UPPSALA
UNIVERSITET

UPTEC X 20025
Examensarbete 30 hp
November 2020

Time prediction and process discovery of administration process

Johanna Öberg

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Time prediction and process discovery of administration process

Johanna Öberg

Machine learning and process mining are two techniques that are becoming more and more popular among organisations for business intelligence purposes. Results from these techniques can be very useful for organisations' decision-making. The Swedish National Forensic Centre (NFC), an organisation that performs forensic analyses, is in need of a way to visualise and understand its administration process. In addition, the organisation would like to be able to predict the time analyses will take to perform. In this project, it was evaluated if machine learning and process mining could be used on NFC's administration process-related data to satisfy the organisation's needs.

Using the process mining tool Mehrwerk Process Mining implemented in the software Qlik Sense, different process variants were discovered from the data and visualised in a comprehensible way. The process variants were easy to interpret and useful for NFC. Machine learning regression models were trained on the data to predict analysis length. Two different datasets were tried, a large dataset with few features and a smaller dataset with more features. The models were then evaluated on test datasets. The models did not predict the length of analyses in an acceptable way. A reason to this could be that the information in the data was not sufficient for this prediction.

Handledare: Helene van Ettinger-Veenstra
Ämnesgranskare: Jan Komorowski
Examinator: Pascal Milesi
ISSN: 1401-2138, UPTEC X 20025
Tryckt av: Uppsala

Populärvetenskaplig sammanfattning

Artificiell intelligens (AI) och maskininlärning är två begrepp som dyker upp i många sammanhang och används allt mer i dagens samhälle. Maskininlärning är en teknik inom AI och kanske den mest använda just nu. Den handlar om att skapa modeller som beskriver verkliga processer, genom att mata en modell med data och låta modellen själv få lära sig mönster i datan, som kan vara för svåra eller tidskrävande för en människa att upptäcka. Det som skiljer maskininlärning mot traditionell programmering är att maskininlärning inte kräver någon explicit programmering av hur modellen ska fungera eller någon kunskap om hur verkligheten är. Det lär sig modellen själv med hjälp av datan. Maskininlärning har visat stor potential inom många applikationer och en modell kan bland annat användas för att klassificera objekt eller prediktera framtida utfall i olika processer.

Många organisationer och företag har stora datamängder insamlade och skulle potentiellt kunna ha stor nytta av maskininlärning. Men det finns en del krav som behöver uppfyllas för att lyckas med maskininlärning, till exempel att ha tillräckligt mycket data och av god kvalitet. Nationellt Forensiskt Center (NFC) är en del av polismyndigheten i Sverige och genomför forensiska analyser. Organisationen har en stor mängd insamlad data om alla ärenden från 2015 till 2020 som NFC önskar kunna använda för maskininlärning. I det här projektet undersöks det hur denna ärendedata kan användas för maskininlärning och om en maskininlärningsmodell kan prediktera analystid för en analys i ett ärende. Projektet syftar till att bredda kunskapen om vad som krävs för att få ett framgångsrikt maskininlärningsprojekt.

Process mining är annan teknik som är data-baserad och som kan utnyttjas av organisationer som har data relaterad till en process, som en produktionsprocess eller, som i det här fallet, en ärendehanteringsprocess. Process mining kan ge insikter i hur den tänkta processen ser ut i verkligheten, hitta varianter av processen som en del av produkterna eller ärendena tar och visa avvikelser och förseningar i processen. I det här projektet har det undersökts hur NFC's ärendedata kan användas för process mining för att se om det kan vara givande för organisationer att använda. En mjukvara användes för att beräkna en processmodell från datan, vilken sedan kunde analyseras. NFC's data med tidsstämplar för de olika stegen i processen var mycket väl anpassad till process mining, mjukvaran var enkel att använda och processmodellen visade en stor mängd processvarianter. Sammanfattningsvis verkar process mining kunna vara ett mycket bra verktyg för organisationer med processer. Men att förstå processvarianterna till fullo kräver tid och stor insikt i organisationens verksamhet, något som en organisation bör vara medveten om innan process mining används.

