DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING, SECOND CYCLE, 30 CREDITS

STOCKHOLM, SWEDEN 2017

Analysis of Emergency Medical Transport Datasets using Machine Learning

JOSEFINE LETZNER

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF COMPUTER SCIENCE AND COMMUNICATION

Analysis of Emergency Medical Transport Datasets using Machine Learning

Josefine Letzner
jletzner@kth.se

June 29, 2017

Supervisor Carmenta: Christoffer Rydberg
Supervisor KTH: John Folkesson

Abstract

The selection of hospital once an ambulance has picked up its patient is today decided by the ambulance staff. This report describes a supervised machine learning approach for predicting hospital selection. This is a multi-class classification problem. The performance of random forest, logistic regression and neural network were compared to each other and to a baseline, namely the one rule-algorithm. The algorithms were applied to real world data from SOS-alarm, the company that operate Sweden's emergency call services. Performance was measured with accuracy and f1-score. Random Forest got the best result followed by neural network. Logistic regression exhibited slightly inferior results but still performed far better than the baseline. The results point toward machine learning being a suitable method for learning the problem of hospital selection.

Analys av ambulanstransport medelst maskininlärning

Sammanfattning

Beslutet om till vilket sjukhus en ambulans ska köra patienten till bestäms idag av ambulanspersonalen. Den här rapporten beskriver användandet av övervakad maskininlärning för att förutsåga detta beslut. Resultaten från algoritmerna slumpmässig skog, logistisk regression och neurala nätvärk jämförs med varanda och mot ett basvärde. Basvärdet erhölls med algorithmen en-regel. Algoritmerna applicerades på verklig data från SOS-alarm, Sveriges operatör för larmsamtal. Resultaten mättes med noggrannhet och f1-poäng. Slumpmässig skog visade bäst resultat följt av neurala nätverk. Logistisk regression uppvisade något sämre resultat men var fortfarande betydligt bättre än basvärdet. Resultaten pekar mot att det är lämpligt att använda maskininlärning för att lära sig att ta beslut om val av sjukhus.

Contents

Terminology

Adaboost Short for adaptive boosting. Refers to the process of combining several models into a weighted sum to increase performance.

Bagging Short for bootstrap aggregation. Bagging refers to the process of creating new datasets by sampling uniformly with replacement to create several datasets. The purpose with this method is to obtain better models by training them on the sampled data sets.

Boosting The process of training models sequentially, making use of information gained by the previous models. Each observation has an assigned weight which tells the classifier how important this observation is. The performance of the previous classifier is evaluated and the weights updated based on where the classifier performed poorly. The models are combined to give the final result.

Classification Accuracy A common performance evaluation criteria where the result of a classifier is given by the percent of observations that are assigned to the correct class.

Cross validation Scheme to validate a model by splitting the data into parts and iteratively using one part for testing and the remaining parts for training.

Ensemble method Methods that create several classifiers and combine these by taking a weighted average of their predictions. A example of a ensemble method is random forest.

Feature A chosen variable considered by the model.

Label The class a certain observation is assigned to in the raw data.

Variable Input that describe the data, the features are selected among the variables.

Note: the use of the words variable, feature and label may be used differently in other texts.

Keywords: Machine-learning; EMS management; Multi-class classification; Decision support.

1 Introduction

Predicting future events is useful in many situations and is found in a great variety of areas such as predicting sales rates [1], in social science [2] or predicting the reliability of infrastructure [4]. In this report classification algorithms from the machine learning domain are used with the aim of predicting to which hospital the ambulance should take the patient. A accurate model could work as decision support for ambulance staff dealing with decision making in emergency situations. Saving even minutes in response times for ambulances could save many lives. Since emergency response is such an important function in society, the idea of using algorithms to reduce response time and make closer to optimal decisions is not new. This can be read about in the background, Section 2. Problems more closely related to this project are found in Section 2.1, "Hospital Selection". The remaining sections contain theory, method and results.

The research question is formulated as follows:

"Can a supervised machine learning method be used to predict which hospital ambulance staff will decide to drive their called out ambulance to?"

Further questions that are answered include:

- Are there any non-obvious attributes that affect the decision?
- Which attributes influence the decision more than others and does this conform with empirically gained knowledge? In either case is there a reasonable explanation for why or why not?
- Can the results be thought to be significant? In other words, what are the performance of the algorithms compared to a baseline algorithm? Are these results enough to draw any conclusion on whether machine learning is a suitable approach for solving this and related problems?

2 Background

To understand the problem of hospital selection it is useful to get a deeper understanding of the full process, from emergency medical service preparedness to the point where the patient is delivered to a hospital. This section hence begins with an overview of what has been done before the decision in question is being made and some associated issues. After that follows a section with the actual topic. The impatient reader may skip to the "Hospital selection" headline, Section 2.1. For a more detailed analysis of the problem the reader is refereed to the paper *"Modeling Requirements for an Emergency Medical Service System design Evaluator"* by Sung and Lee [13]. For further reference on the use of machine learning for solving emergency related problems the paper *"Data Mining and Machine learning in the context of disaster and crisis management"*, Zagorecki et al. [16] is relevant.

The process of emergency medical service can be split into several steps, an overview of this is seen in Figure 1. Every part of the process has been studied and analyzed. Some studies even start before a call has been done, namely by predicting factors that might affect the decision and preparedness of the emergency service vehicles. In Zhou and Matteson's [14] paper, the authors present a time-series model for modeling how emergency call rate per hour varies over time. This was done in order to predict the demand for ambulance services.

The use of predictive models is found in other emergency areas as well and machine learning is not unfamiliar in this context. Angalakudati et al. [4] used

a maximum likelihood approach to predict which assets will interrupt electrical networks due to failure during extreme weather conditions such as storms. This was done to aid the company providing electricity and gas in the region with their preventative emergency planning effort.

Another aspect of preparation is to look for an optimal deployment of the emergency vehicles. Investigations of deployment of ambulances can either be done prior to any service demand or dynamically. The paper "A Distributed Convergent Solution to the Ambulance Positioning Problem on a Street map Graph" [3] is a good example of studying the problem prior to demand. In the study Wang et al. investigated how Voroni regions[1] can be used to deploy ambulances in a efficient and distributed manner. Examples of analyses of the dynamical problem will be presented later.

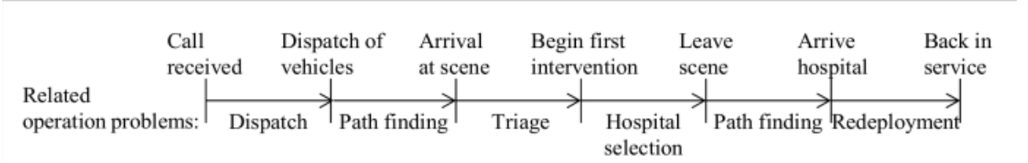


Figure 1: Overview of emergency service response. The image was taken from the paper *Modeling Requirements for an emergency medical service system design evaluator* by Sung and Lee [13].

The above papers cover many aspect of the "Related Operational problems" part and "Call Received" in Figure 1. The next steps are concerned with dispatching (deciding which ambulance to assign to a case) of vehicles, path finding and what happens when personnel arrive at the accident site. When gathering information to aid the path finding problem, machine learning methods have proven useful. Several algorithms that will also be considered in this project have been used. An example is Moussavi-Khalkhalis and Jamshidis [15] paper where, among other algorithms, multi-layer perceptron and principle component analysis were used to predict highway traffic flow. The algorithms were tested both for real time and for off line traffic flow prediction on a highway in Minnesota. The authors concluded that preprocessing of the data (here with principle component analysis) increased the learning pace and that networks trained on this data were better at generalizing than network trained on unprocessed data.

Glenn et al. [5] looked into how to make a efficient decision about how to use the available resources in case of two-site major incidents. The authors found that even when working with a single incident, interest conflicts for the resources arose. The main target with the study was to investigate how the allocation of resources affect the time taken for the most acute patient to be transported from the accident site to a hospital. They looked at what could be a best allocation of resources in these cases. An agent based simulation was used to investigate these matters. The conclusion was that preparedness is an important factor for compensating for the trade offs that have to be made when sharing resources between multiple sites. Like already said these trade offs also have to be done when deciding about which hospital to drive to, as doctors, care places in hospital wards and number of ambulances are limited resources. The authors summed up by stating that there was no resource allocation method found that was strictly

[1] A Voroni diagram is a split of the plane into subregions. The split is done by positioning points called "seeds" and a region consist of all points in space that lie closer to this seed than to another seed.

better than the Pareto optimal solution [2].

2.1 Hospital Selection

It is easy to draw parallels between the dispatching problem and the subsequent problem of choosing a hospital. In both cases the trade off between time and efficient resource usage have to be considered. Gong et al. [17] investigated the dispatching and routing of emergency vehicles in disaster environments. They used data fusion for this task. In the abstract the authors stated that *"The key factors that affect the dispatching of ambulances to patient locations include patient priority, cluster information, and distance. Similarly, those affecting the dispatching of ambulances from patient locations to hospitals include waiting time estimates at hospital emergency rooms, hospital capacities, injury type, and distance."*. Two methods were tested, iteratively applying a shortest path algorithm and to use brute force, by finding multiple paths and evaluating the results. It was found that both strategies clearly outperformed the approach of always sending a predetermined vehicle responsible for the area, as is the current standard [18]. In the patient delivery problem, when taking status of hospital into account, they found that the waiting time for the ambulances to deliver their patients was reduced.

Apart from the mentioned papers, this project was also inspired by Andersson and Värbrand's [18] report. The study was especially relevant since it was made as part of a joint project called "Optimized Ambulance Logistics" between SOS alarm and Linköpings university. Andersson and Värbrand developed a quantifiable measure for level of preparedness based on weights for each region. The weights described demand for ambulance services and were based on either number of calls or number of people currently resident in the region. They also developed an algorithm for the less acute cases. This was done by calculating which ambulance to send out that would have least effect on overall preparedness and was within a reasonable distance from the accident cite. The next step was to solve the optimization problem of dynamic relocation in case level of preparedness reached below a certain threshold value. This was done by mathematically modeling the problem and solving this with a search tree heuristics. The whole process of obtaining calls, dispatching and relocating was simulated. It was found that preparedness could not be kept above the desired threshold at all times, but that taking measures for improvement had a significant impact of overall preparedness level. The data used were travel times collected from fire engines. They were however expected to be representative for ambulances as well.

[2] Pareto optimal solution corresponds to a allocation where it is impossible to make any improvement for one site without making it worse for another.

