Applying neural networks and support vector machines in geographic information systems for determining suitable areas to build railroads and other infrastructural projects within InterCity-Østfold (Project 3)

Ronald Mathew Baysa Payabyab
FYS-STK4155 - Applied Data Analysis and Machine Learning
University of Oslo

December 12, 2020

Abstract

This project studies the potential of applying machine learning to create land subsidence prediction maps for site planning and engineering projects. The study area consists of the InterCity-Østfold area where a railtrack between Fredrikstad and Råde was proposed where heavy risk of ground deformation activity related to infrastructural projects might occur. It will emphasizes on using machine learning algorithms related to non-linear kernels such as the Kernel Ridge Regression and Support Vector Regression and its hyperparameters, but will also study some of the concept of neural networks. All three methods appears to generate the same range of R2-scores between 0.45 and 0.51 where SVR gaining the highest value (0.5101). They do however manage to predict the locations of reasonable susceptible areas with exception of neural networks. Most of the higher predicted values lays around the urbanized areas with heavy infrastructures or nearby water bodies for KRR and SVR. It was also discovered that every kernel used in the analysis could achieve optimal results as long as the other hyperparameters are tuned properly. Applying machine learning for infrastructural projects has potential to be used in the industry, but there are some aspects of human judgment that simply could not be replaced by machine learning, and thus should only be recommended as a compliment to other evaluations.

1 Introduction

The application of artificial intelligence for spatial processing and analysis have recently been under development within the GIS-community. Many machine learning models such as logistic regression, random forests, support vector machines and neural networks has been applied in geographic information systems to create maps for different purposes from predicting susceptibility of gully erosion (Shahabi et al., 2019) to even determining the most profitable location to establish a hotel (Yang, 2015). The main reasons why machine learning models are getting more widely used in the fields of spatial analysis and engineering are the remarkable performance, flexibility and accuracy in modeling and predicting phenomena whereas knowledge based models depend highly on expert's judgment and are associated with uncertainty. On the other hand, rigid systems such machine learning may not always function as the designers have intended since there are aspects of human decision-making that are difficult to automate (Sui, 1994). Therefore an automated strategy that can simulate the experts' learning and reasoning process would be highly highly desirable.

For this purposes, the aim of the project is to use machine learning algorithms to create a map predicting suitable places for site planning within the municipalities of Råde and Fredrikstad based on the risk of ground subsidence. The problem will be considered as a regression problem since it is preferred to have a continuous risk scale. The target map/value is a unitless scale from 1 to 10 based on a knowledge-based model for estimating ground subsidence, where higher values indicates more risk of ground subsidence. The report will briefly present the background of machine learning in GIS and the study area. Then the concept of neural networks, kernel ridge regression and support vector machines (regression) will be introduced. An explanation of how the data set are created and the hyperparameters and tools used to perform the chosen machine learning methods will also be explained, and the final maps will be created from those three machine learning algorithms based on the most optimal hyperparameters after tuning. The resulting maps will be evaluated and discussed if the predictions are reasonable with the original map layer (target value).

2 Background

2.1 GIS, decision making, and machine learning

One of the critical aspects of applying spatial analysis for decision making processes is that it involves evaluation of geographic events based on chosen criteria and the decision maker's preferences to the given set of criteria. This is particularly troublesome in larger engineering problems where experts involved in a project have different opinions of what criteria to involve and how important each criterion is relative to each other. Karlsson et al. (2017) for instances used spatial analysis in GIS to determine susceptibility of natural hazards for road

planning projects. The susceptibility assessment was subjected to uncertainties due to more or less subjective, and partly inconsistent expert judgement of criteria used to determine natural hazards.

Since the processes of natural hazards are quite complex in nature and requires huge amount of data to represent each conditional criteria properly, data mining approaches has gradually been implemented in decision-making processes for GIS-environments. Different machine learning methods such as artificial neural networks, support vector machines, and logistic regression has already found its way in spatial analysis. Lee et al. (2004) used ANN to weight criteria of landslide susceptibility in Yongin, South-Korea. Determining the weights of criteria that potentially causes landslides were investigated in order to compare performane between data mining approaches, and ordinary spatial analysis where the decision maker applies weights to the criteria subjectively. Similarly, Bagheri et al. (2019) also used ANN to predict and create an groundwater-induced subsidence map in Iran. The study indicates that using ANN as a machine learning approach were effective enough to be used as complement to radar measurements of the ground.

2.2 Study area

The purpose of this project is to predict the most suitable places for site planning and building projects within the InterCity-Østfold area based on the risk of ground subsidence. Ground subsidence is a gradual settling or sinking of the Earth's surface, and has particularly been troublesome in city areas and other places with heavy infrastructure. For instances, the Norwegian Road Authorities has documented gradual land subsidence related to larger infrastructural projects in Bjørvika, Oslo which has caused precautions when building a new tram line in the area (Vegvesen, 2016).

The risks of ground subsidence is also prevalent in the former county of Østfold as there are observed to be some subsidence activity in the city center of Fredrikstad, due relatively high clay concentration in the subsurface. The subsurface in the area are affected by old faults or fraction zones of deep weathering that has the potential to transform rocks into clay minerals which poses a threat to building new infrastructure in the area. This project will in particular focus on the InterCity-Østfold zone where Bane NOR is planning to build a 16-17km double railtrack between Haug in Råde municipality and in the city of Fredrikstad. The goal of the InterCity project is to shorten the travel time between Oslo and Råde by 40 minutes and, 8 minutes between Fredrikstad and Råde. This will cover a larger geographical area where the placements of new train stations should also be taken into consideration for risk assessment along with building tunnels and rail tracks.

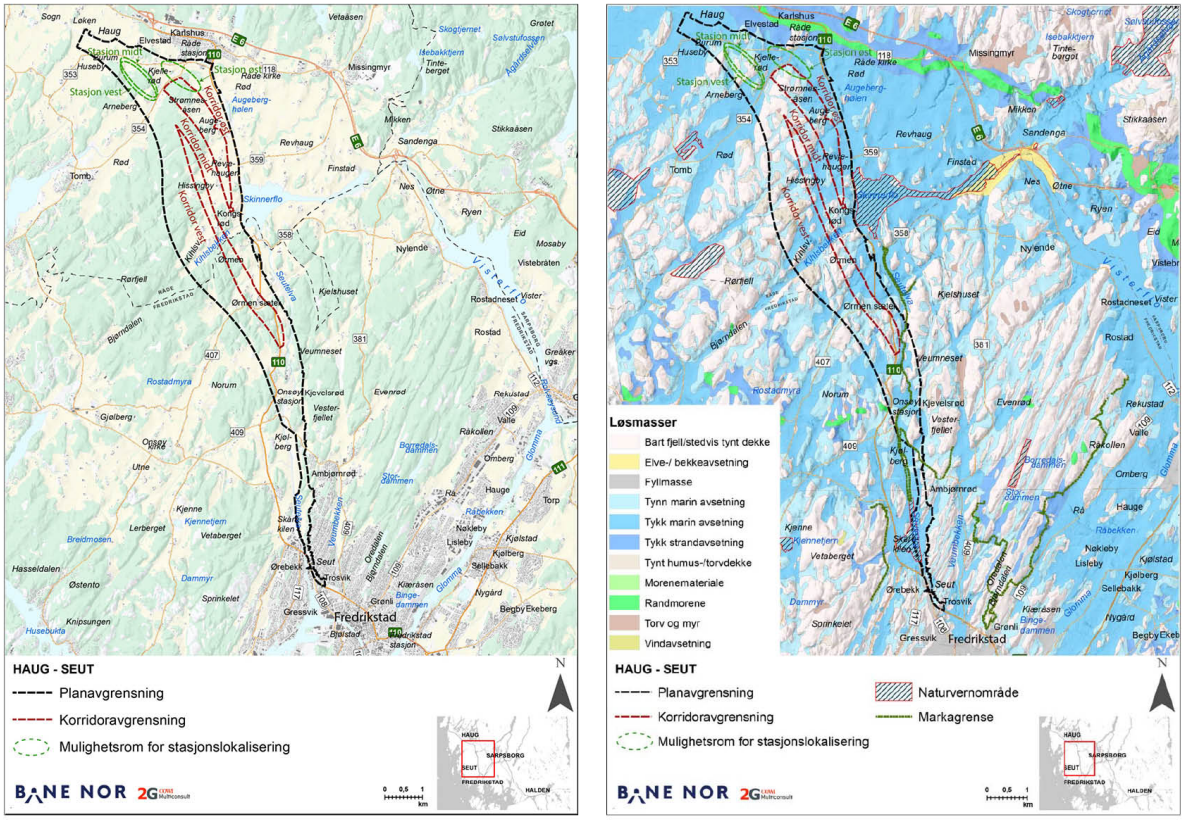


Figure 1: The planned railtrack between the municipalities, and their geological conditions. A majority of the study area contains marine clay deposits, which makes it more susceptible to ground deformations. Source: BaneNor (2018)

The entire study area consist of the three municipalities of Råde, Fredrikstad and parts of Sarpsborg, which has a total land mass of $811 km^2$. The terrain is hilly with elevation varying between 0 to 70 m.a.s.l with some hills extending to 100 m.a.s.l. Most of the InterCity area are still experiencing a post-glacial rebound, meaning that the land masses are still heaving upwards after glaciers have retreated during the last ice age. In addition to the presence of soft marine clay, this makes it challenging to find a suitable place to build railroads due to the recurring risk of subsidence or uplift. Physical radar measurements are taken in the area to document the rate of subsidence or uplift, but has proven to be challenging due to the change of subsidence happen at a very slow rate. An attempt to use machine learning techniques to predict the deformation of ground based on seven different factors was thus currently under development to verify the subsidence rate of the area.

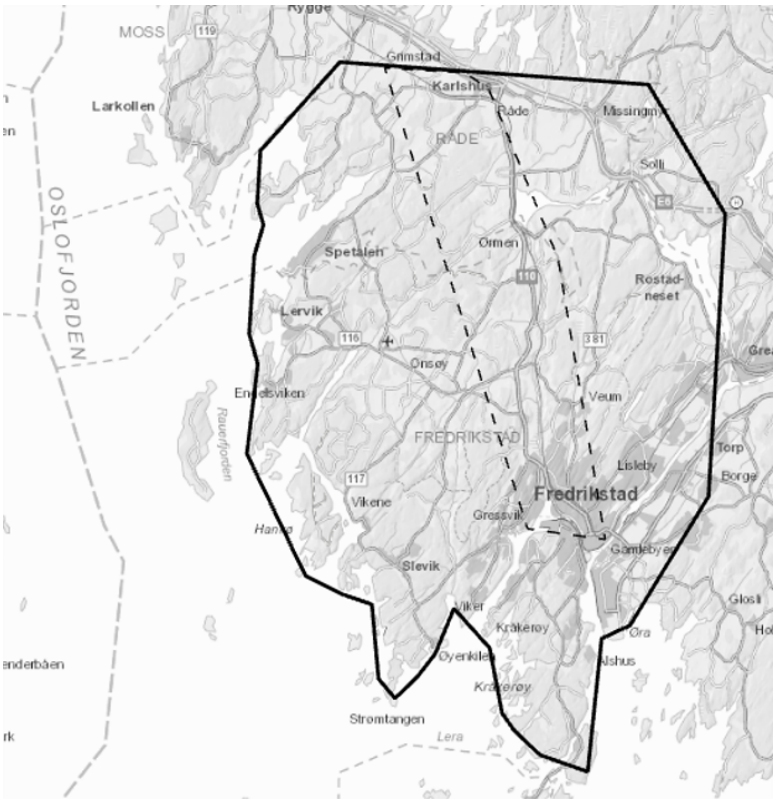


Figure 2: Extent of the data set and the study area presented in GIS. The area within the black line represents the area with data available for all features. The area inside the dashed line represents the actual study area.