Maskininlärningsdelen av projektet visade att maskininlärning går att applicera på datan, men att datan inte innehöll tillräckligt med faktorer som påverkar analystid som kunde användas till att prediktera analystid på ett bra sätt. Prediktionerna blev väldigt otillförlitliga. Ett mindre dataset med mer faktorer testades också, för att se om dessa faktorer kunde förbättra prediktionen. Men datasetet var för litet för maskininlärning av den här typen. Ett lyckat maskininlärningsprojekt kräver en stor mängd av bra, högkvalitativ data som är representativ för datan som modellen ska användas för i framtiden.

Table of contents

Abbreviations

AI	Artificial Intelligence
BI	Business Intelligence
CV	Cross-validation
MAE	Mean Absolute Error
MPM	Mehrwerk Process Mining
NFC	National Forensic Centre
RMSE	Root Mean Square Error

1 Introduction

The use of Artificial Intelligence (AI) is increasing rapidly within all kinds of industries and AI shows promising results in many applications. A popular type of AI is machine learning – the development of computer models that can adjust to data without being explicitly programmed. Machine learning models are typically designed to find patterns in data or to predict future outcomes based on historical data (Gruson *et al.* 2019, Géron 2017). One such industry that has started to use AI is the Business Intelligence (BI) industry. BI can be described as the tools and methods that organisations use to analyse information to optimise decision-making (Larson & Chang 2016). Most organisations acquire large amounts of business-related or process-related data every year, and traditional BI is widely used. With AI, the BI methods could be even more helpful for organisations and give insights in the data that are not possible to get with traditional BI. But how easy is it to implement AI, and what type of data is required?

Different techniques for process-related machine learning predictions have been developed during the past years, and the applications for these techniques are many. Examples are prediction of the remaining running time of the process, prediction of the next activity and its timestamps, and prediction of the case outcome. Research has shown that using Long Short-Term Memory recurrent neural networks to predict the next activity and its timestamp of a case can be successful (Tax *et al.* 2017).

Another field that is growing within BI is the process mining field. The goal of process mining is to extract process-related information in data. The most common step in process mining is *process discovery*, *i.e.* to create a process model that describes the true process and reveals bottlenecks and deviations from the expected process (van der Aalst 2016). Process discovery and visualisation of processes can be made in many ways; there are successful examples of both specific process discovery algorithms and machine learning algorithms (Vartianien 2017).

This project aims to investigate how machine learning and process discovery can be applied to an organisation's administration data, by using data from the Swedish National Forensic Centre (NFC). NFC is an expert organisation within the Swedish Police Authority that performs forensic analyses and investigations. NFC receives orders from the Police Authority and judicial authorities of forensic analyses to be performed. Each order (called *case*) can consist of one or several analyses. The case administration process is complex and NFC handles many thousands of cases every year. This has resulted in a large amount of data in NFC's administration database, with information about all the cases and timestamps of the process activities.

In the coming years the Swedish Police Authority, which NFC is a part of, aims to shorten the pre-trial detention times and increase the efficiency of police investigations. For NFC, this means to have a more efficient case administration and time planning. To do this it is essential to have a high understanding of the administration process. NFC would need a way to discover and visualise the process, and also to be able to predict how much time an analysis will take to complete. These abilities would greatly help NFC achieve the efficiency goal.

To succeed with this project, I have had support from the IT consulting company Drake Analytics. Drake Analytics is specialised in BI and data analytics, and has both the right competence and the right software for process mining and training machine learning models.

1.1 Project aims and limitations

The large amounts of process data at NFC could give NFC valuable information about the administration process and possibly give a way to predict the analysis time. The project's goals are (1) to discover and visualise the case administration process using the process discovery tool Mehrwerk Process Mining (MPM), and (2) to investigate if and how the data can be used for machine learning predictions by training and testing machine learning models. MPM was selected as process discovery tool since it is an advanced but easy tool to understand and to use. The second goal is specified to predicting the time it takes to execute an analysis (*i.e.* the analysis length), not the complete case length. The machine learning predictions will be made on two different data sets – one containing more information but fewer examples than the other data set – to see if the predictions become better with more information.

The impact goal of the project is to help NFC, but also to understand more generally how organisations similar to NFC can use machine learning and process discovery on their data. My hope is that this project can increase the knowledge of what is needed for a successful machine learning project.

In this short time frame it is unlikely that the project will result in a machine learning model that can be implemented directly at NFC, but the project will answer if and how the data at NFC can be used for time predictions, and if this idea is something to continue to work on for the organisation.

