UiA University of Agder

A Machine Learning Approach to Predict Accidents using Real Spatiotemporal Road Data

JONAS GUNNENG HEER
EINAR MARTIN EGELAND JOHNSEN

SUPERVISOR

Assoc. Prof. Morten Goodwin, Ph.D.
Ph.D. Candidate Jahn Thomas Fidje

University of Agder, 2019
Faculty of Engineering and Science
Department of ICT

Abstract

Traffic accidents bring a significant cost to society. In 2017, 106 people died in traffic-related accidents in Norway alone, and a total of 664 people got severely injured [1]. Identifying the alterable parameters that influence the level of road safety is clearly valuable when building new road infrastructure and when implementing other countermeasures. Traffic accident prediction can be divided into classification and regression problems. This thesis looks at arguably the most relevant, binary classification of accident / no-accident from spatiotemporal data.

The problem is particularly challenging due to the imbalanced classes, spatial heterogeneity, and the non-linear relationship between dependent and independent variables. Inspired by a recent paper, we investigate the importance of spatial heterogeneity in traffic accident prediction using Norwegian accident, road, and weather data. [2]

This thesis examines the potential of Decision Tree (DT), Random Forest (RF), Neural Network (NN), and Gradient Boosting Machine (GBM) models for this purpose. Using the DT, RF, and GBM model, we also identify features with the most predictive power.

Experiments show that we can predict accidents with a precision of 33% and a recall of 88%, and that the GBM model achieves the highest recall and F1-score. We find that the annual average daily traffic, hour of the day, and day of the week are the features with the greatest predictive power, while speed limit is of little relevance. To the best of our knowledge, these are the first machine learning models that accurately predict accidents in a Norwegian setting.

Preface

This thesis is made as a completion of the master education in communication and information technology (ICT), at the University of Agder, Norway.

This project has received support and contributions from several individuals throughout its lifespan, and the authors would like to thank in particular our supervisor Associate Professor Morten Goodwin, and Ph.D. Candidate Jahn Thomas Fidje.

Grimstad, 24th of May 2019.

Table of Contents

List of Figures

Listings

List of Tables

Chapter 1

Introduction

Traffic accidents bring a significant cost to society. In 2017, 106 people died in traffic-related accidents in Norway alone, and a total of 664 people got severely injured [1]. Multiple factors influence the occurrence and severity of accidents, including the involvement of alcohol and other drugs, as well as weather, traffic-flow, vehicle, and road conditions [10, 11]. Identifying the alterable parameters that influence the level of road safety is valuable for building new road infrastructure or when implementing other countermeasures.

The existing literature in the field of traffic safety research is rather extensive and include papers on both traffic volume and traffic accident prediction, as well as traffic accident severity analysis [2, 12, 13]. Several statistical and machine learning approaches have been examined, including a Decision Tree (DT) model presented by Abellán et al. used for predicting the severity of an accident, as well as Poisson, Negative Binomial, and Negative Multinomial regression models by Caliendo et al. used for accident prediction [12, 14].

Several factors have been found to have predictive power, such as road-related features like curvature [14], median width [15], and weather-related features like precipitation and temperature [10, 16]. Human-related factors like whether or not someone is driving over the speed limit or driving under the influence are in large part addressed in self-driving vehicle research.

This thesis examines 25 different features of traffic accidents in the Norwegian road transportation system, investigates the importance of spatial heterogeneity in traffic accident prediction, and compares our findings with previous work in the field.

1.1 Problem statement

The economic and social impact of traffic-related accidents justifies time and effort spent in research. In Norway, in 2016, 135 people were killed, and 656 were severely injured in traffic-related accidents. The economic cost of these accidents was over 33 billion Norwegian kroner. [17]

The occurrence of traffic accidents, in more technical terms, can be seen as a function of factors such as high speed and driving under the influence. Other non-human factors can be divided into temporal, weather, and road related factors [2]. In this thesis, human factors are treated as constant.

Traffic accident prediction is divided into two types: classification and regression. This project looks exclusively at binary classification (accident / no-accident) for a time slice of one hour on a given road segment. The problem is a challenging one due to the imbalanced classes, spatial heterogeneity, and the non-linear relationship between dependent and independent variables. [2]

This thesis investigates the predictive power of 25 different features regarding traffic accidents in the Norwegian road network, which, to the best of the author's knowledge, has never been done before. Inspired by a recent paper by Yuan et al. this thesis also investigates the importance of spatial heterogeneity in attempts to produce comparable results with Norwegian accident, road, and weather data [2].

1.1.1 Research questions

1. To what extent is it possible for a machine learning algorithm to predict accidents accurately?
2. Which machine learning approach is best able to predict accidents in

the Norwegian road network?

3. Which features have the most significant impact on traffic accident prediction in the Norwegian road network?

4. Is speed limit one of the features with the most significant impact on traffic accident prediction in the Norwegian road network?

5. How much impact does spatial heterogeneity have related to traffic accident prediction in the Norwegian road network?

1.2 Contributions

This thesis investigates the importance of 25 different features, determined by four different machine learning models: DT, Random Forest (RF), Neural Network (NN), and Gradient Boosting Machine (GBM), using Norwegian accident, road, and weather data.

Experiments are run for each of the four models, optimizing for F1-score favoring recall. Equivalent tests are conducted with varying numbers of spatial features[1] added to determine its importance.

1.3 Thesis outline

Chapter 2 introduces, in brief, the theory behind the machine learning concepts applied in the experiments. Chapter 3 provides an overview of the current state-of-the-art in the field. Chapter 4 presents our data and where it comes from. Chapter 5 details our approach. Chapter 6 presents our experiments and compare the results with state-of-the-art. Finally, in chapter 7, we conclude our work and propose future work.

[1]Features derived from geographical proximity between accidents. See Section 5.5.

Chapter 2

Background

2.1 Classification

Classification is the task of predicting class labels based on observations. It is a subcategory of supervised machine learning and can be either binary or multi-class classification tasks. Binary classification is based on the differentiation between two classes, with values such as {True, False}, {0, 1} or other values to represent the two classes. An example of binary classification might be whether or not an accident happens at a specific time at a specific road, and might have the values {accident, no accident}. Multi-class classification has the same concept as binary classification but with the number of classes > 2. An example of a multi-class task is recognizing handwritten numbers. The MNIST database of handwritten digits contains handwritten images with one of the numbers from 0 to 9 [18], as shown in Figure 2.1. Given an image, the classifier predicts one of the values from 0-9, hence it is a multi-class task with the number of classes equal to 10. [19]

2.2 Neural Network (NN)

NNs are inspired by how our brain works, and the concept of neurons first appeared in the paper "A Logical Calculus of the Ideas Immanent in Ner-

Figure 2.1: Image showing samples from the MNIST database of handwritten digits [3].

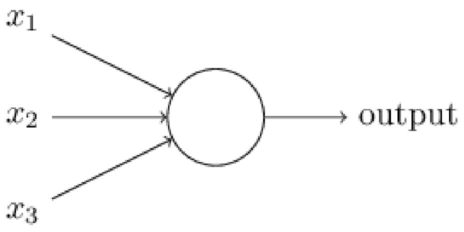

Figure 2.2: Percepteron [4].

vous Activity" by W. S. McCulloch and W. Pitts [20, 19]. Over the years it has been developed, and today NNs are commonly used in machine learning. A NN consists of multiple parts, and many different architectures have been proposed such as Recurrent NN and Convolutional NN [21, 22]. However, regular NNs are built up of many neurons, which together form the network. Such a neuron has n inputs, and a simple model is shown in Figure 2.2. [4]

Each of the inputs x have different values, often between 0 and 1, and the output is based on an activation function, such as the Sigmoid function $\frac{1}{1+e^{-z}}$. $z = w * x + b$, where w refers to the weight between the input and the neuron, x refers to the input and b refers to the bias added to each of the weights. When there are multiple of these neurons, we have a NN, as

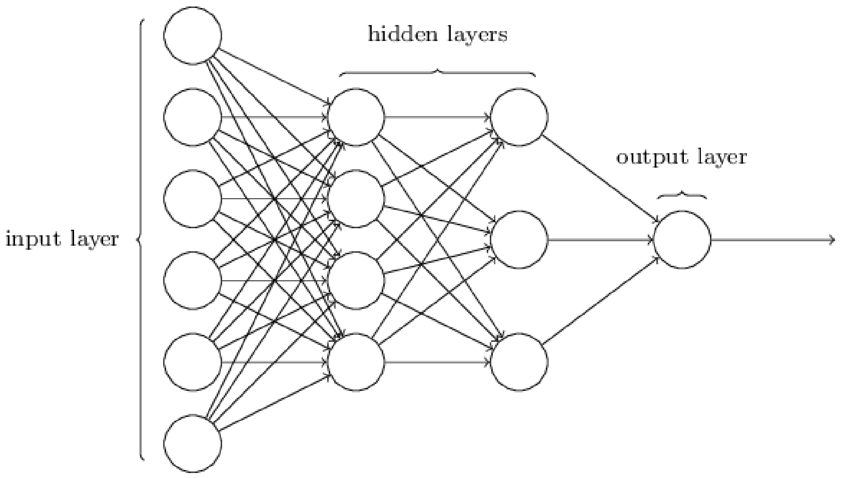

Figure 2.3: An example of a Neural Network structure [4].

shown in Figure 2.3. [4]

The first part of Figure 2.3 is the input to the NN. The hidden layers refer to the layers of neurons between the input and the output layer. Based on an optimization function, such as Stochastic Gradient Descent [23], the network's weights are trained to classify between different inputs. An example is the images in the MNIST database of handwritten digits [18], where the NN learns from 60,000 images how to predict digits in unseen images. An example network structure is shown in Figure 2.4. The network has an input of 784 neurons (28x28 pixels), and one hidden layer with 15 neurons, and the output layer of 10, one for each of the numbers 0-9. [4]

2.3 Decision Tree (DT)

DT is a widely used supervised classification technique and can be applied to any domain. A DT is a tree-like model which given an input predicts a target variable. DT can be split into two main types, which are Classification and Regression Trees. Classification Trees are predicting an outcome in the form of a label, such as {True, False}, while Regression Trees predicts another outcome in the form of a real number, such as the number of people living in a city. There are multiple DT algorithms, but ID3 (Iterative Dichotomies 3), C4.5 and CART (Classification and Regression Tree) are widely used.

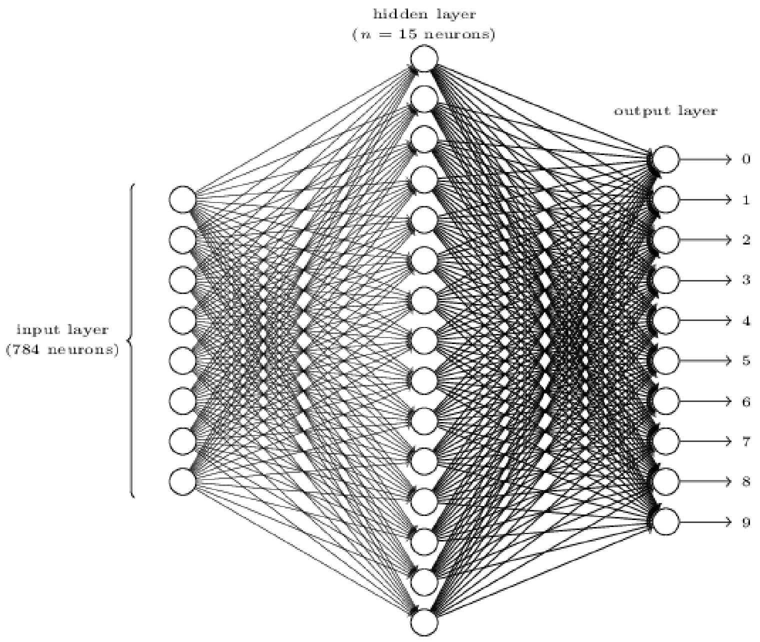

Figure 2.4: An example network structure for the MNIST dataset [4].