2.3 The algorithms

2.3.1 Neural network regression

A neural network is an information-processing approach approach that is inspired by the way biological nervous system process information. It consist of a large number of interconnected processing elements called neurons which are working together to resolve a specific problem. Each neuron responds to the weighted inputs it receives from the other neurons. The neural network architecture are therefore referred to the total arrangement of nodes between the layers.

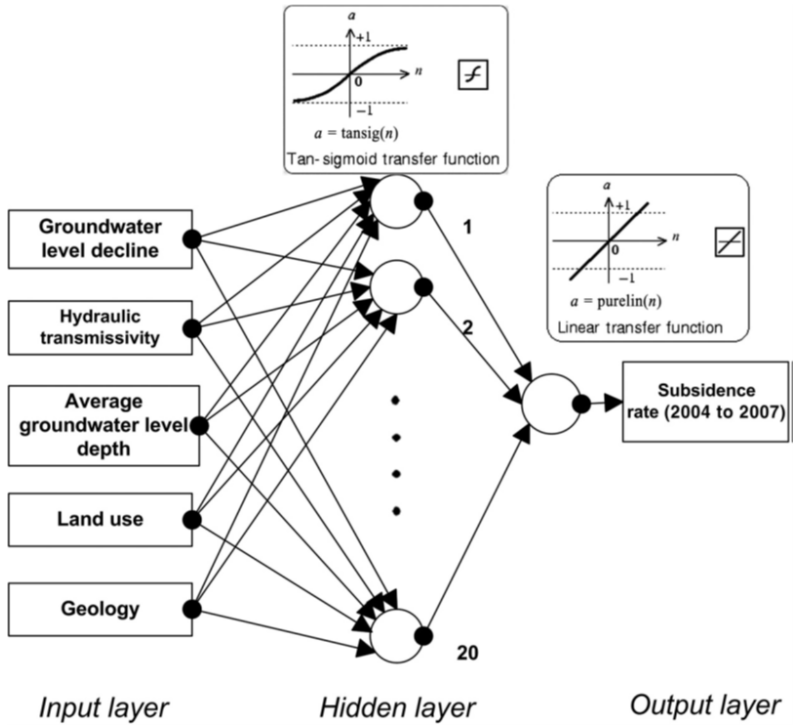


Figure 3: An example of the neural network architecture for predicting land subsidence using 5 criteria that potentially could influence the risk. Source: Bagheri et al. (2019)

The neural network have multiple layers that consist of an input layer, hidden layers and an output layer. The last two process the input they receive from the input layer by multipying each input by a corresponding weight, summing the product, and then processing the sum using a non.linear transfer function (activation function) to produce the result. The aim is to a model of the data-generating process in order for the neural network to predict and produce outputs from inputs that it has not previously seen. Learning the system involves adjusting the connections that exist between the neurons which are often achieved using a backpropagation algorithm. The backpropagation algorithm has two distinct passes, namely the forward pass and the backward pass. In the forward pass, the weights are unchanged throughout the network and the function signals are calculated on a neuron to neuron basis. The function signal , $y_j(n)$, appearing at the output of a neuron j can be described as:

$$y_j(n) = z[v_j(n)] \tag{1}$$

where $v_j(n)$ represents the induced local field of neuron j which is defined as:

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) \tag{2}$$

m stands for the total number of inputs applied to neuron j, while $w_{ji}(n)$ is the synaptic weight connecting neuron i to neuron j, and $y_i(n)$ is the input of neuron j or the functional signal appearing on the output of neuron i. The function $z(.)$ is the activation function and

is normally a non-linear function applied to the weighted sum of inputs before the signal propagates to the next layer. For this particular project, the sigmoid function is used for the hidden layers, while the linear function is used for the output layer, which are respectively expressed as:

$$z(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

$$z(x) = x \tag{4}$$

Then, the output $y_j(n)$ is compared with the desired response. In brief, the backpropagation algorithm updates the weights based on patterns until one epoch has been dealt with. The weights are adjusted accordingly to the respective errors computed for each pattern introduced to the network. Thus, the forward phase begins at the first hidden layer by introducing it to the input vector and ends at the output layer by calculating the error signal for each neuron of this layer. The backward pass on the other hand, starts at the output layer by passing the error signals backward through the network, and recursively computing the local gradient δ for each neuron, described as:

$$\delta_j = z_j'[v_j(n)] \sum_k \delta_k(n) w_{kj}(n) \tag{5}$$

The local gradient then applies a correction $\triangle w_{ji}(n)$ to the weight w_{ji} defined as:

$$\triangle w_{ji}(n) = \alpha \triangle w_{ji}(n-1) + \eta \delta_j(n) y_i(n) \tag{6}$$

where η is the learning rate parameter and α is the momentum constant. The local gradient and $z_j'[v_j(n)]$ relies on the activation function associated to hidden neuron j. Additionally, the local gradient requires knowledge of the error signals for all neurons that is situated in the layer located on the right side of hidden neuron j.

2.3.2 Kernel Ridge Regression and the kernel trick

Before diving into support vector machines, we will first introduce a similar method called Kernel Ridge Regression (KRR). KRR is non-parametric form of Ridge regression with an applied kernel function.

The aim is to learn a function in the space induced by the respective kernel k by minimizing a squared loss with a squared norm regularization term. A kernel function, $K : X \times X \to \mathbb{R}$, is a symmetric and positive definite (strictly positive for every non-zero column vector of real numbers) function. There are many types of kernels such as the Gaussian RBF (Radial basis function) and the polynomial kernel. These kernels are also applied in SVM which will be discussed later. The importance of the kernel functions comes from the property that every positive definite kernel K is related to a mathematical space, H_k (Reproducing kernel Hilbert space, RKHS). such that applying K to two feature vectors, X_1,X_2 is equivalent to projecting these feature vectors into H_k by a projection function, ϕ and taking their inner product by:

$$K(X_1, X_2) = \langle \phi(X_1), \phi(X_2) \rangle_{H_k} \tag{7}$$

This is also known as the "kernel trick". If input features are involved in the equation of a statistical model only in the form of inner product, then it is possible to replace the inner products in the equation with calls to the kernel function. The kernel function will behave in a way as if we had projected the input features into a higher dimension and taken the inner product there without having to perform the actual projection.

KRR is simply an expansion of Ridge regression with an application of the kernel trick. If we have a kernel such that $K_{i,j} = k(X_i, X_j)$. Then the equations for KRR can be defined as:

$$\begin{aligned} \hat{y}' &= \sum_{i=0}^{n} \alpha_i k(X_i, x') \\ \alpha &= (K + \lambda I)^{-1} Y \end{aligned} \tag{8}$$

Where k is the kernel, α represents the weights, and K is the kernel matrix. Additionally, in the function space we want to fit a function $f(x)$ that minimizes:

$$\hat{f}(x) = min_f \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda ||f||^2 \tag{9}$$

$||f||^2$ is the penalty term that penalizes rough functions. The minimizer will be in the form of $\hat{f}(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i)$.

2.3.3 Support Vector Machines (Regression)

Support vector regression (SVR) is another non-linear function that applies the kernel trick. KRR and SVR are quite similar to each other such that they both learn a linear function in the space induced by a chosen kernel which correspond to a non-linear function in the original space. They do on the other hand differ in the loss function. SVR differs by optimizing with a quadratic ϵ-insensitive loss function. The goal of SVR is to create a hyperplane defined by a slope vector w and the intercept b. The purpose is to find two hyperplanes $wx + b = -\epsilon$ and $wx + b = \epsilon$ that covers most of the training data. In other words, most of the data points are bounded in ϵ bands of the optimal hyperplane.

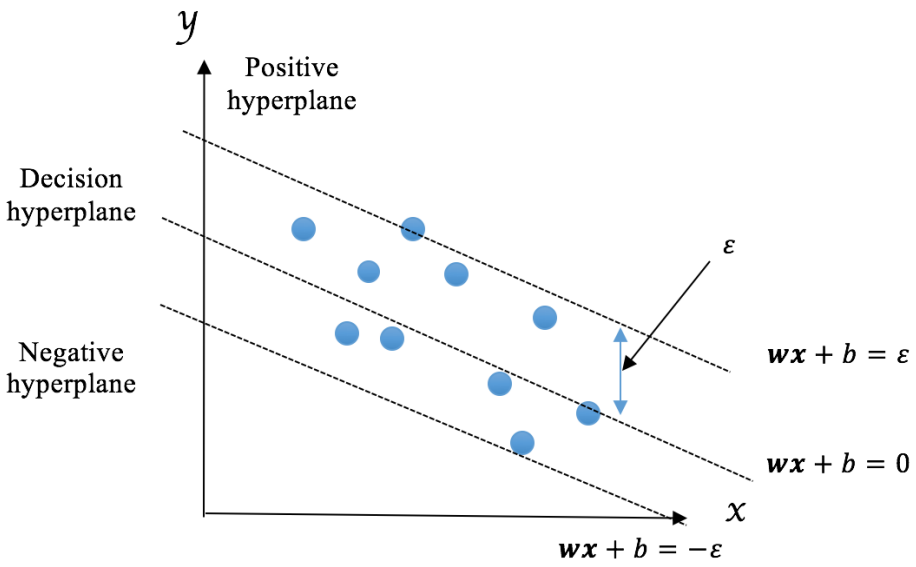


Figure 4: The hyperplanes represented in SVR. It is represented in terms of support vectors, which are training samples that lie outside the boundary of the tube (Liu, 2020).

Support vector machines (SVM) for regression problems mostly uses the same principles as of classification problems, maintaining all the main features that characterize the algorithm. Since the output is a real number it becomes difficult to predict the information at hand, which has infinite possibilities. In the regression case, the margin of tolerance, ϵ, is set in approximation to the SVM. The general purpose of the SVM is the same regardless of cases with minimizing the errors and individualizing the hyperplane, which in turn maximizes the margin.