2.2 Related Procedures, Algorithms and Motivation of Choice

The related articles described were different to this project in the way that they:
1. Were either based on a slightly different problem.
2. The solution that they presented had not been applied to empirical data, but relied on simulations.
To avoid the estimation of how similar another study is to this project, and what implications the differences might have had, the choice of algorithms did not rely on any specific previous work but was inspired by more generally drawn conclusions. The papers by Caruana, Niculescu-Mizil [7] and Lim et al. [8] both describe empirical studies made on supervised classification algorithms. In the first study ten supervised machine learning algorithms such as state vector machines, artificial neural networks, regression models and random forest were compared. The tests were conducted on eleven binary datasets with eight performance evaluation criterias. In case hyperparameters were present different settings for these were tested. The conclusion was that, on average, the best performance was achieved by bagged trees, random forests and neural nets. However it was found that even the models that had poor average performance, such as Naive Bayes and logistic regression, had best performance on some metrics on two of the data sets.

Bauer and Kohavi [6] made a comparison of different sorts of bagging and boosting techniques used for classification. Although this is not the most recent study it gave interesting insights into classification problems. The datasets for experiments for these studies shad similarities to the data set used in this project, for instance that the number of classes and attributes were few compared to the data available for training. Bauer and Kohavi discussed different evaluation criterias and noted that for voting algorithms it is important to restrict the number of sub classifiers, especially for sequential learners as Adaboost. Without this restriction the running time of the algorithms may otherwise turn out unfeasible (running up to several weeks) for practical purposes. Another conclusion was that bagging did not increase error in any data set but reduced it in all cases. Larger trees however showed, as expected, to have higher variance but lower bias, and the opposite for small trees.

The second study by Lim et al. [8] tested a greater number of algorithms but many of these were similar, being tweaked versions of decision trees or statistical methods. The performance was measured in terms of mean error and mean rank error. The algorithms were tested on 16*2 (doubled by adding noise) data sets. Consideration was taken to running time (scalability) and for the trees also how many subtrees the algorithm generated. Similarly as in Caruana's and Niculescu-Mizil's [7] study no single algorithm was found that outperformed the other methods, but again tree models were among the best performing algorithms. A difference in results was that this study, contrary to Bauer and Kohavis [6], showed that logistic discriminant analysis[3], performed second best. This considerable difference between the papers is believed to be because in the second study the methods were only evaluated based on two performance measures instead of eight. However regardless of the reason behind this difference logistic regression was chosen for this project since it is a simple yet powerful classifier which shows up in most machine learning experiments. A logistic regression model can capture even non-linear relationship in a linear fashion. It also gives probability estimates between the categorical dependent variable and the independent variables, information that might give valuable insight about the problem.

Considering the empirical aspect only, it seemed reasonable to test a tree model.

[3]Logistic discriminant analysis is a less flexible and robust version of logistic regression

For this random forest was chosen. To get a model complementary to random forest, multilayer-perceptron[4] was tested. The neural network is mainly different to random forest in the way this algorithm learns. Neural nets also have the advantage of having infinitively adjustable complexity of the model. These things will be clear after reading the theory section.

[4]A feed forward artificial neural network.

3 Theory

This section provides an overview over methods and algorithms. The intuitive understanding of classification algorithms is that they create decision boundaries that split the data space as can be seen in Figure 4. In the case of multi-class classification the models rely on several splits that are combined to find the final region for each class.

3.1 Baseline and Overfitting

Sometimes simpler models outperform, or perform equally well, as more complex models. This happens if the data is to noisy or there is to little data available for training [32]. Other reasons might be that the relationships in the data are actually simple but a complex model is used, or that the problem is to hard too learn for any algorithm. These are conditions under which a model might overfit. Overfitting refers to the case when a model specializes too much on the training data and hence looses the ability to generalize, which gives poor test data results. It is hence reasonable to start with a simple classification model with few parameters that provides a simple model. The usefulness of this approach is both to get initial insights about the degree of difficulty of the problem, and to get a benchmark for performance, a baseline. For this purpose the one rule algorithm was used. This algorithm works as a decision tree with depth one. The first step of the method is to create a frequency table for every feature [26]. A table contains the values each feature can take on and the corresponding predictions that would be made if this value was used as splitting criterion. The value of a feature that maximizes the number of correct classifications is chosen as a rule for each feature. These rules are reduced to one rule by choosing the feature that gives maximum number of correct classifications.

Attempts were made to find a better baseline algorithm. One tested strategy was that of always selecting the closest hospital. This strategy could, as expected, not catch the uneven distribution of hospital labels. The performance was hence poor, close to a fourth of the performance of the one rule algorithm.

3.2 Random Forest

K ={1,2,...N} - possible outcome classes
$\mathbf{x} = [x_1, x_2, ...x_F]$ - input features for a data point
θ_f - threshold value for feature f
fr_k- fraction of data points belonging to class k
V_f - set of values each feature can adopt
S - set with all data points
S_v - subset that has been split on a feature taking on a value v from set V
F- total set of features

Random forest is an ensemble method that combines several decision trees into one model [25]. A single tree is created by iteratively splitting the data into subsets which creates a binary tree structure which can be followed sequentially when classifying new data points, Figure 2. For each split two decisions have to be made: which feature to split on and what threshold value that should be used for this feature, i.e create one subset for $x_f \leq \theta_f$ and another for $x_f > \theta_f$. This process continues until a certain stop criteria has been met or the remaining data points cannot be split further. To choose which feature to split on, either the change in entropy, known as information gain or Kullback-Leibler divergence is maximized, or the Gini impurity of the remaining set is minimized. Information gain is a measure of how much our knowledge increases by doing a split, by knowledge we

mean how pure each set is. A set is as pure as it can be if it only contains data points that belong to a single class. Gini impurity is a measure of disorder among the data points. It is calculated as the probability of mislabeling an observation if it was randomly labeled according to the distribution of all the classes in the set. Mathematically these quantities are defined as [10] (p 49, 666):

$$Entropy(S) = -\sum_{t \in K} fr_k log_2 fr_k \tag{1}$$

With this definition of entropy information gain for splitting on a feature f is given as the entropy before the split, minus the average entropy of the resulting subsets.

$$Information\ gain(S, F) = Entropy(S) - \sum_{v \in V_f} \frac{|S_v|}{|S|} Entropy(S_v) \tag{2}$$

Gini impurity is defined as:

$$Gini = \sum_{t \in K} f_t(1 - f_t) \tag{3}$$

Determining the threshold value θ_f for a split can be done in several ways. A common approach is to sort the values in ascending order and choose the mid point as split value. A conventional stop criteria is to stop when the information gain falls below a predetermined value. If computational time and resources are sufficient, or the feature space is small, the tree might be grown unlimited until no further split is possible. For further details on popular tree algorithms the reader may search for the ID3, C4.5 and C5.0 algorithms.

A decision forest is obtained by using bootstrap aggregation. This means that the selection of data points for each decision tree is done randomly with replacement, resulting in non-disjoint subsets. The same approach is used for feature selection. For each tree a subset of features is randomly selected and these features may be selected multiple times for different trees. The final model is used as a classifier by presenting the input vector of an observation to every tree which outputs a class. The final class prediction is obtained by majority voting among the individual trees.

To avoid the problem of trees only choosing splits that classify data points from the most frequently occurring classes correct, the importance of a class can be balanced with class weights. A simple way to do this is to assign weights inversely proportional to the class frequency. The weights change the way the impurity measure is calculated which in turn affects the split. More on this matter and other balancing techniques can be found in the paper by Chen et al. [25].

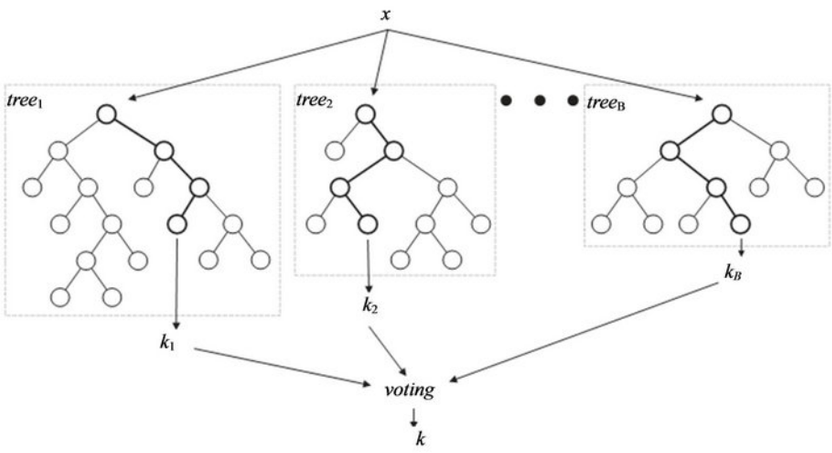


Figure 2: The overall approach for using a trained random forest model. Each tree is presented with the features of the data point x. They output a class which in the image is denoted as the variables $k_1, k_2, ...k_B$. The final answer, k, is obtained by doing majority voting. Nguyen et al. [24]

Multinomial Logistic Regression

K ={1,2,...N} - possible outcome classes
$\mathbf{x^i} = [x_1^i, x_2^i, ...x_F^i]$ - input features for a data point i
$y^i \in \{1, 2, ..., N\}$ - outcome for data point i.
$\mathbf{w^i} = [w_1^i, w_2^i, ..., w_F^i]$ - weights, one for each input feature
$\mathbf{w_0}$ - intercept, value when predictors are equal to zero
J($\mathbf{w}$) - cost function
α - hyperparameter that determines learning rate
Pr - probability
M - total number of data points
C - constant

The idea of logistic regression is to estimate with what probability a data point belongs to the different categories. These probabilities are assumed to be exponentiated linear combinations of the independent (input) variables [29]. The weights are obtained as in Equation 6 and 7, which will be explained later in this section. The probabilities are calculated by using a sigmoid curve as in Equation 4.The graph of this equation is found in Figure 3. For every input value, the output of this function remains within the interval [0,1], which is desirable for modeling probabilities.

$$Pr(y^i = k|\mathbf{x^i}) = \frac{e^{w_0^i + w_1^i x_1^i + ... + w_F^i x_F^i}}{1 + e^{w_0^i + w_1^i x_1^i + ... + w_F^i x_F^i}} \tag{4}$$

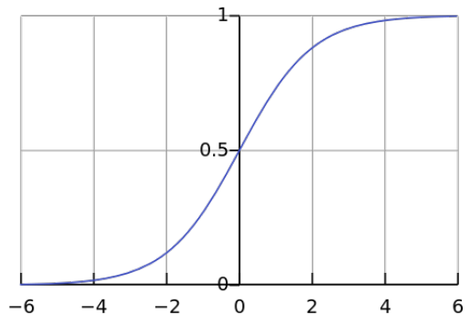


Figure 3: This image shows the sigmoid curve with the characteristic S-shape. As can be seen, for every value of x the output y stays within the interval [0,1]. This property makes the sigmoid function suitable to use for probability estimation. Retrieved from wikipedia, `https://commons.wikimedia.org/wiki/File:Logistic-curve.svg` (March 2017).

