorrectly classifies these values. By doing this, the model will usually be better in regards to classifying this data point. However, the value has to be tuned properly as a too-heavy weight may significantly impact the model performance when classifying data points outside of the attack window, as these are viewed as less valuable, and thus getting them wrong is not as much penalized.

Similar to the Random Forest model, these parameters were tuned using the Optuna optimizer, where to score it attempted to maximize the F1 score. For the parameter search, it was given the space, which may be viewed in this table 4, to work with, as to limit the scope somewhat but still give it the necessary room to find the optimal parameters for each dataset.

For each of the three datasets, the optimizer was run for 120 trials, allowing for a high enough number of attempts to be made to discover the optimal parameters. After completing this, each parameter's final values were set to the values seen in this table 5.

Hyperparameter	Lower Bound	Upper Bound
learning_rate	0.01	0.2
max_depth	3	10
min_child_weight	1	10
colsample_bytree	0.1	1
subsample	0.1	1
n_estimators	100	1000
scale_pos_weight	1	100

Table 4: Hyperparameter tuning ranges for XGBoost model optimization using Optuna

Parameter	Timespan	Value
learning_rate	3 Days	0.09883213236405164
	7 Days	0.14078860702168294
	14 Days	0.15749306775707073
max_depth	3 Days	10
	7 Days	10
	14 Days	10
min_child_weight	3 Days	2
	7 Days	1
	14 Days	3
colsample_bytree	3 Days	0.6724936189691573
	7 Days	0.940364662978833
	14 Days	0.7907067447668132
subsample	3 Days	0.9997693052145172
	7 Days	0.9167289219473476
	14 Days	0.9999071332761309
n_estimators	3 Days	847
	7 Days	763
	14 Days	844
scale_pos_weight	3 Days	61.436103392358085
	7 Days	37.82147002353914
	14 Days	7.702137744746685

Table 5: Parameters found to be most optimal by Optuna for the XGBoost model for each dataset

6 Results

Following are the results for the two models, Random Forest and XGBoost, after being trained on the three datasets described in the method section with the parameters outlined previously in this thesis. The results of each model will be focused on three main categories, which will be used for the later analysis and discussion of the results. The first category of metrics will be the predictive accuracy of the model. The metrics that fall under this category all relate to the model's ability to make predictions about the test dataset after being trained on the test set. The metrics that have been considered for this are as follows:

- **Accuracy:** This is the proportion of the total number of correct predictions to incorrect predictions; this is given by the formula:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

- **Precision:** The proportion of positive cases, if the datapoint was in an attack window, correctly identified in the test set. This is given by the formula:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

- **Recall:** Proportion of correctly classified positive cases, the number of sheep in the attack window, and how many the model correctly classified. The formula for this is:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

- **F1 Score:** This is the harmonic mean of precision and recall, which attempts to balance the two values while punishing extreme values more. The formula is the following

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

- **Specifiicity:** This is the proportion of correctly identified negative cases, data points that were not in the attack window; the equation for this metric is:

$$Specificity = \frac{TN}{TN + FP} \tag{5}$$

where

TP : True Positive

FP : Fale Positive

TN : True Negative

FN : False Negative

The second set of information that will be considered is the feature importance of each model. The feature importances are a score given to each feature that was used to train the model, quantifying how important that feature was in the predictions made by the model. The score is given by a number between 0 and 1, where the higher the score, the more important that feature was in the model's decision-making. The final set of information that will be considered for the results of each model is the confusion matrix. This is a simple 2D matrix that displays the model's performance while classifying the test data by showing the number of True Positives, False Positives, True Negatives, and False Negatives for its classifications.

6.1 Random Forest Results

Dataset: 3 Days Attack Window

These are the results of the Random Forest model that was trained on the dataset with the most narrow criteria for the attack window. First is the model's predictive accuracy, which can be viewed in this table 6.

Metric	Value
Accuracy	0.884
Precision	0.429
Recall	0.895
F1 Score	0.580
Specificity	0.883

Table 6: Performance metrics of the model on the dataset with an attack window of 3 days

The second part of the results for the Random Forest model that is of value for evaluation are the feature importances; they may be viewed in this table 13, with the importance score for each feature being given along the x-axis.

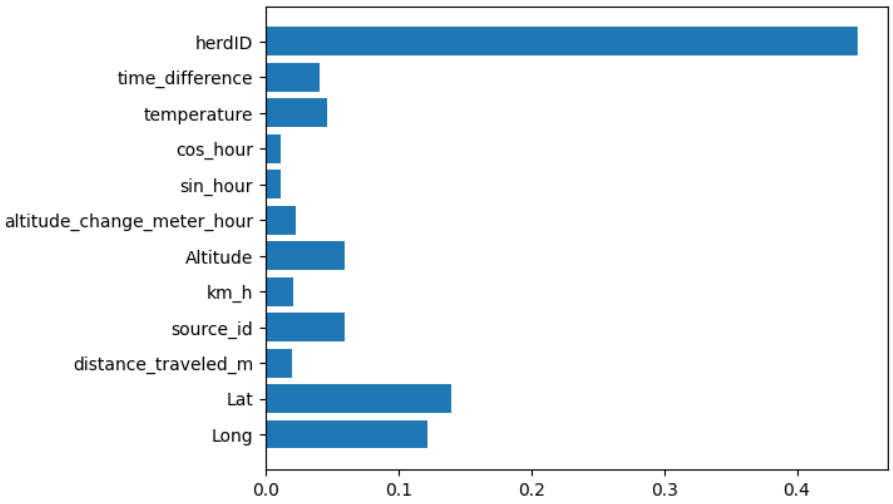


Figure 13: Plot showing the feature importances for the random forest model trained on the 3-day attack window dataset

Finally, the confusion matrix displays the true label and what the model classified each datapoint as 14. The left side indicates the correct label for each datapoint, "false" being outside an attack window and "true" being inside, and the lower side indicates what the model classified that datapoint as being.

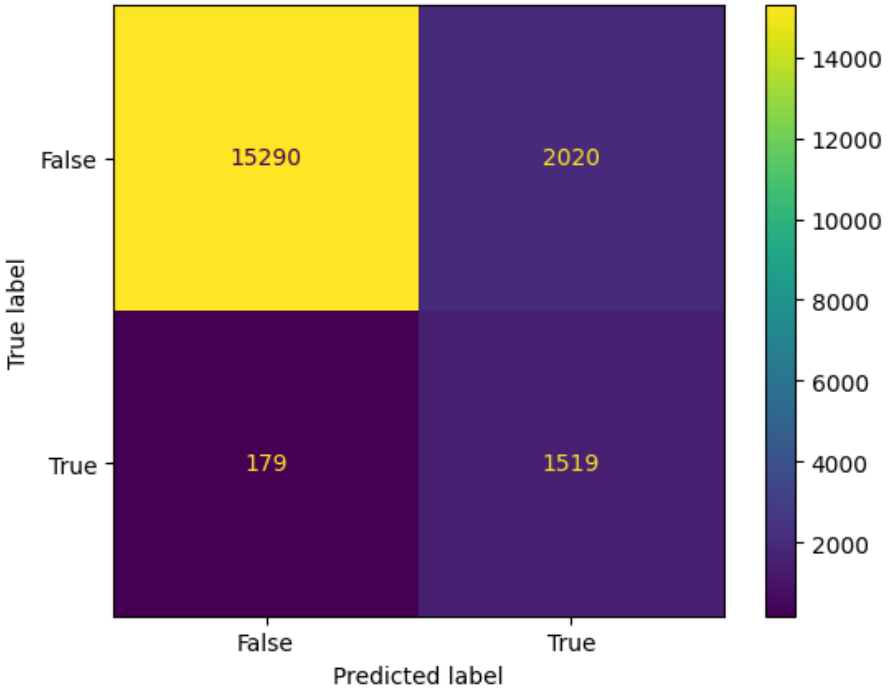


Figure 14: Plot showing the confusion matrix for the random forest model trained on the 3-day attack window dataset

Dataset: 7 Days Attack Window

The results of the predictive accuracy for the random forest model trained on the medium strictness of criteria for the attack window can be seen here 7.