If we have a set of training data where x_n is a multivariate set of N observation with observed resonse values y_n, the linear function can be described as:

$$f(x) = x^T w + b \tag{10}$$

The function $f(x)$ should be as flat as possible by minimizing:

$$J(w) = \min \frac{1}{2}||w||^2 \tag{11}$$

subject to all residuals having a value less than ϵ, giving:

$$\forall n : |y_n - (x_n^T w + b)| \leq \epsilon \tag{12}$$

However, it is also possible that no function $f(x)$ exists to satisfy these constraints for all points. To deal with such constraints, slack variables ξ_n and ξ_n^* has to be introduced for each point. This is similar to applying a soft margin in SVM classification, because the slack variables allow regression errors to exist up to the value of ξ_n and ξ_n^* and still satisfy

the required conditions. With slack variables included to the objective function, the former minimization function can be upgraded to:

$$j(w) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \tag{13}$$

with the following constraints:

$$\begin{aligned} y_i - wx_i - b &\leq \epsilon + \xi_i, \\ wx_i + b - y_i &\leq \epsilon + \xi_i^*, \\ \xi_i, \xi_i^* &\geq 0 \end{aligned}$$

The equation above represents the primal formula. The constant, C, is a positive numeric value that controls the penalty imposed on observations that is situated outside the margin ϵ and helps prevent overfitting. This determines the trade-off between the flatness of $f(x)$ and the amount up to which deviations larger than ϵ are tolerated. The loss is measured based on the distance between the observed value y and the ϵ-boundary formally described as:

$$L_c = \begin{cases} 0 & if \quad |y - f(x)|, \\ |y - f(x)| - \epsilon, & otherwise \end{cases}$$

This portrays an ϵ-insensitive error measure, ignoring errors of size less than ϵ (Hastie et al., 2001). There is a rough analogy that with the support vector classifiers, where points on the "correct" side of the decision boundary and far away from it are ignored in the optimization. In regression cases, those low error points are the ones with small residuals.

The equations above discusses the primal problem in an SVM, but is also possible to express a related problem, called the dual problem. The solution to the dual problem normally gives a lower bound to the solution of the problem, but under certain conditions it might even have the same solution as the primal problem (Géron, 2017). To obtain the dual formula, a Lagrangian function is constructed from the primal function by applying nonnegative multipliers α_i and α_n^* for each observation of x_n. This leads to the dual formula as an attempt to minimize:

$$L(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha^{(i)}\alpha^{(j)}t^{(i)}t^{(j)}\mathbf{x^{(i)^T}} - \sum_{i=1}^{N}\alpha^{(i)} \tag{14}$$

subject to $\alpha^{(i)} \geq 0$ for $i = 1, 2, \ldots, N$.

Once the vector $\hat{\alpha}$ that minimizes the equation is found, it would be possible to compute $\hat{\mathbf{w}}$ and the bias $\hat{b}$ that minimizes the primal problem by using:

$$\begin{aligned} \hat{\mathbf{w}} &= \sum_{i=1}^{N}\hat{\alpha^{(i)}}t^{(i)}\mathbf{x}^{(i)} \\ \hat{b} &= \frac{1}{n_s}\sum_{i=1}^{N}(t^{(i)} - \hat{\mathbf{w}}^T\mathbf{x}^{(i)}) \end{aligned} \tag{15}$$

when $\hat{\alpha}^{(i)} > 0$.

Some data sets might be too complex to be described in a linear kernel. In such cases, the Lagrange dual formula can be extended to nonlinear functions. A non-linear SVM regression model can be obtained by replacing the dot product $t^{(i)}t^{(j)}$ with a non-linear kernel function $G(x_i, x_j) =< \phi(x_i), \phi(x_j) >$ where $\phi(x)$ is a transformation that maps x to a high-dimensional space (Géron, 2017). Common kernels are listed below:

Kernel name	Function
Linear	$G(x_i, x_j) = x_i^T x_j$
RBF	$G(x_i, x_j) = \exp(d(x_i, x_j)^2/2l^2)$
Polynomial	$G(x_i, x_j) = (1 + x_i^T x_j)^q$

The radial basis-function kernel (RBF) function is a stationary kernel. It is parameterized by a length scale parameter $l > 0$. l represents the scale of the kernel and $d(.,.)$ is the Euclidean distance. q in the polynomial kernel function on the other hand stands for the number of degrees in the polynomial.

As mentioned in kernel ridge regression, kernels are functions that are capable of computing the dot products $\phi(x_i)^T$ and $\phi(x_j)$ based on the original vectors x_i and x_j without having to compute the transformation ϕ. If we apply the formula for $\hat{\mathbf{w}}$ into the decision function of a new instance, it is possible get an equation with only dot products between input vectors. Using a kernelized SVM allows us to operate the original feature space without computing the coordinates of the data in a higher dimensional space. The idea is mapping the non-linear separable data set into higher dimensional space where we can find a hyperplane that can separate the samples. Using the kernel trick, the prediction can be expressed as:

$$\begin{aligned} h_{\hat{\mathbf{w}}^T,\hat{b}} = \hat{\mathbf{w}}\phi(\mathbf{x}^{(n)})) + \hat{b} &= (\sum_{i=1}^{N} \hat{\alpha^{(i)}} t^{(i)} \mathbf{x}^{(i)})^T \phi(\mathbf{x}^{(n)}) + \hat{b} \\ &= \sum_{i=1}^{N} \alpha^{(i)} t^{(i)} (\phi(\mathbf{x}^{(i)})^T \phi(\mathbf{x}^{(n)})) + \hat{b} \\ &= \sum_{i=1}^{N} \alpha^{(i)} t^{(i)} G(\mathbf{x}^{(i)}, \mathbf{x}^{(n)}) + \hat{b} \end{aligned} \tag{16}$$

subject to $\hat{\alpha}^{(i)} > 0$.

Since $\alpha^{(i)} \neq 0$ only for support vectors, making predictions involves computing the dot product of the new input vector $\mathbf{x}^{(n)}$ with only the support vectors instead of all the training

instances. The bias, $\hat{b}$, computed using the same kernel trick:

$$\begin{aligned}\hat{b} =& \frac{1}{n_s}\sum_{i=1}^{N}(t^{(i)} - \hat{\mathbf{w}}^T\mathbf{x}^{(i)}) \\ =& \frac{1}{n_s}\sum_{i=1}^{N}(t^{(i)} - (\sum_{j=1}^{N}\hat{\alpha}^{(j)}t^{(j)}\phi(\mathbf{x}^{(j)}))^T\phi(\mathbf{x}^{(i)})) \\ =& \frac{1}{n_s}\sum_{i=1}^{N}(t^{(i)} - \sum_{j=1}^{N}\hat{\alpha}^{(j)}t^{(j)}G(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}))\end{aligned} \tag{17}$$

In a finite input space, if the Kernel matrix is positive semi-definite, then the matrix element K can be a kernel function (Géron, 2017).

2.3.4 Cross validation

When a model has been trained, it is not certain if the model will have the desired accuracy and variance in the production environment. It is necessary to test the data set on unseen data for evaluating the model performance of any machine learning model. Cross validation is a technique that are used to test the effectiveness of the model, and is also a type of resampling procedure to evaluate if the data is limited in the model.

One of the more common methods of cross validation is the train test split approach. The data is randomly split into training and test sets. The model training is then performed on the training set and the test set is used for validation. This method is often acceptable if the data set are relatively large, but does however have the possibility of high bias if data is limited due to lack of information about the data not used for training.

Another approach is to use the K-fold cross-validation technique to reduce the bias. It ensures that every observation from the original data set has the chance to appear in the training and testing set. The entire data is then split randomly into K number of folds, and then the model is fitted using the $K-1$ folds and validate the model using the remaining Kth fold. This process are repeated until every K number of folds have served as the test set.

2.3.5 Error metrics

When performing machine learning analysis on data, it is typical to provide a performance measure for regression problems as it gives an idea of how much error the system makes in its predictions, with a higher weight for large errors. The equation below shows a mathematical

formula of how to compute the mean squared error (MSE):

$$MSE(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y})^2 \tag{18}$$

The MSE provides a description of the squared difference between the predicted data and true data. This is also defined as the cost function the ordinary least square utilizes, meaning that the MSE minimizes the data with respect to the data it was trained on.

Another common metric used in performance measurement is the coefficient of determination (R2 score), expressed as:

$$R2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2} \tag{19}$$

The R2 score gives an idea of how many data points fall within the results of the line formed by the regression equation. The higher score, the higher the percentage of points the line passes through the data points, which indicates that there is a better goodness of fit for the observations. The R2-score is useful to find the likelihood of future events falling within the predicted outcomes.

3 Methodology

3.1 Processing the data sets

Table 1: The data used in the susceptibility assessment, the data sources, and the associated factor classes for the landslide susceptibility mapping in the study area.

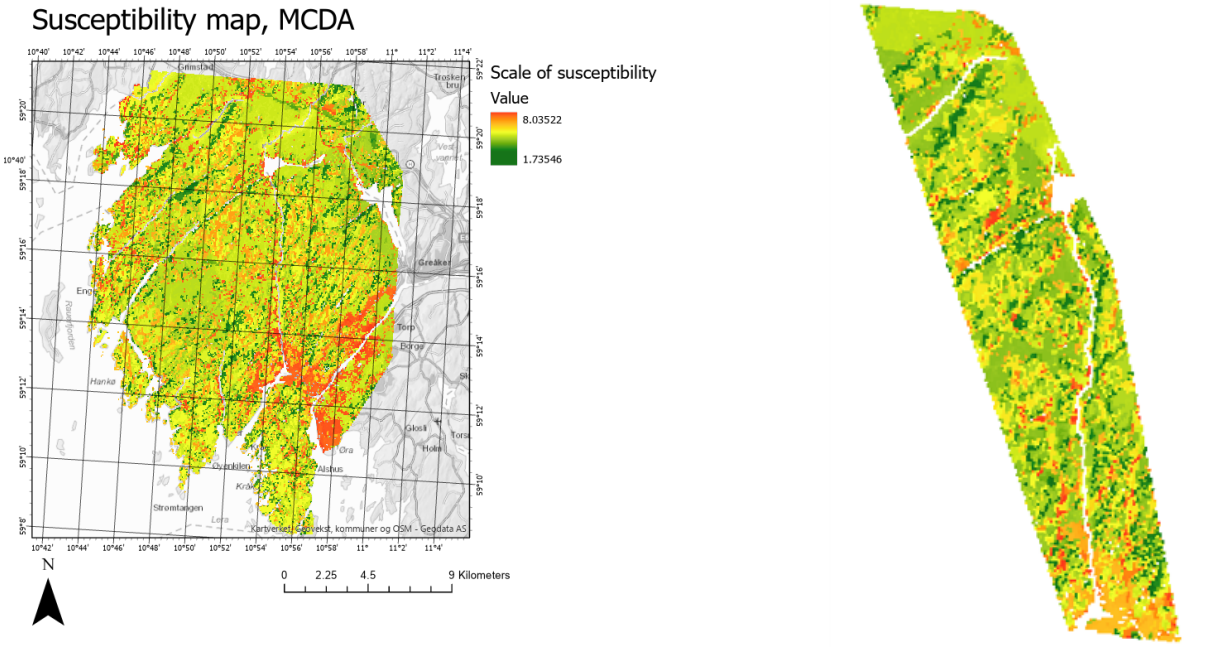
Data layer	Source data	Data type	Scale/Original resolution
Slope	DEM obtained from Kartverket	Raster	50 m
Distance from major streams	River data from NVE	Line	1:50 000
Land cover and land use	LANDSAT-8 OLI	Raster	50 m
Risk of deep weathering	Multiconsult and Norwegian Road Authorities	Polygon	1: 150 000
Lithology	NGU	Polygon	1:50 000
Topographic Wetness Index	DEM obtained from Kartverket	Raster	50 m
Water table altitude	Multiconsult's database	Point	1:50 000