The prediction step of a binary logistic regression model (output 0 or 1) is done by plugging in the x values of a data point into the model. If $Pr(y^i = 1|\mathbf{x^i}) > 0.5$ the data point is classified as belonging to class 1, otherwise 0.

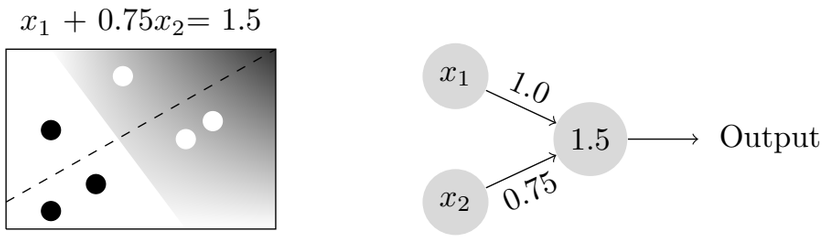


Figure 4: The image to the left shows how a binary dataset could be split by two different decision boundaries, one colored gray/white and the other as a dashed line. The right image shows how the colored decision boundary translates to a set of nodes and weights. The equation for this decision boundary is given above the left image. It can easily be seen that the dashed decision boundary is not as good as the colored, as the split it provides does not separate the data equally well. In the image on the right x_1 and x_2 are the input nodes. The coefficients, 1.0 and 0.75 are the corresponding weights for these variables. The network provides a binary output where 1.5 is the limit. This means if the weighted sum of x_1 and x_2 are larger than 1.5 the output is 1 which implies that the point lies in the gray area. If the output is smaller than 1.5 a point lies in the white area, and if it is exactly 1.5 it is part of the decision boundary.

Multinomial logistic regression is a generalization from the binary case. This generalization can be done in different ways, here a "One v.s the rest scheme" was used. This approach employs $k-1$ independent binary logistic regression models. Each model compares the probability of the output belonging to a certain class with one other class that was chosen as pivot, hence "one v.s the rest". Amongst the set of probabilities in Equation 5, the last class N was used as pivot. The output of the multinomial logistic regression model is the class that got highest probability given the input $\mathbf{x^i}$.

$$
\begin{aligned}
Pr(y^i = 1|\mathbf{x^i}) &= \frac{e^{\mathbf{w_1^{(i)T}x^i}}}{1 + \sum_{k=1}^{N-1} e^{\mathbf{w_k^{(i)T}x^i}}} \\
Pr(y^i = 2|\mathbf{x^i}) &= \frac{e^{\mathbf{w_2^{(i)T}x^i}}}{1 + \sum_{k=1}^{N-1} e^{\mathbf{w_k^{(i)T}x^i}}} \\
&\vdots \\
Pr(y^i = N|\mathbf{x^i}) &= 1 - \sum_{k=1}^{N-1} Pr(y^i = k|\mathbf{x^i})
\end{aligned}
\tag{5}
$$

When training a logistic regression model the goal is to find the weights $\mathbf{w}$ that best classify the data points. One way to find these is by minimizing the negative log likelihood. In other words, the optimal weights are considered those that maximize the hypothetical probability that the observed data would yield a specific outcome. The log likelihood is minimized by differentiating this function and searching for solutions. The reason log likelihood is considered instead of the likelihood as it is, is because the likelihood is a product of probabilities. Products are hard to differentiate, by taking the logarithm of a product it is turned into a sum, which is desirable for the differentiation step. The differentiated function yields the same result because the log function is a monotonic function, in other words it is a continued increasing function. To get different solutions, for instance when looking for solutions were many weights are 0, constants and additional functions can be added to the function under consideration. This function is then called a cost function. For the binary case, in scikit-learn, the cost function to minimize looks as in Equation 6.

$$
J(\mathbf{w}) = C \sum_{i=1}^{M} log(e^{-y_i(\mathbf{x}_i^T\mathbf{w}+c)} + 1) \tag{6}
$$

The solutions to the cost function are searched for by using gradient descent. With this method the weights are iteratively updated with the derivative of the cost function, Equation 7. The derivative is scaled by a hyperparameter that determines the "learning rate", α. This parameter determines how fast the solution converges to the minimum. A large α may converge faster and saves computational time by not having to do as many iterations. However, a too large α value may miss the optimum, as the solution may oscillate due to the steps taken in each iteration being too large.

$$
\min_{\mathbf{w}} J(\mathbf{w}) \Rightarrow \textit{update } w_f\textit{: } w_f = w_f - \alpha \frac{d}{dw_f} J(\mathbf{w}) \tag{7}
$$

3.3 Neural Network

Neural networks are somewhat related to logistic regression. One can think of neural networks as several logistic regression models combined. A Neural network model is divided into layers where each layer is a set of nodes [31]. Each node uses a function to transform the input. The nodes are connected with weights, so the input to one node from a previous layer becomes a linear weighted sum of the inputs, as seen in Figure 4.

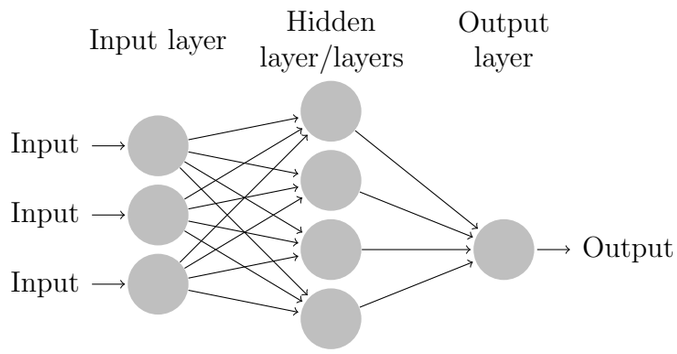


Figure 5: This is a example of a simple neural network with one hidden layer.

The difference to logistic regression is seen when the complexity of a model is increased by adding hidden layers as in Figure 5. At each middle node an activation function maps the input to output. Common functions include the sigmoid function, tangens hyperbolicus (tanh) and the identity function. In this study, models with tanh for activation function had best performance. If σ denotes the sigmoid function tanh can be written as:

$$tanh(x) = 2\sigma(2x) - 1 = \frac{2}{1 + e^{-2x}} - 1 \tag{8}$$

The tanh function maps every input to a number between [-1,1]. To invoke the information that the output is a categorical distribution, a softmax function, Equation 9, is used in the final layer. This function maps a multidimensional vector with arbitrary values into a vector where the values range from [0,1] and the sum of the vector adds to 1. The class that has the highest probability is chosen as prediction.

$$softmax(x_k) = \frac{e^{x_k}}{\sum_{k \in K} e^{x_k}}, (k = 1, 2, ..., N) \tag{9}$$

The training of a neural network consists, as for logistic regression, in finding optimal values of the weights. There are several options for doing this. Here back propagation in combination with a stochastic gradient-based optimizer. Backp ropagation works by calculating the response to a data point, and depending on how far off this result was from the true label (again this is done with a cost function), the weights are updated with some gradient method. The weights are updated starting from the ouput layer and going thru the network back to the input layer, hence back propagation.

3.4 K-fold Cross Validation

In general, the more data a algorithm trains on, the better it performs on testing data. At the same time, it is good to estimate the performance of a classifier while training as this gives an estimate of how good the model generalizes. To address this problem and make efficient use of the data, k-fold cross validation was used for training.

Cross validation works by splitting the training set into k parts, called folds. Each fold contains approximately the same number of data points. One of the folds is used for testing and the remaining $k - 1$ folds are used for training, Figure(6). The process is iterated with different folds for testing and this is done until all folds have been used as test sets, with the remaining folds used for training. The

error is calculated as the average test error on the individual test folds. Following this approach the whole data set can be used for testing (reducing test error variance) while at the same time avoiding over fitting since the training and test sets are disjoint each time. Five- or ten-fold cross-validation is often suggested as a standard. Here five-fold validation was used to reduce training time.

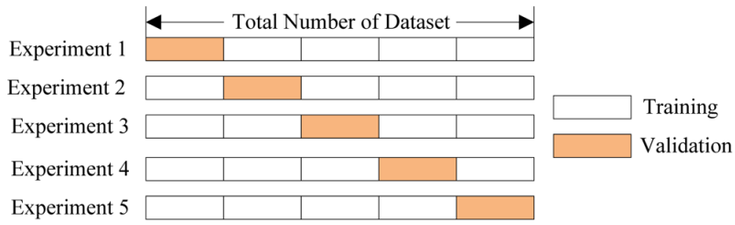


Figure 6: Example of five-fold cross validation. Zhang and Wu [23].

The splits were made with stratified sampling, the number of occurrences of each class in every fold may however not have been exactly the same as the numbers of folds may not be proper divisor for the count of each class. It has been found that stratified sampling has the potential of improving the precision of an estimate as it takes imbalances among the label distribution in the data into consideration [19]. The procedure is described in [20] pp. 102-103.

3.5 Feature Selection

Feature selection is the process of removing features that are unimportant to the learning algorithm. Potential benefits of doing feature selections include reducing overfitting, hence improve performance, and shorten training time. Guyon and Elisseeff[12] describe various methods and aspects on how to do feature selection. Their suggested procedures seem to agree well with the current standards. They conclude that a unifying theory on the subject is lacking and therefore a combination of techniques should be used to get, as they describe it "...a few baseline performance values". The paper treats the, for this problem highly relevant, question on whether or not to remove redundant features. Features that are perfectly correlated obviously do not add any information [12]. Hence perfectly correlated features in the data were removed.

Elisseff and Guyon found that features that are only somewhat redundant, can add information when regarded together. Even if completely useless on their own, features can prove useful when considered with other features, regardless of if this other feature is useful in itself or not. Hence, to get a broader perspective of the importance of the features, they were examined with three different methods. First with univariate feature selection which considers every feature on its own. Thereafter with stability selection and mean decrease accuracy which both are methods that consider features together. The results of these methods are found in Section 5.

In univariate feature selection each feature is tested individually. This means a model is built based on one feature at a time. Each feature in itself often does not have much predictive power and this is the reason why univariate feature selection results in small scores compared to other feature selection methods.