There are many reasons to use DTs, such as their ability to give inductive inference about data. [24, 25, 26]

When a DT is created using the ID3 algorithm, both entropy and information gain is used to select how to split the tree. The attribute with the highest information gain is used to split first, then it continues to split on attributes with the highest information gain until there is nothing more to learn, and a leaf node represents a decision. An example is shown in Figure 2.5. [24, 25, 26]

2.4 Random forest (RF)

The definition of RF defined by Leo Breiman in 2001:

> "A random forest is a classifier consisting of a collection of tree-structured classifiers {h(x,θk), k=1, ...} where the {θk} are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x." [27]

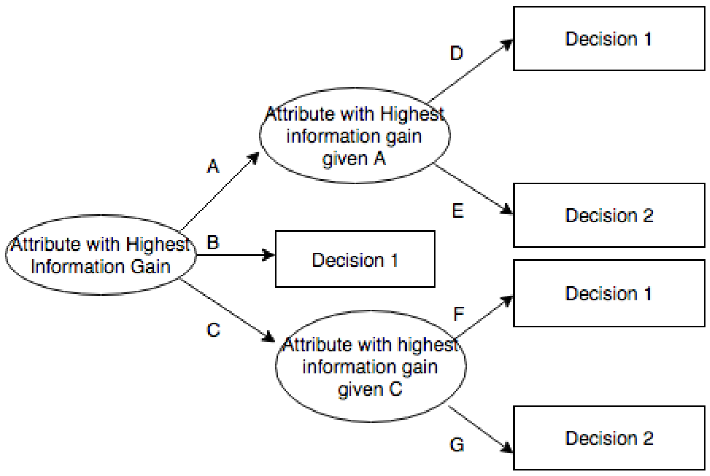
Attribute with Highest Information Gain
A
B
C
Attribute with Highest information gain given A
Decision 1
Attribute with highest information gain given C
D
E
F
G
Decision 1
Decision 2
Decision 1
Decision 2

Figure 2.5: Decision Tree example [5].

As the definition states, a RF is a classifier based on multiple tree-structured classifiers. All of the individual trees vote for a class label for the given input x, and the most popular vote is chosen as the class label. Each of the trees in the forest is created by choosing a random vector from an even distribution. The random vector contains the subset of features from the training set and uses those features to generate the tree. A simplified example is shown in Figure 2.6. [27]

The dataset is used to create four different trees, with the subset of features $\{$N1, N2, N3, N4$\}$ respectively. Each of the trees predict a class label, and the final class is chosen by majority voting. Since the given example contains $B = 1$, $C = 2$ and $D = 1$, C is chosen by majority vote and is the class predicted by the RF algorithm. [27]

2.5 Spectral clustering

Spectral clustering is a clustering algorithm, which tries to cluster data based on a similarity graph. The algorithm is popular in modern clustering, and often outperform traditional clustering algorithms, such as k-means. [9]

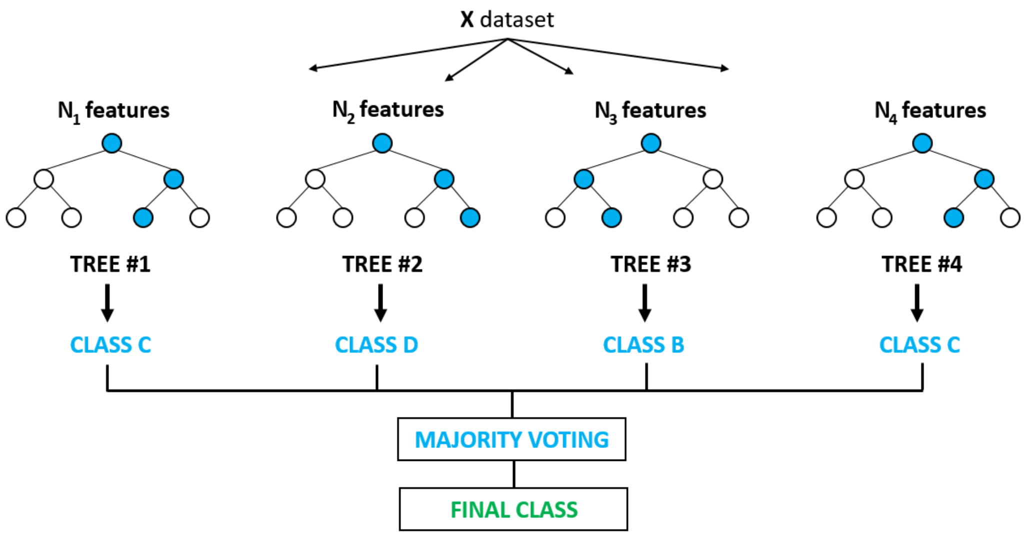

Figure 2.6: Overall structure of the Random Forest algorithm [6].

The spectral clustering algorithm according to Ng, Jordan, Weiss [28], is shown in Listing 2.1.

```
Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number $k$ of clusters
to construct.

1. Construct a similarity graph by one of the ways described in
   Section 2. Let $W$ be its weighted adjacency matrix.
2. Compute the normalized Laplacian $L_{sym}$.
3. Compute the first $k$ eigenvectors $u_1, ..., u_k$ of $L_{sym}$.
4. Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing vectors $u_1, ..., u_k$
   as columns.
5. Form the matrix $T \in \mathbb{R}^{n \times k}$ from U by normalizing the rows to
   norm 1, that is set $t_{ij} = u_{ij} / (\sum_k u_{ij}^2)^{1/2}$.
6. For $i = 1, ..., n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding
   to the $i$-th row of $T$.
8. Cluster the points $(y_i)_{i=1,...,n}$ with the
   $k$-means algorithm into clusters $C_1, ..., C_k$.
```

```
Output:  Clusters  $A_1, ..., A_k$  with  $A_i = \{j | y_j \in C_i\}$.
```

Listing 2.1: Spectral clustering algorithm. Taken from *A Tutorial on Spectral Clustering* by Ulrike von Luxburg [9].

The algorithm is based on a similarity graph, which can be computed in multiple ways, but the overall goal of the similarity graph is to model relationships between the data points. After the similarity graph W is computed, the Laplacian is computed. As with the similarity graph, there exist multiple ways to construct the Laplacian, which can be both normalized and unnormalized. However, in the algorithm by Ng, Jordan, and Weiss, the normalized Laplacian is computed by the formula:

$$Lsym := D^{\frac{-1}{2}} L D^{\frac{-1}{2}} = I - D^{\frac{-1}{2}} W D^{\frac{-1}{2}}.$$

Based on the normalized Laplacian, the first k eigenvectors are computed. The new matrix $T \in \mathbb{R}^{n \times k}$ then contains the k first eigenvectors as columns, and each data point corresponds to a row in the matrix. Then the k-means algorithm is used to cluster the data points into clusters $C_1, \ldots, C_k$. [9]

2.6 Gradient Boosting Machine (GBM)

GBM is a powerful machine-learning technique, which shows great success in many different applications. The idea behind GBM is to fit new models to provide a better prediction for the target. The user can specify the loss function, which the base-learners are trying to be correlated with the negative gradient. It is also possible to specify the base-learners and there exist multiple models, which include linear models, smooth models and DT. DTs are commonly used, and both XGBoost and TF Boosted Trees are libraries where one can use DT as base-learner [29, 30]. By using DTs as the base-learner, many trees are added together to optimize an objective function. An example of two CARTs is shown in Figure 2.7, where two CART tries to complement each other. The Figure shows the probability of a person liking a computer game, and as a single DT is usually not strong enough in practice, multiple trees are used to optimize an objective function. [31, 32]

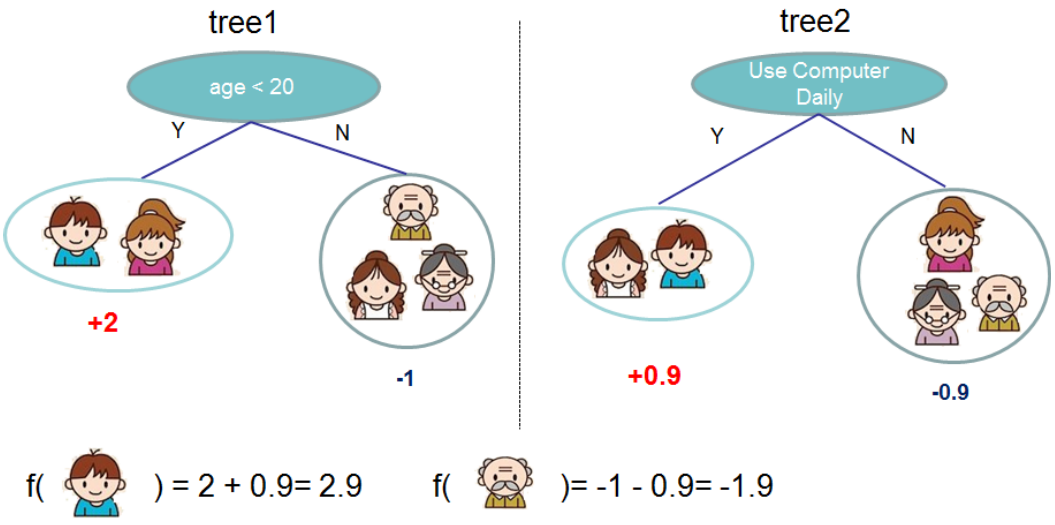

Figure 2.7: Example of Decision Tree Ensembles by Tianqi Chen [7].

Chapter 3

State-of-the-art

Traffic accidents are a big problem for society, and thus has been a topic of interest for both researches and governments for a long time. Several attempts have been made to identify the factors involved with traffic accidents, so that measures can be made to reduce the number of, and the impact of traffic accidents. The literature can be roughly separated into two types, namely classification and regression problems. [2]

3.1 Regression

Classification models aim to classify a given road segment into binary classes: accident / no-accident. Regression models, on the other hand, try to predict the number of traffic accidents on specific roads or regions. Regression analysis is also used for identifying correlations between attributes and the accident risk. [2]

Hadi et al. proposed several accident prediction models, including Poisson and Negative Binomial regression models. In their work, they considered both multi-lane and two-lane roads of urban and rural designation. They concluded that accident rate increases with increasing Annual Average Daily Traffic (AADT) on roads having higher levels of traffic, while it decreases with AADT on roads with lower traffic volumes. [33, 14]

Similarly, Caliendo et al. developed Poisson, Negative Binomial, and Negative Multinomial regression models for predicting the frequency of accident occurrence, although only multi-lane rural roads were considered. They considered variables such as stopping sight distance and pavement surface characteristics and conclude that their results can be used as a reference for engineers in adjusting or designing multi-lane roads. [14]

Eisenberg et al. used a Negative Binomial regression approach to study the relationship between precipitation and traffic crashes. They found a negative relationship between monthly precipitation and monthly fatal crashed, and a positive relationship in their daily level analysis. They claim their findings can be used beneficially in forming policy interventions. [16]

In a recent paper by Yuan et al., a Convolutional Long Short-Term Memory Neural Network model was used to predict accidents. They make use of road, weather, time, satellite and traffic data for the state of Iowa (USA) between 2006 and 2013. They highlight the challenge of spatial heterogeneity in traffic accident prediction, i.e., the factors that lead to traffic accidents in an urban environment might be very different from those in a rural environment. They incorporate the spectral structure of the road network into their model through eigen-analysis. They divide the state of Iowa into a grid, where each cell is a $d \times d$ square region and learns a model to predict the number of accidents in each cell for a given time slot. They use 31 features grouped into seven different categories, where each feature is converted into a three-dimensional ($64 \times 128 \times 1$) feature tensor. A new model is trained for each cell by using the data inside by using sequences of 14 days, where the target is the last 7 days. The performance of the model is measured by using mean squared error, root mean square error, and cross entropy, and the results are promising, giving lower loss than other methods tested, such as Deep Neural Network (DNN) and Decision Tree (DT). [34]

3.2 Classification

Much work has also been done on classification models. While regression models are used for finding relationships between variables and predicting the number of, for instance, traffic accidents, some problems take the form of placing data into a given number of classes.