Metric	Value
Accuracy	0.872
Precision	0.438
Recall	0.905
F1 Score	0.590
Specificity	0.868

Table 7: Performance metrics of the model on the dataset with an attack window of 7 days

This is the resulting feature importances of that model 15.

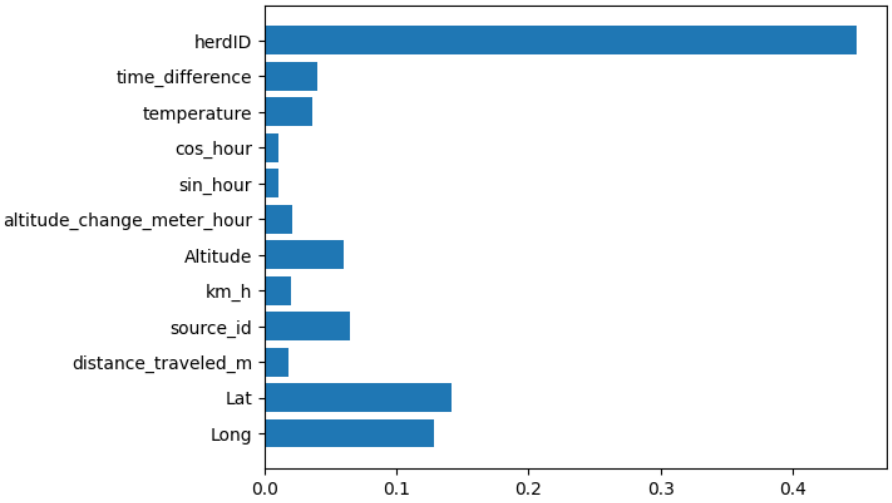


Figure 15: Plot showing the feature importances for the random forest model trained on the seven-day attack window dataset

And finally, the confusion matrix for that model 16.

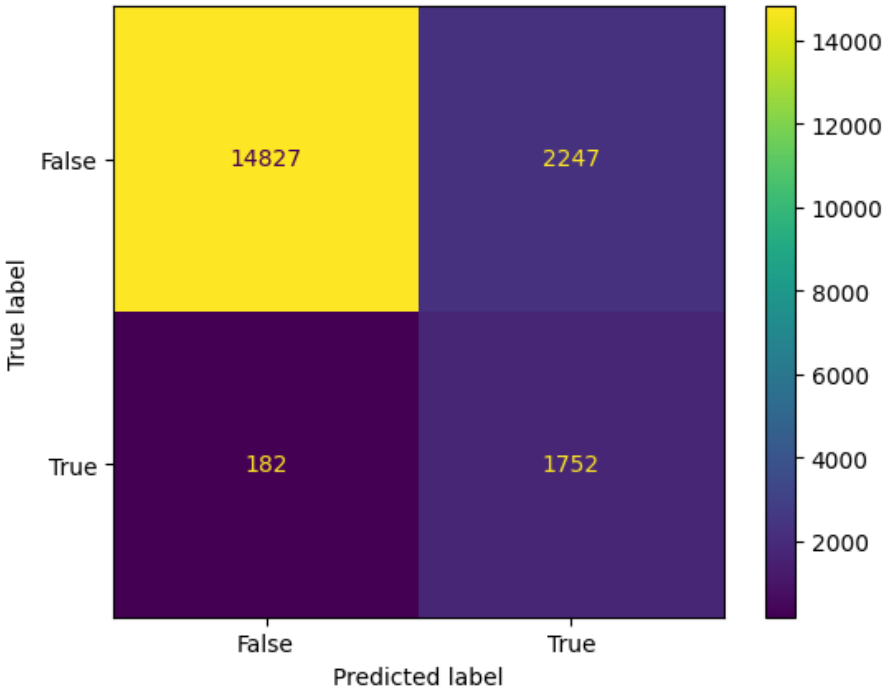


Figure 16: Plot showing the confusion matrix for the random forest model trained on the seven-day attack window dataset

Dataset: 14 Days Attack Window

Here is the table with the predictive accuracy for the random forest model trained on the dataset with the least strict criteria for the attack window, that window being a span of 14 days 8

Metric	Value
Accuracy	0.899
Precision	0.549
Recall	0.883
F1 Score	0.677
Specificity	0.901

Table 8: Performance metrics of the model on the dataset with an attack window of 14 days

The feature importances for this model may be viewed in this plot 17.

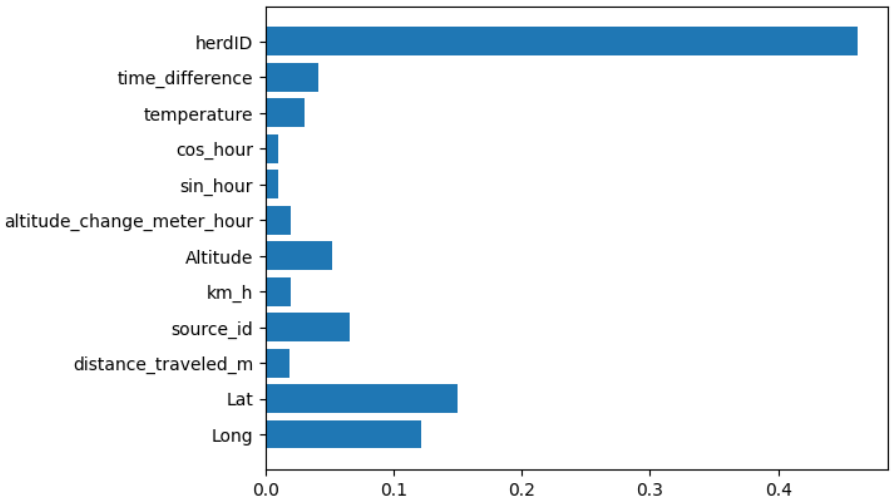


Figure 17: Plot showing the feature importances for the random forest model trained on the 14-day attack window dataset

And lastly is the confusion matrix for the classification made by the same model compared to their actual labels, displayed in this figure 18.

Figure 18: Plot showing the confusion matrix for the random forest model trained on the 14-day attack window dataset

6.2 XGBoost Results

Dataset: 3 Days Attack Window

As XGBoost and the random forest algorithm are in the same family of models, we will be using similar metrics to evaluate them. For the XGBoost model trained on the 3-day dataset, the values of the model's predictive accuracy are listed in this table 9.

Metric	Value
Accuracy	0.943
Precision	0.808
Recall	0.480
F1 Score	0.602
Specificity	0.988

Table 9: Performance metrics of the XGBoost model on the dataset with an attack window of 3 days

Further, we will inspect the feature performances to see what the model valued as the most important feature for its predictions, as seen in this plot 19. Like before, the name of the feature is given along the y-axis, and the importance value is given along the x-axis.