2 Background

In this section, theory and concepts needed to understand the other sections of the report are presented, including the case administration at NFC, BI, process mining and machine learning. The three last concepts are all parts of the data science field, and are closely related and partially overlapping. Data science is an interdisciplinary field that includes all types of data extraction, preparation, transformation, visualisation, data mining and prediction techniques (van der Aalst 2016). All information about NFC in this report has been given to me by employees at NFC and I have their consent to use it.

2.1 The case administration at NFC

The cases at NFC consists of one or sometimes several forensic analyses to perform. At the receiving of each case NFC presents a time plan for the administration of the case to the client. The time it will take to complete a case depends on factors such as the number and type of analyses, the available resources and the current work load of the organisation.

2.2 Business Intelligence

Large amounts of data are produced today and stored in information systems, and a main challenge for many organisations is to extract information from the data. BI can be defined as an umbrella term for methods and tools for organisations to access and analyse information, in order to optimise decision-making. It often begins with collecting raw data and applying business context to the data to create information. Then, the information is used for decisions and actions that can provide business value (Larson & Chang 2016).

BI has been highly affected by the rapid emerge of "Big Data" and AI during the last years. Traditional BI focus on descriptive analytics, with questions like "what has happened, how is the situation today?", but it is starting to be replaced by predictive and prescriptive analytics that uses machine learning predictions, that can answer the question "what will happen?" (Larson & Chang 2016).

The BI software used in this project is Qlik Sense. To the Qlik Sense platform the user can import data sets from files or from enterprise information systems. The platform

provides search, selection and filter functions, and the user can calculate visualisations and diagrams of any subset of the data (Meyer *et al.* 2019).

2.3 Process mining

Process mining combines data mining with business processes. Data mining is the analysis of large data sets to find summaries and relationships. The goal of process mining is to extract process-related information, and it is done by using historical data recorded by enterprise information systems. Either a process model is discovered from the data, or process models are applied as reference models. With process mining techniques an organisation can visualise and analyse a process, and discover bottlenecks and deviations from the expected process path (van der Aalst 2016).

Most processes leave traces or recorded information in databases, called *event logs*. An event log can be described as the set of events belonging to the execution of cases (*i.e.* process instances). The events represent activities or steps in the process, and are characterised by properties like activity name and timestamp. Event logs can also store additional information of the events, like the resources executing the activity (Tax *et al.* 2017, van der Aalst 2016). In this project, NFC's data can be seen as event logs, where each row consists of events with names and timestamps. Table 1 shows a small example of an event log based on NFC's data. If a start event and a stop event belong to the same activity, like *analysis started* and *analysis completed* in Table 1, the duration of that activity can be measured (van der Aalst 2016).

Table 1: An example of a small event log based on the order administration data of NFC. Each row represent an event (activity) of the process for a certain case.

Case-ID	Timestamp	Activity name	Activity resource
1	2020-01-20	Case created	Name 1
1	2020-01-22	Material received	Name 1
1	2020-01-23	Analysis started	Name 2
1	2020-01-29	Analysis completed	Name 2
1	2020-02-05	Case completed	Name 1
2	2020-01-17	Case created	Name 3
2	2020-01-18	Analysis started	Name 3

To use event logs to create a model of the underlying process is called *process discovery*, and that is the first step of process mining (van der Aalst 2016). It is the only step that will be used in this project, and it will be made by MPM. A process model is often in the shape of a directed graph, *i.e.* a set of nodes (activities or events) connected by edges (transitions between events) (Vartianien 2017).

2.3.1 Mehrwerk Process Mining

The process mining tool Mehrwerk Process Mining (MPM) is used in this project. Mehrwerk is a German company working with Qlik Sense that has developed MPM. MPM is implemented as a Qlik Sense extension; a tool deployed on the Qlik Sense platform. MPM combines BI with process mining algorithms and visualisations. Event logs are used as data to discover the process. The data can be imported to Qlik Sense from CSV files and transformed into event log shape by an MPM algorithm. The right format of event logs for MPM is a table with the columns *Case ID*, *Activity type*, *Activity username*, *Activity start timestamp*, and *Activity end timestamp*, where the Case ID column contains the IDs of the cases that flow through the process. Activity username is optional (Meyer *et al.* 2019).