Chang et al. compared Poisson and Negative Binomial regression models with a Neural Network (NN) when predicting freeway accident frequency in Taiwan. They conclude that NN is a consistent alternative method for analyzing freeway accident frequency compared to traditional statistical methods. They also point out the fact that the statistical models have a predefined relationship between dependent and independent variables, which may lead to incorrect results if violated. [35]

Lin et al. compared the k-nearest neighbor algorithm with a Bayesian network, in addition to introducing a Frequent Pattern tree (FP tree) based variable selection method. Their work is based on accident data collected on an interstate highway in Virginia (USA). They point out that available data often are very complex and noisy, hence the need for good variable selection. Their best model is an FP tree based Bayesian network with an accuracy of 61.11%. [36]

Yuan et al. compared four classification models: Support Vector Machine, DT, Random Forest (RF), and DNN. They framed their problem as a binary classification of accident / no-accident on a given road segment in a one hour time window and achieved an impressive accuracy of 95.12% with their DNN model. They contribute their attention to spatial heterogeneity and class imbalance as being the primary reasons for their excellent results. They also point out that in much prior research, rather simple data sets were used, e.g., lacking weather data, whereas they included detailed road network information and hourly weather data. It is important to note that the authors themselves point out the fact that the results based on different data sets cannot be compared directly. Regardless, to the best of our knowledge, their work has produced some of the most impressive results ever presented. [2]

In a non-scientific medium, Daniel Wilson at Esri used a similar approach as Yuan et al., where he used 7 years of accident data from Utah (USA). He also framed the problem as a classification problem with accident / no-accident on a road segment within a one hour time window. The data includes weather, time-related features, static features, human factors, and graph derived features, and he also takes class imbalance into account. He tested multiple models such as DNN and Gradient Boosting Machine and achieved promising results. [37]

Chapter 4

Data

4.1 Data sources

Two data sources are used, one for road and accident data, and another for weather data. The data is cleaned up and joined according to matching geographical positions of a given road and a number of weather stations, more details will follow in Section 4.2.

4.1.1 Accident data

The traffic accident data is acquired from the Norwegian Public Roads Administration (NPRA). The data contains detailed information on 268,514 traffic accidents from today all the way back to 1975 [8]. Figure 4.1[1] shows the location of all these accidents, placed on top of a map of Norway. This data is collected as a single dsv[2] file from NPRA's *Vegkart* service [8]. Each accident is recorded with 70 data points, for instance: date, time, temperature, road id, number of people involved, exact location, temperature, and speed limit [39].

[1]Accidents are drawn on map from GADM [38].

[2]delimiter-separated values.

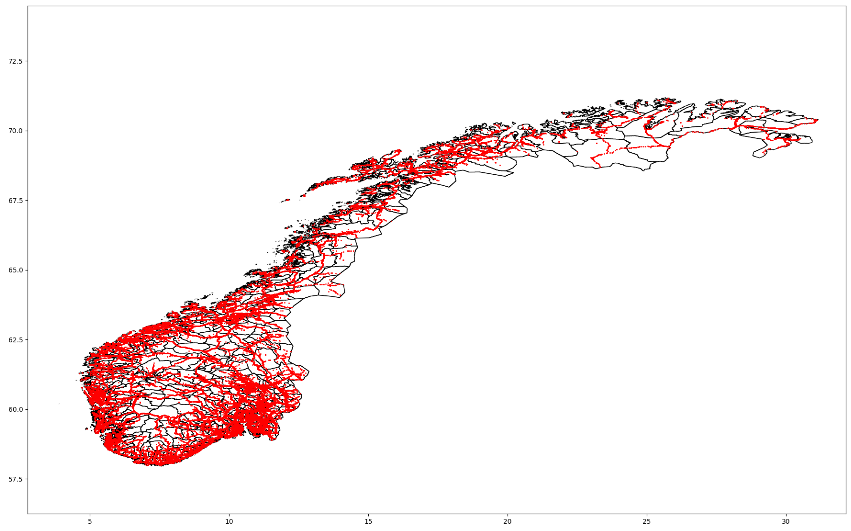

Figure 4.1: All accidents in Norway from 1975-2019 [8].

4.1.2 Road data

Road information is also provided by NRPA. The available data represents Norway's official road network, including both geometry and topology. The data is acquired through the National Road Database REST API [40]. A lot of information is stored in road objects, e.g., speed limit, traffic amount, and tunnels. The database currently contains 399 different road objects [41]. This data source is necessary in order to generate negative samples (no-accident), see Section 4.3.

4.1.3 Weather data

Historical weather data is acquired from the Norwegian Meteorological Institute, through the Frost API. The API provides data from weather stations across Norway, note that not all stations provide the same kinds of data. Some examples of measurements are air temperature, grass temperature, precipitation amount, and wind gust speed. [42]

4.2 Combining data sources

All traffic accidents in Norway from 1975 up until today are used as a starting point, totaling 268,514 accidents. Even though most of the accidents contains all the features we are interested in on their own, including road and weather data, we need a way of obtaining the same features when we generate negative samples. This is why we gather not just accident data, but road and weather data from other sources as well.

To ensure consistency between the positive and negative samples, we decide to not use the road and weather data from the accident data. As an example, we do not want temperature readings for a given position at a given time to be different depending on the data source. From the available data sources, several data points are used as the basis of our features. Some of the most notable are[3]:

- **date:** accident date (YYYY-MM-DD). Acquired from accident data.
- **time:** time of accident (HH:MM). Acquired from accident data.
- **geometry, point:** geographical location of accident. Formatted as well-known text (wkt). Acquired from both accident and road data.
- **road reference:** road identifying string. Acquired both from accident and road data.

These are foundational in matching the different data sources. For instance, using the road reference on a given accident, we can acquire the speed limit, road length, and the number of connected intersections through the road data source. Similarly, when gathering air temperature through the weather data source, we use both time and place information when querying the Frost API[4].

For each accident, a circular polygon[5] is drawn around it with a radius of 4 kilometers[6]. The accident air temperature and precipitation amount are

[3]The names do not necessarily map one-to-one with named in the different data sources

[4]The Frost API does not accept well-known text. We need to convert the accident point geometry to lat-long as a result. A lot of similar work is done, but not stated in detail.

[5]Made up of 20 vertexes

[6]This number is solely based on Yuan et al. using 4×4 km cells [2]

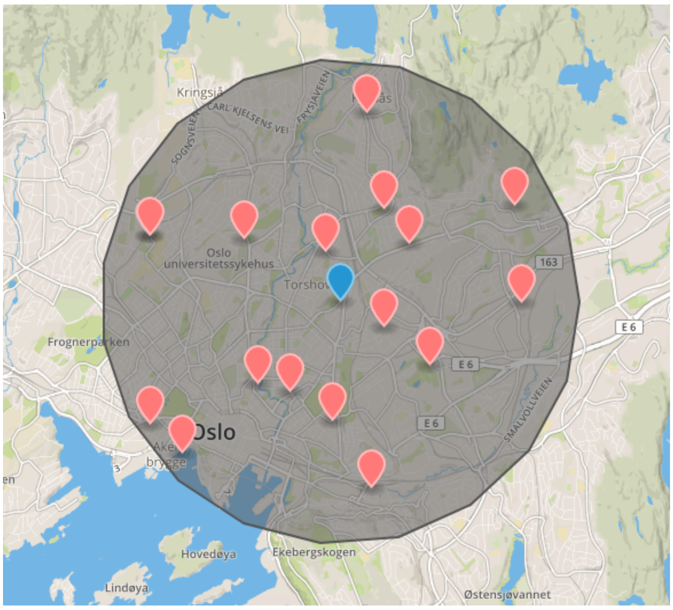

Figure 4.2: Accident with sensors which is inside the polygon with radius of 4 kilometers.

calculated as an average of all weather stations within the polygon. If the polygon contains no weather stations, or solely weather stations with no air temperature and precipitation measurements, the accident is discarded. See Figure 4.2[7].

Time slot granularity is set at one hour[8], meaning that an accident that took place at, for instance, 14:13 is rounded to 14:00. Weather data is collected from this hour[9] so that the accident at 14:13 gets air temperature and precipitation reading between 14:00 and 15:00. If the hour contains more than one reading, the average is used. Other weather-related features are considered, like grass temperature and wind gust speed, but their inclusion would reduce the number of samples drastically.

After combining all data sources, we are left with 14,492 accidents. The main reason for the drop from 268,362 to 14,492 accidents is a lack of nearby weather stations. An alternative method of gathering weather data is pro-

[7]Visualized using geojson.io [43].

[8]The same as used by Yuan et al. [2]

[9]The Norwegian Meteorological Institute stores its data in Coordinated Universal Time (UTC) time, while the NPRA uses Norwegian local time. This is corrected for when sampling.

posed in future work, Section 7.2. Some samples are also lost due to missing data points in the accident and road data. We found Average Annual Daily Traffic (AADT) to be the road related feature that cost us the most samples.

4.3 Negative sampling

In order to train a classifier, negative samples are needed. Given the one-hour time slot, every hour of every day on a road with no accident can be treated as a negative sample. This would cause a major class imbalance in favor of no-accident. Instead, we perform sampling as proposed by Yuan et al.: generating samples at a ratio of 3x that of positive samples[10].

For each positive sample, we generate three negative ones by randomly changing the value of only one feature among hour, day, and road segment. If a generated sample exists as a positive sample, it is discarded, and a new one is generated. The idea is that changing each of these three features represent varying levels of difficulty, from changing the road (hard) to changing the hour (hardest). After negative sampling, we are left with a total of 52,835 samples. Figure 4.3[11] shows the location of all positive accidents in blue, and all negative accidents in red, placed on top of a map of Norway.

4.4 Preprocessing

After collecting all the raw data we need from our data sources, we need to do some filtering and feature transformation. We start with some simple transformations, like transforming a date into an integer $i \in [0, 365)$, and calculating road segment length based on wkt LineStrings. This makes it easier to plot the data and make some observations. Figure 4.4 shows a histogram of some early features.

Some of the immediate observations we can draw from Figure 4.4 are:

[10]An error during sampling resulted in under-sampling of negative road samples: approx. 10,000 not approx. 14,000

[11]Accidents are drawn on map from GADM [38].

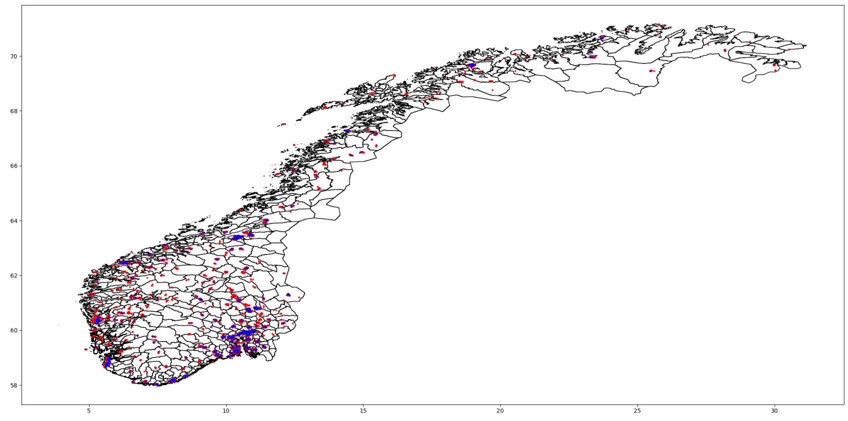

Figure 4.3: Accidents used in our dataset, where the red are negative samples and the blue are real accidents.

- Accident count spike around 7-8 a.m. and 3-4 p.m.
- Most of the accidents happens on roads with a 50 km/h speed limit. This might indicate that most accidents happen in urban areas.
- Number of pitch and reversible lanes are zero for all accident. Indicating that these are quite rare[12].

These observations seem to agree with common knowledge and suggest there is more to the data than mere noise. A machine learning model should, therefore, be able to learn from the data.

Missing values have largely been dealt with during data acquisition; this could have been done in the preprocessing pipeline. As of now, removal of invalid precipitation the only one added to the pipeline. As stated in the Frost documentation [45], we convert precipitation values of -1 to 0. Other negative precipitation values are removed altogether.

Most of the features go through some form of encoding. Road type is the only nominal feature and goes through one-hot-encoding. The number of different lanes are extracted from a single field and encoded as six different features (count of each field type). Labels are simply encoded as 1 (accident) and 0 (no-accident).