The table shows the features used to create the data set. All the features are originally represented as a map layer with varying scales, resolution and data type from various sources. In order make the data set as consistent as possible, all features are converted to raster data with a resolution of 50 m in ArcGIS where they are stored in a common geodatabase. Editing and manipulation of each features are performed in Python by using the arcpy-module, which are connected to the ArcGIS-software. It was also necessary for all the map layers to have the same map reference system. In this case, European Terrestrial Reference System 1989 (ETRS89) UTM 32N was used as a datum. Some of the map layers were also preprocessed and modified using tools in GIS to create the map layer. The slope was calculated in a planar method with 3×3 neighborhood cells. For each cell, the maximum rate of change is calculated in value from a given cell compared to its neighboring cells. In other words, the maximum change in elevation is defined as the distance between a given cell and its eight cell neighbors, which also identifies the steepest downhill descent from the cell. Mathematically, this is expressed as:

$$slope = \arctan(\sqrt{\frac{\partial z}{\partial x}^2 + \frac{\partial z}{\partial y}^2} * \frac{180}{\pi}) \tag{20}$$

The equation shows that the rate of change of the surface in the horizontal ($\frac{\partial z}{\partial x}$) and vertical ($\frac{\partial z}{\partial y}$) direction from the cell center determines the slope, while $\frac{180}{\pi}$ converts the value from radians to degrees. The distance to major river streams are given in meters. The further away from the river, the higher the value. Land use, risk of deep weathering, and lithology are all represented in classes.

Land use have five classes in total, while risk of deep weathering are represented as classes in a scale from 1 to 5, where 5 indicates the highest risk. Lithology contains all the types of soil that can be found in the region. Each type of soil are represented in a class from 1 to 10 based on grain size and permeability. Similarly to the slope, the topographic wetness index (TWI) is also derived from the DEM. It estimates how much moisture the soil contains in a given area. This is based upon how water drains upslope depending on the steepness of the terrain. The water table altitude indicates the depth where the ground is fully saturated with water. The map layer of the water table altitude is defined by piezometric wells and drill samples in the area performed by Multiconsult in order to measure the pore pressure in the ground. Every single drill sample are stored on a database and also contains a data report that presents geotechnical surveys in field. The raw data files from the drill samples are given in m.a.s.l. It is stored on a private database and are not yet published in the national database for ground surveys (Scheibz, 2020). The final product of the map layer is a result of the difference between the elevation and the measured depth to groundwater table.



(a) Full map layer

(b) Closer look to the study area

Figure 5: The target value of the analysis. This is a pre-made map estimating places that are suitable for site planning and infrastructural projects by using the methofs of MCDA. This is represented as a scale from 1 to 10. On this map, the highest value do not exceed 8.0355 and do not have lower pixel values than 1.73546

The target value in the analysis is a pre-made map of the area showing where ground subsidence are most likely to occur based on the same criteria using a method called multicriteria decision analysis (MCDA). In short, MCDA is a general framework for supporting complex decision-making situations with multiple and, quite often, conflicting objectives that stakeholders and/or decision-makers value differently. In this case, we are interested to determine places that are suitable for site planning and other infrastructural projects. The importance of each criterion were ranked relative to the other criteria, and based upon the rankings, each criterion will be given a weight which resembles the importance in percentage. For this particular map, the criteria were given the following weights:

Table 2: List of criteria and their applied weights. The sum of the weight column is 100 % in total.

Criteria	**Weight (%)**
Land use and land cover	33.5858
Slope	24.3126
Risk of deep weathering	16.5127
Lithology	10.4140
Altitude of groundwater table	6.8511
Topographic Wetness Index	5.0288
Distance to river streams	3.3295

The values of target value y would thus be dependent on the weights assigned. y is determined by the expression $\sum_{i=1}^{N} L_i w_i$ where w_i represents the weights and L_i are the criteria which in turn are entire map layers where each cell in the layer contains a given value. The code for creating the target value and extracting the pixel values are found in **test_script.ipynb**

3.1.1 Data conversion and extraction of data points

The data sets are created based on randomly picked pixel values of seven map layers representing the features and the target value along with their latitude and longitude stored on a geodatabase in ArcGIS. ArcGIS is a platform for organizations to create, manage, share, and analyze spatial data. It consists of server components, mobile and desktop applications, and developer tools.

Approximately 9532 randomly distributed pixel values are extracted in the study area and imported as a csv-file to create the design matrix and target value. It was decided to limit the study area nearby the stretch between Fredrikstad city center to Råde defined in figure 1 to reduce the runtime when converting the data set into non-spatial data. The geographic positions are necessary in order to convert the predicted data back into GIS spatial data. In this case the number of data points to be converted back to GIS is far smaller than the original data set. Since only the prediction of the test data set were taken into consideration, only 2183 data points remains for predicting the susceptibility map layer.

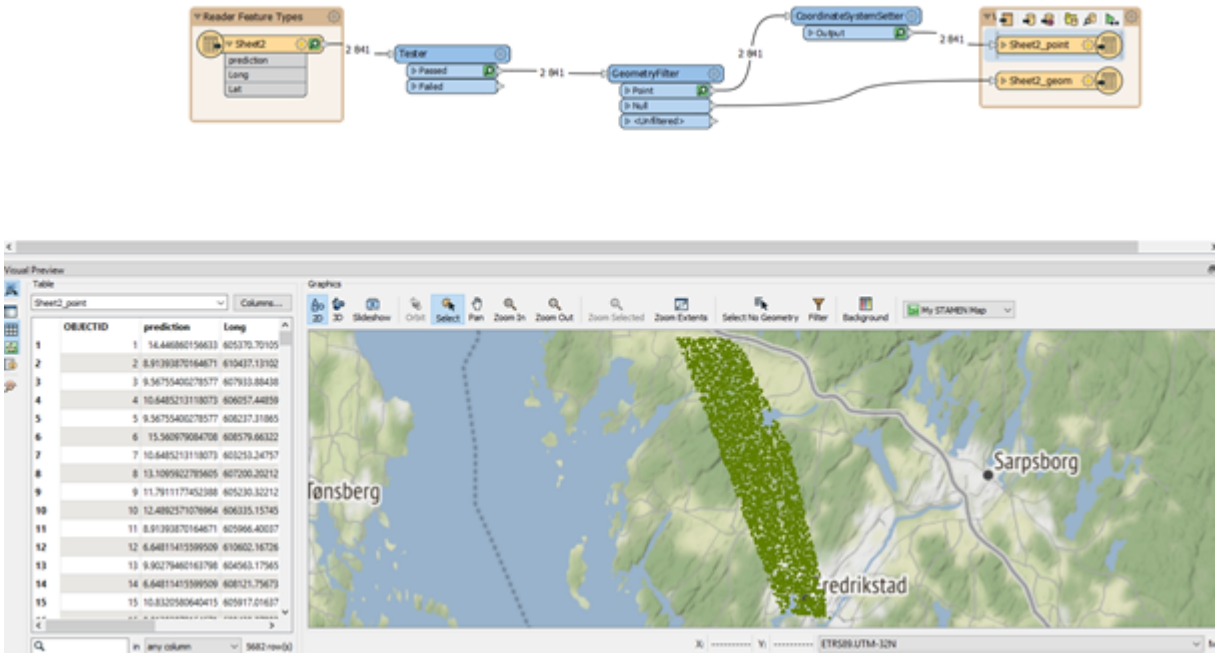


Figure 6: Converting the predicted test data back to a geodatabase file format in FME. Transformers (blue boxes) were used to remove null-values and assigning proper map projection in GIS.

The FME-software was used to convert the geospatial point data into a csv-file and vice-versa. FME is a data integration software tool that easily converts one data type to another. Some additional tools/transformers were also used within FME to remove every row that contains null-values in at least one of the criteria during the conversion process. When the point data set are converted back to the GIS-platform after being trained on a machine learning algorithm, the 2183 predicted test data point are randomly distributed across the defined study area.

To create the final map layer, the data points underwent an interpolation through the techniques of Inverse Distance Weighting (IDW) using tools available in GIS. IDW is a mathematically deterministic method that assumes that closer values are more related than further values with its functions. IDW is chosen as a method due to its flexibility, where it is possible to specify a search radius to determine how many known points should be included to estimate an unknown cell value in the area. In a mathematical sense, the estimation of value z at a location $\mathbf{x}$ is considered as a wighted mean of nearby observations given as:

$$\hat{z}(\mathbf{x}) = \frac{\sum_i^n w_i z_i}{\sum_i^n w_i} \tag{21}$$

where

$$w_i = |\mathbf{x} - \mathbf{x_i}|^{-\beta}$$

$\beta \geq 0$ and $|.|$ corresponds to the euclidean distance. The inverse distance power, β, determines the degree to which the nearer points are preferred over more distant points, and the number of surrounding points, n, decides whether a global or local weighting is applied.

3.2 Neural network

The data set were solved as a regression problem using an object-oriented function. The general neural network architecture contains a feedforward and backpropagation algorithm. A stochastic gradient descent was added for optimizing the training process. The code for the neural network regression problem can be found on **neural_net_regression.py** and the definition of the cost function and activation function are stored in **Regression.py**. Sample runs are performed in **Regression_run.py**. For the regression case, the mean squared error will be used, giving an output error of:

$$\delta^l = \frac{1}{n_{ouput}}(h^l - y)z'(x^L) \tag{22}$$

For more in-depth description about the infrastructure of the neural network and the code implementation, we simply refer to the previous project of the author (Payabyab, 2020).

Due to the size of the data set, only a selected number of hyperparameters were tuned. 30 % of the data set were used for testing while the rest for training. Only sigmoid were tested as an activation function and the structure was set to only contain one hidden layer with 5 neurons. The batch size and number of epochs are fixed with the values 20 and 100 respectively. Only the regularization parameter and learning rate were tuned and tested for $\lambda \in [10^{-5}, 10^{-1}]$ and $\eta \in [10^{-5}, 10^{1}]$ respectively.

3.3 Linear SVM and non-linear kernelized functions

3.3.1 KRR

The code for KRR utilizes Sklearn's functionalities. The data set are split into training and testing sets where 70 % of the data are used for training. Four parameters are used to for grid searching, namely:

Kernel:	"poly", "rbf"
Degree:	3, 4, 5
α:	1, 0.001, 0.0001
γ:	None , 1.0, 0.001

Table 3: Table showing set of values tested for hyperparameter tuning in KRR

The degree-parameter will only be defined in the polynomial kernel and will otherwise be ignored. Additionally, the data set are split into 5 folds by default. The optimal parameters will be decided upon the lowest MSE-value and will be used to fit the data set into a prediction value. The entire code for performing Kernel Ridge Regression can be found in **ridge_kernel.py**.

3.3.2 SVR

Using SVR in the data set has been performed both on Sklearn and Tensorflow. On Sklearn, a simple grid search was used to find the optimal sets of parameters. After finding the optimal parameters, the SVR-program was run 40 times to calculate the R2-score and MSE for each run. The mean value of R2 and MSE were then calculated, and the predicted values were transferred into spatial data.