Stability search is a relatively new method where subsets of features are created and a model is trained on these features. Here a L1-penalized, Equation 10, logistic regression model was used and the target was to minimize the log loss function over the whole dataset by adjusting the parameters of the model. The final result is

obtained by checking how many times a certain feature was selected as important.

$$L_1(w) = \sum_{i=1}^{f} |w_i| \tag{10}$$

In mean decrease accuracy (MDA) a model is first trained on the full dataset. Thereafter the data for a feature is shuffled randomly, one feature at a time, and a new model is trained. The score is calculated as performance difference between the model trained on the full set of features, compared to the model trained with one missing feature, Equation 11. If the shuffled feature is unimportant the score will be close to zero, as performance on the full dataset will be the same or close to the same to the performance with one shuffled column.

$$MDA = \frac{\textit{Performance on full dataset-Performance with one shuffled column}}{\textit{Performance on full dataset}} \tag{11}$$

3.6 Performance Evaluation

The baseline of classification algorithms for performance measure is classification accuracy [21]. This metric is defined as:

$$Accuracy = \frac{\textit{Number of correct predictions}}{\textit{Number of total predictions}} \tag{12}$$

This approach is simple and clearly comparable between models. However it suffers from a serious drawback known as the "Accuracy Paradox" which says that models with a given level of accuracy may have greater predictive power than models with higher accuracy[21]. This is because accuracy assumes a relatively uniform distribution and equal cost for each misclassification. The use of classification accuracy as the single method of evaluated should therefore be avoided [22]. To address this f1-score was chosen as a complementary metric for performance measure, Equation 13. F1-score is complementary to accuracy because it is partly a measure of *precision*, which is a estimation of the random error as compared to accuracy which measures the systematic error. The f1-score can be calculated either globally, or for each label, with the unweighted mean as final value. The second approach is known as *f1-macro*. This means the score is influenced equally by rare categories as by frequently appearing categories. F1-macro was chosen, as this metric would be harder to get a good score on with the uneven distribution at hand (this was also seen during testing). Another reason for choosing f1-score is because, as accuracy, it is a commonly occurring performance measure. It is therefore easy to understand and the results are more easily relatable to what is found in other studies.

PRE_i - precision for class i
REC_i - recall for class i
$|K|$ - Number of classes
True Positive (tp): Correctly classified as the class of interest
True Negative (tn): Correctly classified as not the class of interest
False Positive (fp): Incorrectly classified as the class of interest
False Negative (fn): Incorrectly classified as not the class of interest

$$f1_i = 2 * \frac{PRE_i * REC_i}{PRE_i + REC_i} \tag{13}$$

$$PRE_i = \frac{tp_i}{tp_i + fp_i},\ REC_i = \frac{tp_i}{tp_i + fn_i} \tag{14}$$

$$f1_{macro} = \frac{f1_1 + ... + f1_i}{|K|} \tag{15}$$

Final estimates

As a final tool for evaluation and analysis confusion matrices were used. These are best described with an example as in Figure 7. A confusion matrix gathers the results in a table with correctly classified data points, wrongly classified data points and what type of error this was.

		Prediction	
		Cat	Dog
Actual	Cat	15	35
	Dog	40	10

Figure 7: Simple two class confusion matrix. Along the diagonal, starting in the left upper corner, are the correct predictions. In this case the model predicted cat 15 times when there was actually a cat and dog 10 times when there was actually a dog. The remaining entries are wrong classifications, for instance there were 40 instances where a model predicted cat when it was actually a dog. The image is retrieved from wikipedia, `https://en.wikipedia.org/wiki/Confusion_matrix`, (March 2017).

The final accuracy and f1-scores were calculated on the test set with repeated hold-out. This is done by creating subsets by randomly selecting samples from the testing set. The model is tested on every subset and the results is the mean performance on all test sets. Five subsets were used and each subset was 2/3 of the size of the full test set.

4 Data and Method

The data had been gathered by SOS-alarm. The data set was given accessed to through Carmenta AB, a Swedish software company developing geographical information system-tools for mission-critical systems.

4.1 The Dataset

The raw data contained close to 66 000 observations from SOS's database. An observation consists of information on a ambulance transport that has been made. Most observations were from the year of 2016 and the rest from 2017. The ambulances had driven to 86 different areas where there was either a hospital, a nursing home or other health care facility, these places are from now on referred to as labels. It should be noted that not all areas had a one to one mapping between label and health care facility. Some areas for instance grouped together two or more nursing homes or care centers. How this was dealt with will be described later.

About 100 observations had important data missing, such as no given location. By inspection no correlation between these incomplete observations and some particular label was found, and so these data points were removed. Observations that described pre-ordered transport between hospitals were also removed, as these were not considered to lie within the problem domain. After removal of this data about 62 000 observations remained. Finally labels that appeared less than 15 times were grouped to one category named "OTHER". The number 15 was chosen because there was no area specifications available for labels that had fewer observations than this. This approach resulted in 40 classes remaining. The distribution of labels is depicted in Figure 8. The initial information given about each observation was given as seen in the list below. The bold texts indicate features that were kept. A full list of features to which the models were applied is found in next section, Table 1.

- **Xcoordinate**, **Ycoordinate** - The position for a accident site given as longitude and latitude.
- **Time and date** for when the ambulance was called out.
- **CallcenterID** - Identification number for which call center that received the call
- **CaseIndex1**, **CaseIndex2**, **CaseIndex3**, Index1Text, Index2Text, Index3Text - Numbers and texts that described what kind of accident that had happend, index1 being the most general description and 3 being the most specific. The index 1 number and text were present for all data points but was set to none for the other fields. A example could look like:[5]

	Number description	Text description
Index1	23	"Breathing problems"
Index2	15	"Asthma, not responding to medication"
Index3	None	None

- **Casepriority**.
- **Additional information** - For the cases were a single label had been used to group together nursing homes or care centers a descriptive text of this particular location was given. It was also present for describing different

[5]Note that these Figures and problems are made up and have nothing to do with the real numbers or texts used by SOS-alarm.

departments of a single hospital. This information is here called ”Additional information”, the reason this is stated as a feature and not being part of the labels will be explained later.

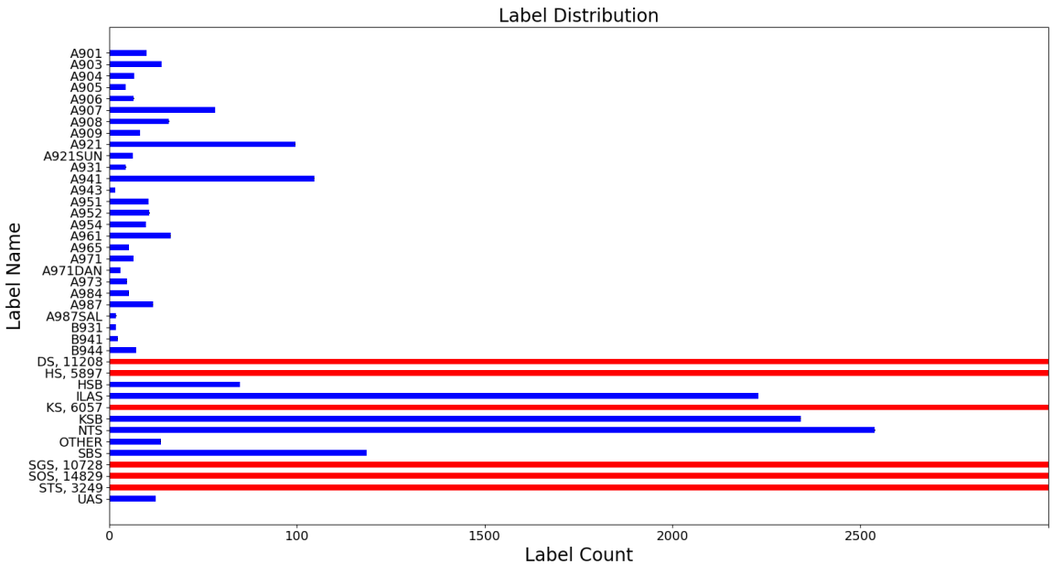


Figure 8: The image shows how the labels were distributed. To make the seldom occurring labels visible in the plot, the scale has been limited to 3000. The labels which occurred more often than this are indicated with red bars. The exact count for these is found next to the label name. [24]

4.2 Feature Transformation and the MapQuest API

To make the information in the data easier to learn from for the algorithms, some changes were made in how the data was displayed. An overview of the preprocessing, training and testing is found in Figure 9. The final 47 features are found in Table 1.

The time and date feature was split into two, one feature for minute of the day and another for day of the year. Time and date describe periodic quantities. Many algorithms rely on the assumption that the data can be modeled with a Gaussian distribution, but this distribution is inappropriate for continuous variables such as periodic variables [11]. To address this a sin/cos transformation was applied to these features [10] (p105-106).

The transformation that had most significant impact on performance was to add the distances from the accident site to every hospital. The location was given as longitude and latitude. The hospital sites were found by manually searching the area each label denoted for a hospital, care center or other care facility. Google maps [27] was used for this. This tool also lended itself for geocoding which is the process of transforming an address to coordinates, here to longitude and latitude. This approach may have introduced noise, mainly since the labels that were grouped together did not get the specific distance to that particular place but an approximate location somewhere in between these care facilities. Also for some labels (all of them that were bundled into the other category) there was no explanation on what area a certain label referred to. Hence it was not possible to get a address for these labels. For these observations the distances were set to 0.

Feature Name	Range	Unit	Description
Xcoordinate	-	°	Longitude
Ycoordinate	-	°	Latitude
CaseIndex1	{1,2,...,35}	-	-
CaseIndex2	{1,2,...,315}	-	-
CaseIndex3	{1,2,3,4,5}	-	-
Casepriority	{1,2,3,4}	-	-
Additional info	{1,0}	-	-
sin Time	-	Radians	Day of the year
sin Day	-	Radians	Minute of the day
cos Time	-	Radians	Day of the year
cos Day	-	Radians	Minute of the day
A901-(lon, lat)	-	km	Distance
A903-(lon,lat)	-	km	Distance
⋮	⋮	⋮	⋮
UAS-(lon, lat)	-	km	Distance
A903 - time	-	minutes	Travel time
A903 - time	-	minutes	Travel time
⋮	⋮	⋮	⋮
UAS - time	-	minutes	Travel time

Table 1: What the feature looked like after preprocessing. The features "Label name" - (lon, lat) describe the calculated distance between a location and that specific care facility. The features colored in blue describe the travel time from the accident location to that care facility. These features were only used on a test subset where the MapQuest API was used to calculate distances.

The first method for getting distance estimations was to calculate the great-circle distances [6]. This was done with the Haversine-formula which can be found in Appendix A. The second approach was to use MapQuest's distance matrix API [28]. The API gave both an estimate of travel times and more accurate distance calculations than the haversine formula. The API was used on a subset of 10 330 data points since this approach was time consuming compared to using the Haversine-formula. The difference in results between training the algorithms on input with the approximate great-circle distance compared to the more exact MapQuest distances can be seen in the results section, Section 5. It should be noted that the MapQuest API was not always successful in calculating the distances; it failed for about 400 out of 10330 data points. This happened when the route search time was extensive (terms of usage for this API was restricted) or the address of the location was not found. The search was then canceled and 0 was returned both for distances and travel times. These distances were replaced with the great-circle distance.