[12]A road can contain six different lane types: ordinary, bus, turn, bike, pitch, and reversible. A detailed description can be found in the NPRA handbook [44]

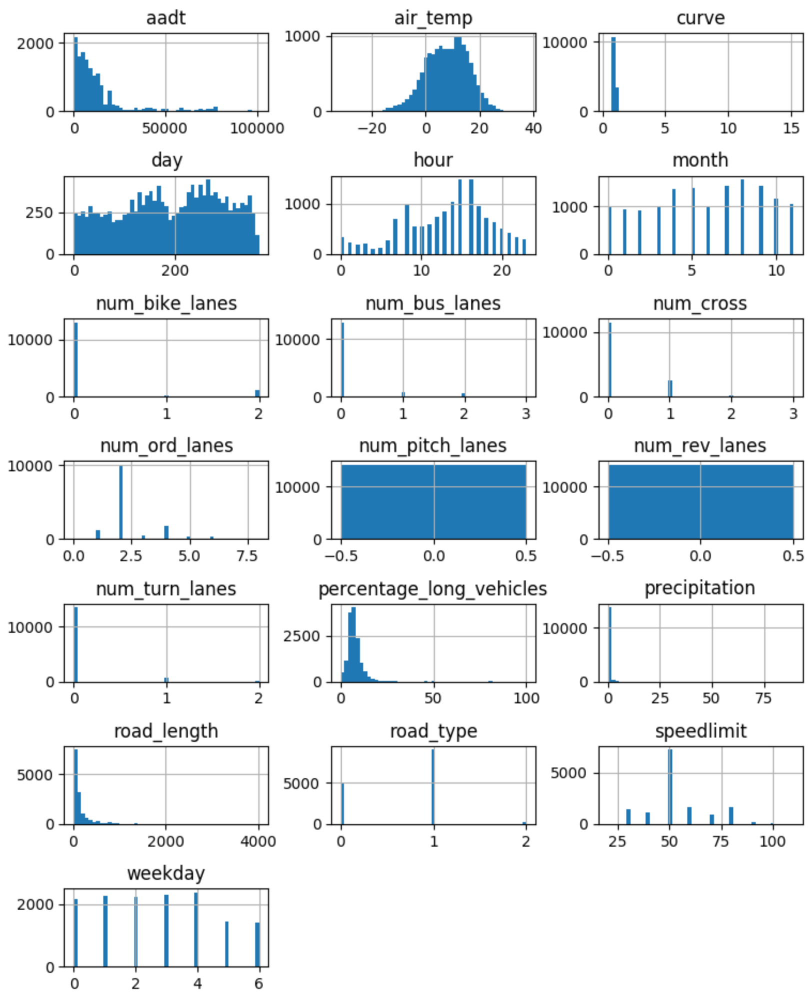

Figure 4.4: Histogram of feature values for the positive samples. The y-axis denotes crash count.

Feature	Description	Feature type
AADT	Annual Average Daily Traffic	Human
percentage_long_vehicles	Percentage of AADT being long vehicles. Vehicles longer than or equal to 5.6 meters is considered a long vehicle [?]	Human
road_length	Length of road segment	Road
hour_x	Cosine component of hour	Temporal
hour_y	Sine component of hour	Temporal
weekday_x	Cosine component of weekday	component
weekday_y	Sine component of weekday	Temporal
day_x	Cosine component of day (of the year)	component
day_y	Sine component of day (of the year)	Temporal
month_x	Cosine component of month	component
month_y	Sine component of month	Temporal
speedlimit	Speed limit of road segment in km/h	Road
curve	As proposed by Yuan et al.: $curve_{road} = \frac{length(r)}{dist(r.start, r.end)}$	Road
air_temp	Air temperature in °C	Weather
precipitation	Precipitation amount in mm across time slot	Weather
num_cross	Number of roads connected to road segment	Road
x0_channeled_road	Channaled road: true/false (result of one-hot-encoded road type)	Road
x0_ordenary_road	Ordinary road: true/false (result of one-hot-encoded road type)	Road
x0_roundabout	Roundabout: true/false (result of one-hot-encoded road type)	Road
num_ord_lanes	Number of ordinary lanes	Road
num_bike_lanes	Number of bike lanes	Road
num_bus_lanes	Number of bus/taxi lanes	Road
num_turn_lanes	Number of turn lanes (lane that transitions into another)	Road
num_pitch_lanes	Number of pitch lanes (a type of parking area)	Road
num_rev_lanes	Number of reversible lanes (lanes that can change direction)	Road

Table 4.1: All *default* features, 25 total.

All of the temporal features are inherently cyclical. This is not properly represented if we leave, for instance, values for hour at [0, 24). We need the difference between 0 and 23 to be the same as the difference between 5 and 6. One way of doing this is to encode the temporal features as a pair of sine and cosine values:

$$\forall e \in f.e_{sin} = sin\left(\frac{2\pi e}{max(f)}\right)$$
$$\forall e \in f.e_{cos} = cos\left(\frac{2\pi e}{max(f)}\right)$$

This method ensures a smooth transition from one day to another. All the remanding features are numerical by default and need no further encoding. Table 4.1 presents all features after preprocessing. These features are referred to as *default* features throughout this report.

A final optional step in the pipeline is feature scaling. Either standard scaling or min-max scaling can be used. Whether or not this is used is specified for each model in Section 5.4.

The data is partitioned into 80% training data and 20% test data. We use stratified sampling[13] for splitting the data based on sample type. This means that the proportion of positive samples and negative hour, day, and road are equal in both the training and test set.

[13]We use the scikit-learn implementation [46].

Chapter 5

Approach

Our approach is inspired by a recent paper by Yuan et al. [2]. They evaluate four different machine learning models: Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), and Deep Neural Network (DNN). They achieve excellent results with all of them except for the SVM. Because of this, we decide to test DT, RF, and Neural Network (NN) model as well. Additionally, we elect to use Gradiant Boosting Machine (GBM).

This chapter presents our assumptions and environment. It also describes our models mentioned above, the metrics used to measure their performance, and hyperparameter search results. Finally, it details the creation of spatial features.

5.1 Assumptions

In order to use the data from The Norwegian Public Roads Administration, some aspects of the real world have to be simplified. One simplification is to assume that peoples behavior is not represented in the data collected. However, this simplification is not true since peoples behavior does affect accidents, and the driver causes many of the accidents collected. This includes the driver not paying attention, speeding, driving under the influence, and ailment. These factors are not always known, nor public, and it is therefore not possible to exclude these from the data. Therefore the dataset is created without taking human behavior into account and is likely to affect the results. However, with time and new technology these factors are more likely to have less impact. With self-driving cars, human factors can safely be ignored.

Another assumption is that the accidents in the dataset are representative of the road network in Norway. As there exist many accidents without the corresponding weather data or missing other vital features, these are not present in the final dataset. Because of this, not every road segment with an accident from 1975-2019 is present, and therefore there are parts of the road network which are not present in our data. By looking at the accidents in Norway from 1975-2019, as shown in Figure 4.1, the overall road network of Norway is present. By looking at the accidents in our dataset in Figure 4.3, the number of accidents are reduced. It is however assumed that the accidents collected are representative of the road network, as there are accidents present in the overall road structure.

A third assumption is that the weather data collected is correct. During the creation of the dataset, there were some unusual negative precipitation amounts. After contacting the owner of the data we learned that there was some bad time-series data with noise in the interval from -0.3 to +0.3. The noise was seen on certain stations with a simple variant of the algorithm from Geonorm measurements. They had fixed some of them, but there was still some noise in their data. These data were filtered out, and not used in the dataset, but some noise may still exist.

5.2 Environment

The dataset is created to capture the Norwegian infrastructure and to represent it as realistic as possible. When looking at other environments, there are some differences in our data worth mentioning. The Norwegian Road structure is relatively small compared to other countries, with a small amount of multilane-highways. The structure might, therefore, be different from other road networks and impact the data collected. The Norwegian population is also small compared to many other countries. The population is most likely to have an impact on the number of traffic accidents compared to other countries, as there are fewer people there is also most likely fewer accidents. One example is the number of accidents used by Yuan et al. in

Iowa from 2006-2013, where they matched 415,153 accidents [2]. During the same period in Norway, there were 51,123 accidents [47]. The Nordic climate is also different from many other parts of the world, and could therefore have an impact on both the number and location of accidents.

5.3 Metrics

We select seven metrics to measure model performance:

- **Accuracy:** Classification accuracy [48].
- **Balanced Accuracy:** Average of recall obtained on both classes [49].
- **Average Precision:** Summarizes a precision-recall curve [50].
- **Precision:** The ratio $\frac{tp}{tp+fp}$ where tp is the number of true positives and fp is the number of false positives [51]. Intuitively, precision is the ability of the classifier to not label as positive a sample that is negative [52].
- **Recall:** The ratio $\frac{tp}{tp+fn}$ where tp is the number of true positives and fn is the number of false negatives [53]. Intuitively, recall is the ability of the classifier to find all the positive samples [52].
- **F1:** Weighted average of precision p and recall r: $2\frac{(p \times r)}{(p+r)}$. The precision and the recall are equally important. [54]
- **ROC-AUC:** Area Under the Reciever Operating Characteristic Curve [55]. A ROC curve plots recall agains the false positive rate, and the AUC summarizes the plot into a single number. A perfect classifier will have a ROC-AUC equal to 1. [56]

All of these metrics tell us something about how our models are performing. Unfortunately, we cannot reach perfect scores for all of them. The precision-recall tradeoff is a well-known manifestation of this: increasing precision reduces recall, and vice versa [56]. If you are attempting to diagnose cancer, you probably want to optimize for recall, which means that you are willing to accept a few false positives in return for finding more positive samples. On the opposite end, if you are creating a video streaming service for children, you probably want the model responsible for finding appropriate movies to be optimized for precision. You don't want horror movies intended for adults to be labeled as children movies.

We chose to optimize for F1-score, favoring recall over precision. We are willing to classify more negatives as positives, in return for finding a larger portion of the positive samples. We want to identify as many potentially dangerous roads as possible.

5.4 Models

This section presents the four machine learning models used: DT, RF, NN, and GBM, as well as parameter search results. Classification results are presented in Chapter 6.

5.4.1 Decision Tree (DT)

For DT we use the decision tree classifier in scikit-learn [57]. This implementation uses Classification and Regression Trees (CART) as introduced by Leo Breiman [58]. The implementation required no data normalization, but missing values is not supported. As a result of using a white-box model, we can easily extract feature importance. Results are presented in Chapter 6.

Exhaustive hyperparameter search is performed using the grid-search cross-validation model in scikit-learn [59]. We select F1 as the scoring metric, and $k = 5$ for the number of folds. Table 5.1 shows the hyperparameter space for the DT model, and Table 5.2 shows the search results.

Parameter	**Description**	**Values**
max_depth	Maximum tree depth	[1, 10]
criterion	Function to measure split quality	{entropy, gini}
min_samples_split	Minimum number of samples required to split and internal node	[2, 10]
min_samples_leaf	The minimum number of samples required to be at a leaf node	[1, 10]
class_weight	Weights associated with each class	(1 for positive sample and range(0.1, 1.01, 0.1) for negative sample) **OR** (0.726, 0.273)
max_features	The number of features to consider when looking for the best split	$\{\sqrt{n\_features}, log_2(n\_features), n\_features\}$

Table 5.1: Decision Tree hyperparameter space.

Parameter	**Value**
max_depth	6
criterion	entropy
min_sample_split	2
min_sample_leaf	9
class_weight	positive class: 1 negative class: 0.3
max_features	n_features

Table 5.2: Hyperparameter search results for Decision Tree model with F1 as scoring function using all training data.

5.4.2 Random Forest (RF)

For RF we use the random forest classifier in scikit-learn [60]. Unlike the original publication [27], this implementation combines classifiers by averaging their probabilistic prediction instead of letting each classifier vote on a single class [61]. Just like the DT, this model does not require any data normalization, and feature importance is easily accessible.

Exhaustive hyperparameter search is performed using the grid-search cross-validation model in scikit-learn [59]. We select F1 as the scoring metric, and $k = 5$ for the number of folds. The trees that make up this model uses the parameters presented in Table 5.2. As a result, the number of trees to use is the only hyperparameter left to search. The optimal number of trees was found to be 180 over a space of [1, 1500] trees.

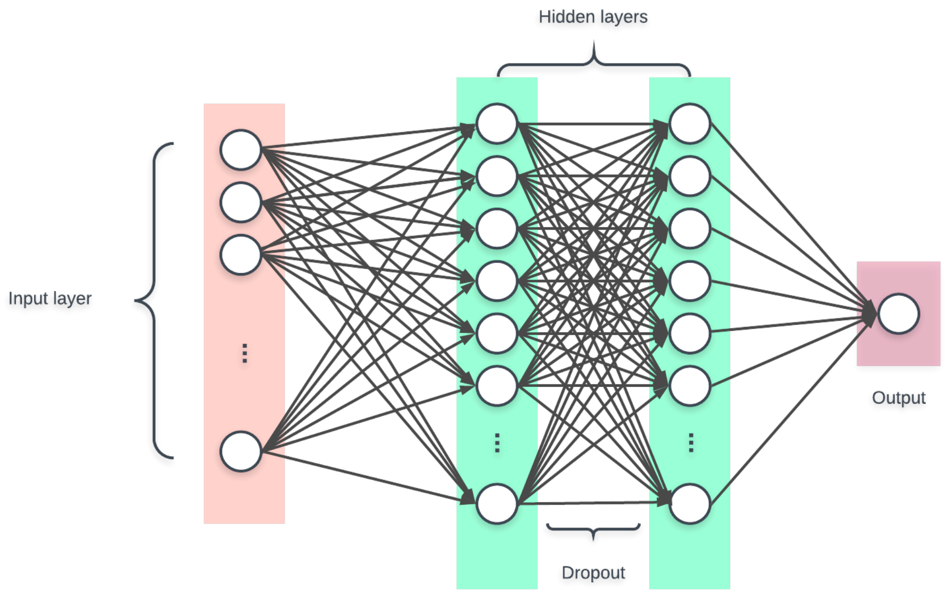

Figure 5.1: Neural Network structure overview.

