ENGINEERING MATHEMATICS
AND COMPUTING LAB

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Comparison of Machine Learning Methods for Predicting Employee Absences

Alejandra Jayme, Philipp D. Lösel, Joachim Fischer, Vincent Heuveline

Preprint No. 2021-02

Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)

www.emcl.iwr.uni-heidelberg.de

Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)
ISSN 2191–0693
Preprint No. 2021-02

Affiliation of the Authors

Jayme, A.[a], Lösel, P. D.[a], Fischer, J.[b], Heuveline, V.[a]

[a] *Engineering Mathematics and Computing Lab (EMCL), Interdisciplinary Center for Scientific Computing (IWR), Heidelberg University, Germany*
[b] *Mannheim Institute of Public Health, Social and Preventive Medicine, Universitätsmedizin Mannheim, Germany*

Impressum

Heidelberg University
Interdisciplinary Center for Scientific Computing (IWR)
Engineering Mathematics and Computing Lab (EMCL)

Im Neuenheimer Feld 205,
69120 Heidelberg
Germany

Comparison of Machine Learning Methods for Predicting Employee Absences

Alejandra Jayme, Philipp D. Lösel, Joachim Fischer, Vincent Heuveline

April 30, 2021

Abstract

Employee absences cannot be avoided but excessive and uncontrolled absences affect not only the companies and employees but also impact the economy, government and society. Though actual losses are hard to compute, absenteeism has been estimated to cost billions in direct and indirect costs. Addressing employee absences is difficult because the underlying reasons and causes are complex and not straightforward. Compounding this, companies do not have tools to analyze and predict the future risk of employee absences, relying instead on retrospective data that may not be relevant to the current situation at hand. In this study, we show how machine learning methods can be used to predict employee absence risks. Results show that Neural Networks give best accuracy (77%) over Random Forest (72%) and Support Vector Machines (62%). The effect of training data size and varied feature sets on the models' performances were also tested. Also, a method allowing for ranking the sensitivity of a Neural Network to each feature is presented.

1 Introduction

Employee absenteeism is defined as an employee's incapacity to work, with or without good reason. The range of direct and indirect costs of absenteeism is enormous. For example, according to the annual report by the Federal Ministry of Labor and Social Affairs (Bundesministerium für Arbeit und Soziales, BMAS) and the Federal Institute for Occupational Safety and Health (Bundesanstalt für Arbeitssicherheit und Arbeitsmedizin, BAUA) in 2007 the lost of employment was 1.2 million years [1]. This translates to €40 million in loss of productivity by labor costs and €73 billion loss in gross value added, 1.6% and 3% of the gross national income respectively. These numbers were derived from the inability to work certificates of all compulsory and voluntary members of only 5 federal associations of statutory health insurers. And not included here are incurred costs by the industry, loss of potential output, medical treatment costs, administration costs incurred by the firms and insurance companies.

Workplace absenteeism affects not only the employers and employees but also the insurance companies, the government and society as a whole. The costs are shouldered by all the stakeholders, albeit unequally. The employee often has reduced income and could have additional expenses like health care services. Moreover, frequent or long-term absenteeism can cause loss of jobs or strained relationships with colleagues. Employers, on the other hand, incur costs from sick pay, lost productivity, scheduling, replacement personnel and inferior service or product quality. This has a negative effect on the company's competitive position. The government is affected by the increased costs of medical treatment and social security. Furthermore, the reduction in labor force has a negative effect on the national economy. For the society, it is important that the people are healthy and can contribute to the gross national product up to retirement age.

It is clear that even a small reduction in workplace absences is beneficial to all parties involved and could save a lot of money. But statistics and research on the causes and levels of absenteeism are lacking. At the very least, it is widely known that sickness is a major reason for absences from work. However,

absenteeism is a complex phenomenon that is influenced by many other factors than health. The decision to attend or resume work are related to the real and perceived conditions of work, anticipated job demands, management attitude, social norms in the peer group and community, economic pressures, job satisfaction, motivation, personal beliefs, behavior and life events.

Traditionally, the attempts undertaken by companies to combat absenteeism are of 3 types: procedural measures aimed at monitoring and controlling absenteeism, preventive work-oriented measures aimed at reducing the work-related causes in the area of safety, health and well-being and preventive person-oriented measures aimed at supporting employees to work and live in a safe and healthy way [6]. To analyze and implement such measures, companies collect medical, health insurance and administrative data and conduct employee surveys and interviews.

The key characteristics of successful measures against absenteeism include a systematic approach, assessment based on needs, focused on active worker participation and regular evaluation [6]. In general, forecasting the development of work ability and related risks and simulating the effects of measures would be helpful tools in implementing and designing such measures.

In this regard, artificial intelligence (AI) can play a significant role. AI is the concept of building machines that mimic human cognitive functions like knowledge representation, planning, learning, decision making and problem solving. A subset of AI is Machine Learning (ML), which has computers learning automatically from data. Machine learning algorithms build a model based on sample (training) data to make predictions or decisions without being explicitly told how to do so. Most of the applications of machine learning fall under the following functions: computer vision, text and speech recognition, natural language processing, sentiment analysis, predictive analysis and learning associations.

Machine learning can also be used in mitigating workplace absenteeism. The employee data collected by companies can be used as training data to predict the absenteeism risk level of individual workers. Analysis can be further made into which and how variables affect this risk. Moreover, machine learning models can be used as a simulation tools to quantify how planned measures may affect absenteeism.

In this study, we compare three popular machine learning models for predicting employee absence risk: Random Forest, Support Vector Machine and Artificial Neural Networks. Two datasets provided by the occupational health consultant HealthVision GmbH, headed by Prof. Dr. med. Fischer, are used to train the models. These datasets contain information such as employees' health and behavioral history, demographic and work structure data and results of life and job satisfaction surveys. In almost all test cases, we find that Neural Networks have the best accuracy while Random Forests gave consistent results and Support Vector Machines always had the worst performance.

2 Methodology

2.1 Learning Models

In this section we describe the machine learning models considered in this paper in order to predict absence risk. We compare three models: Random Forest (RF), Support Vector Machine (SVM) and Artificial Neural Networks (NN). We focus this study on supervised learning for classification problems [5], where after learning from labelled employee data the model specifies into which category (e.g. low or high absence rate) previously unseen employee belongs to.

2.1.1 Random Forest

Random Forests (RF) [3] are made up by decision trees or estimators that operate as an ensemble (see Figure 1). In classification problems, each tree gives a class prediction and the class with the most votes becomes the model's final prediction. How random forests operate is simple and intuitive; it is based on the concept that a large number of models working as a group will outperform any individual model's performance. The important key here is low correlation among the trees and also low variance in their individual predictions. To ensure this, two techniques are employed: *bootstrap aggregation* and *feature randomness*[4].

Bootstrap aggregation or bagging is the method of training each tree on a sample of the data set with replacement. So each tree trains on a different version of the data set resulting in different trees.

Moreover, feature randomness means that the trees are trained on only a random subset of features, further diversifying the trees. Combining diverse trees cancels out some errors yielding a better model.

How the nodes and splits are decided in constructing the tree depends on a selection criteria. The most common measures are *entropy* and *information gain*. Entropy is a measure of the randomness in the information being processed. For a random variable X, it is given by

$$E(X) = -\sum_{i=1}^{n} P(x_i) \log P(x_i) \tag{1}$$

where $x_1, \ldots, x_n$ are the possible outcomes that occur with probabilities $P(x_i), \ldots, P(x_n)$. Information Gain measures how well a feature separates training samples according to their target classification. It is calculated by calculating the entropy of the data set before and after a transformation.

Using either of this measure, the steps how to build a decision tree is as follows:

1. Start at the root node with the complete training data sample and feature subset.
2. With each iteration of the algorithm, calculate the entropy or information gain for each remaining feature (and corresponding feature values).
3. Select the feature with the smallest entropy or highest information gain.
4. Split the training sample according to the selected feature to produce a subset of the sample.
5. Repeat from step 2 on each training sample subset and remaining features until all features have been selected.

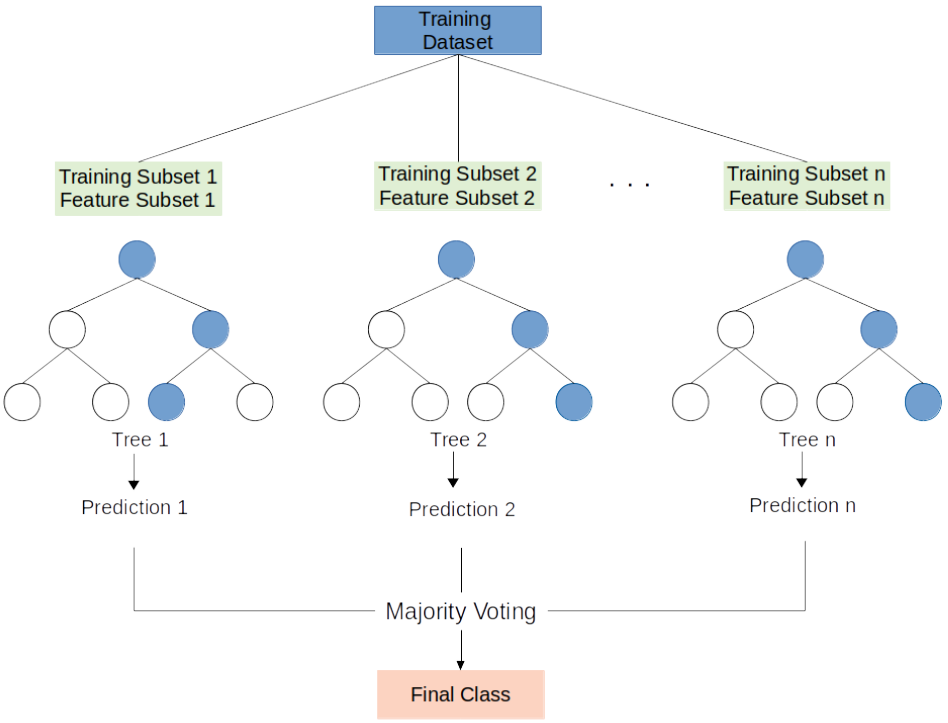

Figure 1: Random Forest

2.1.2 Support Vector Machine

Support Vector Machines (SVM) [2] is based on finding a hyperplane or a set of hyperplanes that divides a data set into desired classes (see Figure 2). The data points closest to the hyperplane are called *support vectors*. Intuitively, the farther away the support vectors are from the hyperplanes (i.e. the margin) while still being in the correct side, the more confident is the classification. Therefore, the goal of SVM is to find the hyperplane that separates these critical points with maximal margin.