Kernel:	"poly", "rbf"
Degree:	3, 4, 5
C:	0.1, 1.0, 10.0
γ:	"scale", "auto"
ϵ:	0.01, 0.1, 1.0

Table 4: Table showing set of values tested for hyperparameter tuning in SVR

The table above shows the parameters and its values used to perform a grid search of the optimal set of hyperparameters. Unlike KRR, The main γ-parameters in Sklearn are either defined as "scale" or "auto". When "scale" is chosen, it uses $\frac{1}{n*var(X)}$ as an argument for γ, where n is the number of features in the data set, and $var(X)$ is the variance of design matrix X. On the other hand, if "auto" is chosen, γ is only defined as $\frac{1}{n}$. After finding the optimal kernel in the grid search, the C-parameter and γ-parameter will be looked more closely. Intuitively, the γ-parameter defines how far the influence of a single training example reaches, with low values meaning far and high values meaning close. C represents the penalty just like for kernel ridge regression. The results of R2-score and MSE will be represented as a heatmap. The values explored are relatively small with $C = [10^{-2}, 10^{1}, 10^{3}]$ and $\gamma = [10^{-9}, 10^{-5}, 10^{-2}]$. This is due to the size of the data set as any larger set of values will cause runtime problems. The general grid search using Sklearn are found in **svr_basics.py** and the generation of heat maps to check the correlation between the hyperparameters C and γ on the accuracy are found in **SVR_parameter.py**. A comparison of time execution between KRR and SVR are also measured along with their respective learning rate with increasing training size in **svr_vs_krr.py**

The program in Tensorflow does on the other hand investigate the linear kernel. The code is available on **SVR_tensorflow.py** It will include a loss function that is defined from scratch. The support vector machine are reduced to a linear regression problem to see how it compares

to the non-linear kernel functions. An optimizer will be added to the linear SVR-problem. The main focus is to check how the choice of optimizer, learning rate and the number of epochs could affect the loss function, L_c. The loss function L_c has been discussed as of chapter 2.3.3. In addition, the ϵ-parameter was also tuned to find the optimal R2-score. As mentioned earlier, the ϵ-parameter in SVR works with the ϵ-insensitive hinge loss. The value of ϵ itself defines a margin of tolerance where no penalty is given to errors. The larger ϵ is, the larger error we admit in the solution. Contrarily, if $\epsilon \to 0_+$ every error is penalized.

4 Results and discussion

4.1 Neural Networks

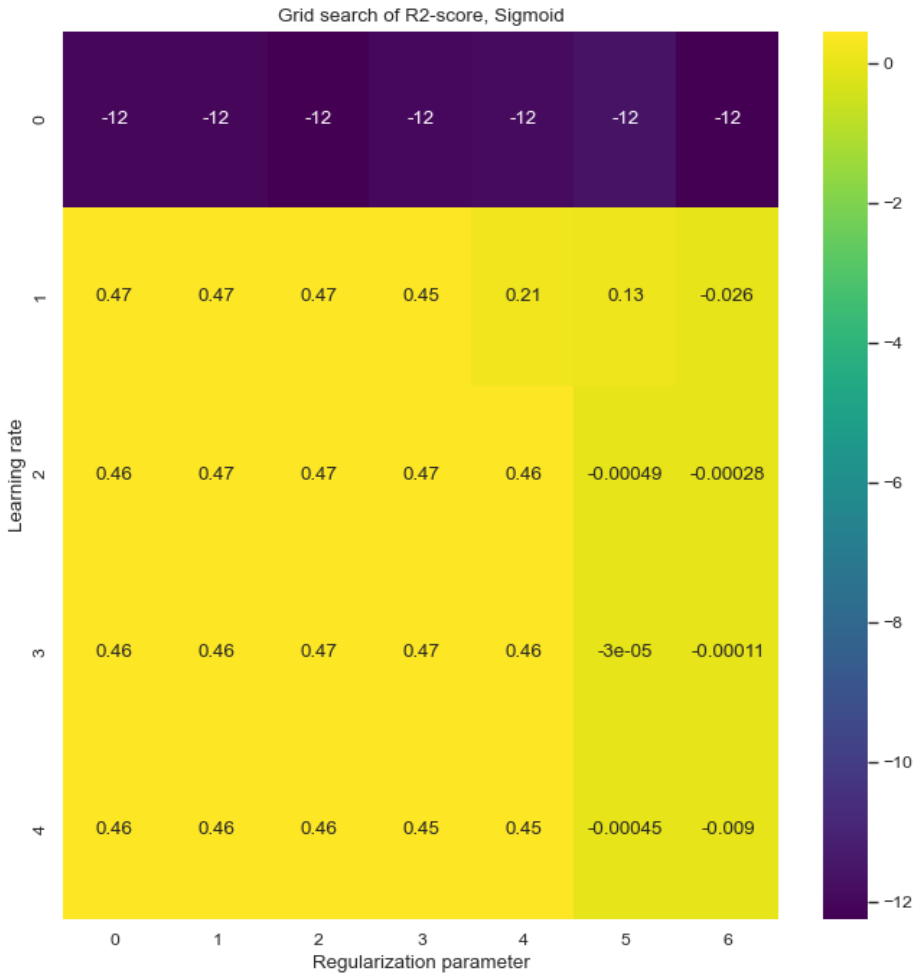


Figure 7: Grid search of optimal R2-score based on learning rate and regularization parameter.

The results from the grid search shows that one could safely exclude having a learning rate of $\eta \leq 10^{-5}$ as it would produce bad results regardless of the value of the regularization parameter. The other learning rates in the analysis appears to $\eta \in [10^{-4}, 10^{-2}]$. $\eta = 10^{-1}$ can also be taken into consideration, but do produce slightly lower R2-scores. The heat map also shows that the regularization parameter, λ, should be relatively small within the range $\lambda \in [10^{-5}, 10^{-2}]$.

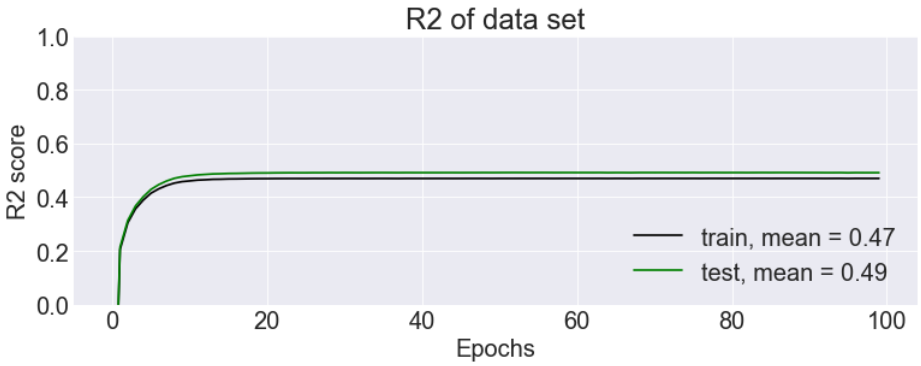


Figure 8: Plot showing the relation between R2 and number of epochs using sigmoid as activation function. The R2-score appears to flatten after approximately 15 epochs.

Based on the heat map, it was decided to choose $\lambda = 0.0001$ and $\eta = 0.001$ for predicting the layer, and then transfer the results and convert them into spatial data. The figure above showcases that it does not require too many epochs before the R2-score remains stable. It might pinpoint that the R2-score will remain around 0.5 when using KRR and SVR even after tuning.

4.2 Kernel Ridge regression

Table 5: The error metrics using RBF-kernel with 5 folds

Gaussian RBF			
α	γ	**MSE**	**R2**
1	None	0.5543	0.4269
1	1	1.6359	-0.6681
1	0.001	0.5073	0.4761
0.001	None	0.8739	0.0983
0.001	1	2.1337	-1.2009
0.001	0.001	0.4994	0.4830
0.0001	None	1.9360	-0.9912
0.0001	1	4.5292	-3.6717
0.0001	0.001	0.4994	0.4828

Table 6: The error metrics of the polynomial kernel

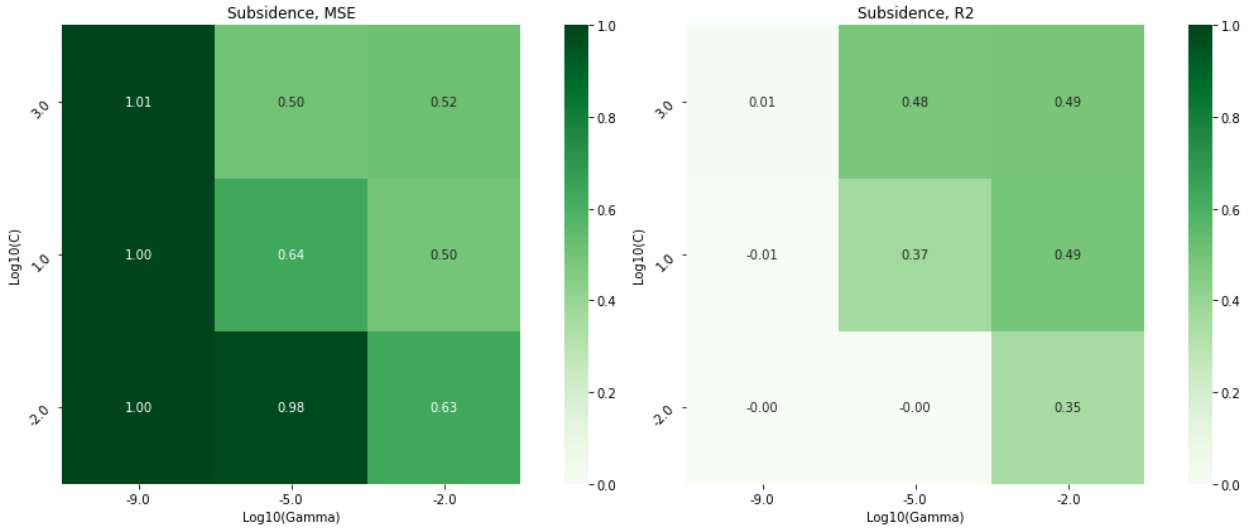
Polynomial kernel				
α	γ	**Degree**	**MSE**	**R2**
1	None	3	0.5087	0.4735
1	1	3	0.5096	0.4726
1	0.001	5	0.5023	0.4801
0.001	None	3	0.5096	0.4726
0.001	1	3	0.5096	0.4726
0.001	0.001	4	0.4994	0.4831
0.0001	None	3	0.5096	0.4727
0.0001	1	3	0.5096	0.4727
0.0001	0.001	3	0.4996	0.4828

Only the Gaussian RBF and polynomial kernels were tested for KRR. The linear kernel were also tested, but the MSE value was too high (24.3768) to be taken into further consideration, indicating that the features in the data set are too complicated to be solved linearly.