5.4.3 Neural Network (NN)

The NN model is implemented using Keras and is based on a DNN by Yuan et al. [62]. The network structure is shown in Figure 5.1. The hidden layers consist of 50 neurons each, with a 10% dropout rate. The network itself is not as deep as the one proposed by Yuan et al., as the dataset is smaller. [2]

The final node consists of a Sigmoid output: $\frac{1}{1+e^{-y}}$ [24], which produces an output between 0 and 1. Before the training data is fed to the input layer, each sample is scaled using the following formula: $z = \frac{(x-\mu)}{s}$, where μ is the mean of training samples and s is the standard deviation [63].

Between each layer, the rectifier function is used as the activation function: $rectifier(x) = max(0, x)$ [64]. To avoid overfitting, the dropout is added between the layers. Adaptive Moment Estimation (ADAM) is used as the optimizer function [65], with the default parameters proposed by the authors: $\alpha = 0.01$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$

Parameter	Description	Values
max_depth	Maximum tree depth [67].	{4, 6, 8}
gamma	Loss reduction needed to make a partition on a leaf node [67].	{0.2, 0.5, 1}
colsample_bytree	Column ratio used when a tree is constructed [67].	{0.5, 0.8, 1}
min_child_weight	Instance weight needed in a child [67].	{1, 5, 10}
scale_pos_weight	Weights associated with positive class [67].	{2, 2.5, 3, 4, 4.5, 5}
subsample	Ratio of training samples used [67].	{0.5, 0.8, 1}
learning_rate	Learning rate refered as eta [67].	{0.01, 0.15, 0.3}
n_estimators	Number of trees used [67].	{50, 80, 100, 200}

Table 5.3: Gradient Boosting Machine hyperparameter space.

As the loss function, the binary crossentropy formula is used:

$$\frac{1}{N}\sum_{n=1}^{N}[-y_n log(q_n) - (1-y_n) log(1-q_n)].\ [66]$$

Since the dataset is uneven and contains about 73% negative samples, a boosting of positive samples is chosen. The weight used for positive and negative samples is 1 and 4, respectively, which means that positive samples are 4 times more important.

5.4.4 Gradiant Boosting Machine (GBM)

The GBM model is implemented using the eXtreme Gradient Boosting (XG-Boost) library [31]. XGBoost is a scalable machine learning algorithm for tree boosting, the impact of which has been widely recognized in a number of machine learning challenges [29]. The GBM uses DTs as base-learners, which is a common base learner [32]. An objective function, consisting of training loss and an regularization term, is used for optimization. The objective function in XGBoost with loss l and regularization Ω is defined as:

$$\text{obj} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \qquad .$$

We use the default logistic loss as our loss function. Table 5.3 shows the hyperparameter space for the GBM model, and Table 5.4 shows the search results.

Parameter	**Value**
max_depth	4
gamma	1.
colsample_bytree	0.8
min_child_weight	5
scale_pos_weight	4
subsample	0.8
learning_rate	0.15
n_estimators	50

Table 5.4: Hyperparameter search results for Gradient Boosting Machine model, optimizing for F1-score.

5.5 Creating spatial features

Spatial heterogeneity often refers to an uneven distribution of events or relationships across a region or landscape [68]. Meaning that events or relationships in one region are not necessarily the same in a different region, i.e., accidents in a city may have different properties than accidents in the countryside. According to Yuan et al., spatial heterogeneity is a problem with earlier research on traffic accident prediction. They tend to apply classical prediction on the data, and therefore ignore problems such as and spatial heterogeneity. Therefore an approach similar to the one presented by Yuan et al. is used, in order to address the problem. [2]

The algorithm used to create new features, which represents the spatial heterogeneity in the data (called SpatialGraph features by Yuan et al.), is similar to spectral clustering mentioned in Section 2.5. However, instead of using k-means to cluster the points in the last step of spectral clustering, this step is skipped. It would be interesting to train a model for each cluster, i.e., training the model solely on the data belonging to each cluster, where each cluster represents a region. However, as mentioned by Yuan et al., this is difficult as the data is limited and each model would get few positive accidents. Instead, the resulting top k-eigenvectors are added as additional features for each accident. This means that closely related accidents have similar features to represent the spatial heterogeneity. The algorithm is based on Yuan et al. and Von Luxemburg[1] [9]. [2]

[1]scikit-learns SpectralEmbedding is used in the implementation [69].

```
Algorithm

Let each row of the Similarity matrix $S$ represent a
data point.

1. Create similarity matrix $S \in \mathbb{R}^{n \times n}$ and degree matrix
   $D$, with degrees $d_1, \ldots, d_n$ on the diagonal.

2. Compute normalized laplacian  $L_{norm} = D^{\frac{-1}{2}} L D^{\frac{-1}{2}}$.

3. Compute the first $k$-eigenvectors of $L_{norm}$.

4. Let $T \in \mathbb{R}^{n \times k}$ be the matrix containing the
   $k$-eigenvectors as columns.

5. Let each row of $T$ represent the new
   features for each corresponding data point.
```

Listing 5.1: Spatial feature creation algorithm. Based on Yuan et al. and Ulrike von Luxburg [2, 9].

To create the Similarity matrix, the Gaussian similarity function $s(x_i, x_j) = exp(\frac{-||x_i - x_j||^2}{(2\sigma^2)})$ is used. It is often used as the similarity function with a fully connected graph. The Euclidean distance is computed on the points (x_i, x_j) pairwise, where each point is the position (latitude, longitude). As σ is a free parameter, we test multiple vales of σ to see its effect. We test σ by using the GBM model with values $\{0.5, 1, 1.5, 2\}$, but we see very little difference by changing it. Since the results are similar, we choose to use $\sigma = 1$ throughout the experiments in Chapter 6. [9]

We use the algorithm to generate k = 10, 20, and 30 spatial features for use in our experiments, see Chapter 6.

Chapter 6

Experiments

This chapter presents our results for the four models: Decision Tree (DT), Random Forest (RF), Neural Network (NN), and Gradient Boosting Machine (GBM). For each model, we compare the validation and test results and present feature importance if possible. Table 6.1 shows the baseline results for a naive classifier guessing positive and negative proportional to the amount of positive and negative samples. This should give an idea as to how good our results turn out.

The metrics used for measuring model performance is described in Section 5.3. Each model is trained four times, once with the *default* features as described in Table 4.1, and three times with 10, 20, and 30 spatial features added, see Section 5.5. Validation is done using 5-fold cross-validation, and the test results are based on a single training and classification cycle using a static seed for all models.

6.1 Decision tree (DT)

Figure 6.2a shows the validation results, and Figure 6.2b shows the test results. We see that the impact of spatial features is minimal and that the validation and test results are very similar.

Looking at the confusion matrix in Figure 6.1 we can see that the model is able to label about 85% of the positive samples correctly but at the cost

Metric	Score
accuracy	60.3%
balanced_accuracy	50.1%
average_precision	27.5%
ROC-AUC	50.1%
recall	27.4%
precision	27.6%
F1	27.5%

Table 6.1: Baseline results.

Metric	Default	k=10	k=20	k=30
accuracy	46.3%	46.9%	47.0%	47.2%
balanced_accuracy	58.5%	58.5%	58.5%	58.3%
average_precision	35.2%	35.1%	35.1%	35.1%
ROC-AUC	62.6%	62.5%	62.5%	62.4%
recall	85.5%	84.2%	84.1%	82.9%
precision	32.0%	32.1%	32.1%	32.1%
F1	46.6%	46.5%	46.5%	46.2%

a Validation results.

Metric	Default	k=10	k=20	k=30
accuracy	47.6%	46.4%	46.4%	47.2%
balanced_accuracy	58.2%	58.1%	58.1%	58.4%
average_precision	34.8%	34.7%	34.7%	35.1%
ROC-AUC	62.2%	62.1%	62.1%	62.7%
recall	82.0%	84.0%	84.0%	83.3%
precision	32.2%	31.9%	31.9%	32.2%
F1	46.2%	46.3%	46.2%	46.4%

b Test results.

Table 6.2: Decision Tree validation and test results for *default* features and $k = 10, 20, 30$ spatial features.

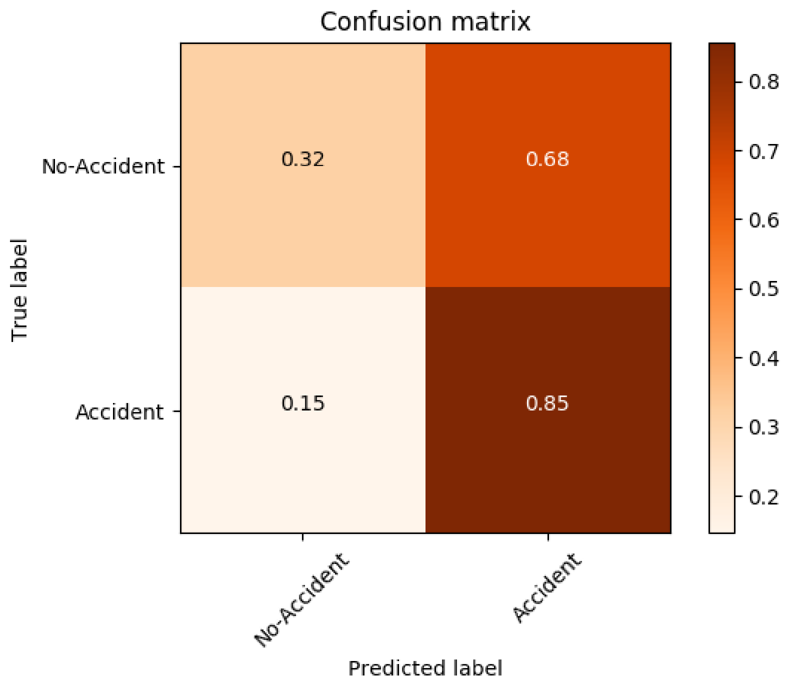

Figure 6.1: Decision Tree CM validation results with *default* features.

of mislabeling 68% of the negative samples. This means that the model is only accurate 32% of the time. Figure 6.2 shows a precision-recall curve for the DT model. We see that the precision gets worse at a steady pace as recall increases until the model classifies every sample as positive.

In Table 6.3 we see the top 15 features sorted by importance to the model. We see that the top six stays the same regardless of spatial features and that the top spatial features come in at number seven for the model with 30 spatial features. It seems like the spatial features overall provide very little information.

Looking at Table 6.4, we see that the negative road samples were the easiest to classify correctly, followed by hour and day. Initially, it was speculated that negative hour samples would be harder to classify than negative day samples, but this does not seem to be the case.

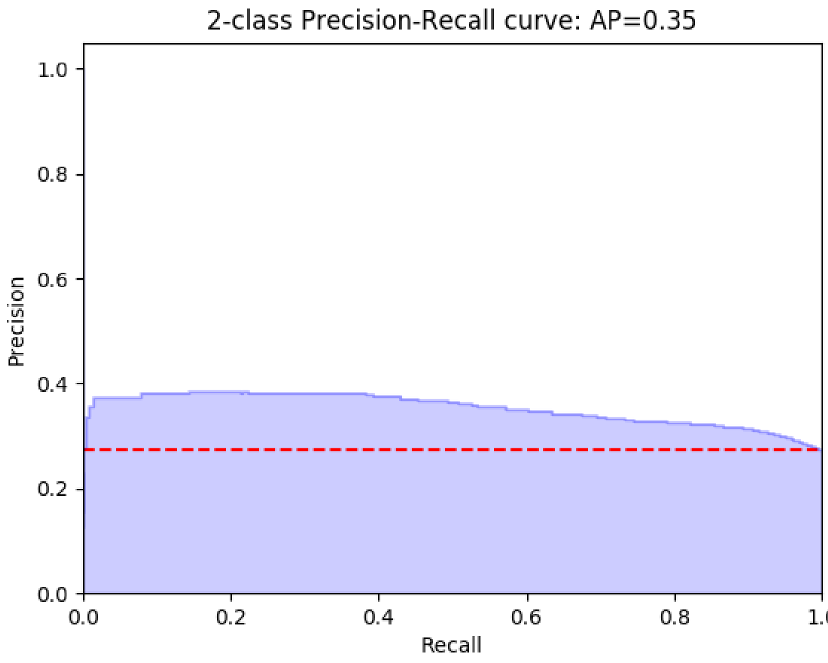

Figure 6.2: Decision Tree precision-recall curve for test prediction with *default* features.