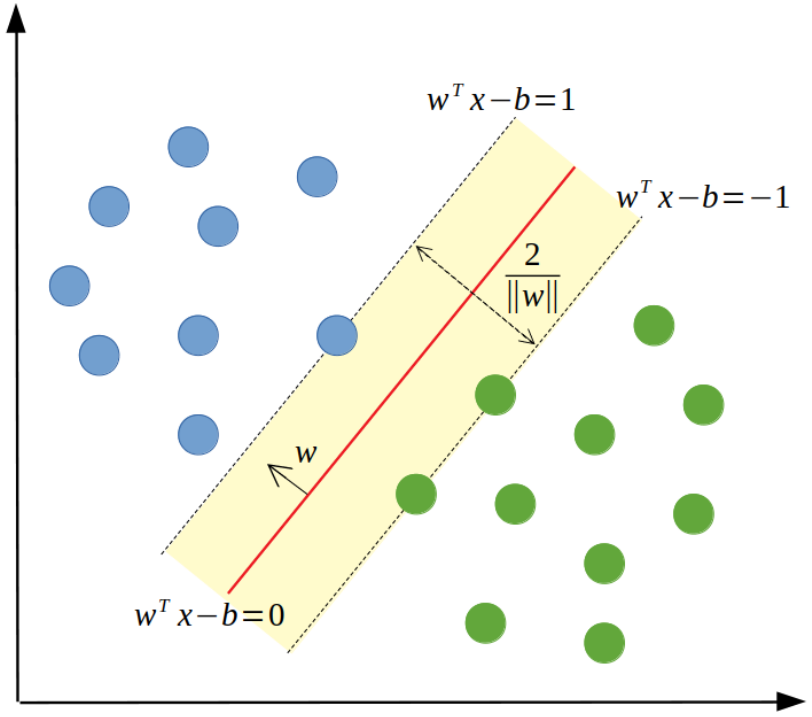

Figure 2: Support Vector Machine

Consider a training dataset of n points with the form $(\boldsymbol{x_1}, y_1), \cdot, (\boldsymbol{x_n}, y_n)$ and $y_i = 1$ or $y_i = -1$. If this dataset is linearly separable, two parallel hyperplanes that separate the two classes of data can be selected such that the distance between them is as large as possible. The maximum-margin hyperplane is the hyperplane that lies halfway between them. With a normalized dataset, these hyperplanes can be described with

$$\boldsymbol{w^T x} - b = 1 \quad \text{(anything on or above this hyperplane has class label = 1)} \tag{2}$$

$$\boldsymbol{w^T x} - b = -1 \quad \text{(anything on or below this hyperplane has class label = -1)} \tag{3}$$

where $\boldsymbol{w}$ is a normal vector to the hyperplane and b is some constant.

The distance between these two hyperplanes is $\frac{2}{\|\boldsymbol{w}\|}$. So to maximize the distance, $\|\boldsymbol{w}\|$ must be minimized. To ensure the datapoints are classified correctly, the following constraint is added:

$$\boldsymbol{w^T x}_i - b > 1 \quad \text{if} \quad y_i = 1 \quad \text{or} \tag{4}$$

$$\boldsymbol{w^T x}_i - b < -1 \quad \text{if} \quad y_i = -1 \tag{5}$$

which can be written as

$$y_i(\boldsymbol{w^T x}_i - b) \geq 1 \quad \text{for all} \quad 1 \leq i \leq n \tag{6}$$

However in most finite-dimensional space problems, the classes are not linearly separable. In this case, adding the *hinge loss* (i.e. $l(y) = \max(0, 1 - t \cdot y)$ for intended output $t = \pm 1$ and classifier score y) is helpful and the optimization goal becomes to minimize

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max(0, 1 - y_i(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_i - b))\right] + \lambda \|\boldsymbol{w}\|^2 \tag{7}$$

where the parameter λ determines the trade-off between increasing the margin size and ensuring that x_i lie on the correct side of the margin.

The *kernel trick* may be applied to create nonlinear classifiers. In this method every dot product is replaced by a nonlinear kernel function, implicitly mapping the input space into a transformed feature space where the maximum-margin hyperplanes are fit. The more common kernel functions include polynomial, hyperbolic tangent and Gaussian radial basis function.

2.1.3 Artificial Neural Network - Multilayer Perceptron

An artificial neural network (ANN) [2] is a computational model based loosely on the biological brain's network of interconnected neurons. It is composed of connected processing units called *nodes* that receive input x_k from other nodes and compute new output y_k for other nodes. Each node has an *activation function* $f(\cdot)$ applied on the node's input. *Weights* w_{km} and *biases* b_k are assigned to each node; they are adjusted as the network attempts to produce favorable out following some *learning rule*.

There are many types of ANNs which depends on properties, such as the topology of the network, the learning algorithm being followed and the activation function being used. In this study, we use a Multilayer Perceptron (MLP). It is a simple, static feedforward ANN, consisting of an input layer, one or mode hidden layers and an output layer (see Figure 3). The information passes through the network in one direction: from input layer to hidden layers to output layer. The nodes have nonlinear activation functions and learning is achieved by *backpropagation*.

The goal of the MLP is to produce desired outputs given the inputs by finding the weights and biases that give optimal results. This can be done by feeding the inputs to the network, teaching it patterns and letting it adjusts its weights and biases according to some rule. To quantify how well the network output approximates they expected output, an error or cost function is defined that measures the difference between the outputs.

Here we consider a *classification* problem, where the model predicts a class or label (e.g. low, medium or high absence risk) for a given example of input data (e.g. employee data). In an MLP implemented for such a problem, the output layer is composed of M nodes, one for each possible class (e.g. $M = 3$ for low, medium, high), giving a probability that the input point belongs to that class. The class with the highest probability is chosen as the output predicted class. A common error measure for this multi-class classification problem is the *cross-entropy loss or log-loss*

$$E(w) := -\frac{1}{n}\sum_{n}\sum_{c=1}^{M} y_{x_n,c}\ln(p_{x_n,c}), \tag{8}$$

where n is the number of datapoints, w are the weights and biases of the network, M is the number of classes, y is a binary indicator (0 or 1) if class c is the correct classification for input datapoint x_n and p is the predicted probability x is of class c.

Training the network means allowing the network to find the weights and biases w so that the error function $E(w)$ is minimized. A way to find the minimum of such a function is a method called *gradient descent*, where proportional steps are taken to the negative of the gradient of the function at the current point. Given that $E(w)$ is defined and differentiable in the neighborhood of point w, then $E(w)$ decreases faster if it goes from w to the direction of the negative gradient of E at w, $-\nabla E(w)$. That is, the parameters are adjusted such that

$$w_{n+1} = w_n - \alpha \nabla E(w_n) \tag{9}$$

For small enough α, $E(w_n) \geq E(w_{n+1})$. The positive parameter α is called the *learning rate*. In this manner, the error is propagated back to each node with an associated value that reflects its contributions to the original output.

There are three types of gradient descent: stochastic, batch and mini-batch. *Stochastic gradient* descent updates the model after evaluating each sample in the training dataset. The frequent up dates gives immediate insight into the performance and rate of improvement of the model. The model can also learn faster and can potentially avoid local minima. However, frequently updating the model is computationally expensive and can result to a noisy gradient signal. This means the model may have a harder time settling to an error minimum.

Batch gradient descent goes through the entire training dataset and only updates the model after all training samples have been evaluated. One such cycle is called a *training epoch*. This method is more computationally efficient and can result in a more stable error gradient and convergence. However, the stable error gradient can also result in premature convergence to a less optimal set of parameters. Implementation also commonly requires the entire training dataset in memory and can become slow for very large datasets.

Mini-batch gradient descent seeks a balance between stochastic and batch gradient descents. It splits the training dataset into smaller batches and updates the model after evaluating the error for each smaller batch. This method is more robust than stochastic gradient descent and can be more efficiently implemented than batch gradient descent.