The polynomial kernel included the number of degrees as an additional hyperparameter. In the table above, only the optimal degree for each distinct combination of alpha and gamma are included in the table. In general, it appears that the least accurate results occurs when the α-parameter is relatively low and the γ-parameters stays relatively high or not included at all. α is the regularization parameter that are supposed to prevent overfitting of the data set. It appears that it is often necessary to have stronger regularization in this data set in order to improve the accuracy. The best R2-scores all have a γ-value of 0.001 when applying Gaussian RBF. γ itself determines how far the influence of a single training example is relative to the other samples. The tables above shows how important it is to correctly define the γ-parameter. If γ is too small, the model would be too constrained and can not capture the complexity of the function. On the other hand, if γ is too large, then then a single training sample would have too much influence of an area. $\gamma = 0.001$ seems to be the ideal value since the MSE drastically increases if it exceeds that given value.

The error metrics seems to be following another pattern when applying the polynomial kernel. Surprisingly, the R2-score decreases when increasing the polynomial degree. This clearly shows in table 4 where most of the optimal MSE and R2-scores ended up having an optimal polynomial degree of 3. In some cases, the MSE even contained a two-digit number when applying a polynomial degree of 5. Tuning the number of degrees in the polynomial kernel did have a much larger impact in comparison to the α- and γ-parameters. If we combine the the results of applying Gaussian RBF and the polynomial function as the kernel, the optimal MSE-value is reached at 0.4994 with an R2-score of 0.4824 given $\alpha = 0.001, \gamma = 0.001$ using a polynomial kernel of degree 4.

4.3 Support Vector Regressor



(a) MSE (b) R2

Figure 9: Heat maps of the MSE and R2-score of the data set using the RBF-kernel.

The heat maps above shows the MSE and R2-scores of the data set using the RBF-kernel based on the parameters γ and C. The behavior of the model is relatively sensitive towards the γ-parameter as there are clear improvements when $\gamma = 10^{-2}$ compared to $\gamma = 10^{-9}$. The same principles for determining the γ-parameter is the same for SVR as of KRR. When γ is very small, the model would be too constrained and are not able to capture the complexity or the "shape" of the data set. The region of influence of any selected support vector would thus include the entire training set. The model will then behave as if it were a linear model with a set of hyperplanes that separates the centers of high density of any pair of two classes.

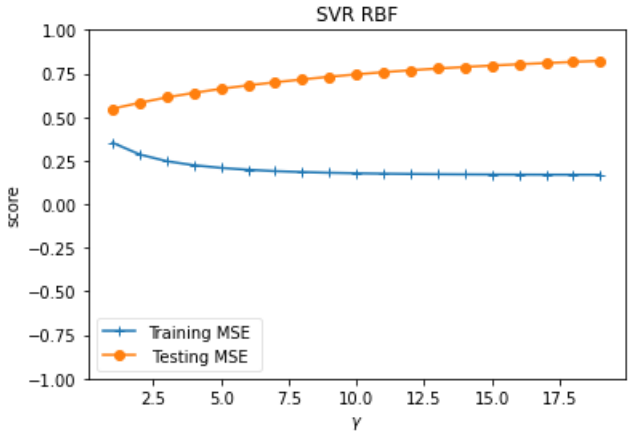


Figure 10: MSE as a function of γ for training and testing set, using RBF-kernel with a fixed C-value (0.1).

However, it also important to not set very high γ-values. If γ is too large, then the radius of the area of influence of the support vectors only includes the support vectors itself and no amount of regularization with the C-parameter could prevent overfitting. The figure above shows that the MSE for the testing set does not improve with increasing γ-value. It is also clear that the MSE for the training and testing set slightly diverge as the γ-values, which indicates signs of overfitting the data set due to the difference between training and testing set.

From the heat maps, it is also observed that for $\gamma = 10^{-2}$, the model performs equally when C becomes very large. It is thus not necessary to regularize the model by adding a larger margin. The radius of the RBF-kernel might be a good structural regularizer on its own. In practice, it should be possible to simplify the decision function by lowering the C-value to use less memory and create models that are faster to predict.

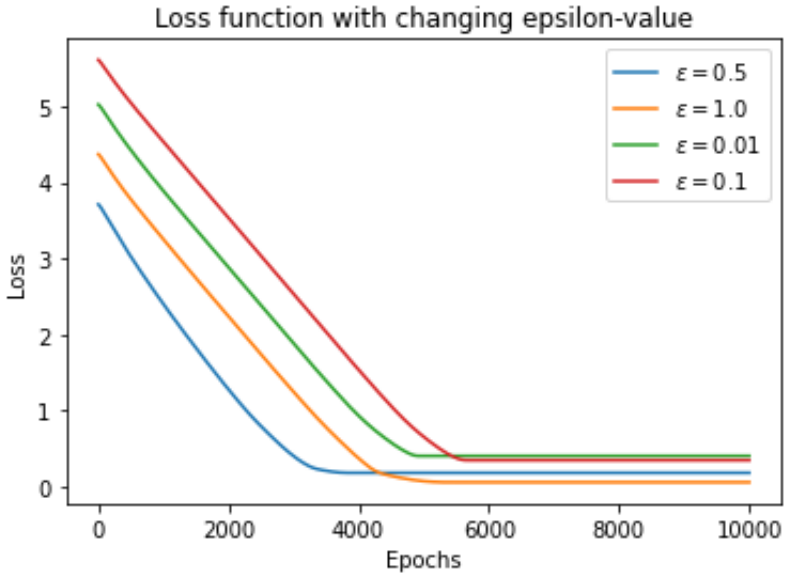


Figure 11: The loss against the number of epochs in linear SVR with different ϵ-values

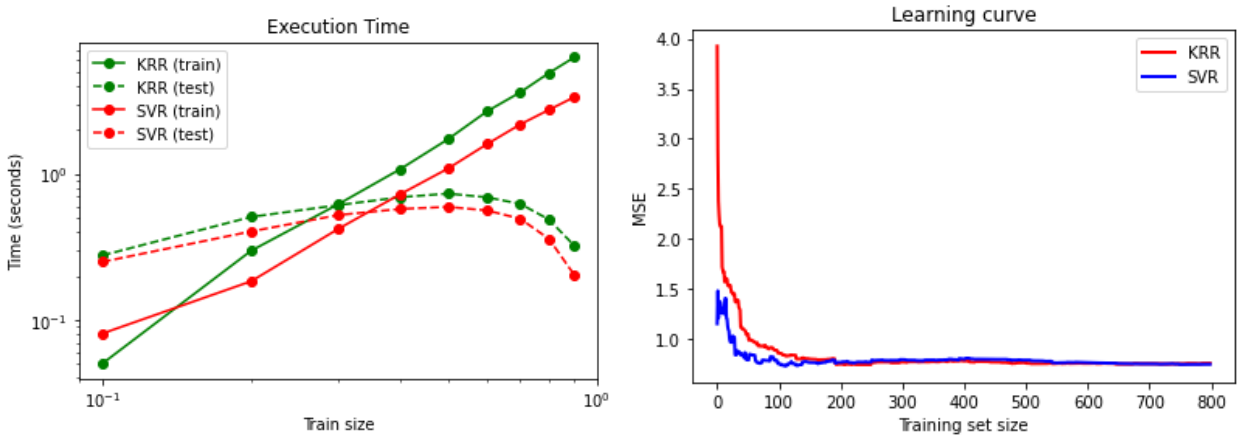
It is observe that the SVR-model has an optimal value when ϵ is between 0.5 and 1.0 for all epochs as any smaller value will yield higher initial loss and the loss would flatten at a much later epoch as seen from figure 11. One of the goals in SVR is to find a curve that minimizes the deviation of the points to it. The margin is defined by the ϵ, and instances that falls within the margin do not incur any cost. It denotes how much error that can be allowed per training data instance. If larger errors are allowed, then there will be fewer support vectors involved and if ϵ is small then the model will have a larger number of support vectors.

Table 7: R2-score of linear SVR by ϵ-value

ϵ	**R2**	**No. support vectors**
0.01	0.3674	6042
0.1	0.4336	3756
0.5	0.5105	2364
1.0	0.3996	1149

The R2-scores also appears to have a correlation to the number of support vectors. Mattera and Haykin (1999) proposed choosing an ϵ which impose the condition that the percentage of support vectors are equal to 50%. This seems to be the case for the data set as the "better" R2-scores are the ones closest to have 50% of its training set as support vectors, namely $\epsilon = 0.1$ and $\epsilon = 0.5$. Smaller or larger numbers of support vectors might also be viable, but depends on the size of data set and noise variance.

4.4 Comparison between KRR and SVR



(a) Runtime

(b) Learning curve

Figure 12: Performances of SVR and KRR

As mentioned earlier, KRR and SVR are similar to each other in a way that they both are able to learn a non-linear function by employing the kernel trick. They only differ by the loss function where SVR utilizes the epsilon-insensitive loss. In this particular data set, it appears that SVR is slightly faster than KRR for most of the defined training sizes, which is a bit unexpected as KRR can be fitted in a closed form. KRR only outperforms SVR for fitting very small training samples. On the other hand, the MSE for KRR are as much as 4 times larger than SVR as shown on the learning curve. When predicting, SVR is without

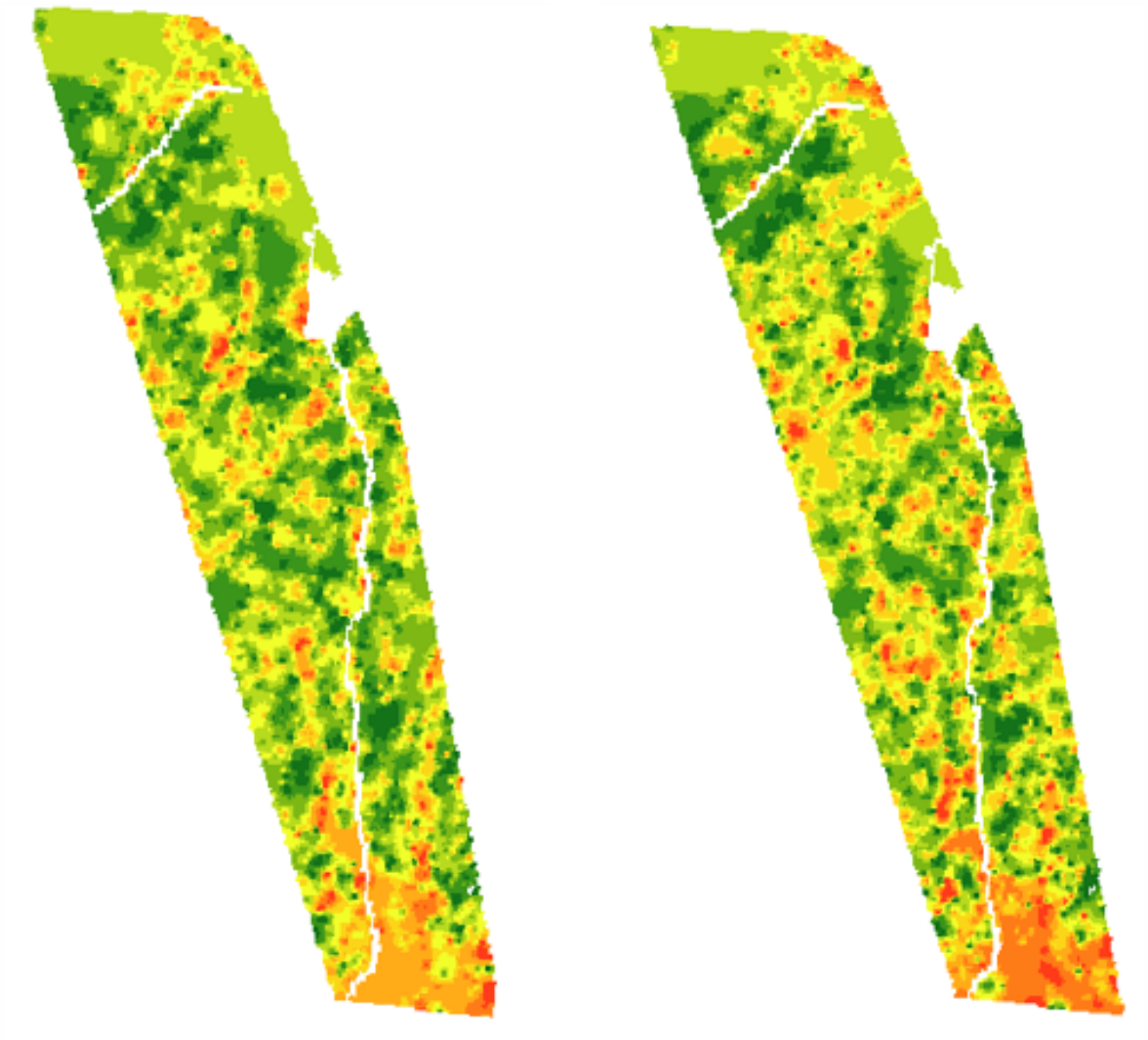
any doubt faster than KRR for all training sample sizes, but the differences are minimal. When measuring the runtime with a grid search of their respective regularization parameter and γ for the RBF-kernel, the time it takes to fit the model are very similar, with 74.995 s, and 74.754 for SVR and KRR respectively. KRR are slightly faster, but with a large and complex data set with 7 features, the choice of method based on runtime are inconsequential.