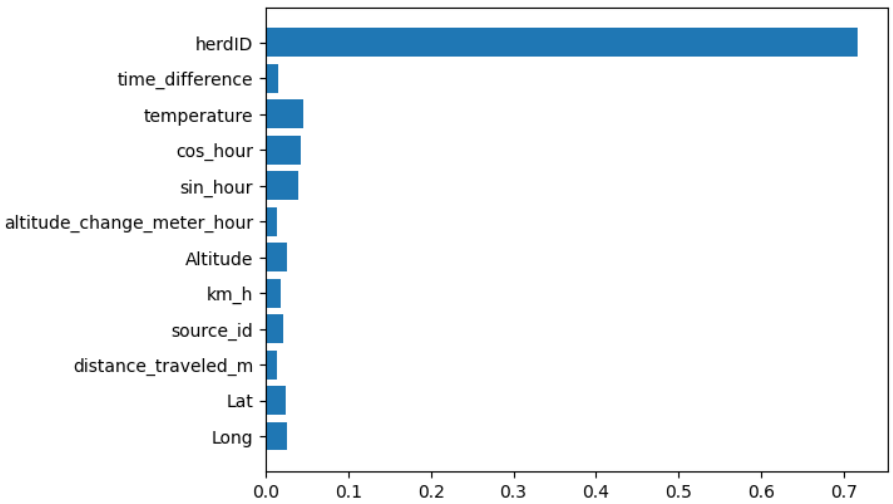


Figure 19: Plot showing the feature importances for the XGBoost model trained on the three-day attack window dataset

Lastly, the confusion matrix displayed the specific number of each true positive, true negative, false positive, and false negative classification made by the XGBoost model 20.

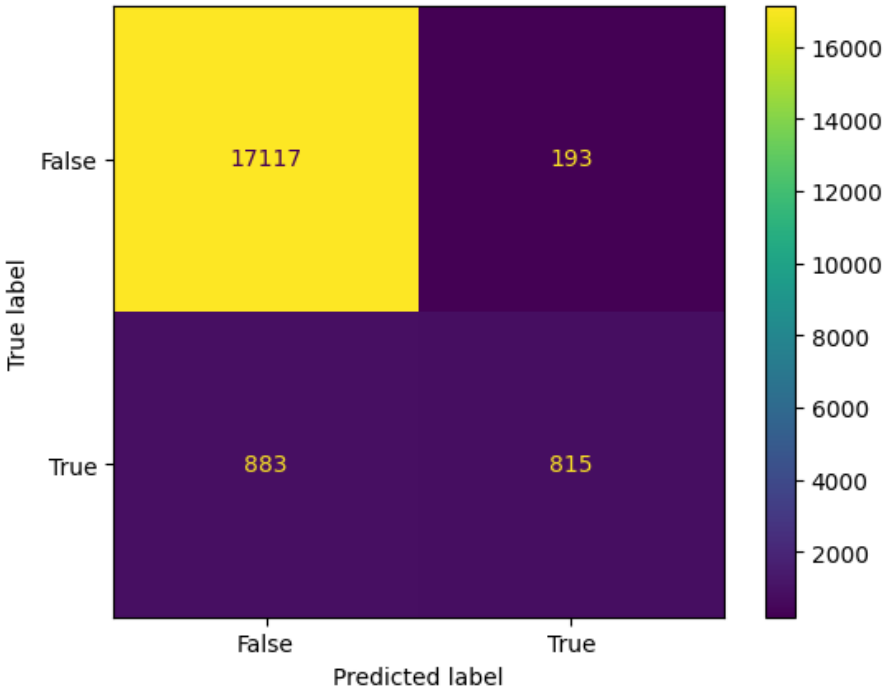


Figure 20: Plot showing the confusion matrix for the XGBoost model trained on the 3-day attack window dataset

Dataset: 7 Days Attack Window

The next XGBoost model that was trained and tested was the one on the 7-day dataset using the parameters optimized for that specific dataset. When tested, the values for that model's predictive accuracy were as can be seen here 10.

Metric	Value
Accuracy	0.939
Precision	0.795
Recall	0.549
F1 Score	0.650
Specificity	0.984

Table 10: Performance metrics of the XGBoost model on the dataset with an attack window of 7 days

The feature importances for the 7-day XGBoost model are as can be seen in this plot 21.

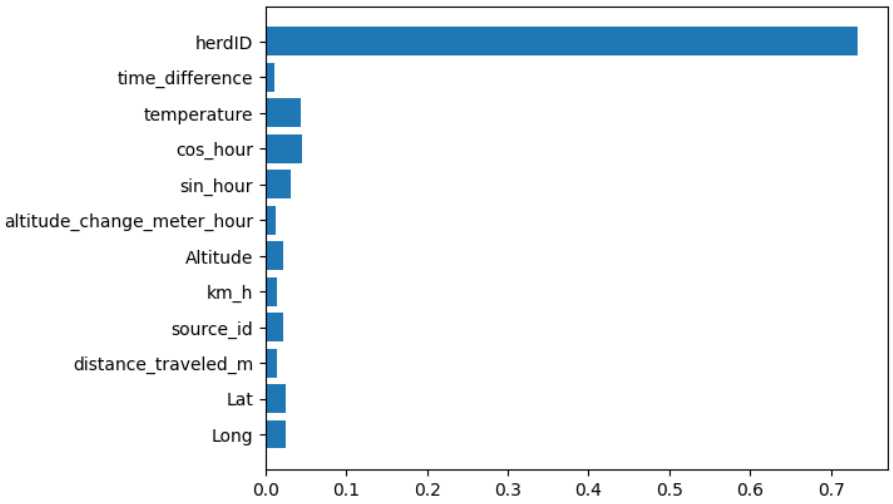


Figure 21: Plot showing the feature importances for the XGBoost model trained on the 7-day attack window dataset

Further, the confusion matrix for the same model is as follows 22

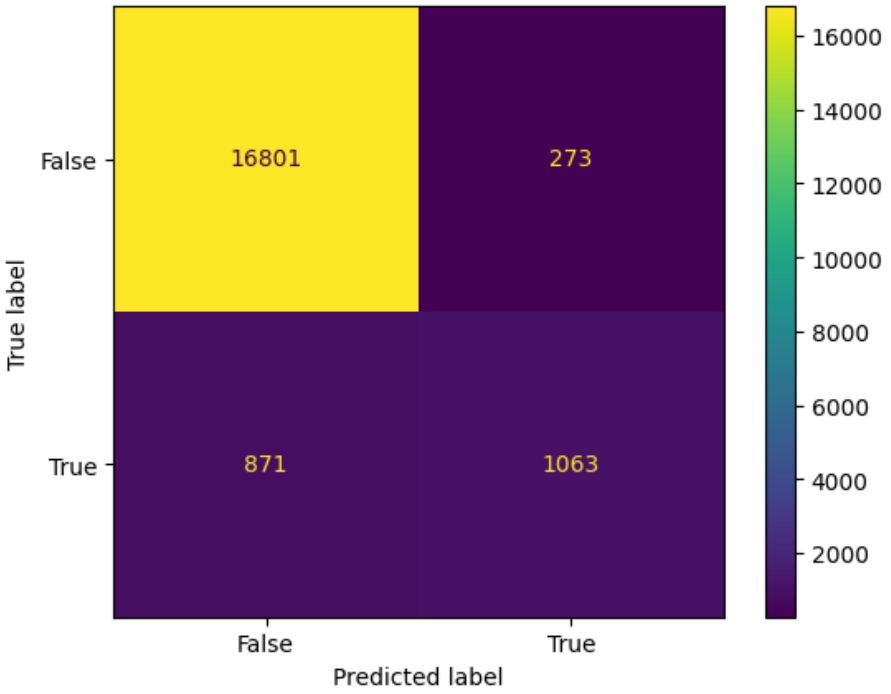


Figure 22: Plot showing the confusion matrix for the XGBoost model trained on the 7-day attack window dataset

Dataset: 14 Days Attack Window

The final results came from the XGBoost model that was trained on the least strict of all the datasets and the one with the most positive classes, the dataset with a valid attack

window of 14 days. This table shows the scores for that model's predictive accuracy when tested against the test set, which was sampled as described in the method part of this thesis 11.