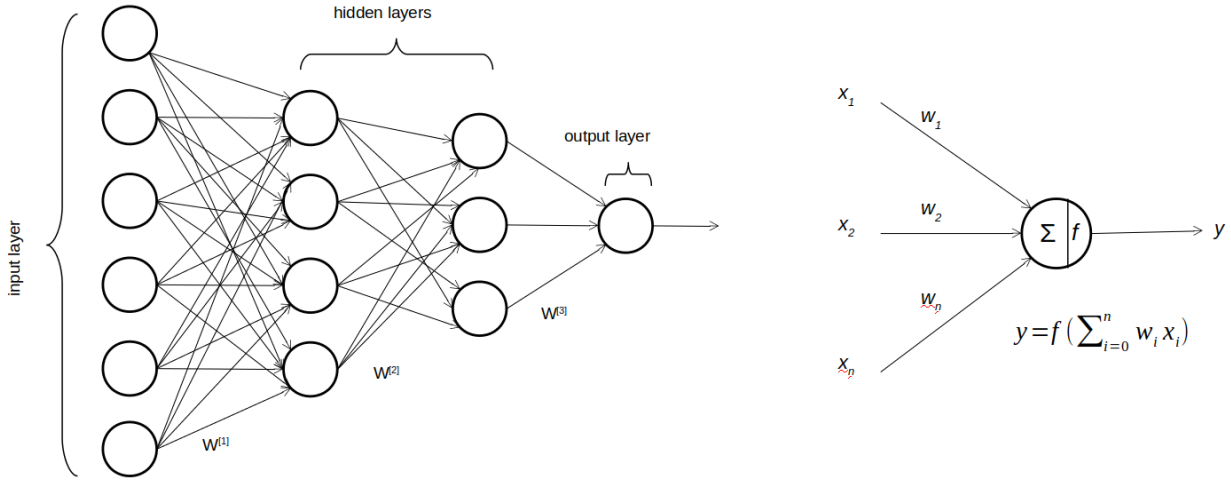

Figure 3: Multilayer Perceptron

2.2 Feature Analysis

In this section we look at two different ways we analyze which features affect the machine learning model. This gives important information for planning data gathering, model changes or even absenteeism measures.

2.2.1 Feature Importance

Feature importance is a *ranking* of features in terms of its usefulness at predicting a target. We use several rankings for comparison:

• *Domain Expert* is a ranking from a person or institute considered an authority in the area of workplace productivity.

• *Random Forest (RF)* is an ensemble learning method that operate by constructing a set of decision trees. Each tree is trained on both randomly selected subset of the data and the features. Each tree

gives a solution and the final solution is the mode of the classes for classification and the mean of the predictions for regression.

During construction (training) of the trees, the features and splits are selected by how much it decreases the *GINI impurity* (Equation 10) after the split. The decrease can be averaged over all features over all trees and is used as a ranking of feature importance.

$$G = 1 - \sum_{i=1}^{C} p^2(i) \tag{10}$$

where C is the number of classes and $p(i)$ is the probability of picking a datapoint with class i.

• *Principal Component Analysis (PCA)* [2] is an orthogonal and linear transformation of data into a new coordinate system such that the greatest variance of the data lies on the first coordinate (called the first principal component), the second greatest variance on the second coordinate and so forth.

Let $\boldsymbol{X}$ be the data matrix of $n \times p$ size, where n is the number of samples and p is the number of features. The transformation is defined by a set of size l (the desired number of principal components) of p-dimensional unit vectors of weights $\boldsymbol{w}$ that map each row of vector $\boldsymbol{X}$ to a new vector of principal component scores given by

$$t_{k(i)} = \boldsymbol{x}_{(i)} \cdot \boldsymbol{w}_{(k)} \quad \text{for} \quad i = 1, \ldots, n \quad k = 1, \ldots, l \tag{11}$$

in such a way that $\boldsymbol{t}$ contains the maximum possible variance from $\boldsymbol{X}$. For this to happen on the first principal component, then

$$\boldsymbol{w}_{(1)} = \underset{\|w\|=1}{\arg\max} \left\{ \sum_i (t_1)^2_{(i)} \right\} = \underset{\|w\|=1}{\arg\max} \left\{ \sum_i \left(\boldsymbol{x}_{(i)} \cdot \boldsymbol{w} \right)^2 \right\} \tag{12}$$

which can be written in matrix form as

$$\boldsymbol{w}_{(1)} = \underset{\|w\|=1}{\arg\max} \left\{ \|\boldsymbol{X}\boldsymbol{w}\|^2 \right\} = \underset{\|w\|=1}{\arg\max} \left\{ \boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{w} \right\} \tag{13}$$

and because $\boldsymbol{w}$ has been constrained to be unit vectors, then also

$$\boldsymbol{w}_{(1)} = \arg\max \left\{ \frac{\boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{w}} \right\}. \tag{14}$$

For the positive semidefinite matrix $\boldsymbol{X}^T\boldsymbol{X}$, the quotient's (in Equation 14) maximum possible value is the largest eigenvalue of the matrix which occurs when $\boldsymbol{w}$ is the corresponding eigenvector.

The subsequent kth principal components can be determined by subtracting the first $k-1$ principal components from $\boldsymbol{X}$

$$\hat{\boldsymbol{X}}_k = \boldsymbol{X} - \sum_{s=1}^{k-1} \boldsymbol{X} \boldsymbol{w}_{(s)} \boldsymbol{w}_{(s)}^T \tag{15}$$

and then

$$\boldsymbol{w}_{(k)} = \underset{\|w\|=1}{\arg\max} \left\{ \|\hat{\boldsymbol{X}}_k \boldsymbol{w}\|^2 \right\} = \arg\max \left\{ \frac{\boldsymbol{w}^T \hat{\boldsymbol{X}}_k^T \hat{\boldsymbol{X}}_k \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{w}} \right\} \tag{16}$$

The full transformation can then be written as

$$\boldsymbol{T} = \boldsymbol{X}\boldsymbol{W} \tag{17}$$

where $\boldsymbol{W}$ is a $p \times p$ matrix of weights whose columns are the eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$ and are exactly the principal components of the data $\boldsymbol{X}$.

$\boldsymbol{X}^T\boldsymbol{X}$ itself gives the *covariance matrix* of $\boldsymbol{X}$

$$C = \frac{1}{n-1} \boldsymbol{X}^T \boldsymbol{X} \tag{18}$$

which is a symmetric matrix that can be diagonalized to

$$C = \boldsymbol{V}\boldsymbol{L}\boldsymbol{V}^T \tag{19}$$

where $\boldsymbol{V}$ is a matrix of eigenvectors and $\boldsymbol{L}$ is a diagonal matrix of eigenvalues λ_i in decreasing order on the diagonal.

Going back to the data matrix $\boldsymbol{X}$, the matrix factorization technique *Singular Value Decomposition* (SVD) gives

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T \tag{20}$$

where $\boldsymbol{U}$, called left singular vector, is an $n \times n$ unitary matrix, $\boldsymbol{S}$ is the $n \times p$ diagonal matrix of positive numbers s_i, called singular values and $\boldsymbol{Z}$, called right singular vector, is a $p \times p$ unitary matrix.

Substituting this in Equation 18, we get

$$C = \frac{1}{n-1}\boldsymbol{V}\boldsymbol{S}\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T = \boldsymbol{V}\frac{\boldsymbol{S}^2}{n-1}\boldsymbol{V}^T. \tag{21}$$

Comparing this to Equation 19, we see that the right singular vector $\boldsymbol{V}$ is also the eigenvectors of the covariance matrix C and therefore the principal components of $\boldsymbol{X}$. Furthermore, the singular values are related to the eigenvalues of the covariance matrix, $\lambda_i = s_i^2/(n-1)$.

Using the first principal component, the importance of the p features can be obtained from the magnitude of the corresponding values in the eigenvector. The larger the absolute values of the eigenvector's components the more the feature contributes to the principal component.

• Also included in the ranking of features are *random rankings*. These are random rankings of randomly chosen features. They are used to show whether the rankings listed above are significantly better than random.

2.2.2 Feature Sensitivity

Feature sensitivity is a ranking of the impact of each feature has on the model's predictions.

We demonstrate for a one-hidden layer MLP, with weights $\boldsymbol{W}$ and biases $\boldsymbol{b}$. The following equations describe the forward propagation of data through the model:

$$\boldsymbol{x} = \text{input} \tag{22}$$
$$\boldsymbol{z_1} = \boldsymbol{W_1}\boldsymbol{x} + \boldsymbol{b_1} \tag{23}$$
$$\boldsymbol{h_1} = f(\boldsymbol{z_1}) \tag{24}$$
$$\boldsymbol{z_2} = \boldsymbol{W_2}\boldsymbol{h_1} + \boldsymbol{b_2} \tag{25}$$
$$\hat{\boldsymbol{y}} = softmax(\boldsymbol{z_2}) \tag{26}$$
$$J = CE(\boldsymbol{y}, \hat{\boldsymbol{y}}) \tag{27}$$

where f is the chosen nonlinear activation function of the nodes in the hidden layers and $softmax$ is the activation function of the nodes in the output layer. The softmax function squashes the input to the output layer into a vector with values in the range $(0, 1)$ that add up to 1, which are then interpreted as class probabilities. J is the cross entropy loss in Equation 8.

The sensitivity s_i of the model to a feature i is the derivative of the error function with respect to the input of the ith neuron in the input layer.

$$s_i = \frac{\partial J}{\partial x_i}(\boldsymbol{x}) \tag{28}$$

This can be calculated by applying the Chain Rule to the partial derivates of the inner layers:

$$\frac{\partial J}{\partial x_i} = \frac{\partial J}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z_2}\frac{\partial z_2}{\partial h_1}\frac{\partial h_1}{\partial z_1}\frac{\partial z_1}{\partial x_i} \tag{29}$$

Calculating s_i for all samples in the test data set, we get an $n \times p$ matrix, where n is the number of samples and p is the number of features. Then the sensitivity ranking can be obtained with the SVD technique described above.