#	Default		$k = 10$		$k = 20$		$k = 30$	
1	AADT	(20.9%)	AADT	(19.9%)	AADT	(19.7%)	AADT	(19.3%)
2	weekday_x	(17.1%)	weekday_x	(16.9%)	weekday_x	(16.7%)	weekday_x	(17.2%)
3	hour_x	(15.7%)	hour_x	(15.5%)	hour_x	(15.3%)	hour_x	(15.5%)
4	weekday_y	(13.8%)	weekday_y	(13.6%)	weekday_y	(13.5%)	weekday_y	(12.1%)
5	road_length	(8.8%)	road_length	(9.3%)	road_length	(9.2%)	road_length	(8.4%)
6	x0_channeled_road	(4.1%)	x0_channeled_road	(4.1%)	x0_channeled_road	(4.0%)	x0_channeled_road	(4.0%)
7	percentage_long_vehicles	(4.1%)	num_ord_lanes	(3.3%)	num_ord_lanes	(3.3%)	spec23	(2.9%)
8	num_ord_lanes	(3.3%)	percentage_long_vehicles	(3.0%)	percentage_long_vehicles	(2.5%)	speedlimit	(2.8%)
9	air_temp	(3.2%)	speedlimit	(2.6%)	speedlimit	(2.5%)	num_ord_lanes	(2.8%)
10	speedlimit	(2.3%)	air_temp	(2.3%)	air_temp	(2.3%)	air_temp	(2.6%)
11	curve	(2.2%)	spec5	(1.8%)	spec5	(1.7%)	percentage_long_vehicles	(2.1%)
12	day_x	(1.5%)	spec6	(1.3%)	spec19	(1.4%)	spec28	(1.7%)
13	hour_y	(1.4%)	curve	(1.2%)	curve	(1.2%)	spec21	(1.3%)
14	day_y	(0.6%)	hour_y	(1.1%)	hour_y	(1.1%)	curve	(1.2%)
15	precipitation	(0.5%)	day_x	(1.1%)	spec6	(1.0%)	hour_y	(1.1%)

Table 6.3: Feature importance on training data for all four Decision Tree models.

Sample type	Retrieved
hour	34.0%
positive	85.5%
day	20.1%
road	45.0%

Table 6.4: Percentage of retrieved samples from each sample type on test data using Decision Tree model with *default* features.

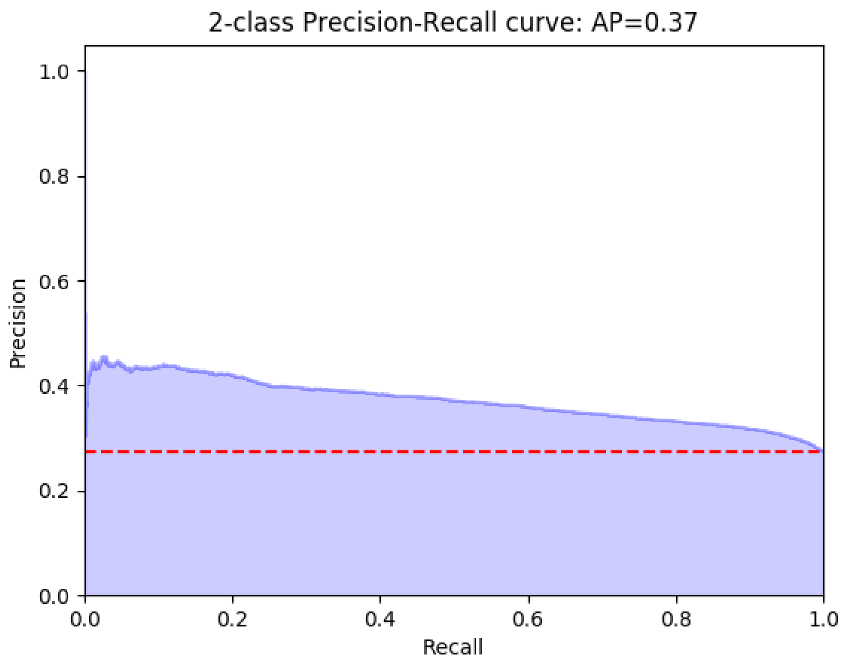

Figure 6.3: Random Forest precision-recall curve for test prediction using *default* features.

6.2 Random Forest (RF)

Figure 6.5a shows the validation results, and Figure 6.5b shows the test results. We see that the impact of spatial features is minimal and that the validation and test results are very similar. The RF model provides slightly better F1-score compared to the DT model.

Table 6.6 shows that most of the improvement comes from correctly classifying more negative roads. Looking at Figure 6.3, we can see that the RF model has a higher precision peak compared to the DT model.

Looking at Table 6.7 we see that the top five features are the same as for the DT model. Overall the improvement from DT to RF is marginal.

Metric	Default	k=10	k=20	k=30
accuracy	48.0%	47.8%	47.8%	47.7%
balanced_accuracy	59.4%	59.4%	59.4%	59.4%
average_precision	37.5%	37.5%	37.5%	37.6%
ROC-AUC	64.1%	64.2%	64.2%	64.2%
recall	84.5%	84.8%	85.0%	85.1%
precision	32.6%	32.6%	32.6%	32.6%
F1	47.1%	47.1%	47.1%	47.2%

a Validation results.

Metric	Default	k=10	k=20	k=30
accuracy	47.9%	47.8%	47.6%	47.4%
balanced_accuracy	59.2%	59.4%	59.4%	59.4%
average_precision	37.0%	37.1%	37.2%	37.4%
ROC-AUC	63.7%	63.8%	63.9%	64.0%
recall	84.5%	85.2%	85.4%	86.1%
precision	32.6%	32.7%	32.7%	32.6%
F1	47.1%	47.3%	47.3%	47.3%

b Test results.

Table 6.5: Random Forest validation and test results with *default* features and $k = 10, 20, 30$ spatial features.

Sample type	Retrieved
hour	35.0%
positive	84.5%
day	21.0%
road	52.6%

Table 6.6: Percentage of retrieved samples from each sample type on test data. Random forest with *default* parameters.

#	**Default**		$k = 10$		$k = 20$		$k = 30$	
1	AADT	(19.6%)	AADT	(19.0%)	AADT	(18.1%)	AADT	(17.3%)
2	weekday_x	(15.8%)	weekday_x	(15.4%)	weekday_x	(15.0%)	weekday_x	(14.6%)
3	hour_x	(13.9%)	hour_x	(13.7%)	hour_x	(13.4%)	hour_x	(13.1%)
4	weekday_y	(12.9%)	weekday_y	(12.6%)	weekday_y	(12.2%)	weekday_y	(11.9%)
5	road_length	(10.2%)	road_length	(9.7%)	road_length	(9.4%)	road_length	(8.9%)
6	curve	(4.4%)	curve	(4.0%)	speedlimit	(3.5%)	speedlimit	(3.4%)
7	percentage_long_vehicles	(4.2%)	speedlimit	(3.8%)	curve	(3.4%)	curve	(2.9%)
8	speedlimit	(4.1%)	num_ord_lanes	(2.8%)	num_ord_lanes	(2.6%)	num_ord_lanes	(2.4%)
9	air_temp	(3.0%)	percentage_long_vehicles	(2.6%)	spec19	(2.5%)	air_temp	(2.1%)
10	num_ord_lanes	(2.9%)	air_temp	(2.6%)	air_temp	(2.3%)	x0_channeled_road	(2.1%)
11	x0_channeled_road	(2.4%)	x0_channeled_road	(2.2%)	x0_channeled_road	(2.2%)	spec23	(1.7%)
12	day_y	(2.4%)	day_y	(2.1%)	percentage_long_vehicles	(1.8%)	spec21	(1.6%)
13	day_x	(1.9%)	day_x	(1.7%)	day_y	(1.8%)	day_y	(1.5%)
14	precipitation	(0.8%)	spec5	(1.6%)	spec18	(1.6%)	spec28	(1.4%)
15	hour_y	(0.8%)	spec6	(1.5%)	spec17	(1.5%)	day_x	(1.2%)

Table 6.7: Feature importance for Random Forest model on all training data.

6.3 Neural Network (NN)

The NN is trained with the network structure discussed in Section 5.4.3, for 15 epochs with a batch size of 30. When training the network, we see that the loss is decreasing, but the validation loss is fluctuating. This indicates that the model is overfitting, which is the case since we are boosting the positive samples. Based on the number of positive samples in each batch we therefore get fluctuating validation loss and variable accuracy. However, training for more epochs gives little difference as the model overfits very early.

The 5-fold cross-validation results are shown in Table 6.8a. The validation results show the F1-score is lower than DT and RF, but has a higher recall. The results are shown in Table 6.8b, where the NN shows better results on the F1-score and recall than the validation set. The spatial features give very similar results as using the *default* parameters. However, the accuracy is a bit higher when using spatial features, and the recall is a little lower. Looking at the accuracy and F1-score we see that when recall is lower, accuracy is higher. This indicates that the spatial features make the network predict more negatives. When the number of positive predictions is lower, the accuracy gets higher since we are more likely to find a negative sample. This makes the recall go down since we are classifying fewer true accidents. However, the validation data is very similar to the test data, indicating that we can classify unseen data by using the NN.

6.4 Gradient Boosting Machine

The validation and test results are shown in Table 6.9a and Table 6.9b. The results from using GBM shows slightly better results over DT and NN in terms of F1-score, recall, and ROC-AUC. GBM show similar results as RF, but with minor differences where the F1-score is slightly lower with certain spatial features. It shows higher recall with lower accuracy, indicating that it predicts more positive samples than RF. The model shows very similar results on both validation and testing data, but with higher F1, recall and ROC-AUC scores on the test data. This shows that the model is capable of learning from the data it has seen and adapt to unseen data. By looking at the spatial features in the GBM model we see that the F1-score is equal

Metric	**Default**	**k=10**	**k=20**	**k=30**
accuracy	45.4%	45.3%	45.8%	45.7%
balanced_accuracy	58.1%	58.1%	58.2%	58.2%
average_precision	36.3%	35.7%	35.4%	35.7%
ROC-AUC	62.8%	62.3%	62.4%	62.5%
recall	86.2%	86.3%	85.4%	85.7%
precision	31.8%	31.7%	31.8%	31.8%
F1	46.4%	46.4%	46.4%	46.4%

a Validation results.

Metric	**Default**	**k=10**	**k=20**	**k=30**
accuracy	44.7%	44.9%	45.3%	45.0%
balanced_accuracy	58.0%	58.0%	58.1%	58.1%
average_precision	36.2%	35.9%	36.0%	36.0%
ROC-AUC	62.8%	62.6%	62.6%	62.8%
recall	87.6%	87.0%	86.5%	87.1%
precision	31.7%	31.7%	31.8%	31.7%
F1	46.5%	46.4%	46.5%	46.5%

b Test results.

Table 6.8: Neural Network validation and test results with *default* features and $k = 10, 20, 30$ spatial features.

Metric	Default	k=10	k=20	k=30
accuracy	46.3%	46.1%	46.8%	46.8%
balanced_accuracy	59.2%	58.8%	59.1%	59.1%
average_precision	36.7%	36.7%	36.6%	36.7%
ROC-AUC	63.7%	63.6%	63.6%	63.6%
recall	87.6%	86.8%	86.4%	86.2%
precision	32.3%	32.1%	32.4%	32.3%
F1	47.2%	46.9%	47.1%	47.0%

a Validation results.

Metric	Default	k=10	k=20	k=30
accuracy	45.7%	46.5%	46.2%	46.5%
balanced_accuracy	58.8%	59.2%	58.9%	59.2%
average_precision	36.8%	37.4%	37.1%	37.0%
ROC-AUC	63.7%	64.2%	63.9%	63.9%
recall	87.9%	87.2%	87.1%	87.5%
precision	32.1%	32.4%	32.3%	32.4%
F1	47.1%	47.3%	47.1%	47.3%

b Test results.