Metric	Value
Accuracy	0.941
Precision	0.799
Recall	0.679
F1 Score	0.734
Specificity	0.977

Table 11: Performance metrics of the XGBoost model on the dataset with an attack window of 14 days

Here, for that same model, are the correlated feature importances plotted for each feature 23.

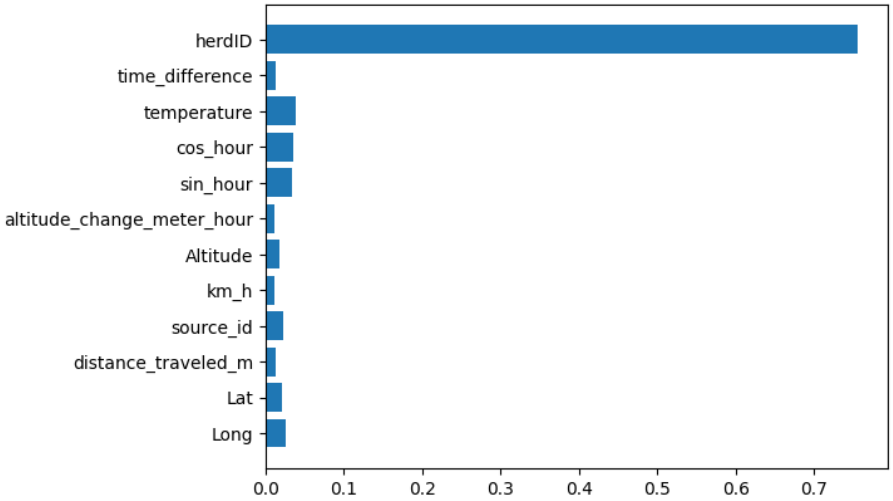


Figure 23: Plot showing the feature importances for the XGBoost model trained on the 14-day attack window dataset

Furthermore, the last of the results extracted from the model when tested is the raw confusion matrix seen here 24.

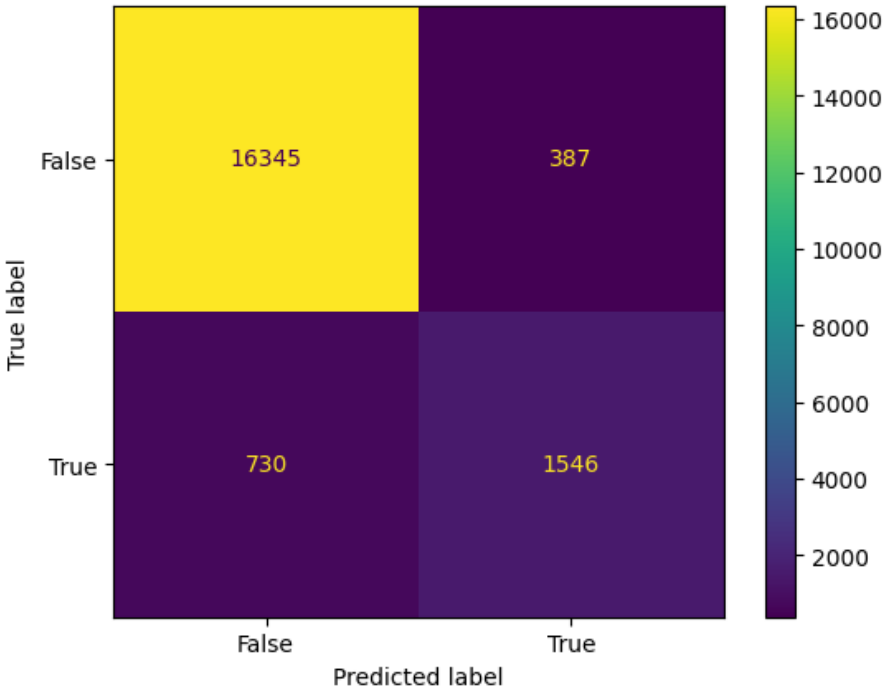


Figure 24: Plot showing the confusion matrix for the XGBoost model trained on the 14-day attack window dataset

7 Discussion

7.1 Data selection decisions

The result of this thesis is highly correlated with the dataset that was used to train the machine learning models, and as such, the decision made concerning that dataset will have a large impact on this. One major decision that was made was in regard to the filtering of outliers and the boundary that was set for something to be an outlier and not. This filtering will have had an impact because at the core of this theory is the detection of anomalies in the data of sheep, and outliers are anomalies. Thus the threshold of where the limit for what is a noise point that needed to be removed and what was a valid outlier that may provide value to the model was not easy to define. I believe that using tried and tested methods, such as the z-score, in combination with relaxing the limit from the commonly used 3 to 6 and 4 for latitude and longitude, respectively, helped limit the number of valid points that were removed while still filtering out the worst of the noise points. As can be viewed visually in figure 8, there are still points in the data that seem to be an outlier from visual inspection but were not touched as to err on the side of caution.

Further removal of points to specify which data is interesting continued with the decision to remove the data points from before the sheep went out to the pasture, being on the farm, and after they were gathered. While they may have given some insight into the general normal behavior of the sheep, this was done, as outlined in the preprocessing section, to ensure that the data and relationships between the data points reflected a situation when the sheep were in an unsecured environment and not as protected as they would be while on a farm. Another boundary that was imposed on the data was the timespan; the datasets for each year had some differences in the beginning and end times of the dataset. This limitation was set so that the totality of the dataset reflected a similar timespan; in consequence, it was decided to limit the dataset to a date range where all datasets were active; this meant cutting off some of the data that fell outside this span. Doing this made the dataset's time range known and easier to handle, as all the subsets had a set beginning and end, which was equal for all the years.

7.2 Feature generation decisions

DBSCAN and herdID One of the decisions made was to label each sheep with which herd it belongs. This decision was made because, as outlined in the theory section, sheep are herd animals and will communicate with each other and act in unison. They sleep together, graze together and react to predators as a flock. As such, it was likely that

it would be valuable for the model to know which herd each sheep belonged to, as the actions made by one sheep are likely to relate to those taken by other sheep in the same herd. The major decision for the DBSCAN model itself was the values of epsilon, which in this case translates to how close a point has to be to each other to be classified as belonging to that herd. The value chosen, 0.045, which equates to 5km, was good due to the daily distance traversal of sheep, as mentioned when the value was first discussed.

Another reason I believe this to be a correct value is due to the fact that the home ranges of the sheep are sufficiently dispersed in the terrain that there is little overlap between the herds; this fact allows for a larger margin of error when it comes to the epsilon value, lowering the risk of misclassification. While that being stated, it is not certain that misclassification did not occur if a sheep strayed into another flock's home range. However, it was verified that no sheep was present in multiple herds, so this problem is also unlikely unless a sheep's unique identifier was somehow also changed. It should also be noted that for any real-life application, there would likely not be any need to run a clustering algorithm to identify the herds, as one would then know for certain which sheep belong to which herd and be able to label them as such with no risk of misclassification.