Some algorithms, such as neural networks might consider values that are in a greater range more than values that appear in a smaller range . This trait may be desirable in some cases but not for this project so the features were scaled with Pythons standard scaler which removes the mean and scales to unit variance.

The last feature that was dealt with was the text description referred to as "Additional info". Like stated earlier it was found that this text described when the target location was a certain department of a hospital often for children, nursing homes for elderly or a health center somehow specialized. This information could be found from the descriptive texts (index1 Text, index2 text, index 3 text). It

[6]Great-circle distance is the shortest distance between two points on the surface of a sphere.

was hence considered safe to assume that some additional information was available for the SOS caller, for instance age or if the person was currently staying at some particular care establishment. This information was summarized by binarizing the presence or absence of a descriptive text.

After preprocessing, the data was split into a hold out and a training set. The final performance estimation was done on the hold out set. The data points in the hold out set were hence disjoint to the training set so that the final results were based on the performance on data points never previously seen by the algorithms. The split was made with stratified sampling meaning that the label proportions were kept. From the total set, 90% was used for training and the remaining 10% was used as hold out set.

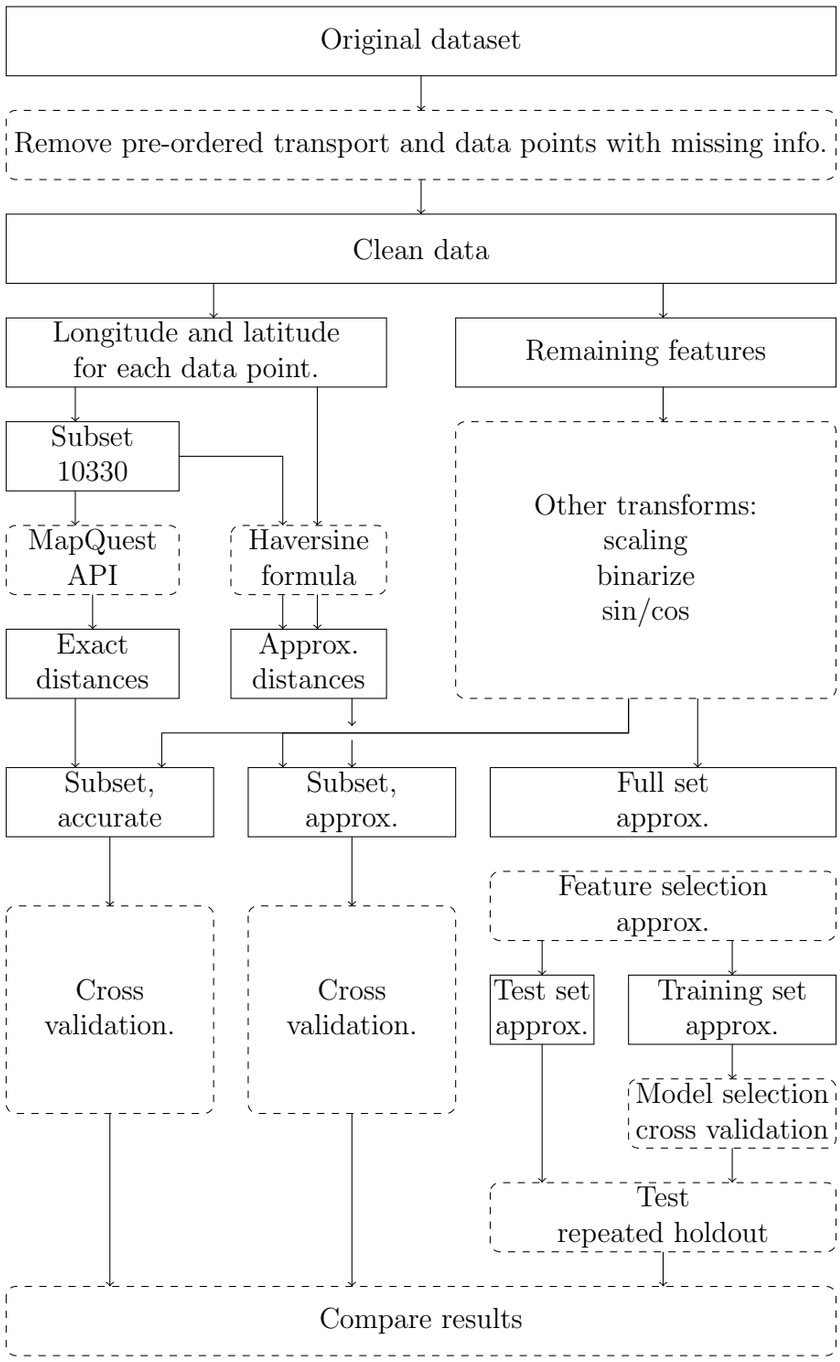


Figure 9: Overview of how preprocessing, training and testing was done. Boxes with dashed lines describe changes that were done to the data, whereas the boxes with continuous lines describe data. Starting from the original dataset some ineligible observations were removed. After this, feature transformation and feature engineering was done on the "Clean data". The most important preprocessing step was to use the location of every observations to calculate the distances from this place to each health care facility. This was done with the aid of MapQuest API and the Haversine formula. The remaining features went through transformations summarized in the "other transforms" box. By following this procedure three sets were obtained, one set made up of all observations where the distances had been calculated as the great-circle distance. And two subsets with 10330 observations each were the distances had been calculated with the MapQuest API for one data set and as the great-circle distance for the other. Part of the full dataset was used for model selection as described in Section 4.3 and for training. The remaining part was used for testing. Finally the results of each model on the subsets were evaluated and compared.

4.3 Hyperparameter Tuning

Estimators, such as the models used in this project, have parameters that are not learned from the data but describe properties of a model. For example in case of a random forest model these specify how many trees that should be grown and how many features each tree should use. These parameters are called hyperparameters and determine the complexity of a model and how it should train. Which hyperparameters that are optimal hence depend on the data, and so these parameters have to be searched for for each model. Since it is computationally impossible to try every combination, another way of finding suitable hyperparameters had to be used. This was done by first manually varying parameters and looking at the performance for some different settings and extremes to find what seemed to be suitable values. This knowledge was combined with the recommendation of scikit-learn to find plausible intervals for testing. The intervals were evaluated with grid-search where every combination of the grid is tested. The performance was, as described in a previous section, evaluated with five-fold cross validation. Hyperparameters not present in the grids had default values as set by scikit-learn. Links to the scikit-learn documentation, that gives a closer explanation to the hyperparameters and their possible values, is found in the Appendix.

Grid for random forest

Max features:	3	7	10	15	35
N-estimators:	10	40	60	70	130
Max depth:	1	4	10	unlimited	
Class weight:	none	balanced			

Best: Max features - 15, N estimators - 60, Max depth - unlimited, Class weight: none

Grid for Logistic Regression

Solver:	newton-cg	sag	liblinear (not combined with multinomial)	
Multi class:	ovr	multinomial		
Max iter:	50	100	200	300
Class weight:	balanced,	None		

Best: Multi class - ovr, Solver - liblinear, Max iter - 100, Class weight - balanced

Grid for Neural Network

Hidden layers:	(5,)	(10,)	(15,)	(30,)	(50,)	(100,)	(5,5)	(10,10)	(15,15)
Activation:	identity	logistic	tanh	relu					
α:	0.1	0.001	0.0001	10^{-6}	10^{-7}				
Max iter:	100	200	400	600					

Best: α - 0.001, Activation-tanh, Hidden layers -(100,), Max iter - 200

5 Results

The result section commences with bar charts obtained with the feature selection methods. After this the effect of feature transformation and hyperparameter selection is shown. The following "Best Scores and Types of Error" section show the results of the trained models when applied to a hold out set. The final plot, Figure 17, is the outcome of the models when the problem was made less fine grained. This was done by removing observations of labels that were not real hospitals but nursing homes, care centers or similar. This reduced the number of labels from 37 to 10, with 9 labels being for the Stockholm region and one for Gotland. The limit was set to labels that occurred in less than 500 observations, as no label with frequency below this threshold was considered to be a "real" hospital. With these restrictions applied, close to 3500 points were removed in total from training and hold out set.

5.1 Feature Importance

For mean decrease accuracy and the univariate feature selection method a random forest classifier was trained with the following settings:
Max features - $\sqrt{number\ of\ features} = 7$
N-estimator - 30
Max depth - unlimited
Class weight - none
The estimations were obtained by using the full training set and five fold cross validation.

The stability selection was done with a randomized logistic regression model as explained in Section 3.5. Default settings were used, these can be found in the Appendix, "Scikit documentation" section. Based on these scores CallcenterID was removed from the dataset when training the final models. For the following plots scores along the y-axis are percentage for the accuracy metric (blue bars), and f1-macro score for the green bars.

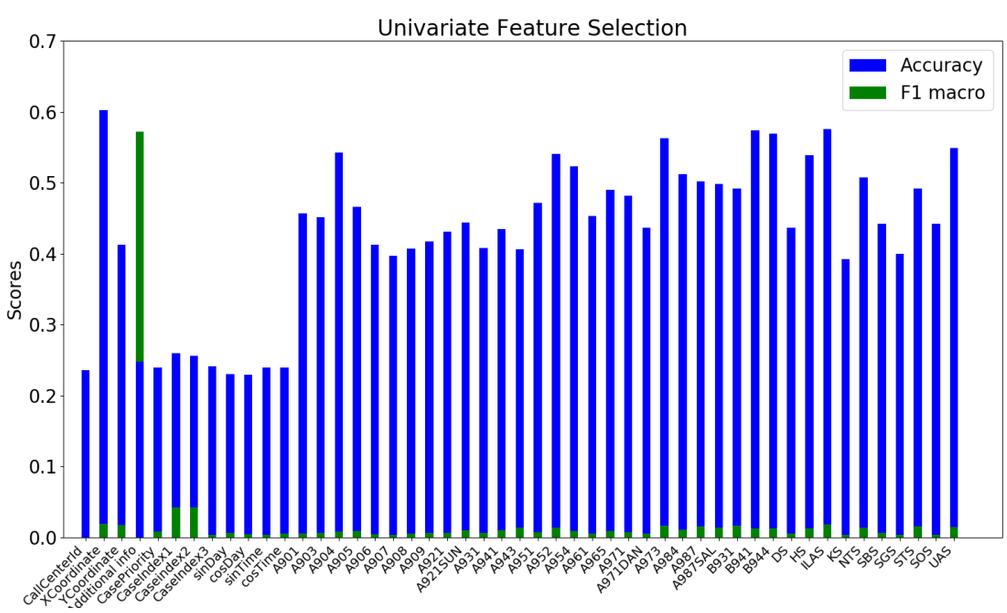


Figure 10: Scores when a single model was created to evaluate each feature.