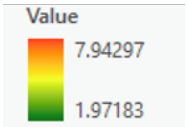
4.5 Susceptibility maps

Figure 13: Predicted map layer of KRR and SVR. The results are presented as a unitless scale from 2 to 8.

(a) KRR

(b) SVR





The figures above shows what the optimal predicted data looks in a map format. The predicted data are defined in a unitless scale from from 1 to 8 where lower values indicates that the area are more suitable for site planning while higher values shows areas not recommended for infrastructural projects based on the 7 criterion/features. The maps for KRR and SVR appears to look similar where the highest values are concentrated in the southern area.

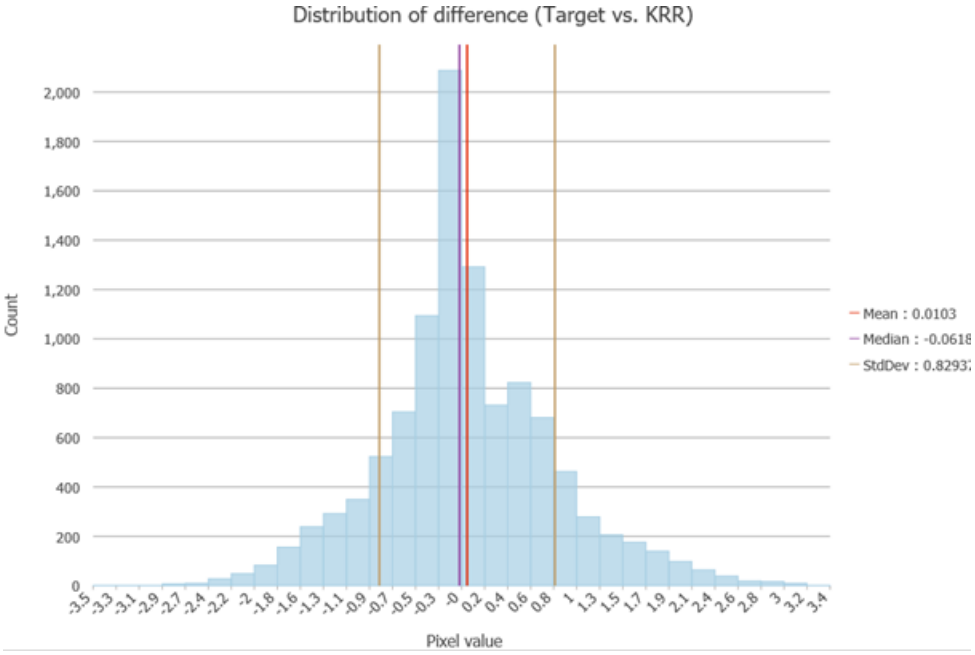


Figure 14: The difference between the true value and the predicted value in KRR

The difference between the target value and the predicted value from KRR appears to be fairly distributed with a mean value of 0.0103 and a median value of -0.0618. There appears to be no huge anomalies and outliers in the prediction, and the predicted map layer has managed to maintain the "shape" of the original map layer to some extent, especially for the southernmost area. It can be concluded that the majority of the pixels in the predicted layer have a relatively small difference of less than ± 1.0

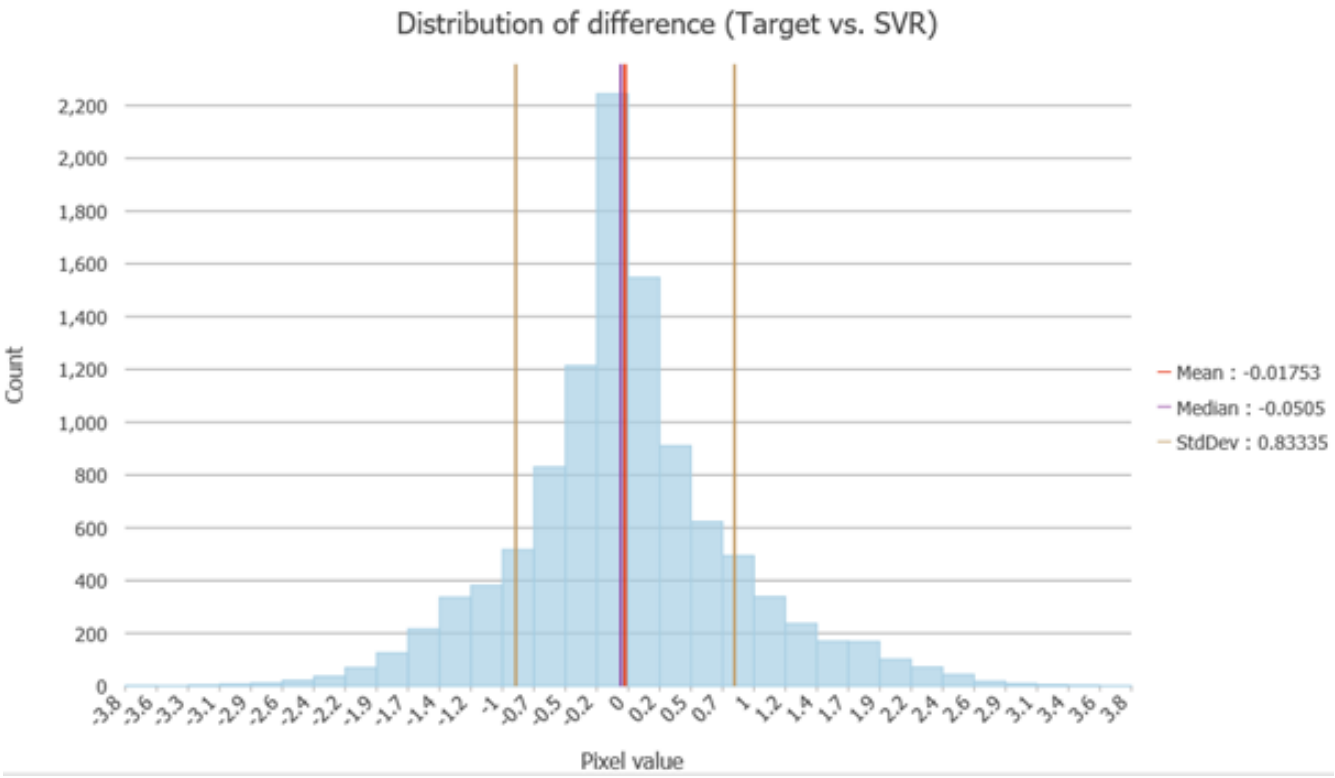


Figure 15: The difference between the true value and the predicted value in SVR

SVR also shows a similar error distribution as KRR, but has a negative mean value of -0.01753, which indicates that the map layer slightly predicts higher values than the target value. The median value is also negative (-0.0505) but also closer to zero. On the other hand, the standard deviation is very similar to KRR and the overall distribution also follows suit with KRR, which explains the similarities of the susceptibility maps.

All in all, there are little differences between the performances of KRR and SVR. The models are proved to be stable with regard to the differences in spatial data as there are only minor differences between the maps from figure 13a and 13b where SVR somewhat slightly rates the southernmost area as higher risk for subsidence.