3 Results

Two datasets provided by HealthVision GmbH were used in the experiments conducted for this study. These datasets include features such as employees' medical measurements, demographic data, work-structure data, self-reported life- and work characteristics and health- and behavioral history. The first dataset (DS1) has 3307 samples and 58 features. The output (i.e. number of days absences) was categorized into 5 classes (i.e. 0, up to 2, up to 5, up to 14, more than 14 days absences per year). The second dataset (DS2) has 10,013 samples and 48 features with 5 output classes (i.e. up to 5, up to 14, up to 28, up to 42, more than 42 days absences per year). Both datasets have imbalanced output classes; to mitigate this, Synthetic Minority Oversampling Technique (SMOTE) was used to balance the number of samples per class. Standardization was also used to rescale the data. The final datasets have 6,285 and 13,250 samples respectively.

Three machine learning models are compared: Random Forest (RF), Support Vector Machine (SVM) and Artificial Neural Network (NN). Each model has several hyperparameters to be tuned for optimized performance. However, in this study minimal calibration was done: for RF only the number of estimators, for SVM only the kernel and for NN the activation function, solver and hidden layer size.

Because the classes are balanced, it is sufficient to evaluate the models using accuracy as the metric. Furthermore, stratified k-folds cross-validation was also used for a more generalized assessment of the model performance. Model training times are also given.

In the first section, we look into the general accuracy and training times of the basic (i.e. small) configuration of the models. In the succeeding sections, we look at specific aspects such as the size of training data, the size of the model, the number of features and feature importance.

3.1 General Accuracy and Training Times

Here we compare the performances of the model with very minimal parameter tuning. For Dataset 1, the configurations of the three models are as follows: 100 estimators for Random Forest (RF), 3 layers of 100 nodes for the Neural Network (NN) and radial basis function (RBF) kernel for the Support Vector Machine (SVM). Dataset 2 is a bigger dataset, so the model sizes are increased: 200 estimators for RF and 5 layers of 100 nodes for the NN.

RF and NN gave almost similar accuracies for both datasets, with RF performing slightly better for Dataset 2 (69% vs 67%). SVM performed worst at 60% and 61%. In all the test conducted, SVM always performed worst. Moreover, it is significant to note that NN takes approximately 6 times longer to train than RF. NN always takes longer to train regardless of other conditions.

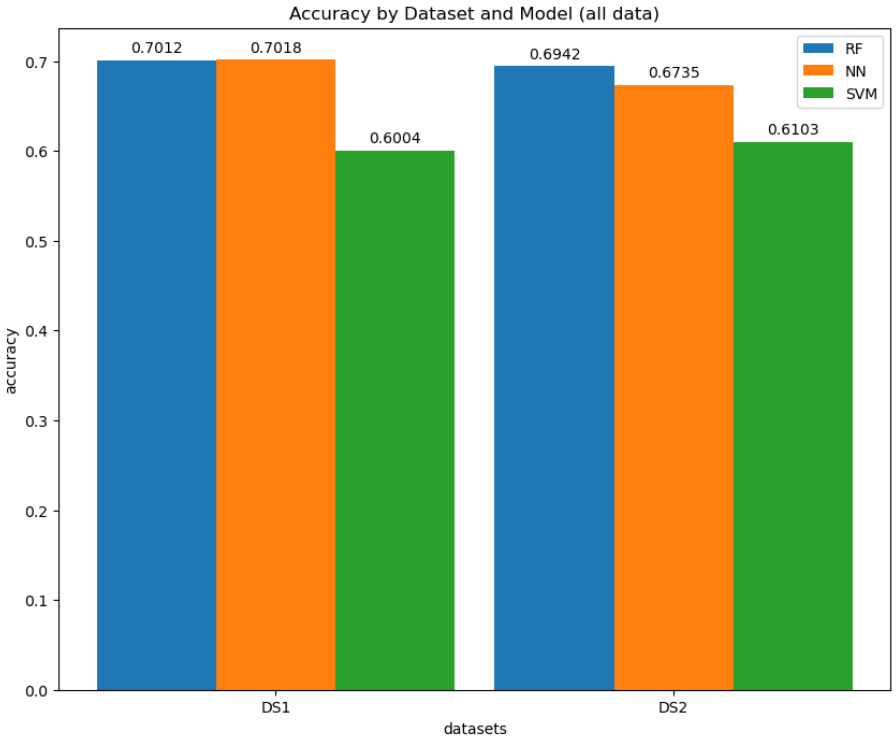

Figure 4: Model Accuracy per Dataset

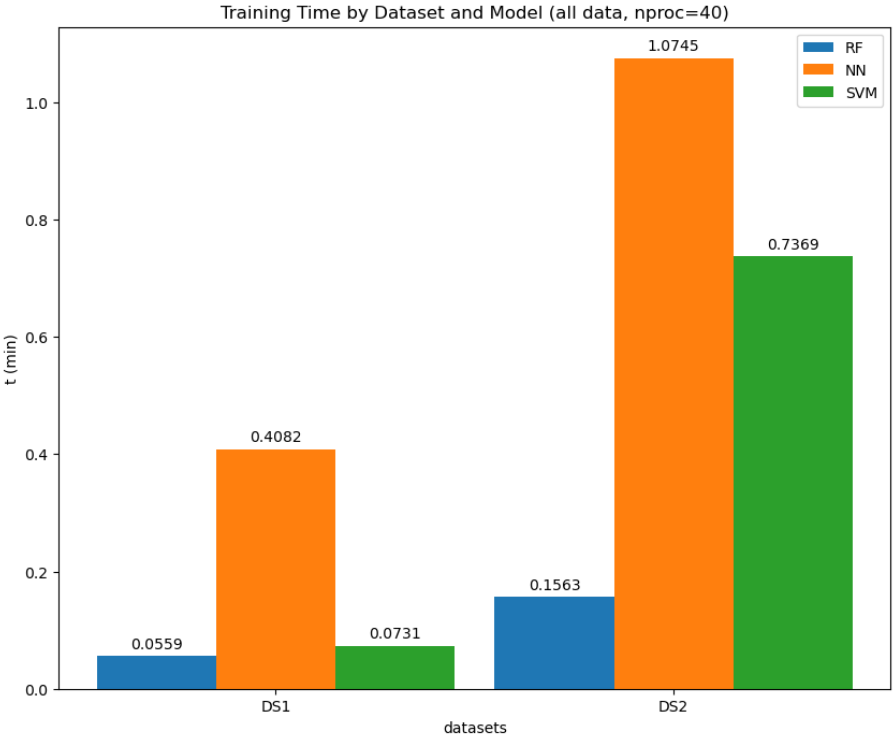

Figure 5: Model Training Times per Dataset

3.2 Accuracy by Model Size

Random Forests and Neural Networks are built up by components, trees or estimators and nodes or neurons respectively. Intuitively, it is expected with more components (ie. bigger model) the better the model's performance.

Figure 6 shows that for Random Forests, the performance trend does increase with the number of estimators but the increase is not simply linear. Furthermore for Dataset 1, from 400 to 5000 estimators the model's accuracy falls within the very narrow range of 71.5% and 72.1%. The smallest model with only 100 estimators already gives 69.7% accuracy. The same behavior is observed for Dataset 2.

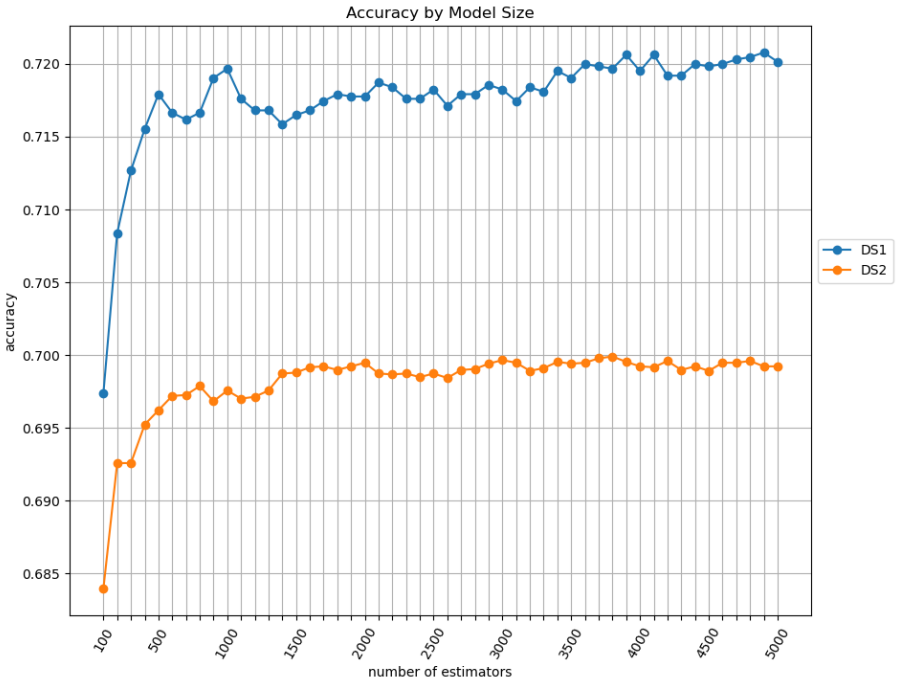

Figure 6: Random Forest Model Accuracy by Number of Estimators

For Neural Networks, the accuracy range for varying model size is wider, 68.7% - 76.8% for Dataset 1 (excluding nlayers = 15) and 66.6% - 74.2% for Dataset 2 (Figure 7 and Figure 8 respectively). This shows that choosing the appropriate model size is important for evaluating the model's performance. It also shows that it is possible for the model size to be too big to be effective, e.g. 15 layers for Dataset 1 (see Figure 7). Furthermore, training times for Neural Networks also become significantly longer for bigger models (see Figure 9). The trade off between the accuracy gain with bigger models should be considered.