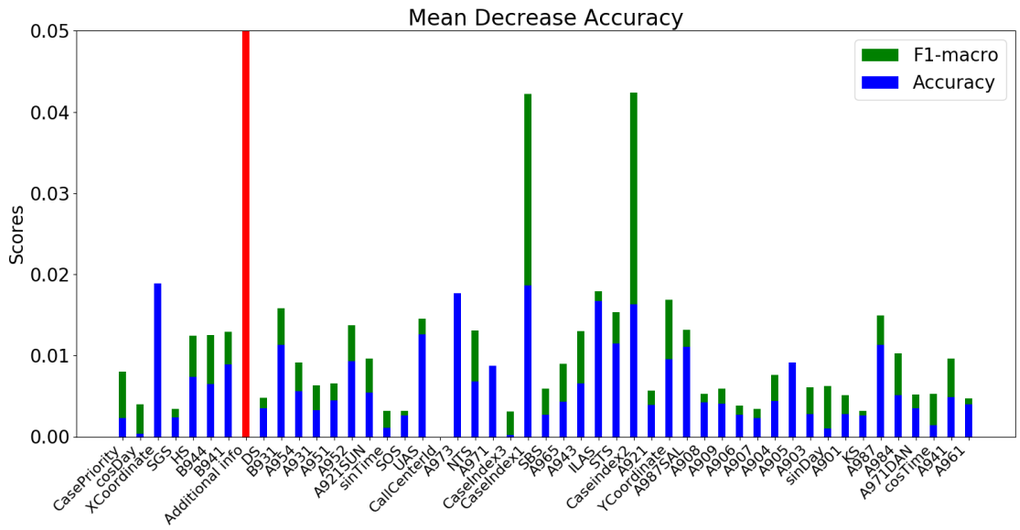


Figure 11: For mean decrease accuracy the most important feature was "Additional Info". This feature received a accuracy score of 0.0691 and a f1-score of 0.572.

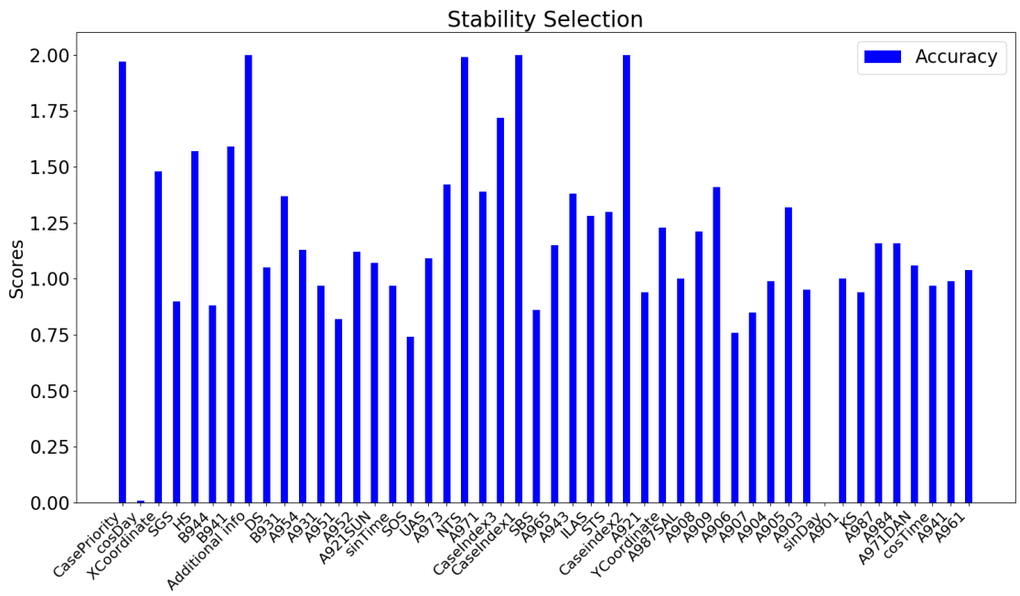
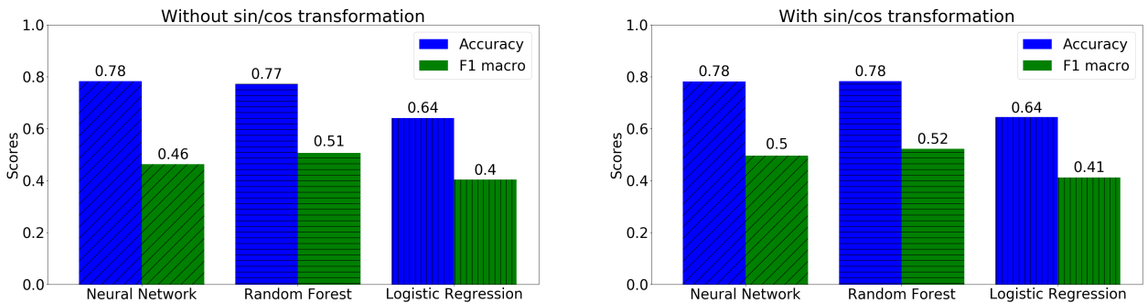


Figure 12: The stability selection did not measure f1-score but gave an estimate of the feature importance for accuracy score.

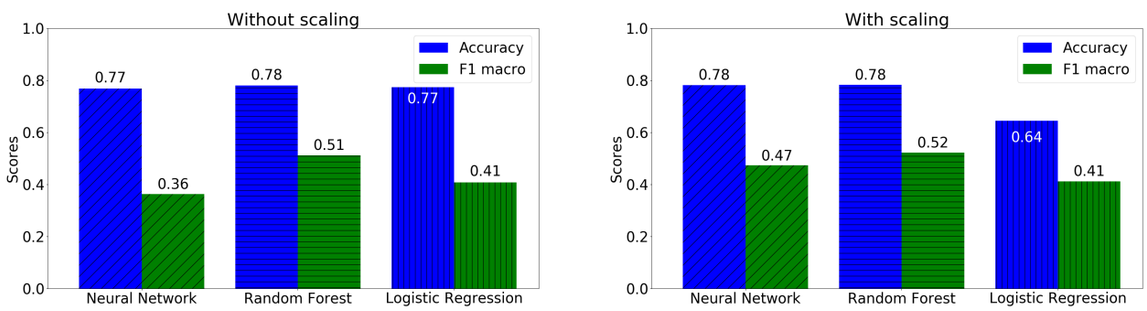
5.2 Transformations and Convergence

In order to investigate if the sin/cos transformation and scaling of the feature was useful, tests were run on the training set with default values for the hyperparameters. The results with and without sin/cos transformation is seen in Figure 13. The effect off scaling on the different models is displayed Figure 14. The results showed that the neural network experienced a slightly better performance when the sin/cos transformation was used. For scaling neural network showed a significantly better performance with scaling, logistic regression had a barely noticeable loss in performance and the result for random forest was nearly unchanged. Hence for the final models sin/cos transformation was used to transform the time and date feature for all models. Scaling was applied to all features for the neural network and random forest models but not for the final logistic regression models.



(a) **(b)**

Figure 13: Result of training default models on data with (a) and without (b) sin/cos transformations for the periodic time and date features.

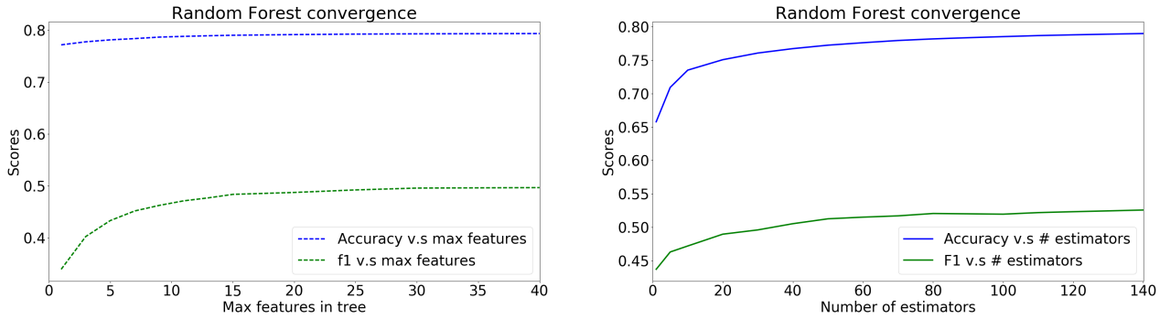


(a) **(b)**

Figure 14: Result of training default models on data where all features have their original range (a) and where all features have been scaled to unit variance (b).

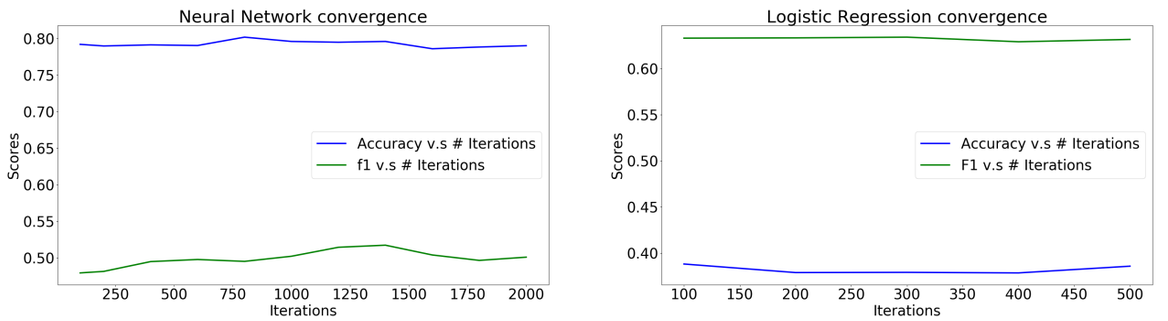
Convergence

Both logistic regression models and neural networks rely on optimization techniques that iterate to find the optimal solution. Similarly the number of trees in a forest and number of features used in each split affects how computationally expensive it is to train a model. As the numbers of iterations or number of trees grow large the calculations become time consuming. Hence, to save time, the number of iterations, number of trees and features were limited when tuning the hyperparameters during the grid search. To ensure that the solution for the final models would converge to a optimal value the number of iterations, features and trees were sought for with the aid of the training set. The results can be seen in Figure 15 and 16. The target parameter was varied as can be seen on the x-axis of these plots and the remaining hyperparameters were held constant, set to the best values found during the grid search.