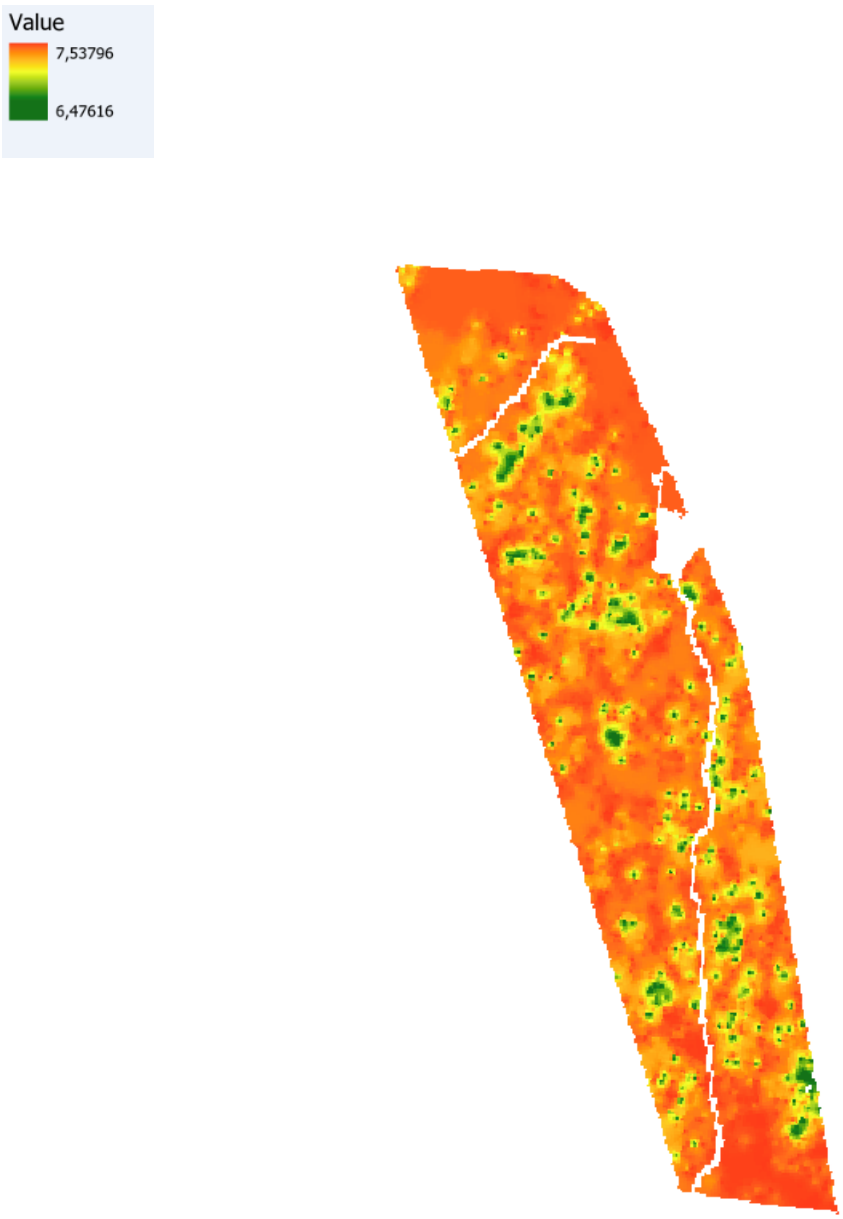


Figure 16: Predicted map layer from Neural network

The results from neural network on the other hand surprisingly showed much more different results compared to the other maps. While KRR and SVR have a range from approximately 1.0 to 8.0, the predicted values from neural network have much more narrower range, marking the majority of the area as unsuitable for site planning. On the other hand, the greener areas and also the least susceptible to land subsidence appears on the same spot as of all the other maps. They do however have much higher values as none of the pixel values the map layer are under 6.0.

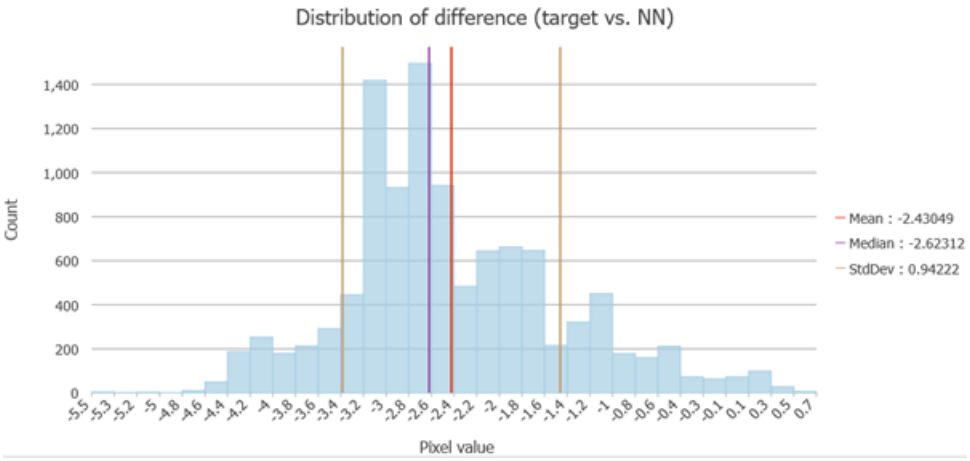


Figure 17: Distribution difference between target and NN

The difference between the target map and NN are significantly higher with a mean difference of -2.4305 and median value of -2.6312, which confirms that the neural network algorithm has a tendency to overpredict data. On the other hand, it should also be noted that less hyperparameters are tuned in the neural network due to the data set being computationally expensive to calculate. Although neural networks usually have an advantage for larger data sets compared to other traditional learning algorithms, it is quite difficult interpret exactly how the map layer turned out different than the other algorithms. It is thus necessary to explore a wider range hyperparameters other than the penalty value and the learning rate.

4.6 Evaluation of machine learning for site planning

Table 8: Optimal error metrics for each ML-algorithm

R2-score		
Neural network	**Kernel Ridge regression**	**Support vector regressor**
0.4914	0.4831	0.5105

Throughout the analysis of predicting suitable places for site planning, the R2-score remained relatively adequate within the range $0.45 - 0.51$ even after tuning and optimizing the hyperparameters for each machine learning algorithm with SVR achieving the highest R2-score. The differences are however quite small, and each kernel used for both SVR and KRR has the potential to generate optimal values as long as $\gamma \geq 10^{-2}$. Even the linear kernel can be utilized if choosing appropriate ϵ-values. This is most likely due to the complexity of the original map/target value since it is a unitless scale based on multiple criteria where each criterion were weighted based on subjective judgment. The goal of the project was to check

how well machine learning algorithms could replicate the original map based on the same 7 criteria and maintain the "shape". Both Kernel Ridge Regression and Support Vector Regressor managed to generate acceptable maps where they managed to predict the least suited areas for site planning due to the risk of ground subsidence. Maps created from those machine learning algorithm agreed with the original map layer that the least suitable areas for infrastructural projects are placed nearby major rivers and lakes and the central city area of Fredrikstad (the southernmost part of the map). Ground subsidence is in fact prevalent in urban areas as groundwater extraction in connection with urbanization and population growth is the main cause of severe land subsidence in central city areas.

On the other hand, the resulting map from neural network gave much more different results where most of the area are labeled as high risk areas for ground subsidence. Neural networks gave much higher values than KRR and SVR, but somehow managed to predict areas that have the least risk of land subsidence as seen from the green spots in figure 16 even if the predicted values for those areas are still high. Within the GIS-environment it still unclear what kind of neural network architecture and training algorithm should be used for a particular application (Sui, 1994). The selection of a particular type of network and the configuration of the neural network architecture are determined by neural network users in an ad hoc manner. Besides these issues, the local minima problem, the possible network paralysis, and unpredictable convergence rate might also have threatened the successful application of neural networks.

5 Conclusion

To summarize, three susceptibility maps were created by the methods of Neural networks, kernel ridge regression and support vector regressor. The purpose was to check how well machine learning algorithms could reproduce the original map layer based on seven criteria that are a factor for causing land subsidence. Although the R2-scores remained in the middle, KRR and SVR managed to maintain the shape of the original map to some extent, pointing out urbanized areas and locations nearby major river streams as the most susceptible areas. SVR gave the highest R2-score, and also ran slightly faster than KRR, although insignificant to make any major impact. SVR do however scale larger training sets better than KRR. Three kernels were used to find an optimal R2-score and MSE, namely the Gaussian RBF, polynomial and the linear kernel. All of them had the potential to generate acceptable R2-scores, making it difficult to decide which kernel to use. The other hyperparameters do however have a preferred value. It turns out that the most optimal γ-values for the RBF-kernel should be equal or larger than 0.001, but less than 1.0. The polynomial kernel on the other hand appears to favor either the 3rd or the 4th degree while generating much worse results otherwise.

The neural network on the other hand also gave similar R2-score as KRR and SVR as long as the learning rate $\eta > 10^{-5}$ and a regularization parameter $\lambda \in [10^{-5}, 10^{-2}]$. Any lower

learning rate will generate worse R2-scores regardless of the λ-parameter chosen. The shape of the map were much more different than the other two maps, marking most of the area red and unsuitable for site planning where all values exceed 6.0 and the runtime appears to be longer than KRR or SVR. It did however also predict the location of the areas with least risk of land subsidence. The anomaly is also evident from figure 17 where the mean difference between the target and the map produced from neural network was -2.43 which more than 2.0 units lower than for KRR and SVR. However, only a limited number of hyperparameters were tuned for the neural network, and would need more adjustments before we conclude the performance of neural network. The use of machine learning for site planning, infrastructure and engineering projects has the potential to make a more objective suggestion of suitable areas than the expert's judgment alone which are prone to uncertainty, but based on how the results differ from method to method, it would be more beneficial to use machine learning as a compliment or guidance to other decision-making processes rather than using it alone to evaluate safe places to build new infrastructure.

6 Further improvements

There were some issues with runtime when applying the neural network algorithm. Ideally, more hyperparameters should be tuned in the project. Additionally, it would be interesting to use another map layer as a target value for predicting land subsidence. Radar measurements from satellites have recently been used in more infrastructural projects such as the area of Bjørvika, Oslo, and would perhaps be easier for the machine learning algorithm to predict than a map layer generated from a unitless decision-making process as it yields a unit of sunsidence rate in mm/year. An individual regression analysis of each criterion and the target value would also be beneficial to investigate in order to check the correlation, and the result could eventually indicate if some criteria/features are insignificant to predict land subsidence. More features were also considered to be added such as rainfall intensity, permeability and hydraulic conductivity, but due to the lack of such data in the study area, such map layers could not be produced.

References

Bagheri, M., Dehghani, M., Esmaeily, A. and Akbari, V. (2019). Assessment of land subsidence using interferometric synthetic aperture radar time series analysis and artificial neural network in a geospatial information system: case study of rafsanjan plain, *Journal of Applied Remote Sensing* **13**: 1.

BaneNor (2018). Fakta om prosjektet.
URL: *https://www.banenor.no/Prosjekter/prosjekter/ostfoldbanen2/haug-seut/fakta-om-prosjektet/*

Géron, A. (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 1st edn, O'Reilly Media, Inc.

Hastie, T., Tibshirani, R. and Friedman, J. (2001). The elements of statistical learning, *Aug, Springer* **1**.

Karlsson, C., Kalantari, Z., Mörtberg, U., Olofsson, B. and Lyon, S. (2017). Natural hazard susceptibility assessment for road planning using spatial multi-criteria analysis, *Environmental Management* **60**: 823–851.

Lee, S., Ryu, J.-H. and Park, H.-J. (2004). Determination and application of the weights for landslide susceptibility mapping using an artificial neural network, *Engineering Geology* **71**(3): 289 – 302.
URL: *http://www.sciencedirect.com/science/article/pii/S001379520300142X*

Liu, Y. (2020). *Python Machine Learning By Example: Build intelligent systems using Python, TensorFlow 2, PyTorch, and scikit-learn*, 3rd edn, O'Reilly Media, Inc.

Mattera, D. and Haykin, S. (1999). Support vector machines for dynamic reconstruction of a chaotic system, *Advances in kernel methods: support vector learning* pp. 211 – 241.

Payabyab, R. M. B. (2020). Using logistic regression and neural networks for forecasting crude oil prices and detecting depression among the rural population in siyaya county, kenya.

Scheibz, J. (2020). Private communication.

Shahabi, H., Jarihani, B. A., Tavakkoli Piralilou, S., Chittleborough, D., Avand, M. and Ghorbanzadeh, O. (2019). A semi-automated object-based gully networks detection using different machine learning models: A case study of bowen catchment, queensland, australia, *Sensors* **19**: 4893.

Sui, D. (1994). Integrating neural networks with gis for spatial decision making, *Operational Geographer* .

Vegvesen, S. (2016). Erfaringsrapport bjørvika - bygging av gateanlegg i oslo bysentrum, *Technical Report 525*, Statens Vegvesen, region øst.

Yang, Y. (2015). Hotel location evaluation: A combination of machine learning tools and web gis, *International Journal of Hospitality Management* **47**.