Weather data The decision to include weather data was based on the fact that sheep will behave differently given temperature and weather. As the dataset contained few features other than the location of the sheep, it was necessary to include as much external information as possible to aid the model. As noted in the section on the preprocessing where weather data was generated, only one measuring station was used for all the herds due to the lack of more suitable stations. Due to this reason, the actual temperature in the location and altitude that the actual data point for a given sheep recorded may deviate somewhat from the value attributed to it. While this is the case, it is unlikely that the deviation is so great that it will significantly impact the behavior of the sheep, even if there is a degree difference in temperature. As this is a fact, the decision to still include the weather data was made to provide the model with more information about the circumstances that may affect the sheep's behavior, other than the presence of predators.

Altitude data The next external dataset that was gathered was the altitude of the sheep. As can be seen in figure 11, there is a notable change in altitude for different times of the day, which may help the model detect deviations and aid in classifying what is normal behavior and what may be behavior in the presence of predators. This data was, in turn, used to generate information about the changes in altitude with the feature that described the total altitude change for one sheep between two data points. As one of the defense mechanisms of sheep is the seek higher ground, it was decided that a sudden change in

the altitude may indicate that the sheep and been spooked or otherwise was behaving anomalously. While that may be a fact, there is one challenge with this data that makes it demanding to capture this behavior of suddenly seeking higher ground, which is the fact that the timespan of each datapoint recording for one sheep often was hours apart, which makes it hard to capture any sudden changed as opposed to standard changes that happen over hours. The challenges with this sparsity of data recording will be discussed more closely later in this section.

Velocity data The next feature generated was the movement velocity using the haversine formula and the time between two data points. The major decision made here was to set the cut-off point to 55km/h. As noted in the preprocessing section, this stems from the theoretical maximum speed of 45km/h and allows for the case that some sheep may achieve even higher bursts of speed. Although this was set as the max velocity, this was more of a matter of procedure than having any real impact on the dataset. The reason this had little impact is because the theoretical maximum speed of 45km/h is for a short burst of speed that cannot be maintained for any significant length of time. However, as mentioned in the paragraph above, one of the challenges with the data is the fact that it is very temporally spaced and, as such, does not capture these slight sudden changes. Thus if any such instances were to occur, it would likely not be captured, and the velocity feature is more likely to capture cases where a sheep chooses to flee an area over greater lengths of time but at a lower speed, which may still be indicative of a predator's presence.

7.3 Generation of predator attack data

One of the main sources of uncertainty and possible error stems from the labels, how they were gathered, processed, and the decision made. The data the labels were generated using did not stem from the original dataset and had no proper interaction with it. As such, through the process outlined in the method chapter, a link between the predator data and the sheep dataset had to be estimated. The challenge is that any labeling error will introduce incorrect data and lower the accuracy of the model by labeling data points as being in the attack window when they should not have been or labeling some as not being in the attack window when they were. Unfortunately, when dealing with a dataset that is as limited as this is, any such incorrect labeling may greatly impact the results. This challenge is further increased by the fact that the data that was labeled through this process was also used to test the model. While a test dataset is usually a way to verify the actual performance of the data, in this case, even a good performance on the test data may not translate to good performance in a real scenario if the decision that was made

when generating the labels were wrong, and thus the test data is not valid. While this is an unfortunate circumstance, there was no other way to approach this as no dataset held the ground truth in this case existed.

It is believed that the choice to set a day limit on the uncertainty allowed for a time span when a sheep was killed helped somewhat mitigate the uncertainty of the positive values. Setting a narrow timespan makes it less likely that a data point has been incorrectly labeled as being in the attack window, although it is still possible. As such, this must be kept in mind when drawing conclusions from the results found. Also, allowing for three datasets with different criteria for this attack window will allow for even further insight into the effect this had, which will be looked at more in-depth when discussing the specific results of each model later in this section.

7.4 Random Forest Results Discussion

As may be interpreted from the results of the random forest model, one may be optimistic about the performance of the model and the possibility that one may be able to detect if a sheep is close to a predator attack through the use of information easily gathered from the tracker and other sources.

Predictive accuracy and confusion matrix The three datasets had slightly different but still relatively similar values when it came to the predictive accuracy metrics. At first glance, one may be optimistic about the model's performance when looking at its high accuracy, recall, and specificity values. While these are high, it is important to note the precision and the F1 score values, which are likely the most valuable metrics for this dataset. In the instance of this dataset and with the model trained, the model's performance when it comes to accuracy and specificity is not especially interesting as they deal with negative case labels, i.e., when the sheep are not in an attack window. As the majority of the dataset is of this class, and this is seen as the normal state for the sheep, that being not near a predator attack, it is not unusual for this metric to be high. In fact, while it may seem high, one may expect it to be higher, as with a dataset where there are only 10% true labels, the model could easily label every instance as false and achieve a better accuracy score than this model did. As this is not the case, show clearly the value of adjusting the class weights to penalize the model highly for guessing the true label wrongly, which is the minority class. The benefit of changing the weights is also evident in the recall score, which is surprisingly high, with a recall score of an average of 0.894 across the three models, meaning that they all functioned well when classifying instances where a datapoint was in an attack window.

Looking at the confusion matrix for the model trained on the 14-day dataset, of 2,276 data points in the attack window in the test dataset, it was able to correctly identify 2,011 of them, which in itself is a good performance. The challenge is that it misclassified 1,651 data points that were not in the attack window as being in one, a likely byproduct of the adjusted class weights. One of the metrics all of the models did not achieve very well in was the precision metric, the reason for this is that the model classifies a high number of false cases as true. The model trained on the dataset of 3 days attack window only achieved a precision of 0.429, which means that for any given data point classified as true, it is more likely to be a false positive than a valid indicator of a predator attack. One of the better and most succinct ways to look at the models is through their F1 score, which is a good indicator of the model's overall performance, being the harmonic mean of precision and recall. With an F1 score between 0.580 and 0.677, the model performs somewhat well while classifying data points but still has many misclassifications.

Feature importance Across all the models, the results of the feature importance were similar; in fact, they were close to identical for each of the models. This similarity between models was unsurprising as they are all random forest models trained on the same type of data, only with some changes in the labels of some data points. As such, it is understandable that the models would assign similar weights to each of the features. One interesting point, however, is that of the features across all three models, the feature that was valued the absolute most was the herd, meaning which herd a sheep belonged to. This feature had a significantly higher value than any of the other features, with an average value of 0.468, compared to the second most important features, Latitude and Longitude, with average importance scores of 0.144 and 0.122, respectively. One possible reason for this is that as the sheep move in, they communicate within and often act as one within any herd. Thus a great deal of information about a sheep at a given moment can be translated to other individuals in the same herd, making it an important factor for identifying sheep's behavior. For the rest of the features, the next most important feature set was the latitude and longitude, the geographical location of the sheep. As predators are also somewhat restricted to a geographical location and have a preferred terrain, it is not unusual that the importance of the geographical location is an interesting feature. It is worth noting here that the inclusion of this as a feature and their usefulness will likely rely on the model being trained on historical data for one area, and if trained on a completely new dataset with no information regarding previous predations, it is unlikely to be as, if at all, valuable. Worthy of note are also the features of altitude and temperature that had some importance for all datasets; as discussed in the theory chapter, the sheep tend to be less at risk when at higher altitudes and tend to deviate from normal behavior when temperatures rise too high, which may be something the models were able to learn and

use in this case.