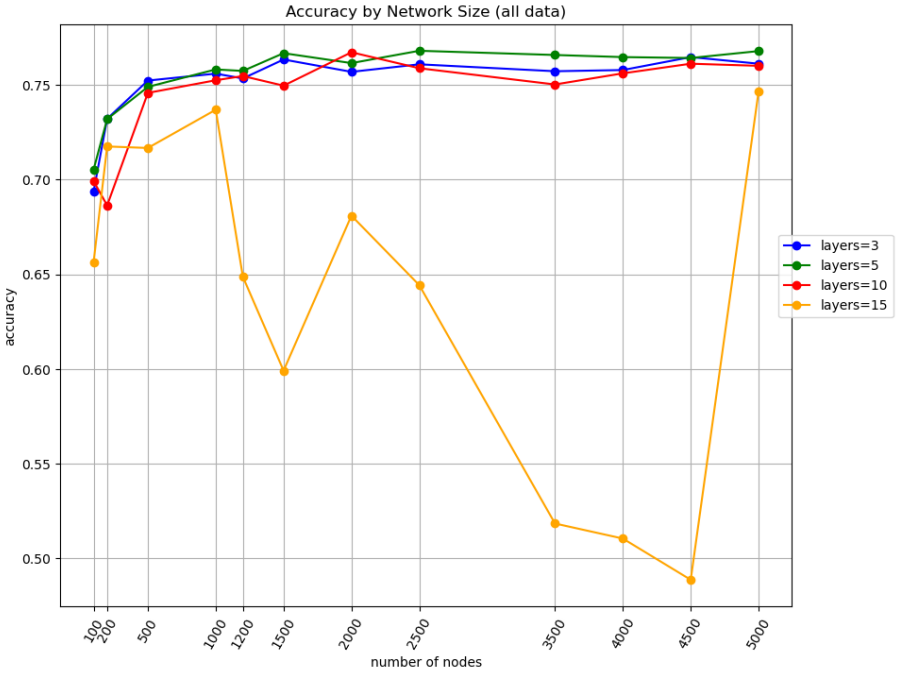

Figure 7: Neural Network Model Accuracy by Network Size (DS1)

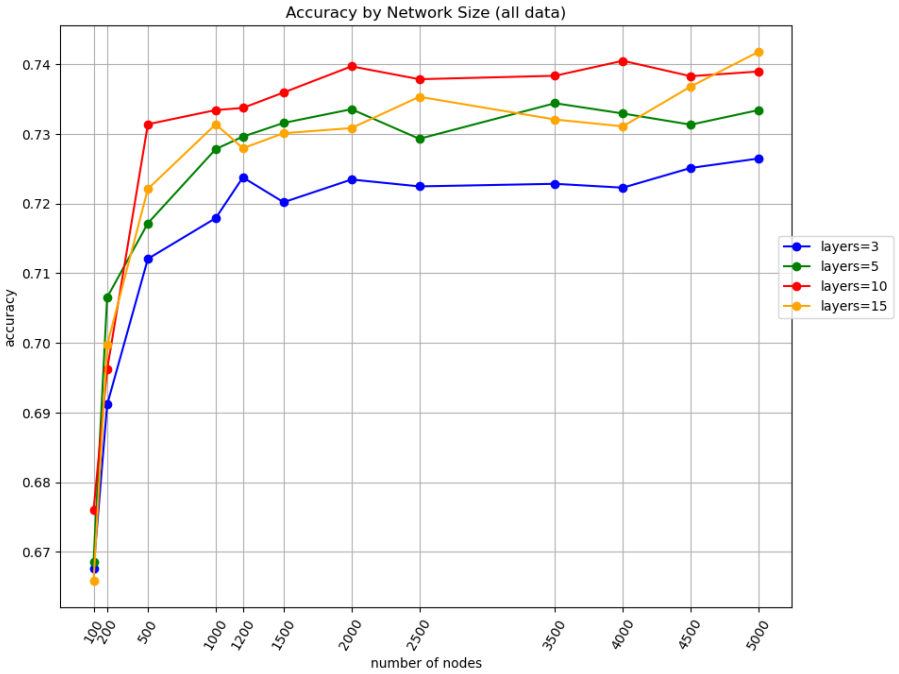

Figure 8: Neural Network Model Accuracy by Network Size (DS2)

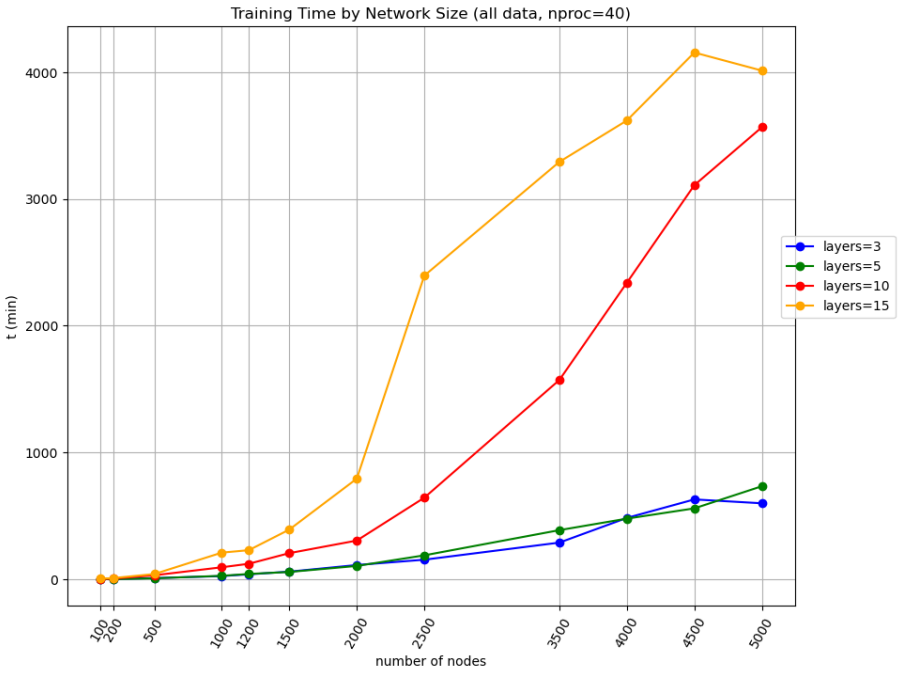

Figure 9: Neural Network Model Training Time by Network Size (DS2)

3.3 Accuracy by Training Data Size

An equally important factor in machine learning models is the training data size, in terms of the number of samples and the number of features. In general, most machine learning models benefit with more training data but RF work more reliably with less data compared to NN and SVM. We see this in Figures 10 and 11 where RF performs better up to about half the original dataset. We keep the class balance with each subset of the dataset. The graphs are also annotated to show the sizes of the model that gave these best accuracies.

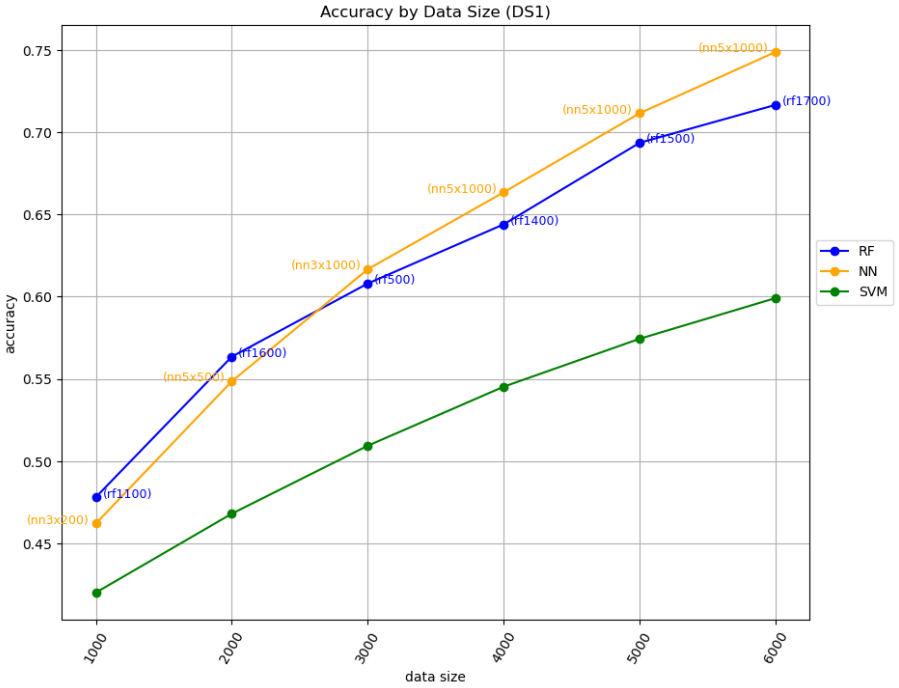

Figure 10: Model Accuracy vs Training Data Size (DS1)