The MPM algorithms transform the event log into a process model. The order that the activities are performed in can be seen in the model, and it is visible if the activities are performed in alternative orders (alternative process variants). It shows the average duration time for the activities and the lag time between activities (Meyer *et al.* 2019).

2.4 Artificial Intelligence and machine learning

The mimicking of human behaviours and human thoughts for actions is a key characteristic of AI. AI has been shown to be very useful for complex tasks like weather forecasting and face recognition. Machine learning can be seen as a subfield of AI and includes the developing of computer models (Gruson *et al.* 2019). Normally, the data used for machine learning consists of rows and columns, where each row is called an *instance*, and where each column is called a *feature* and represents some characteristic of the instance (Géron 2017).

With machine learning, the model do not require explicit rule descriptions. This is the main difference between machine learning and traditional model programming. The model adjusts to fit the training data that it is provided with, and can then be used to predict outcome from new data (Gruson *et al.* 2019). A machine learning model can

learn from the data which features that are informative and most important for predicting correctly (Géron 2017).

2.4.1 Supervised machine learning

There are three main types of machine learning: supervised, unsupervised and reinforcement learning. In this project only supervised learning is used. In supervised learning, each instance in the data that the model is trained on has been labelled with the instance's "answer", the correct outcome. The labels can be manually assigned or come from measurements. The features are the input and the label the output when using the model for predictions. The model is "fitted" or "trained" on the training data, by predicting the data, measuring how "wrong" it was using the correct answers, and then adjusting itself (its model parameters) to perform better next time. This is repeated until the model is well fitted to all of the training data. A separate test data set can be used to measure the performance of the model, to assess how well it can generalise to new, "unseen" data. The test set is normally not used until at the end of the project (Géron 2017).

Two common supervised learning tasks are *classification* and *regression*. A regression example is to predict a numeric value (*target* value) such as the price of a house, given a set of features, *e.g.* the size of the house and number of bathrooms (Géron 2017). Since the task in this project is to predict a natural number, *i.e.* the analysis length, only regression models and no classification models will be used. An example structure of a data set used for regression is shown in Table 2.

Table 2: An example structure of a data set with three features used for machine learning regression. The prediction target is the price of a house.

INDEX	FEATURES			TARGET
ID	Bathrooms	Distance to city	Colour	Price
1	2	5	Blue	100,000
2	3	20	White	75,000

2.4.2 Requirements

The requirements for supervised machine learning are (a) a definition of the task to perform, (b) a performance metric for evaluating the model's performance, (c) a data set of examples to learn from with known predictions, and (d) one or several models to be tested (Gruson *et al.* 2019). Machine learning also requires sufficient quantity

and quality of the data. Typically thousands of examples are needed, even for simple problems. The training data needs to be representative of the new data that should be predicted, and not contain too much errors, noise or outliers which make it harder for the algorithm to detect the underlying patterns of the data. The data needs to contain enough relevant and not too many irrelevant features.

Another challenge is that the model can be overfitting or underfitting while training. If the model is fitted too good to the data (overfitting), it has learnt the training data too well and will not be able to generalise well. This can happen if the model is too complex in comparison to the complexity or amount of training data. To reduce overfitting, regularisation techniques that constrain the model can be used. If the model is not fitted well to the data (underfitting), it may be too simple to describe the data and learn its underlying structure, and the model will not predict well. In that case a more complex model can be tried, or more informative features should be used (Géron 2017).

2.5 Regression

Regression is about writing the numeric output as a function (model) of the input. The most simple regression method is a linear function, *e.g.* $y = kx + m$, where y is the output value and x is the input feature. There are two model parameters, k and m, that can be adjusted to the data so that the model represents any linear function. To decide which linear function that represents the data points the best, a performance measure needs to be specified. Normally a cost function is defined, that describes how bad the model fits the data, *i.e.* how big the difference is between the training data and the linear model's predictions. During the training, the linear regression algorithm finds the model parameters that minimises the cost function (Géron 2017). How the training and optimisation of the cost function is done will not be explained in this report.

Two common cost functions for regression tasks are Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). RMSE is the square root of the average squared error, while MAE is the "sum of absolutes", the average of absolutes of the errors. RMSE is more sensitive to outliers, since it is affected more by very large errors (Géron 2017).