7.5 XGBoost Results Discussion

Predictive accuracy and confusion matrix The XGBoost model also performed well when it came to the accuracy of its predictions; with an average accuracy of 0.941, one can have reasonable confidence that any randomly selected classification is correct. The challenge, of course, is that similar to the Random Forest model, it struggles when classifying sheep in the attack window. With an average recall of 0.569, it has notable challenges when classifying these labels, almost misclassifying them as often as it classifies them correctly. The models' trouble with this is mirrored in the F1 scores of the models, which range from 0.602 to 0.734, these scores being attributed to both poorer performance of the recall, as mentioned, but also not as strong as hoped performance when it comes to precision. It is noticeable that the use of class weighting worked as intended to some degree, as the model has not ignored the minority class, sheep being in the attack window, there was likely some value to it. That being stated, there is a consideration to be made whether it would have been valuable to adjust the weights even further to increase recall even more, minimizing the number of misclassified "True" classes. By doing this, the model would likely have fewer False Negatives, and more points in an attack window would have been correctly classified, but the trade-off to that would likely have been a sharp increase in the True Positives.

Feature importance in XGBoost Similar to the feature importances of the random forest model, we see a similar trend when it comes to the feature importances of the XGBoost models across all the datasets. The dominant feature is the herd feature; the possible reasons for this are likely similar to the reasoning made for the random forest, which is the fact that sheep act in herds, and thus, information about one sheep can somewhat be generalized from the behavior for all sheep in a herd. However, The XGBoost models did not value sheep's location as much, with most other features being of similar importance. One possibility is that the model was able to extract that information using mostly information about the herd with some combination of other features to a greater degree than the random forest model.

7.6 Difference between models

Both models performed well in their own aspects, with the Random Forest model having a good recall and the XGBoost model having good precision. As these two models are

very similar in how they work, much of the difference in their results likely stems from their hyperparameter tuning, especially the tuning of the weights given to the classes in the dataset. As such, the class weighs something that would have to be fine-tuned to the wishes of any deployer as to what they wish to optimize for if they want more false positives and thus some uncertainty if a sheep actually is close to an attack when the model is classified as such. This difference in class weighting is a case that can be seen in the random forest models, where they have a comparatively high recall bordering 90% but low precision, as such if the model classifies something as not being in an attack window, there is a high chance it is true, but it will, in turn, have a high rate of misclassifying instances when the sheep is not in such a window.

The other alternative, as with the XGBoost model, is the opposite. Here there is a model which has high precision but a lower recall. As it is trained now, this model will ensure higher precision, so it is likely the case when it classifies that a data point is in an attack window. The challenge, of course, is that unlike with the Random Forest models, it is more likely that the model erroneously classifies data points that are in an attack window.

Looking at just the performance metric of the F1 score, we can see that of the two models; the XGBoost performed higher in that regard. However, as the task these models were made for is the detection of anomalous data, missing a sheep that is close to a predator attack is likely a lot more costly, and incorrectly classifying a sheep that is not near one to be one would likely prefer the performance of the random forest model. That being noted, with more hyperparameter tuning, especially regarding class weights and better data, the XGBoost model should be able to do this in a similar, if not better, regard.

7.7 Difference between datasets

There were some differences between the three datasets for the random forest model, but they were not as striking as hypothesized. There is a trend visible in the model's predictive accuracy. This trend is that when the strictness of the criteria for the attack window is loosened, giving the model more positive classes in its dataset, the model tends to perform better when tested. There is an overall increase in the F1 score for each model, with a slight jump from the 3-day dataset to the 7-day dataset and then a more noticeable jump in the f1 score when further loosening the criteria to include a 14-day window.

The same behavior can be seen with the XGBoost model, although the increases in the F1 score are more drastic than with the random forest model. With an F1 score of 0.602 for the model trained on the 3-day dataset and an F1 score of 0.734 on the 14-day dataset, we can observe an increase of 21.9%, which is a nontrivial improvement. This improvement

is mainly due to the increase in recall, as when the criteria are loosened and more positive cases are introduced, the model becomes better at classifying these positive cases.

One of the possible causes for this is that the model has more classes in the attack window and, as such more data to work with, often translating to a more robust model. As the datasets contained a comparatively low ratio of positive to negative cases, any increase in the number of cases will be of help to the model. As both models struggled mainly with identifying the positive cases, i.e., if the data point is in an attack window, increasing examples of this helped the model better learn the patterns associated with these points.

One thing that should be kept in mind when looking at the differences between the datasets is the problems brought up in the section discussing the generation of predator data and splitting it into test and train. As we are sampling the test data from this dataset, we are also decreasing the validity of the test when we are loosening the criteria for the labels due to the fact that we are lowering the confidence in these labels; thus, while it seems the models are performing better, we cannot be certain that translates to actual real-life improvements or is simply due to some pattern present in both the testing and training data. This uncertainty is an unfortunate but unavoidable side effect of the dataset. So while we observe a nontrivial increase in F1 score when going from the strict 3-day dataset to the less strict 14-day dataset, this may not translate to real-life performance. Instead, we may end up with a less robust model due to introducing more noise and lowering the test validity.

8 Conclusion and future work

8.1 Conclusion

This thesis explored the possibility of using machine learning models trained on labeled data from sheep GNSS necklaces to predict predation events and how two models based on the same principle compared in performance. Previous research in this field has mainly relied on either non-machine learning statistical methods, or when machine learning was used, it was implemented as unsupervised methods. This thesis has attempted to fill a gap in previous research by using supervised machine learning to improve the analysis of sheep's movement behavior with respect to predator attacks and to further research in livestock monitoring and loss prevention. For the research questions, we can draw the following conclusions.

Research Question 1: *Can sheep's movement patterns, derived from tracking necklace data, be utilized to predict if a sheep is in the proximity of a predator attack effectively?*

The machine learning models displayed promising predictive accuracy with an F1-score reaching up to 0.734. This non-trivial predictive ability suggests the potential of such models for livestock monitoring and loss prevention. However, further optimization and tuning are necessary to reduce false positives and negatives, thereby enhancing model efficacy.

Research Question 2: *Compared to conventional tree algorithms, is there a significant improvement in the prediction accuracy of sheep predation events when using gradient-boosting tree algorithms?*

A notable difference was observed between the performance of the Random Forest and XGBoost models. The Random Forest model excelled in accurately predicting true labels, while the XGBoost algorithm was more proficient at classifying instances when the sheep were not near predation. In the context of this thesis, accurately predicting true labels is of greater importance than correctly identifying false labels, suggesting that the advanced XGBoost model may not offer a significant advantage over the Random Forest for this specific task. However, the XGBoost model achieved the highest F1-score, indicating superior overall predictive ability when considering both precision and recall. It's important to note that due to the inherent uncertainties and challenges in the dataset, these conclusions are specific to this study and may not extend beyond its scope.

8.2 Future work

More accurate data from sheep One of the major challenges in this thesis was the nature of the data gathered from the sheep. As the dataset was based on GNSS positions, with each data point often being hours apart, it is hard to capture any nuance in changes in sheep's behavior. If a predator were to attack and the sheep sprint off in a different direction at a theoretical maximum of 45km/h for 1 minute and then hides in shrubs or at altitude, this looks no different when averaged over multiple hours to a sheep migrating from one area to another as part of its normal daily activity. The same goes for changes in altitude; there is no proper way to model true abrupt changes in altitude, as any change will end up being averaged out, removing the ease of detecting any flight reaction instead of normal behavior.