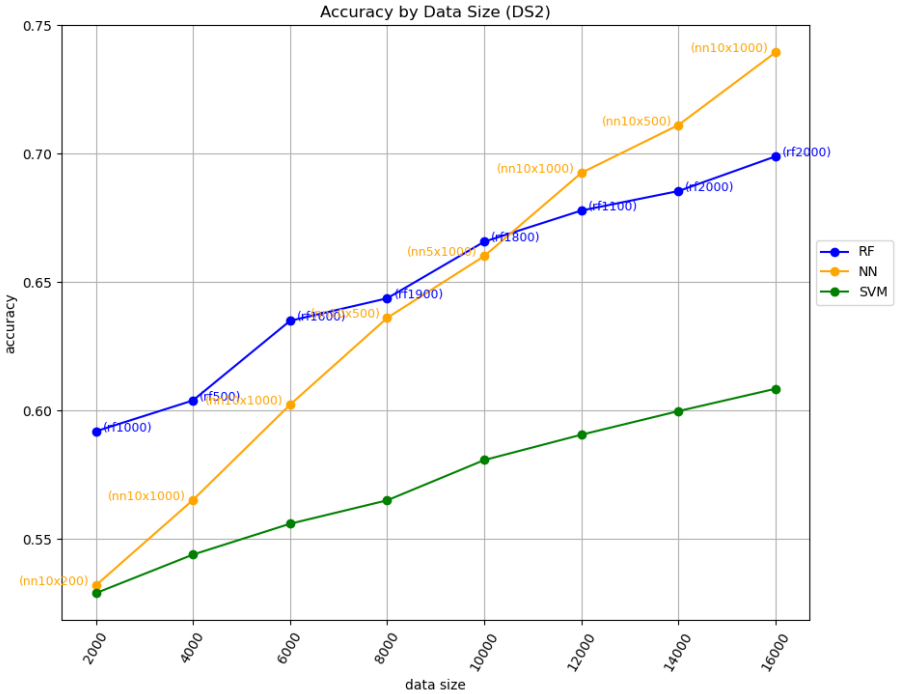

Figure 11: Model Accuracy vs Training Data size (DS2)

3.4 Accuracy by Number of Training Features

The number of features in the dataset also affect the training and performance of the models. But many times it is not possible or practical to acquire data over a lot of features and oftentimes different features carry the same information anyway. So it is prudent to perform feature selection before training the models.

Here we test the effect of the number of training features and also the quality of the feature set. We use two ways to reduce the number of features: by correlation and by ranking.

Correlation is a measure of the association between two variables. This can be positive (when the values of one variable increase so does the other), negative (when the values decrease together) or neutral (the variables are not associated at all). Because no assumption on the distributions of the variable values are made, we use Spearman's rank correlation. Rank correlation quantifies the association between variables using the ordinal relationship between values instead of specific values. Here ρ specifies the degree of association with significance of $\alpha = 0.05$. A feature is randomly chosen from the set of correlated features to represent that set in the reduced training feature set.

For Dataset 1 and NN model (see Figure 12), we see that training with 58 features vs 35 features does not result in a significant increase in accuracy (75.8% vs 74.8%). Training with 44 features (75.3%) gives almost the same performance. For RF model, training with only 35 features even gives the best accuracy (72.1%), while training with the whole feature set is only 71.6% accurate. This demonstrates how features can carry the same information.

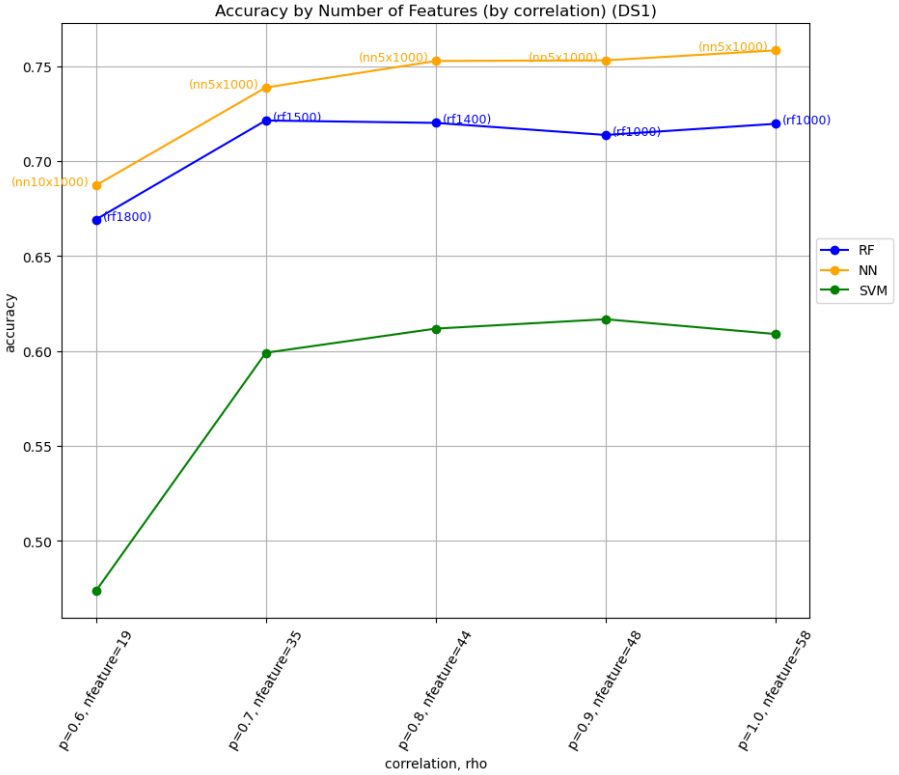

Figure 12: Model Accuracy by Number of Training Features (reduced by correlation) (DS1)

We can also reduce the number of features by ranking the features first and selecting only the subset at the top (e.g. top 10, top 20, etc). Here we use Random Forest feature importance ranking. Figure 13, shows that the NN model gives is most accurate (76.8%) with only the top 30 features. We can further reduce the number of features by removing correlated features first and then ranking, similar to Figure 14 where correlated features with $\rho = 0.8$ are removed. Here the best model performances are 76.3% (40

features) and 76.1% (30 features). This shows that well advised feature selection results in better model performance.

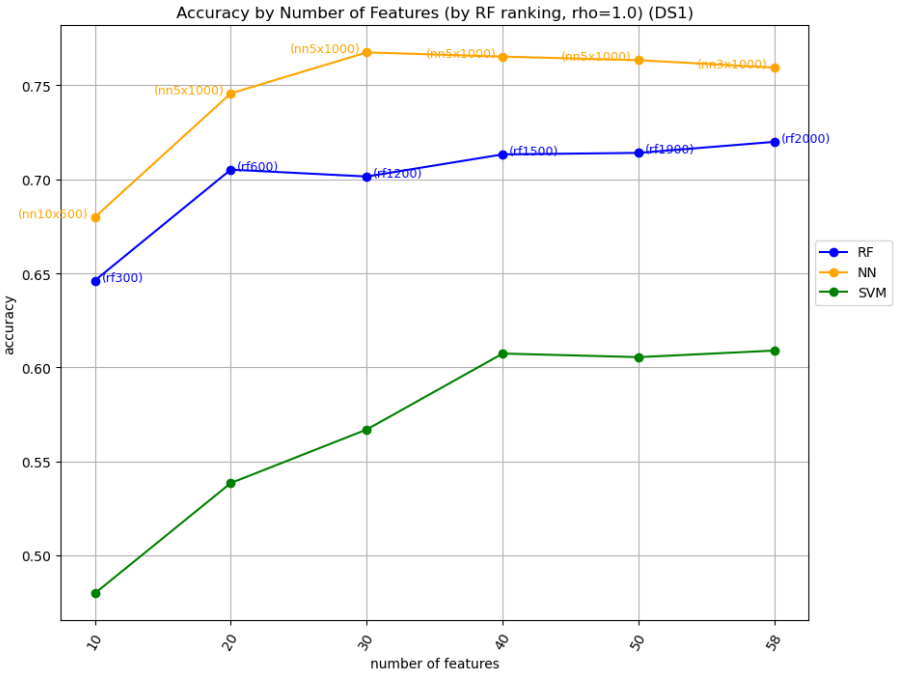

Figure 13: Model Accuracy by Number of Training Features (reduced by ranking) (DS1)

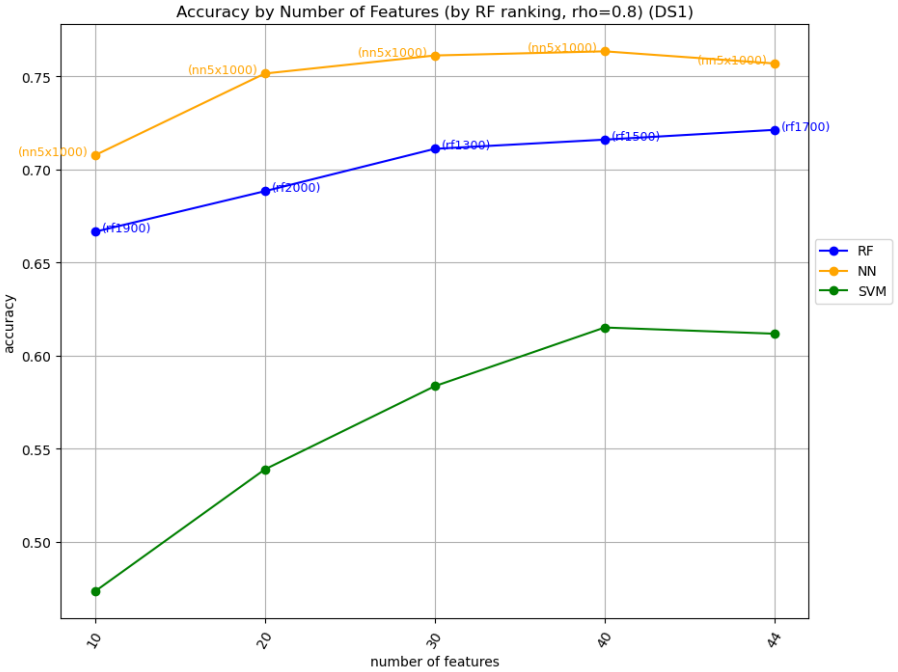

Figure 14: Model Accuracy by Number of Training Features (reduced by ranking) (DS1)

Lastly, we note that training time decreases with increasing number of features. This tells us that training converges faster with more features.