A linear regression model can have more than two parameters, so that the data points cannot be visualised by a 1-dimensional graph. If the parameters are seen as weights, a linear model computes a weighted sum of the input features plus a constant term (called bias term or intercept term). It is described in the equation below,

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \tag{1}$$

where y is the predicted value, θ_i is the i$^{\text{th}}$ model parameter, n is the number of features and x_j is the j$^{\text{th}}$ feature value (Géron 2017).

The features can be both categorical (*e.g.* red, blue, green) and numerical (*e.g.* 1, 2, 3). Most machine learning models work best with numerical features, so categorical features need to be mapped to numerical values. One way is to simply replace each category with a number, *e.g.* red becomes 1, blue becomes 2, and green becomes 3. A problem with this is that the algorithms will assume that nearby values are more similar than distant ones, so that red is more similar to blue than to green. In this case the method *one-hot encoding* can be used. With one-hot encoding new features are created, one for each category of the original feature, and each new feature either contains 1 or 0. So for the example with colours, the feature names can be red, blue and green, and for a red instance the values are [1, 0, 0], while for a green instance the values are [0, 0, 1]. If the number of possible categories of a feature is large, the resulting features of a one-hot encoding will be many, which can decrease the performance and slow down training (Géron 2017).

Numerical features are often scaled. If the numerical features have very different scales the models might not perform well. But scaling the target value, in this project the analysis length, is often not needed. Scaling can be done with *normalisation* (also called min-max scaling) or with *standardisation*. Normalisation uses the maximum and minimum values of all features and results in a range from 0 to 1 for all feature values. Standardisation rescales the values so that all features have a zero mean and the distribution has unit variance. Normalisation is more sensitive to outliers than standardisation (Géron 2017).

Commonly, the training set is pre-processed independently of the test set. The test set or other new data is then transformed in the same way as the training set, before making predictions. The test set is treated as if it was not present during the training. For example, when normalising the test data the same min-max range should be used as for the training set. To ensure this, the pre-processing functions' parameters are often fitted to the training data and then saved (Géron 2017).

When training a first model on the training set, it can be interesting to evaluate the model without using the test set. In this case, the training set can be divided into a training part and a validation part. The model is trained on the training part and evaluated on the validation part. To make this evaluation result more reliable, the training data can be split into *k* distinct subsets, and the model is trained on *k-1* subsets and evaluated on the last subset. This is done *k* times, and each time a different subset is used for validation. This is called *k-fold cross-validation*. After training, *k* different evaluation scores are available and an average of the scores can be taken (Géron 2017).

There are other models than a linear function that can be used for regression. The goal is to find the best performing model, the model that best describes new data. A good approach is to first try different models, and then choose two to five promising models to use for fine-tuning of the hyperparameters. After fine-tuning, the models with their final settings are evaluated on the test set. Hyperparameters are the parameters of a machine learning algorithm that are set by the user; the hyperparameters do not change in the learning process. Fine-tuning can be done by manually changing the hyperparameters, one by one, to find a good combination, or by using a method. Randomised Search and Grid Search are the two methods used in this project. Randomised Search randomly selects and evaluates a given number of combinations from defined sets of hyperparameter values, while Grid Search evaluates all possible combinations of the defined sets (Géron 2017).

2.5.1 Decision Trees and Random Forest regressors

A Decision Tree is a machine learning model that can fit complex data sets and handle both classification and regression. An example tree of a regression task is shown in Figure 1. The model consists of a root node (top), child nodes, and leaf nodes. During training of the model, features are associated with the nodes and the values are adjusted so that the errors are minimised. For regression tasks the value in each leaf node is the average target value of the training instances that are connected to that node. A Decision Tree can be regularised by setting a defined constraint on the minimum number of instances per leaf node. After training, the model can be used to predict new instances. In general, Decision Trees are sensitive to variations in the data, *e.g.* if a some instances from the training data are removed the resulting tree can be completely different (Géron 2017).