Therefore, one of the aspects that can be improved in future work is securing a dataset where data reporting is at a much more frequent pace, optimally a few times each minute. An increased frequency in data measurements would allow for capturing sudden changes in speed and altitude to a degree where one would be able to more easily detect instances where a sheep suddenly accelerate to a sprint to flee something or immediately seeks a higher altitude at speed.

Another reason a dataset with a higher rate of position reporting is that this will allow for better tracking and modeling of a sheep's normal behavior and movement. Suppose the location is reported multiple times a minute each day. In that case, one will be able to model features such as the paths sheep tend to walk, areas where they may move faster or slower, and other insights that could enhance prediction but which, unfortunately, were not possible in this case due to the infrequency and, thus, the roughness of the data.

Another avenue of data that could be useful to gather from sheep in future work would be accelerometer data. If the sheep were fitted with devices that gathered and reported information from an accelerometer, it would be possible to accurately track changes in speed as well as sudden changes in direction to an even finer granularity than with a GNSS system. By incorporating this data, one could model sudden changes in the sheep's behavior, allowing for an even more accurate model to be built.

More accurate predator data One of the primary limitations of this thesis was the nature of the data gathered regarding sheep that had been predated on, which in turn affected the labeling of the data and, thereby, the accuracy of any model built with that data. One of the major points that can be incorporated into future studies is more accurate predator data which will provide more certain and actionable results. More accurate labeling

of classes may be achieved through multiple means, but one way would be to ensure all sheep in a flock are equipped with GNSS necklaces with frequent reporting. If this were the case, one could easily tell when a sheep is predated by the fact that the sheep would no longer be moving. By combining this with sheep carcasses discovered to be predation victims, one can accurately tell when and where predation happens and which sheep were close when it happened, removing much of the uncertainty in this thesis.

Bibliography

[1] Anna Blix and Odd Vangen. *Sau*. URL: https://snl.no/sau (visited on 27th Apr. 2023).

[2] Animalia. *Slaktestatistikk - småfe*. URL: https://www.animalia.no/no/kjott--egg/klassifisering/klassifisering-av-sau/ (visited on 18th May 2023).

[3] Ivar Mysterud. *Tap av sau i utmark : dødsvarsler-systemet som hjelpemiddel*. nob. 1992.

[4] Rovdata. *Erstatning for sau*. 2023. URL: https://www.rovbase.no/erstatning/sau (visited on 25th Apr. 2023).

[5] D Shackleton and Christopher Shank. 'A Review of the Social Behavior of Feral and Wild Sheep and Goats'. In: *Journal of Animal Science* 58 (Jan. 1984), pp. 500–509.

[6] L. J. Keeling and H. W. Gonyou. *Social Behaviour in Farm Animals*. Ed. by L. J. Keeling and H. W. Gonyou. CABI Publishing, 2001.

[7] A. B. Lawrence and D. G. M. Wood-Gush. 'Home-Range Behaviour and Social Organization of Scottish Blackface Sheep'. In: *Journal of Applied Ecology* 1 (1988), pp. 25–40. ISSN: 00218901, 13652664.

[8] Derek W Bailey et al. 'Mechanisms that result in large herbivore grazing distribution patterns'. In: *Journal of Range Management* 49.5 (1996), pp. 386–400.

[9] *Saueboka*. 1998.

[10] Jon J Nedkvitne. *Beitedyr i kulturlandskap*. nno. Oslo, 1995.

[11] 'Geometry for the selfish herd'. In: *Journal of Theoretical Biology* 31.2 (1971), pp. 295–311. ISSN: 0022-5193. DOI: https://doi.org/10.1016/0022-5193(71)90189-5.

[12] Sauehelsenett. *Sauens atferd*. 2023. URL: https://www.animalia.no/no/Dyr/sauehelsenett/velferd2/sauens-atferd/ (visited on 26th Apr. 2023).

[13] Steven Lima and Larry Dill. 'Behavioral Decisions Made under the Risk of Predation: A Review and Prospectus'. In: *Canadian Journal of Zoology-revue Canadienne De Zoologie - CAN J ZOOL* 68 (Apr. 1990), pp. 619–640. DOI: 10.1139/z90-092.

[14] Inger Hansen, Hanne Solheim Hansen and Frank Christiansen. 'Kartlegging av antipredatoratferd hos ulike saueraser'. In: *Planteforsk Rapport* (1998).

[15] Christian Vucic Toni; Axell. 'Tracking sheep by radio tags and UAV: A field study of Bluetooth round-trip time ranging and multilateration'. In: (2022). URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/3024696.

[16] Muneer Ilyas Qazi Mudassar; Ahmad. 'Smart Farming: An Enhanced Pursuit of Sustainable Remote Livestock Tracking and Geofencing Using IoT and GPRS'. In: (2020). DOI: https://doi.org/10.1155/2020/6660733.

[17] L. Turner et al. 'Monitoring cattle behavior and pasture use with GPS and GIS'. In: *Canadian Journal of Animal Science - CAN J ANIM SCI* 80 (Sept. 2000), pp. 405–413. DOI: 10.4141/A99-093.

[18] Rovdata. *Fakta om jerv*. 2023. URL: https://rovdata.no/Jerv/Faktaomjerv.aspx (visited on 25th Apr. 2023).

[19] Rovdata. *Fakta om gaupe*. 2023. URL: https://rovdata.no/Gaupe/Faktaomgaupe.aspx (visited on 25th Apr. 2023).

[20] Rovdata. *Fakta om kongeørn*. 2023. URL: https://rovdata.no/Konge%C3%B8rn/Faktaomkonge%C3%B8rn.aspx (visited on 25th Apr. 2023).

[21] Audun Stien. *I disse fylkene finner man flest kongeørndrepte lam*. 2019. URL: https://blogg.forskning.no/rovdyrbloggen/i-disse-fylkene-finner-man-flest-kongeorndrepte-lam/1579016 (visited on 25th Apr. 2023).

[22] Rovdata. *Brunbjørn*. 2023. URL: https://rovdata.no/Brunbj%C3%B8rn.aspx (visited on 25th Apr. 2023).

[23] Klima- og miljødepartementet. *Forskrift om erstatning når husdyr blir drept eller skadet av rovvilt*. 2014. URL: https://lovdata.no/dokument/SF/forskrift/2014-05-30-677.

[24] 'Understanding and managing conservation conflicts'. In: *Trends in Ecology Evolution* 28.2 (2013). ISSN: 0169-5347. DOI: https://doi.org/10.1016/j.tree.2012.08.021.

[25] Norvald Kjerstad and Børje Forssell. *GPS*. URL: https://snl.no/GPS (visited on 13th Apr. 2023).

[26] Christopher Hegarty and Elliott Kaplan. 2005.

[27] IBM. *What is machine learning?* 2022. URL: https://www.ibm.com/topics/machine-learning (visited on 24th Apr. 2023).

[28] IBM. *What is unsupervised learning?* 2022. URL: https://www.ibm.com/topics/unsupervised-learning (visited on 24th Apr. 2023).

[29] Martin Ester et al. 'A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise'. In: *Knowledge Discovery and Data Mining*. 1996.

[30] Alexander Hinneburg and Daniel A. Keim. 'Optimal Grid-Clustering : Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering'. In: *Proceedings of the 25 th International Conference on Very Large Databases, 1999*. 1999, pp. 506–517.

[31] Erich Schubert et al. 'DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN'. In: (2017). DOI: 10.1145/3068335. URL: https://doi.org/10.1145/3068335.