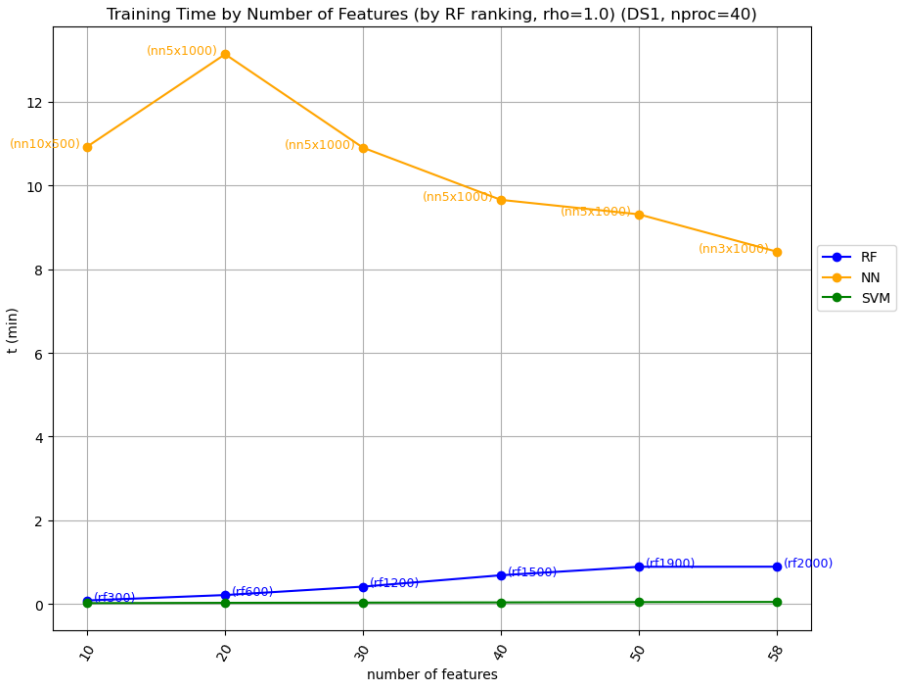

Figure 15: Model Training Times by Number of Training Features (DS1)

3.5 Accuracy by Feature Importance

Here we further look into feature selection by importance. For Dataset 1 , we compare several top 12 feature rankings (see Figure 16). The results in Figure 17 show that Domain Expert ranking (67%) is behind Random Forest ranking (71%) and Random Ranking 3 (69%). This demonstrates that machine learning methods can sometimes find associations in data that are not readily observable to human experts.

Figure 18 also shows that training times significantly vary even with equal number of features.

	Domain Expert	RF	PCA	Random 1	Random 2	Random 3	Random 4	Random 5
1	waiind	waiind	department_mean	siwressource	gpt	trig	framaltdifflim	lhqu
2	ksk15	ksk15	dev_purpose15	hba1c	waiind	wai_norma_exp	audit_c15	dev_jobpredict
3	dev_jobskill	sisubhealth	dev_jobskill	fghv_tob	audit_c15	copstress	demquan	dev_purpose15
4	dev_jobsat15	erkrankind	dev_dememo	slapn15	gender	framaltdifflim	ksk15	healb15
5	copstress	wai_norma_exp	dev_commit15	ksk15	fghv_tob	procama	ldllg	psk15
6	psk15	trig	dev_jobsat15	bluecollar	dev_purpose15	beweg15	tsh	dev_commit15
7	alter_cat	sipressource	dev_jobpredict	alter_cat	tsh	alter_cat	demphys	gender
8	bluecollar	throm	dev_copfuehr	hdl	alter_cat	wolib15	dev_jobsat15	declat15
9	dev_copfuehr	mbauch	sipressource	dev_jobpredict	mdrd4	waiind	hb	rrsysm
10	framaltdifflim	mdrd4	sisubhealth	hb	healb15	ksk15	gpt	procama
11	mets15	declat15	waiind	sisubhealth	dev_dememo	chol	erkrankind	hba1c
12	fghv_tob	chol	siwressource	procama	procamr	mbauch	ssindexdem	erkrankind

Figure 16: Different Feature Importance Rankings (DS1)

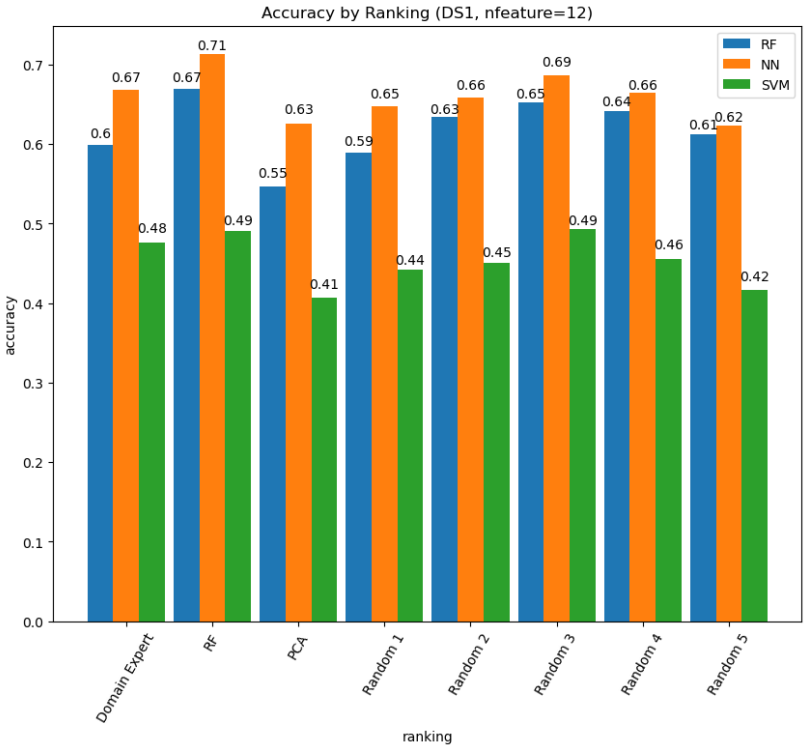

Figure 17: Model Accuracy by Feature Importance Ranking (DS1)

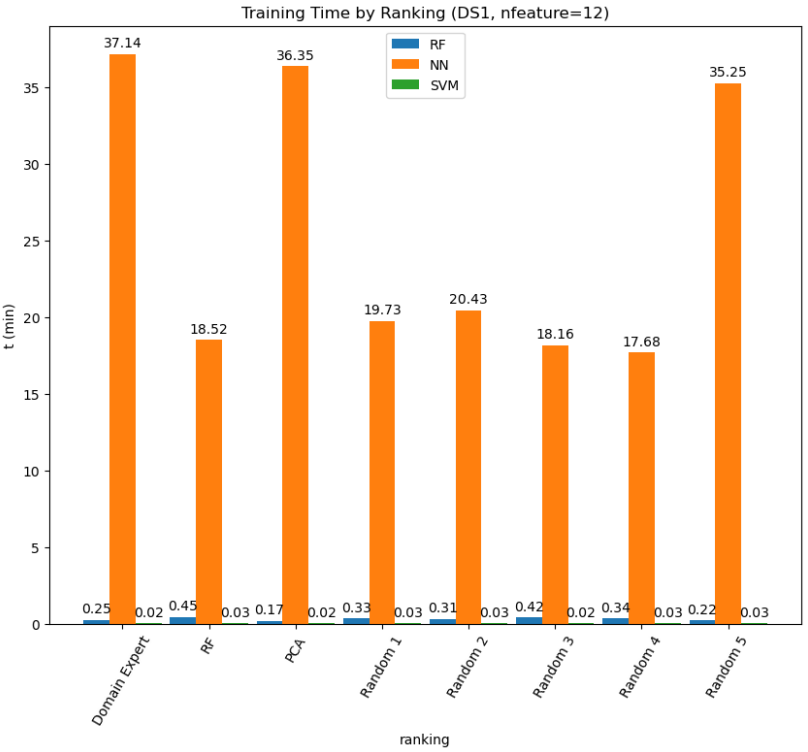

Figure 18: Model Training Time by Feature Importance Ranking (DS1)

3.6 Model Sensitivity to Features

Here we apply the method described in Section 2.2.2 on how to compute a Neural Network's sensitivity to changes in the features. A subset of the sensitivities are listed in Table 1 for Dataset 1.

rank	feature	sensitivity
1	waiind	0.734585
2	fghv_tob	0.220246
3	dev_copfuehr	0.207455
4	dev_jobsat15	0.195193
5	dev_jobskill	0.192048
6	dev_jobpredict	0.165959
7	rrsysm	0.153924
8	exhas15	0.147053
9	healb15	0.142487
10	procamr	0.116064
⋮	⋮	⋮
17	bluecollar	0.089616
⋮	⋮	⋮
38	bmi	0.040838
⋮	⋮	⋮
58	trig	0.001671

Table 1: Model's Sensitivity to Features (DS1)

To demonstrate how this sensitivities can be used, we look at three features: whether employee is smoker or non-smoker (*fghv_tob*), whether employee's job is blue collar or white collar (*bluecollar*) and employee's body mass index (*bmi*). It is known that these factors affect employee's health and work performance in general. Figure 19 shows that as expected smokers, blue collar jobs and high BMI result in more absences. (There were 5 absence labels for this dataset: 0, ≤ 2, ≤ 5, ≤ 14 and > 14 days absent per year. But consider that the first 2 classes belong to low risk and the next 3 as high risk.) Therefore, we can hypothesize that these factors might be significant to an employee's absence risk. Furthermore, based on the sensitivity ranking, we expect that the effect would be *fghv_tob* $>$ *bluecollar* $>$ *bmi*.