(a) **(b)**

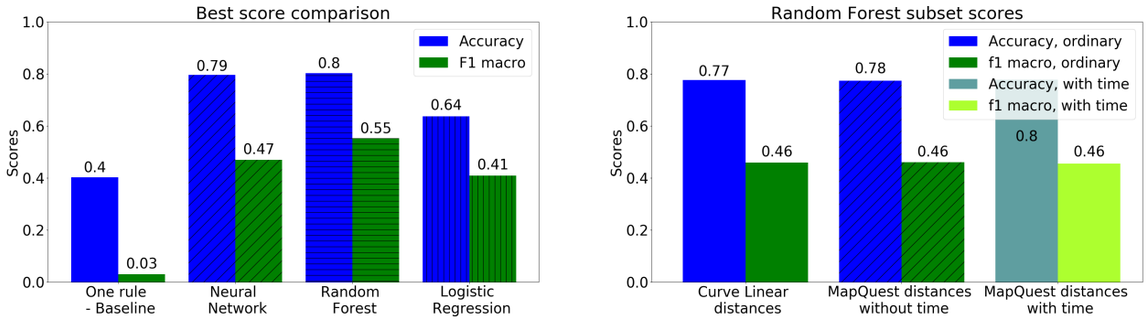
Figure 15: Image (a) shows a graph of f1 score (green) and accuracy score (blue) v.s maximum number of features in each random subset considered when doing a split at a node. The maximum number of features seemed to have reached it's peak already at 15 features and so there is was no need to grow trees larger than this.
Figure (b) shows how the performance of a random forest model changes depending on number of estimators (trees) used in the forest.



(a) **(b)**

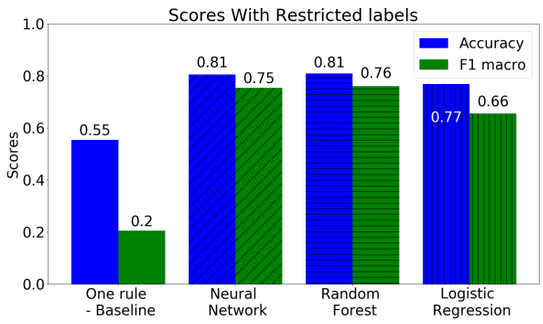
Figure 16: Image (a) displays the resulting accuracy and f1-scores depending on how many iterations the solver used to create a neural net. The accuracy and f1-score did not exhibit their corresponding maximum value at the same point. The most suitable number of iterations chosen for the final model was chosen as 1300 as it was considered more important to get a higher f1-macro score than a higher accuracy score.
The plot of image (b) shows the same as plot (a) but for logistic regression instead of neural network. The performance of a model, with the given hyperparameters, did not change significantly due to change in number of iterations and so 100 iterations as had been found while doing grid search, were used for the final models.

5.3 Best Scores and Types of Error



(a) **(b)**

Figure 17: (a) shows a performance comparison between all algorithms and baseline.
(b) shows how the result of a default random forest model depended on if the distances were calculated with the MapQuest API or as the approximate great-circle distances. The algorithm was applied to a subset of 10330 data points. The last bar in this plot "MapQuest distances with time", shows the result when travel times were also considered.



(a)

Figure 18: As the classes were restricted to hospitals only (care centers, nursing homes and etcetera excluded) the results were clearly improved for all models.

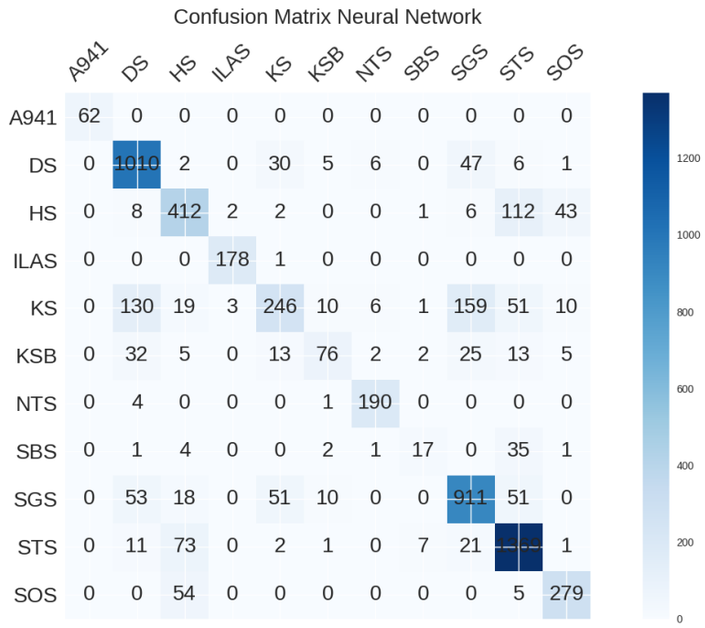


Figure 19: This confusion matrix shows the performance of a tuned neural network on the hold out set. The data points along the diagonal are correctly classified. For readability only the 11 largest classes have been included.

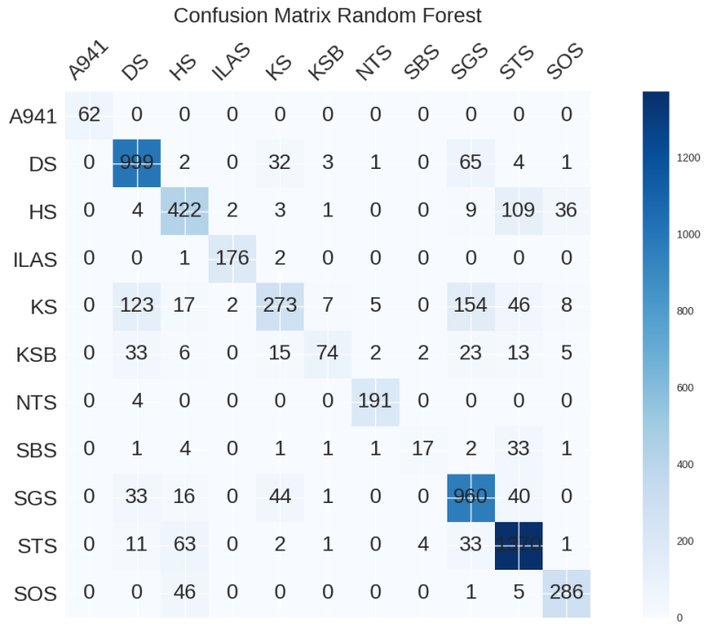


Figure 20: Confusion matrix obtained by running the best random forest model on the hold out set.

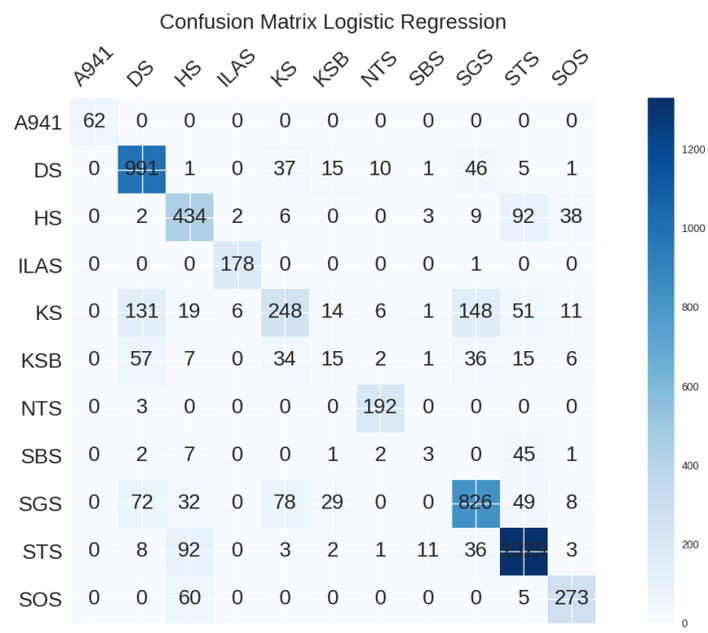


Figure 21: Confusion matrix for logistic regression model.

6 Discussion

In terms of accuracy, random forest and neural network performed equally well. On f1-macro score, the random forest model showed a 0.08 better score than the second best model, which was a neural network. Even though logistic regression could not compete with the other algorithms, it performed significantly better than the baseline which achieved a 40% accuracy and close to zero in f1-score. Another strength of the random forest model was that it had a short training time compared to the other algorithms. This could be addressed by using other solvers for neural network and logistic regression, however this would result in further performance decrease. It should be noted that the f1 scores on the restricted problem Figure 18a, were significantly higher; 0.75, 0.76 and 0.66 for neural network, random forest and logistic regression, more on this matter will be said later in the discussion.

It is known that Random forest is a more robust model than the neural network. This is also seen in how much the neural network responds to changes done in feature representation such as sin/cos transformations and scaling, compared to how the random forest responds to these changes. This suggests that the inferior result of the neural network model may be due to deficits in training and feature transformation. When talking about changes in feature representation it should be noted that the difference in using approximate great-circle distance calculations to using more accurate MapQuest distances was barely noticeable. This suggests that other features than distance are essential for making the decision. It may also be due to missing variables in the data set, which makes the respective importance of the available features seem more arbitrary than is really the case, as mentioned in the feature selection section. And so the distance features, and the accurate representation of these, might have been more important in the presence of more information. Regardless, considering the results of 17, using the approximated distances on the data as it was available for this project did not seem to have any substantial impact on the results.

Looking at the outcome of the feature selection process it is clear that "Additional info" was by far the most important feature for improving f1-score. This comes as no surprise, as this binary value was set to 1 more often for seldom appearing labels compared to larger classes. Also, no feature turned out to be completely useless. Considering this together with the result on the subset score, where training the models on data with more accurate distance received similar results, it is not far fetched to believe that the results of the algorithms could be significantly improved if more features were provided. A look at the confusion matrices, which show that all three algorithms made the same errors, strengthen the theory that more information is needed to improve the result.

When looking at the results one needs to keep in mind that some labels were grouped together to the "OTHER" category due to missing data. However, the decision can still be considered to be quite fine grained since as many as possible labels (as long as there was information) were kept. When the problem was restricted to only contain larger hospitals and removing other types of health care facilities, the problem was noticeably easier for the models to learn, Figure 18a. The settings for hyperparameters used for these models were the same as those used for the unrestricted problem. Hence, these parameters had not been optimized for this task and the results may have been better in case this would have been done. The improvement in f1 score when comparing the restricted and the unrestricted problem came as no surprise as it could be seen from the confusion matrices that the predictions for the larger classes were quite accurate for all models.

One can wonder if it would not have been a better idea to investigate the restricted problem only, since not having done this restriction unquestionably made the analysis of the data more complicated and the alternatives to consider greater. Even the choice of performance evaluation metric had to be chosen with more care. I believe, however, that this approach contributes to a broader perspective on the problem and gives more information for answering the question of whether or not machine learning is suitable for this predictive task.