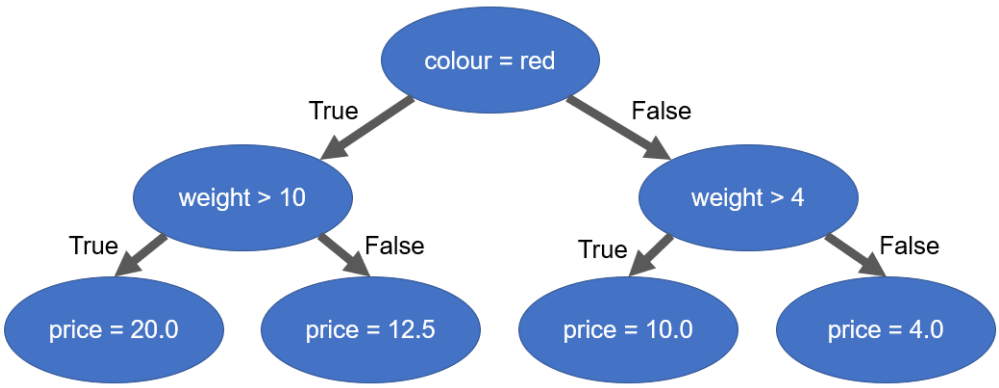

Figure 1: Example of a simple Decision Tree model for a regression task. The features are colour and weight, and the prediction target is price.

A Random Forest is a model that consists of a "forest" of many Decision Trees, and each tree is being trained on random subsets of the features. The prediction of the Random Forest is the average of the prediction from the trees. This is called *ensemble learning*, to create a model from many other models. Random Forests are generally more stable than Decision Trees (Géron 2017).

2.5.2 Feature selection

A trained Random Forest regressor can, in addition to making predictions, also give information about the relative importance of each feature. How much each feature contributes to accurate predictions. The trained model contains importance scores for all features. This can give valuable insights to the problem and data, but the feature importance analysis can also be used to change the learning. One or several of the less important features can be dropped, which can make the model simpler and possibly increase the accuracy. A feature's importance can be estimated by how much the impurity is reduced by the nodes that use the feature, across all trees. The impurity of a node is a measure of how "pure" it is, *i.e.* how many of the training instances that it is connected to belong to the same class or have similar target values (Géron 2017).

3 Material and methods

The project consisted of the following four main activities: examine the data, visualise the process using MPM, pre-process the data to fit machine learning, and train and evaluate different machine learning models. These activities are described in the following subsections, starting with a description of the data.

3.1 Data

Five different data sets were used in this project – four for the machine learning part and one for the process discovery part – and they were all in CSV format. The data sets came originally from the same source. The data fields in the database have been filled in manually by the users at NFC and there are no controls in the database that the values are correct or that all fields are filled in.

For the machine learning part, there were a large data set with all cases created and com-

pleted or cancelled in January 2015 to April 2020 (called the case data set) and a smaller data set with only the fingerprint analyses from October 2019 to April 2020 (called the fingerprint data set) that had additional information about the employee occupancy rate and the analysis queue where the analysis was performed. This additional information was not available for the case data set. Almost all cases and analyses used had been completed; only a few had been cancelled and they had in general short lengths.

There were also two very small data sets with data from May 2020, used in the end of the project. All five data sets are described below.

3.1.1 The case data set

The case data set contained originally 643,527 rows (instances). In this data set each instance represented an analysis (*e.g.* drug analysis, fingerprint analysis and DNA analysis) with a unique analysis ID, a code for the type of analysis, a case ID indicating which case the analysis belonged to, and other features such as the places where the case and analysis were performed and the names of the persons that performed them. In NFC's system each analysis belongs to a case, *i.e.* all cases consist of one or more analyses. It can be seen as a one-to-many relationship. The 643,527 analyses belonged to 530,661 different cases.

After filtering instances, removing features and creating new features from the original ones, the number of instances was 552,448 and the features were 11 (excluding the target; analysis length). The features *case place* and *analysis place* were used to create a third feature called *same place*, with values "Yes" and "No". If one or both of the values were missing the value was set to "No". The feature *no of order codes* was calculated from *final order codes*, and the missing values were set to 0. After the feature selection (see subsection 3.4.3), five features remained for predicting the target. See Table 3 for descriptions of all features.

Table 3: Names, descriptions, data types, and examples of the 11 features of the case data set. The features selected for machine learning prediction are marked with an asterisk. Analysis ID was not used, instead the row number was set as index (instance ID). Some of the features had many missing values. The numbers in parentheses are the numbers of categories.