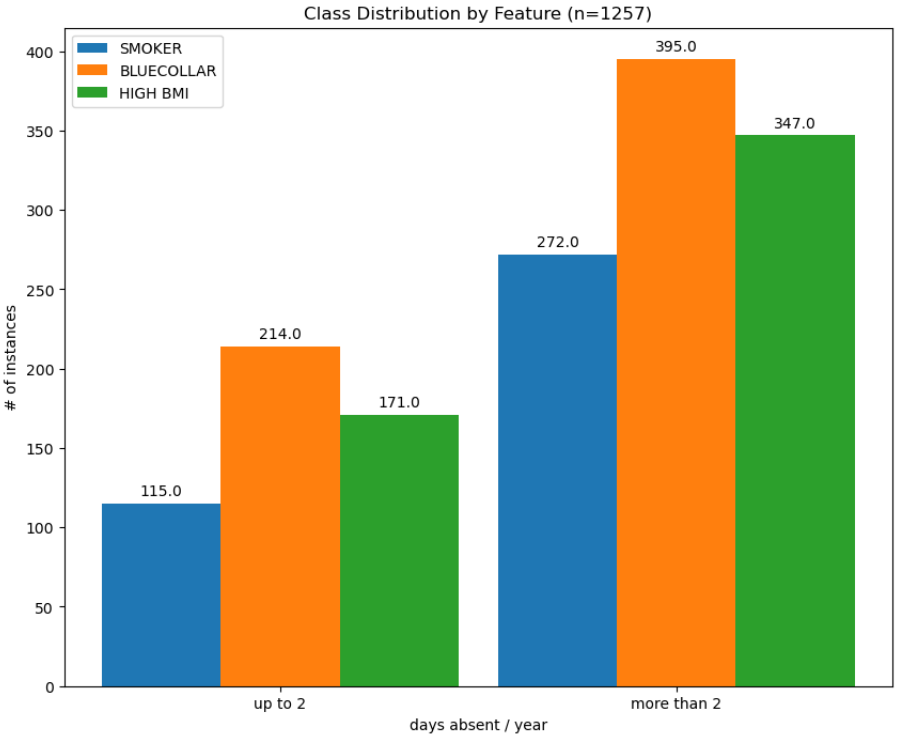

Figure 19: Low and High Absence Risk Distribution by Feature (DS1)

For each of this feature, we shuffle the values on the held-out test data set to represent change in an employee's attribute, i.e. smoker to non-smoker, blue collar to white collar, high to low BMI and vice versa. Then we check how the model reacts to these changes.

In Table 2, for the *fghv_tob* feature, shuffling produced 484 feature changes, of this 61 had class label changes and 49 of these class label changes were correct, i.e. smoker to non-smoker led to the model predicting a lower absence rate and vice versa. This is 80.3% accurate. For *bluecollar* and *bmi*, accuracies were 47.9% and 38.0% respectively. This suggests that if we were to implement measures to mitigate employee absences, it might be prudent to consider the features that the model is highly sensitive to.

feature	# feature changes	# class changes	# correct class change
fghv_tob	484	61 (12.6%)	49 (80.3%)
bluecollar	607	73 (12.0%)	35 (47.9%)
bmi	905	50 (5.5%)	19 (38.0%)

Table 2: Model Response to Feature Change (DS1)

4 Conclusion and Outlook

A comparison of machine learning models for predicting employee absence risk are compared. The training data includes employee medical and behavioral history, job- and life- related surveys and company structure data. The Neural Network model is found to give the best accuracy (77%) over Random Forest (72%) and Support Vector Machine (62%). It also takes at least 6 times longer to train compared to Random Forest and Support Vector Machine. In almost all test cases, Neural Networks perform better except when the training data size is smaller, here Random Forest has slight 1-2% advantage. Random Forest also gives consistent results with accuracy ranging from 68% - 74%. On the other hand, Support Vector Machine always performed worst among the three models.

In addition to training data size affecting accuracy, the set of training features (size and members) also influence model performance significantly. For example, highly correlated features contain the same information and removing redundant features has little impact on the model's accuracy. Neural Networks can also correctly classify the change in absence risk after switching values of features that the model is highly sensitive to. More work could be done towards features' effect on model. This could pinpoint in which areas companies can focus their efforts in reducing absences. Careful selection of features can also be cost effective and also enhance model generalization.

These findings suggest machine learning methods can successfully be used for predicting employee absences and similar health-related problems. Further improvements could be obtained with in-depth data analysis, exhaustive model tuning and rigorous model evaluation.

Acknowledgements

This study was carried out in the scope of the project KIPROSPER funded by the German Federal Ministry of Education and Research (BMBF; KMU-innovativ; 01IS18008).

References

[1] *Absence from work - Germany*. URL: https://www.eurofound.europa.eu/publications/report/2010/absence-from-work-germany.

[2] C. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.

[3] L. Breiman. "Random Forests". In: *Machine Learning* (2001).

[4] T. Dietterich. "An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization". In: *Machine Learning* (2000).

[5] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[6] *Preventing absenteeism at the workplace*. URL: https://www.eurofound.europa.eu/publications/report-summary/2000/preventing-absenteeism-in-the-workplace-summary.

Preprint Series of the Engineering Mathematics and Computing Lab

recent issues

No. 2021-01 Chen Song, Jonas Roller, Ana Victoria Ponce-Bobadilla, Nicolas Palacio-Escat,Julio Saez-Rodriguez, Vincent Heuveline: Spatial Effect on Separatrix of Two-Cell System and Parameter Sensitivity Analysis

No. 2020-01 Saskia Haupt, Nassim Fard-Rutherford, Philipp D. Lösel, Lars Grenacher, ArianebMehrabi, Vincent Heuveline: Mathematical Clustering Based on CrossSections in Medicine: Application to the Pancreatic Neck

No. 2019-02 Nils Schween, Nico Meyer-Hübner, Philipp Gerstner, Vincent Heuveline: A time step reduction method for Multi-Period Optimal Power Flow problems

No. 2019-01 Philipp Gerstner, Martin Baumann, Vincent Heuveline: Analysis of the Stationary Thermal-Electro Hydrodynamic Boussinesq Equations

No. 2018-02 Simon Gawlok, Vincent Heuveline: Nested Schur-Complement Solver for a Low-Mach Number Model: Application to a Cyclone-Cyclone Interaction

No. 2018-01 David John, Michael Schick, Vincent Heuveline: Learning model discrepancy of an electric motor with Bayesian inference

No. 2017-06 Simon Gawlok, Philipp Gerstner, Saskia Haupt, Vincent Heuveline, Jonas Kratzke, Philipp Lösel, Katrin Mang, Maraike Schmidtobreick, Nicolai Schoch, Nils Schween, Jonathan Schwegler, Chen Song, Marin Wlotzka: HiFlow^3 Technical Report on Release 2.0

No. 2017-05 Nicolai Schoch, Vincent Heuveline: Towards an Intelligent Framework for Personalized Simulation-enhanced Surgery Assistance: Linking a Simulation Ontology to a Reinforcement Learning Algorithm for Calibration of Numerical Simulations

No. 2017-04 Martin Wlotzka, Thierry Morel, Andrea Piacentini, Vincent Heuveline: New features for advanced dynamic parallel communication routines in OpenPALM: Algorithms and documentation

No. 2017-03 Martin Wlotzka, Vincent Heuveline: An energy-efficient parallel multigrid method for multi-core CPU platforms and HPC clusters

No. 2017-02 Thomas Loderer, Vincent Heuveline: New sparsing approach for real-time simulations of stiff models on electronic control units

No. 2017-01 Chen Song, Markus Stoll, Kristina Giske, Rolf Bendl, Vincent Heuveline: Sparse Grids for quantifying motion uncertainties in biomechanical models of radiotherapy patients

No. 2016-02 Jonas Kratzke, Vincent Heuveline: An analytically solvable benchmark problem for fluid-structure interaction with uncertain parameters

No. 2016-01 Philipp Gerstner, Michael Schick, Vincent Heuveline, Nico Meyer-Hübner, Michael Suriyah, Thomas Leibfried, Viktor Slednev, Wolf Fichtner, Valentin Bertsch: A Domain Decomposition Approach for Solving Dynamic Optimal Power Flow Problems in Parallel with Application to the German Transmission Grid

No. 2015-04 Philipp Gerstner, Vincent Heuveline, Michael Schick : A Multilevel Domain Decomposition approach for solving time constrained Optimal Power Flow problems

Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)

www.emcl.iwr.uni-heidelberg.de