Table 6.9: Gradient Boosting Machine validation and test results with *default* parameters and $k = 10, 20, 30$ spatial features.

or higher. This can be explained by the increase in accuracy when using spatial features, as well as lower recall, but with higher precision. This is an indication that the features might give some insights, even though they seem to lower the score on the validation data.

Looking at the feature importance of the top 15 features in the training data (full), we can see that the most important feature is hour_x, and that both AADT, weekday_x, and weekday_y are essential. As the number of spatial features grows, the importance of each feature is less important. However, we notice as the number of spatial features increases they become more important than some of the *default* features. However, the maximum importance of a spatial feature is about 3% with k=10. This indicates that the spatial features give some insight, even though they are not the most important in the dataset and have limited impact.

Figure 6.4, shows a similar curve for both DT and RF, indicating similar learning from the data.

#	**Default**		$k = 10$		$k = 20$		$k = 30$	
1	hour_x	(12.8%)	hour_x	(12.5%)	hour_x	(8.0%)	hour_x	(6.0%)
2	AADT	(9.3%)	AADT	(8.5%)	weekday_y	(6.3%)	weekday_y	(5.8%)
3	weekday_y	(8.7%)	weekday_y	(8.2%)	weekday_x	(5.7%)	AADT	(5.5%)
4	weekday_x	(8.5%)	weekday_x	(7.6%)	AADT	(5.6%)	weekday_x	(4.7%)
5	road_length	(6.2%)	speedlimit	(6.2%)	x0_channeled_road	(4.5%)	num_ord_lanes	(3.5%)
6	speedlimit	(5.8%)	num_ord_lanes	(4.0%)	speedlimit	(4.3%)	speedlimit	(3.2%)
7	num_ord_lanes	(5.7%)	x0_channeled_road	(3.8%)	num_ord_lanes	(4.1%)	spec6	(3.0%)
8	x0_channeled_road	(5.5%)	road_length	(3.4%)	road_length	(3.5%)	spec21	(3.0%)
9	x0_ordenary_road	(4.7%)	spec0	(3.0%)	spec15	(2.6%)	road_length	(3.0%)
10	percentage_long_vehicles	(3.0%)	spec4	(2.9%)	spec18	(2.6%)	spec0	(2.7%)
11	month_y	(2.8%)	spec2	(2.9%)	spec6	(2.6%)	spec24	(2.5%)
12	x0_roundabout	(2.8%)	num_cross	(2.5%)	spec9	(2.5%)	month_x	(2.4%)
13	curve	(2.8%)	hour_y	(2.5%)	spec5	(2.5%)	spec2	(2.1%)
14	num_bus_lanes	(2.7%)	spec7	(2.3%)	curve	(2.2%)	spec28	(2.0%)
15	precipitation	(2.7%)	spec6	(2.2%)	spec2	(2.2%)	spec26	(1.9%)

Table 6.10: Feature importance for Gradient Boosting Machine.

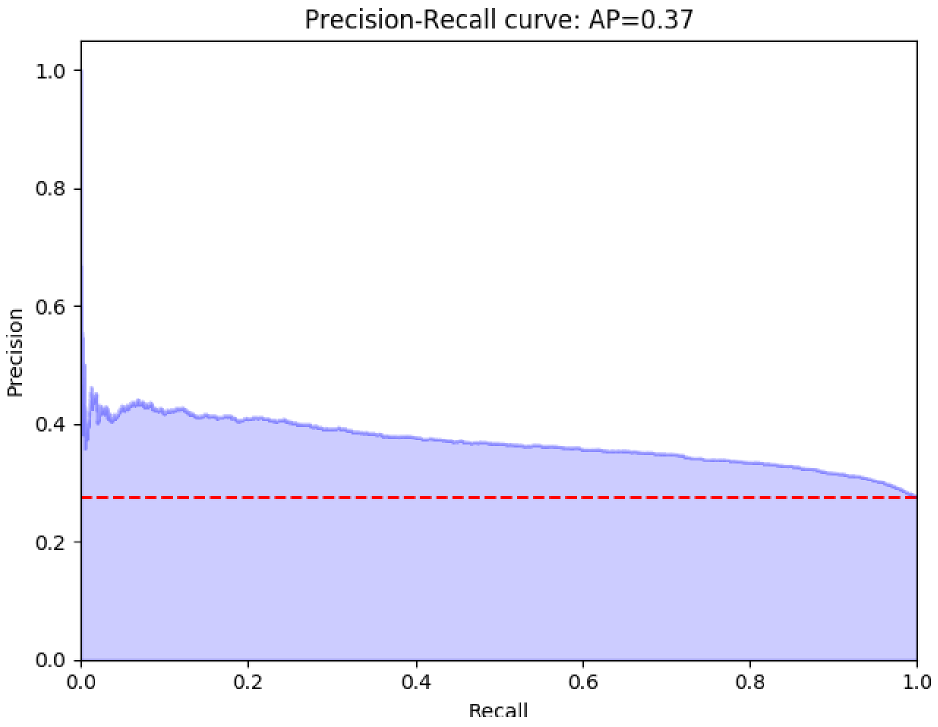

Figure 6.4: Gradient Boosting Machine precision-recall curve for prediction with *default* features

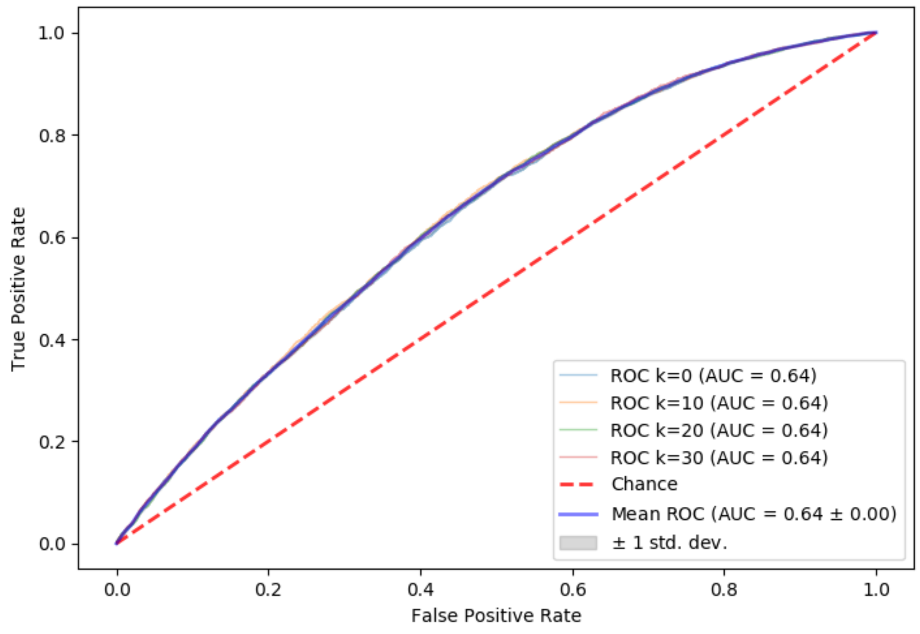

Figure 6.5: Gradient Boosting Machine ROC curve for training data with different k values

Looking at Figure 6.5 we see that the different spatial features are not impacting the ROC-AUC curve, giving very similar values for all values of k.

6.5 Result comparison

In this section we compare our results from the four experiments conducted with the current state-of-the-art and discuss our findings in the Norwegian data.

Table 6.11 shows the different models with k spatial features, where $k = 0$ means only the *default* features are used, see Table 4.1 for details. The table also shows the NN results for Yuan et al., and the GBM results for Daniel Wilson [2, 37]. N/A is used where metrics are not present.

For our data, the color-coded values represent the highest metric value(s) in the test data. As the table shows, RF achieves the highest accuracy, balanced accuracy, and precision score, while GBM achieves the highest ROC-AUC and recall score. F1 and average precision have the same values in RF and GBM. The results are very similar for all models, but RF and GBM are slightly better than DT and NN. Because our goal was to optimize for F1 and recall, the GBM has a small advantage over RF and is considered our best model.

Looking at the results from Yuan et al., they obtain higher accuracy, ROC-AUC, precision, and F1-score. However, their recall is a little lower than ours. This is most likely due to the difficulty in finding real accidents and shows why they have a 4.9% inaccuracy. By looking at the scores with *default* features, we see that our results are more similar to theirs. Their accuracy is comparatively higher, but our F1-score is closer to theirs, since we have a higher recall. Looking at the results with spatial features, we see that their model improves drastically, which is not the case with our experiments. We do, however, achieve the highest balanced accuracy, average precision, ROC-AUC, precision, and F1 values for our models using spatial features, indicating that they improve our models slightly. [2]

Comparing our results to Daniel Wilson, we see that our results are similar, with average precision, recall, and F1 scores being very close. His model has better accuracy, ROC-AUC, recall, but slightly lower precision and F1-score than our model based on their accident data from Utah (USA). [37]

Our experiments for DT, RF, NN, and GBM show very similar results, even though RF and GBM perform the best. Compared to the baseline results in Table 6.1, the results show that all models are capable of approximating the underlying function in the data. Using DT, RF, and GBM, we gain insight into the importance of different features in the dataset. We see that the speed limit is not as important as we first thought and that several temporal features and AADT are essential in predicting accidents. Daniel Wilson also found speed limit to be of little importance [37]. Another interesting insight is that both air temperature and precipitation amount have little impact on the predictions.

Even though our model is not as good as Yuan et al. or Daniel Wilson, it shows promising results and should be a starting point for further research on the Norwegian infrastructure. [2, 37]

	k[1]	Accuracy	Balanced Acc.	Avg. Precision	ROC-AUC	Recall	Precision	F1
DT	0	47.6	58.2	34.8	62.2	82.0	32.2	46.2
	10	46.4	58.1	34.7	62.1	84.0	31.9	46.3
	20	46.4	58.1	34.7	62.1	84.0	31.9	46.2
	30	47.2	58.4	35.1	62.7	83.3	32.2	46.4
RF	0	47.9	59.2	37.0	63.7	84.5	32.6	47.1
	10	47.8	59.4	37.1	63.8	85.2	32.7	47.3
	20	47.6	59.4	37.2	63.9	85.4	32.7	47.3
	30	47.4	59.4	37.4	64.0	86.1	32.6	47.3
NN	0	44.7	58.0	36.2	62.8	87.6	31.7	46.5
	10	44.9	58.0	35.9	62.6	87.0	31.7	46.4
	20	45.3	58.1	36.0	62.6	86.5	31.8	46.5
	30	45.0	58.1	36.0	62.8	87.1	31.7	46.5
GBM	0	45.7	58.8	36.8	63.7	87.9	32.1	47.1
	10	46.5	59.2	37.4	64.2	87.2	32.4	47.3
	20	46.2	58.9	37.1	63.9	87.1	32.3	47.1
	30	46.5	59.2	37.0	63.9	87.5	32.4	47.3
Yuan et al. NN [2]	0	79.7	N/A	N/A	80.4	50.6	61.4	55.4
	10	94.5	N/A	N/A	95.8	85.2	92.2	88.6
	20	95.0	N/A	N/A	96.1	86.6	92.8	89.6
	30	95.1	N/A	N/A	96.1	86.9	93.1	89.7
	40	95.1	N/A	N/A	96.0	86.9	93.0	89.9
Daniel Wilsom - GBM [37]	0	68.6	N/A	38.9	82.8	89.1	31.1	46.2

Table 6.11: Comparison of test results with current state-of-the-art. Scores are listed in percent.

Chapter 7

Conclusion and Future Work

In this Chapter, we discuss our findings, answer the research questions posed in Section 1.1.1, and propose future work.

7.1 Conclusion

This thesis investigates the ability of a Decision Tree (DT), Random Forest (RF), Neural Network (NN) and Gradient Boosting Machine (GBM) model to accurately predict traffic accidents in the Norwegian road system. We also examine the importance of spatial heterogeneity by sourcing features based on accidents' geographical position relative to each other. We present the first models that accurately predict road accidents in a Norwegian setting.

Our results show that DT, RF, NN, and GBM are close in performance, but that their predictive powers are not as good as state-of-the-art. We also find the spatial heterogeneity to be of little importance on the Norwegian data. This could be due to greater distances between the accidents or the fact that we have limited data.

We show that machine learning models can predict accidents with a precision of 33% and a recall of 88%, which is 5 and 61 percentage points better than baseline, respectively. This is however at the cost of lower accuracy than the baseline.