6.1 Conclusion, Ethics and Sustainability

The main question to investigate with this project was to determine whether or not machine learning is a suitable task for predicting, and hence learning, the decision of hospital selection. It seems that machine learning is a possible tool for predicting this choice. The algorithms were clearly able to learn from the data and the results were significantly better than baseline.

In the beginning of the project it was believed that closeness to a hospital was one of the main factors for choosing a hospital. The feature selection suggest that this matters but that many other features, such as the type of incident (caseIndex), play a major role.

Since the lives and well being of patients is the central question, environmental aspects have a limited interests for this particular questions. However, as mentioned in the background, the study by Gong et.al (2009)[17] investigated the dispatching and routing of emergency vehicles in disaster environments, and found that the use of algorithms, at least in theory, could make more efficient decisions than what is done by humans today. Hence, if the use of algorithms for the decision making process was integrated with other knowledge in the field, such as ambulance deployment, this could lead to decreased driving distances for the ambulances.

Even though the algorithms in many cases give convincing results, the question remains if machines (computers) are desirable for critical decision tasks. This question is currently a matter of debate in many areas, for instance "Do airplanes need pilots?" or "Is it wrong to leave supervision of patients in intensive care to machines?". In case the algorithm makes a "wrong" decision, who is responsible?

6.2 Future Work

Although several things were tried in terms of training, a significantly better performance only seems reachable if new features are added. The exception to this would be to spend time on finding optimal weights for the different classes, as it was seen during training that this had a noticeable impact on performance. New features that would be of particular interest could for instance be hospital occupancy and ambulance deployment. To investigate the real usefulness further domain knowledge would be needed, as this could constrain the problem and set hard performance boundaries.

Neither the grid search nor the number of algorithms tested were exhaustive. The procedure of finding the best hyperparameters with two steps, first with a grid search for most parameters and after that to find the most suitable number of trees and iterations, might miss the optimal solution as the optimal value for the different hyperparameters depend on each other. A better way would therefore be to search for these computationally costly parameters together with the other

parameters.

The quality of the distance calculations leaves room for improvement. For instance, by getting the exact coordinates of the hospital driven to, instead of having to search for this addresses by hand. Also, removing the resource constraints and calculating all points with a more accurate distance API that also consider departure time would be interesting. With the current data, reducing the noise in the distance calculations did not seem to have an impact on model performance, this could however change if combined with the other suggested actions as mentioned previously.

Chapter 1

Bibliography

[1] Davenport, "A Predictive Analytics Primer", Predictive Analytics in Practice, Harvard Business Review, 2014. page 1 http://www.sas.com/content/dam/SAS/en_us/doc/whitepaper2/hbr-predictive-analytics-in-practice-107511.pdf

[2] Capelli, "We can't always control what makes us successful", Predictive Analytics in Practice, Harvard Business Review, 2014. page 7 http://www.sas.com/content/dam/SAS/en_us/doc/whitepaper2/hbr-predictive-analytics-in-practice-107511.pdf

[3] Wang, Colledanchise, Marzinotto, Ögren, 2014. "A Distributed Convergent Solution to the Ambulance Positioning Problem on a Streetmap Graph". Elsevier. 47, 91909196.

[4] Angalakudati, Calzada, Farias, Gonynor, Monsch, Papush, Perakis, Raad, Schein, Warren, Whipple, Williams. 2014. *"Improving Emergency Storm Planning using Machine Learning"*. https://www.researchgate.net/.

[5] Glenn, Graham, Duncan, Roger.(2015).*"Agent-based simulation of emergency response to plan the allocation of resources for a hypothetical two-site major incident"*.

[6] Bauer, Kohavi. (1998). An empirical Comparison of Voting Classification Algorithm: Bagging, Boosting, and Variants. Springer Link, Volume 36, pp 105-139.

[7] Caruana, Niculescu-Mizil. (2006). An Empirical Comparison of Supervised Learning Algorithms. Proceeding ICML. pp161-168.

[8] Lim, Loh; Shih. (2000). A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirthy-Three Old and New Classification Algorithms. Machine Learning. Volume 40, pp 203-228.

[9] Bauer, Kohavi. (1998). An empirical Comparison of Voting Classification Algorithm: Bagging, Boosting, and Variants. Springer Link, Volume 36, pp 105-139.

[10] Bishop. (2006). *"Pattern Recognition and Machine Learning"*. 1:st edition. Springer Verlag.

[11] Ghassem, Deisenroth. (2014) *"Analytic Long-Term Forecasting with Periodic Gaussian Processes*. Proceedings of the International Conference on Artificial Intelligence and Statistics" (AISTATS).

[12] Isabelle Guyon, Andr Elisseeff. (2003). *"An Introduction to Variable and Feature Selection"*.. Journal of Machine Learning Research 3, 1157-1182.

[13] Sung, Lee. (2012). *Modeling requirements for an Emergency Medical Service system design evaluator*. Simulation Conference (WSC), Proceedings of the 2012 Winter.

[14] Zhengyi, Matteson. (2015).*"PREDICTING MELBOURNE AMBULANCE DEMAND USING KERNEL WARPING"*. The Annals of applied Statistics, volume 10, nr 4.

[15] Moussavi-Khalkhali, Jamshidis (2014). *"Leveraging Machine Learning Algorithms to Perform Online and Offline Highway Traffic Flow Predictions"*. IEEE Conference Publications, 13:th International Conference on Machine Learning and Applications.

[16] Zagorecki, Johnson, Ristvej. (2013). *"Data mining and machine learning in the context of disaster and crisis management"*. International Journal of Emergency Management 9(4):351 - 365.

[17] Gong, Jotshi, Batta. (2009). *Dispatching/Routing of Emergency Vehicles in a Disaster Environment using Data Fusion Concepts.*Socio-Economic Planning Sciences 43(1):1-24.

[18] Andersson, Värbrand. (2007). *"Decision support tools for ambulance dispatch and relocation."*. Journal of Operational Research Society 58, 195-201.

[19] Zahng, Li. (2007). *"An Empirical Study of Learning from Imbalanced Data"*. ADC '11 Proceedings of the Twenty-Second Australasian Database Conference - Volume 115, p 85-94.

[20] Singh, Singh Mangat. (1996). *"Elements of Survey Sampling"*. ISBN: 978-94-017-1404-4, Volume 15. Springer Netherlands.

[21] Francisco, Carmen. (2014). *"100% Classification Accuracy Considered Harmful: The Normalized Information Transfer Factor Explains the Accuracy Paradox"*. `http://dx.doi.org/10.1371/journal.pone.0084217`.

[22] Provost, Fawcett. (1997). Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distribution. CiteSeerX.

[23] Zhang, Wu. (2012). *"Classification of Fruits Using Computer Vision and a Multiclass Support Vector Machine"*. Sensors, 12(9), 12489-12505.

[24] Nguyen_1, Wang, Nguyen_2. (2013). *"Random forest classifier combined with feature selection for breast cancer diagnosis and prognostic"*. Journal of Biomedical Science and Engineering Vol.6 No.5, Article ID:31887.

[25] Chen, Liaw, Breiman, (2004). *Using Random Forest to Learn Imbalanced Data*. ResearchGate accessed 15, may, 2017. `https://www.researchgate.net/publication/254196943_Using_Random_Forest_to_Learn_Imbalanced_Data`.

[26] Saed Sayad, (2017). *OneR*. Accessed 24 May 2017. `www.saedsayad.com/oner.htm`

[27] Google. (n.d.).[Google map for finding care facilities and corresponding longitude and latitude]. Retrieved 10 April from `https://www.google.se/maps?source=tldsi&hl=en`.

[28] MapQuest (n.d). [MapQuest distance matrix API for finding distances between points.] Retrived at 10 April from `https://developer.mapquest.com/documentation/directions-api/route-matrix/post/`.

[29] Czepiel, (2016).*Maximum Likelihood Estimation of Logistic Regression Models: Theory and Implementation*. https://www.researchgate.net/.

[30] .Scikit-learn documentation page, retrieved at 1 may 2017 .`http://scikit-learn.org/stable/modules/linear_model.html#logistic-regression`.

[31] Schmidhuber, (2015). *Deep learning in neural networks: An overview*. Elsevier. Neural Networks 61, 85-117.

[32] Sammut, Webb, (2010). *Encyclopedia of Machine Learning*. Springer. p 744.

Appendix A

Haversine formula

This formula is a way of calculating the linear distance between two coordinates on a sphere, such as the surface of the earth.

R = earth radius, mean value 6371 000 m was used in this project.
ϕ = lattitude coordinate, λ = longitude coordinate, d = final distance in meters
The angles are given in radians.

$a = sin^2(\Delta\phi/2) + cos(\phi_1) * cos(\phi_2) * sin^2(\Delta\lambda/2)$
$c = 2 * atan2(\sqrt{a}, \sqrt{(1-a)})$
$d = R * c$

Scikit-learn documentation

Below are brief explanations of some of the hyperparameters used in the models. There is also links to the documentation page for the classifiers. The material was retrieved 17 may 2017.

URL to documentation page for random forest classifier:
`http://scikit-learn.org/stable/modules/generatedhttps://preview.overleaf.com/public/ffxmyhfnvnyv/images/b0d6d0af44a7894d159bcd7a9828e3612a2f5c3d.jpeg/sklearn.ensemble.RandomForestClassifier.html`

Max features - What size of a random subset of features that should be considered when doing a split at a node. A small max feature value reduces variance but increases bias, and the opposite when a greater number of features is used.

N-estimators - Number of trees in the forest. Increasing number of trees will often give a better result but also increases computational time.

Max depth - How far a tree should grow until it constructs a leaf node.

Class weight - Weights associated with each class to make the tree consider some classes more than others.

Logistic regression classifier:
`http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`

Solver - Specifies which optimization algorithm to use.

Multi class - Whether to solve the problem by fitting several binary classifers to the problem or fit a single classifier mimimizing the multinomial loss fit across the entire probability distribution.

Max iter - How many iterations the solver algorithm should do.

Class weight - Same as for random forest.

Neural Network:
`http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html`

Hidden layers - How many layers that should be used between the input and the output layer.

Activation - Which activation function to use for the hidden layers.

alpha - Determines step size. Coefficient before regularization term when L2 regularization is used:

$$L_2(w) = alpha * |w|_2^2 \tag{A.1}$$

Max iter - How many iterations the solver algorithm should do. The optimal value of max iter is related to the step size, alpha, since a smaller step size might need more iterations to converge to a final solution.

Randomized Logistic Regression
`http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RandomizedLogisticRegression.html` Settings used: C - 1, Scaling - 0.5, Sample fraction - 0.25, N resampling - 200, Selection threshold - 0.25, Tol - 0.001, Fit intercept - True, Normalize - True, Random state - None