[32] IBM. *What is supervised learning?* 2022. URL: https://www.ibm.com/topics/supervised-learning (visited on 24th Apr. 2023).

[33] Leo Breiman. 'Bagging Predictors'. In: *Machine Learning* (1996).

[34] L Breiman. 'Random Forests'. In: *Machine Learning* (2001). DOI: 10.1023/A:1010950718922.

[35] S. Madeh Piryonesi and Tamer E. El-Diraby. 'Using Machine Learning to Examine Impact of Type of Performance Indicator on Flexible Pavement Deterioration Modeling'. In: *Journal of Infrastructure Systems* (2021). DOI: 10.1061/(ASCE)IS.1943-555X.0000602.

[36] XGBoost. *Machine Learning Challenge Winning Solutions*. 2023. URL: https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions (visited on 23rd Apr. 2023).

[37] Tianqi Chen and Carlos Guestrin. 'XGBoost: A Scalable Tree Boosting System'. In: 2016. DOI: 10.1145/2939672.2939785. URL: http://doi.acm.org/10.1145/2939672.2939785.

[38] Ryan Holbrook. *Feature Engineering | Kaggle Course*. 2023. URL: https://www.kaggle.com/learn/feature-engineering (visited on 20th Apr. 2023).

[39] Parth Gohil. *Feature Transformation*. 2022. URL: https://towardsai.net/p/l/feature-transformation (visited on 20th Apr. 2023).

[40] Dalwinder Singh and Birmohan Singh. 'Investigating the impact of data normalization on classification performance'. In: *Applied Soft Computing* 97 (2020), p. 105524. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2019.105524.

[41] Shipra Saxena. *Here's All you Need to Know About Encoding Categorical Data*. 2022. URL: https://www.analyticsvidhya.com/blog/2020/08/types-of-categorical-data-encoding/ (visited on 22nd Apr. 2023).

[42] Dipanjan Sarkar, Raghav Bali and Tushar Sharma. *Practical Machine Learning with Python*. Jan. 2018. ISBN: 978-1-4842-3206-4. DOI: 10.1007/978-1-4842-3207-1.

[43] SciKit. *Recursive Feature Elimination*. 2019. URL: https://www.scikit-yb.org/en/latest/api/model_selection/rfecv.html (visited on 22nd Apr. 2023).

[44] James Kanter and Kalyan Veeramachaneni. 'Deep feature synthesis: Towards automating data science endeavors'. In: 2015. DOI: 10.1109/DSAA.2015.7344858.

[45] IBM. *What is overfitting?* 2022. URL: https://www.ibm.com/topics/overfitting (visited on 22nd Apr. 2023).

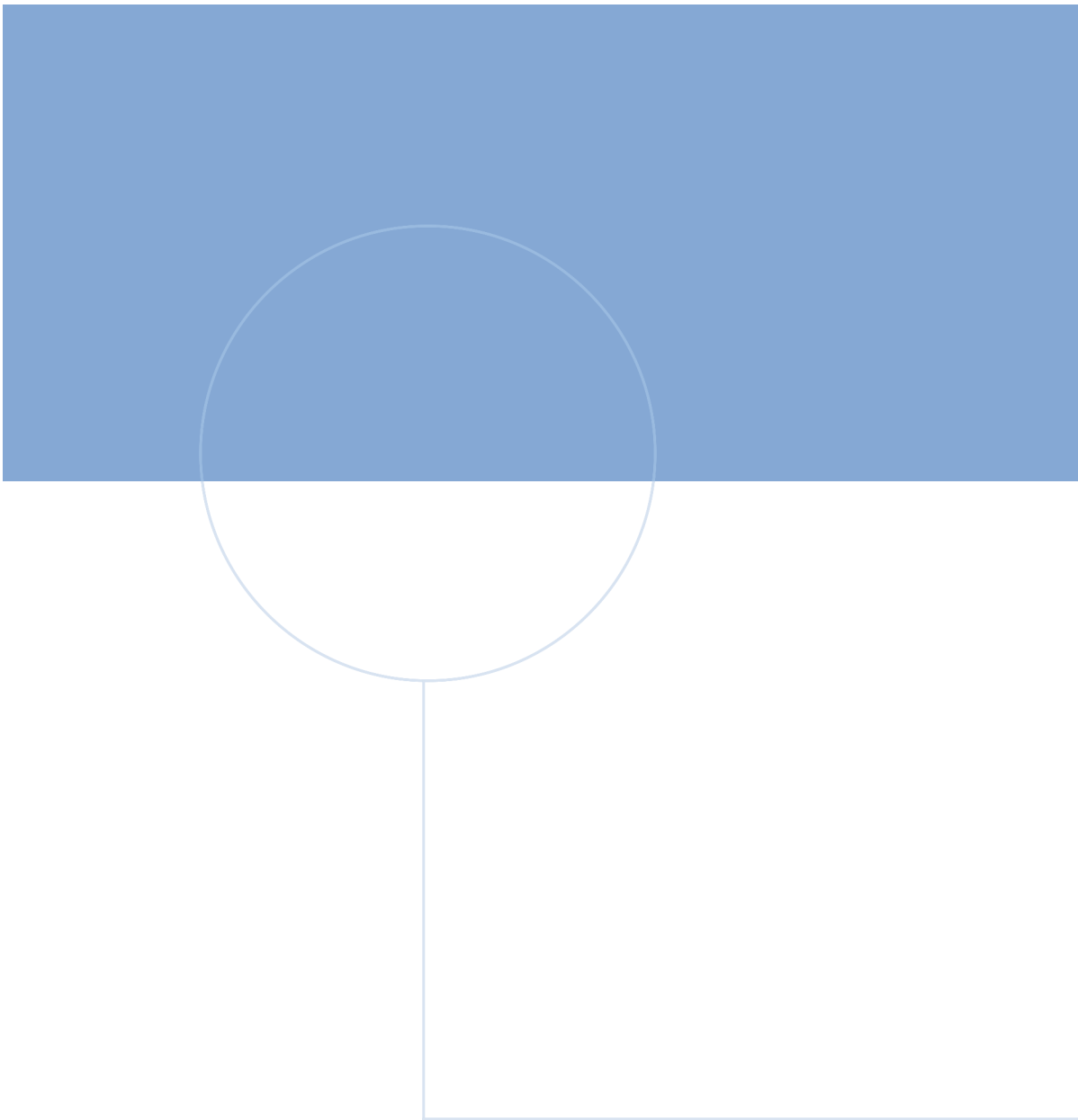
[46] Xue Ying. 'An Overview of Overfitting and its Solutions'. In: 1168.2 (), p. 022022. URL: https://dx.doi.org/10.1088/1742-6596/1168/2/022022.

[47] Haibo He and Edward A Garcia. 'Learning from imbalanced data'. In: *IEEE Transactions on Knowledge and Data Engineering* (2009).

[48] John D Kelleher, Brian Mac Namee and Aoife D'Arcy. *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. MIT Press, 2015.

[49] Javier Cabezas et al. 'Analysis of Accelerometer and GPS Data for Cattle Behaviour Identification and Anomalous Events Detection'. In: *Entropy* 24.3 (2022). DOI: 10.3390/e24030336.

[50] Caitlin A. Evans, Mark G. Trotter and Jaime K. Manning. 'Sensor-Based Detection of Predator Influence on Livestock: A Case Study Exploring the Impacts of Wild Dogs (Canis familiaris) on Rangeland Sheep'. In: *Animals* 12.3 (2022). DOI: 10.3390/ani12030219.



NTNU

Norwegian University of
Science and Technology