We examine 25 different features and find that the feature with the biggest impact varies between Annual Average Daily Traffic (AADT), hour of the day, and day of the week across the different models. In contrast to what may naturally be thought of as highly relevant, the speed limit is not an accurate predictor of accidents.

Our results suggest that GBM is slightly better at predicting accidents compared to the other models, measured by F1 and recall score.

This thesis shows that the introduced models are capable of learning the underlying function and accurately predict accidents in a Norwegian setting. This shows that machine learning models can be used to get an overview of potential accidents in an area and in turn prevent accidents from occurring in the Norwegian Road System.

7.2 Future Work

To improve the model and be able to predict accidents better, there are several approaches one could explore. One possible improvement is to acquire more data, as the data is limited with regards to both accidents and weather data. Another possibility is to acquire more features for the data already collected, such as other temporal related features. Looking more into features related to spatial heterogeneity is another approach, as there was little impact on the data collected.

Acquiring more data can be done in multiple ways, where one option is to be less strict when collecting weather data. As the weather data reduces the number of samples, a way to get more is to allow a larger radius on the polygon shown in Figure 4.2. Another approach is to use a Voronoi diagram on all weather data sensors and interpolate missing values as proposed by Yuan et al. However, this might be challenging as the time-span is relatively high in the Norwegian data, and sensors having weather-series for only parts of the time period. A solution could be to allow higher time resolutions on the weather data and use average values for each hour; however, this might lead to less accurate values. Since Norway have relatively few accidents, another way to acquire more related data is to consider a larger area than just Norway. Combining data from Norway and Sweden could be a solution but at the cost of having to combine potentially conflicting data sources. [2]

Adding more features for the data is also an aspect one could look into. Adding more features about the weather, roads and other temporal features could give more information about accidents. Calendar features such as holidays and vacations might give valuable information as people often travel during holidays such as Easter and Christmas. Other temporal features such as dusk/dawn could also have an impact on the number of accidents. Norway is huge, and different regions may be very different because of this. The northern part of Norway has both midnight sun and polar darkness, and these features can give new insights about particular regions. [2]

As Norway have both rural and urban areas, the spatial heterogeneity is likely to impact the likelihood of accidents. Therefore, looking more into the creation and effect of the features related to spatial heterogeneity could improve the model. A possibility is to look at an urban and rural area, and only use data from these two areas when creating the features, and check if it has an impact on the two areas.

References

[1] S. sentralbyrå, "Veitrafikkulykker med personskade." https://www.ssb.no/transport-og-reiseliv/statistikker/vtu/aar, 2019.

[2] Z. Yuan, X. Zhou, T. Yang, and J. Tamerius, "Predicting traffic accidents through heterogeneous urban data : A case study," 2017.

[3] "Mnist figure." https://upload.wikimedia.org/wikipedia/commons/thumb/2/27/MnistExamples.png/220px-MnistExamples.png.

[4] M. A. Nielsen, "Neural networks and deep learning," 2018.

[5] "Id3 decision tree figure." https://upload.wikimedia.org/wikipedia/commons/4/46/ID3_algorithm_decision_tree.png.

[6] "Random forest figure." http://www.globalsoftwaresupport.com/wp-content/uploads/2018/02/ggff5544hh.png.

[7] T. Chen, "Introduction to boosted trees." https://homes.cs.washington.edu/~tqchen/data/pdf/BoostedTree.pdf, 2014.

[8] S. vegvesen, "Vegkart." https://www.vegvesen.no/vegkart/vegkart/#kartlag:geodata/hva:(~(farge:'0_0,id:570))/@115265,7084476,3, 2019.

[9] U. von Luxburg, "A tutorial on spectral clustering," *CoRR*, vol. abs/0711.0189, 2007.

[10] R. Bergel-Hayat, M. Debbarh, C. Antoniou, and G. Yannis, "Explaining the road accident risk: Weather effects," *Accident Analysis & Pre-*

vention, vol. 60, pp. 456 – 465, 2013.

[11] S. vegvesen, “Dybdeanalyser av dødsulykker i vegtrafikken 2017.” `https://www.vegvesen.no/_attachment/2346577/binary/1267249?fast_title=Dybdeanalyser+av+d%C3%B8dsulykker+i+vegtrafikken+2017.pdf+`, 2018.

[12] J. Abellán, G. López, and J. de Oña, “Analysis of traffic accident severity using decision rules via decision trees,” *Expert Systems with Applications*, vol. 40, no. 15, pp. 6047 – 6054, 2013.

[13] R. Silva, S. M. Kang, and E. M. Airoldi, “Predicting traffic volumes and estimating the effects of shocks in massive transportation systems,” *Proceedings of the National Academy of Sciences*, vol. 112, no. 18, pp. 5643–5648, 2015.

[14] C. Caliendo, M. Guida, and A. Parisi, “A crash-prediction model for multilane roads,” *Accident Analysis & Prevention*, vol. 39, no. 4, pp. 657 – 670, 2007.

[15] D. W. R. Matthew W. Knuiman, Forrest M. Council, “Assosiation of median width and highway accident rates,” *Transportation Research Record*, vol. 1401, 1993.

[16] D. Eisenberg, “The mixed effects of precipitation on traffic crashes,” *Accident Analysis & Prevention*, vol. 36, no. 4, pp. 637 – 647, 2004.

[17] Politidirektoratet, “Tilstandsanalyse 2017, trafikk.” `https://www.politiet.no/globalassets/04-aktuelt-tall-og-fakta/trafikk/tildstandsanalyse-2017.pdf`, 2017.

[18] Y. LeCun and C. Cortes, “MNIST handwritten digit database,” 2010.

[19] S. Raschka, *Python Machine Learning*. Packt Publishing, 2015.

[20] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[21] Z. C. Lipton, “A critical review of recurrent neural networks for sequence learning,” *CoRR*, vol. abs/1506.00019, 2015.

[22] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, and G. Wang, “Recent advances in convolutional neural networks,” *CoRR*, vol. abs/1512.07108, 2015.

[23] X. Cui, W. Zhang, Z. Tüske, and M. Picheny, “Evolutionary stochastic gradient descent for optimization of deep neural networks,” in *Advances in Neural Information Processing Systems 31* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 6048–6058, Curran Associates, Inc., 2018.

[24] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, Inc., 1 ed., 1997.

[25] W. Peng, J. Chen, and H. Zhou, “An implementation of id3 decision tree learning algorithm,” *From*, 2009.

[26] B. Gupta, A. Rawat, A. Jain, A. Arora, and N. Dhami, “Analysis of various decision tree algorithms for classification in data mining,” *International Journal of Computer Applications*, vol. 163, pp. 15–19, Apr 2017.

[27] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, pp. 5–32, Oct 2001.

[28] A. Y. Ng, M. I. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” in *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pp. 849–856, MIT Press, 2001.

[29] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” *CoRR*, vol. abs/1603.02754, 2016.

[30] N. Ponomareva, S. Radpour, G. Hendry, S. Haykal, T. Colthurst, P. Mitrichev, and A. Grushetsky, “Tf boosted trees: A scalable tensorflow based framework for gradient boosting,” 10 2017.

[31] “Xgboost.” `https://xgboost.readthedocs.io/en/latest/tutorials/model.html`.

[32] A. Natekin and A. Knoll, “Gradient boosting machines, a tutorial,” in *Front. Neurorobot.*, 2013.

[33] M. A. Hadi, J. Aruldhas, L.-F. Chow, and J. A. Wattleworth, “Esti-

mating safety effects of cross-section design for various highway types using negative binomial regression," *Transportation Research Record*, vol. 1500, pp. 169–177, 07 1995.

[34] Z. Yuan, X. Zhou, and T. Yang, "Hetero-convlstm: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &#38; Data Mining*, KDD '18, (New York, NY, USA), pp. 984–992, ACM, 2018.

[35] L.-Y. Chang, "Analysis of freeway accident frequencies: Negative binomial regression versus artificial neural network," *Safety Science*, vol. 43, no. 8, pp. 541 – 557, 2005.

[36] L. Lin, Q. Wang, and A. W. Sadek, "A novel variable selection method based on frequent pattern tree for real-time traffic accident risk prediction," *Transportation Research Part C: Emerging Technologies*, vol. 55, pp. 444 – 459, 2015. Engineering and Applied Sciences Optimization (OPT-i) - Professor Matthew G. Karlaftis Memorial Issue.

[37] D. Wilson, "Using machine learning to predict car accident risk." `https://medium.com/geoai/using-machine-learning-to-predict-car-accident-risk-4d92c91a7d57`, 2018.

[38] "Gadm." `https://gadm.org/maps/NOR.html`, 2018.

[39] N. vegdatabank, "Trafikkulykke." `http://labs.vegdata.no/nvdb-datakatalog/570-Trafikkulykke/`, 2019.

[40] S. vegvesen, "National vegdatabank api." `https://www.vegvesen.no/nvdb/apidokumentasjon/#/`, 2019.

[41] NVDB, "Nvdb datakatalog." `http://labs.vegdata.no/nvdb-datakatalog/`, 2019.

[42] N. M. Institute, "What is frost?." `https://frost.met.no/index2.html`, 2019.

[43] "Geojson.io." `http://geojson.io`.

[44] S. vegvesen, "Nationalt vegreferansesystem." `https://www.vegvesen.`

no/_attachment/61505, 2019.

[45] N. M. Institute, "Explanations for some coded negative value elements." https://frost.met.no/concepts#negativevalueexplanation, 2019.

[46] scikit learn, "sklearn.model_selection.train_test_split." https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html, 2019.

[47] S. vegvesen, "Vegkart." https://www.vegvesen.no/vegkart/vegkart/#kartlag:geodata/hva:(~(farge:'0_0,filter:(~(operator:'*3e*3d,type_id:5055,verdi:(~'2006-01-01)),(operator:'*3c,type_id:5055,verdi:(~'2014-01-01))),id:570))/@792280,6992165,2, 2019.

[48] scikit learn, "sklearn.metrics.accuracy_score." https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html#sklearn-metrics-accuracy-score, 2019.

[49] scikit learn, "sklearn.metrics.balanced_accuracy_score." https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html#sklearn-metrics-balanced-accuracy-score, 2019.

[50] scikit learn, "sklearn.metrics.average_precision_score." https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html#sklearn-metrics-average-precision-score, 2019.

[51] scikit learn, "sklearn.metrics.precision_score." https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html#sklearn-metrics-precision-score, 2019.

[52] scikit learn, "Classification metrics." https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics, 2019.

[53] scikit learn, "sklearn.metrics.recall_score." https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html#sklearn-metrics-recall-score, 2019.

[54] scikit learn, "sklearn.metrics.f1_score." https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn-metrics-f1-score, 2019.

[55] scikit learn, "sklearn.metrics.roc_auc_score." https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html#sklearn-metrics-roc-auc-score, 2019.

[56] A. Gron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* O'Reilly Media, Inc., 1st ed., 2017.

[57] scikit learn, "sklearn.tree.decisiontreeclassifier." https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn-tree-decisiontreeclassifier, 2019.

[58] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees.* Statistics/Probability Series, Belmont, California, U.S.A.: Wadsworth Publishing Company, 1984.

[59] scikit learn, "sklearn.model_selection.gridsearchcv." https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html, 2019.

[60] scikit learn, "sklearn.ensamble.randomforestclassifier." https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, 2019.

[61] scikit learn, "Forests of randomized trees." https://scikit-learn.org/stable/modules/ensemble.html#forests-of-randomized-trees, 2019.

[62] "Keras." https://keras.io/.

[63] scikit learn, "sklearn.preprocessing.standardscaler." https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html, 2019.

[64] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (G. Gordon, D. Dunson, and

M. Dudík, eds.), vol. 15 of *Proceedings of Machine Learning Research*, (Fort Lauderdale, FL, USA), pp. 315–323, PMLR, 11–13 Apr 2011.

[65] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[66] A. Buja, W. Stuetzle, and Y. Shen, "Loss functions for binary class probability estimation and classification: Structure and applications," 01 2005.

[67] "Xgboost parameters." `https://xgboost.readthedocs.io/en/latest/parameter.html`.

[68] "Spatial heterogenity." `http://gispopsci.org/spatial-heterogeneity/`.

[69] scikit learn, "sklearn.manifold.spectralembedding." `https://scikit-learn.org/stable/modules/generated/sklearn.manifold.SpectralEmbedding.html`, 2019.

Appendices

A Source code

Source code is available at: `https://github.com/EinarJohnsen/master`