Name	Description	Data type	Example
Analysis ID	Unique ID for each analysis	Num	1
Analysis type*	Type of analysis	Cat (290)	N15
Case ID	ID of the case that analysis belongs to	Num	1
Case type*	Type of case	Cat (3)	Drug
No of analyses*	Number of analyses case is related to	Num	2
Analysis place	Where the analysis is performed	Cat (5)	West
Case place	Where the case is performed	Cat (4)	West
Same place	Yes if analysis place equals case place	Cat (2)	Yes
Order codes	Which analyses that are ordered	Cat	N15,S21
No of order codes*	Number of order codes	Num	2
Priority*	Decides if the case is prioritised	Cat (4)	Yes

num: numerical, cat: categorical

The distribution of different analysis types was very uneven; some types were much more common than other. The high number of different analysis types (290) could be problematic when using machine learning. To decrease this number the values were grouped based on similarity so that similar analyses that were run at the same place belonged to the same group.

Some analysis types occurred often together with certain case types. For example, N-analyses (drugs and doping analyses) occurred most often with case type "1" (drug case). About 10% of all cases had the priority "Yes", and the remaining had "No", "No, down-classified", or "Yes, upclassified", or the value was missing.

In Table 4, the minimum and maximum values for the analysis length and the case length are shown, together with the mean, median and standard deviation. The distribution of the analysis length is shown in Figure 2. The range was large and the distribution uneven; most cases and analyses were short but some were very long. This could make the prediction more difficult, since there are few long examples for the model to learn from.

Table 4: Statistics of analysis length and case length for the raw case data set, in days.

Length	Minimum	Maximum	Mean	Median	Std
Analysis	1	1569	14	4	39
Case	1	1645	38	8	80

std: standard deviation

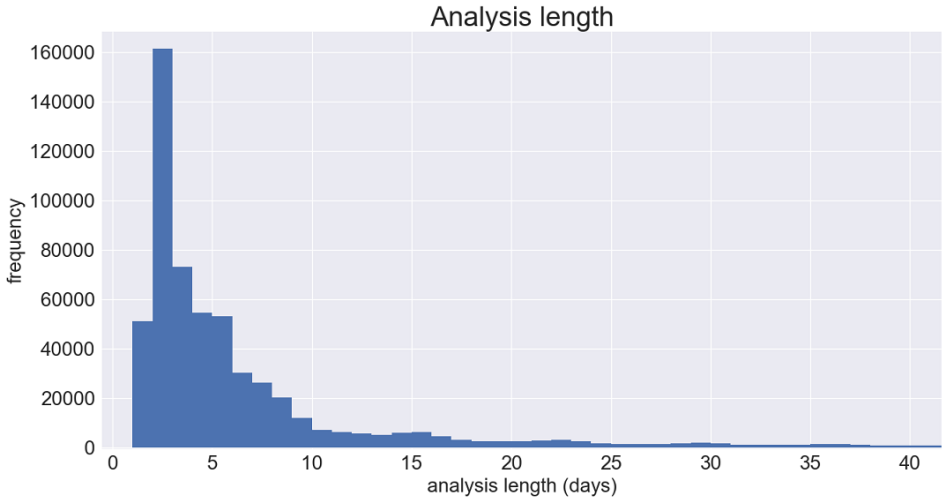

Figure 2: Histogram of the analysis length of the case data set, showing the lengths from 0 to 40 days. The analyses longer than that are not shown in this window.

3.1.2 The fingerprint data set

The fingerprint data set consisted of the instances of analysis type "S21" from October 2019 to April 2020, with all features included in Table 3 plus seven additional features, shown in Table 5. The additional features contained information about the weekly occupancy rate and weekly analysis queue, for the four different regional NFC offices (Linköping, Malmö, Gothenburg, and Stockholm). The code "S21" means fingerprint developing on objects. The data set contained 3,390 rows and 18 columns. The feature *incoming forecast* was 0 for some weeks, but this was probably due to that the values were unknown or not calculated, not that the actual forecasts were 0 cases. The features selected for machine learning prediction of the target value analysis length were *no of analyses*, *no of order codes*, *priority*, and all features in Table 5 except for *available YL to production*, since it had high correlation with *time production*. The features *analysis type* and *case type* had the same values for all instances, and were therefore not selected.

Table 5: Names and descriptions of the seven additional features of the fingerprint data set, that were not available for the whole case data set. There were no missing values for these features, and all were of numerical data type.

Feature name	Description
Time available	Average working rate for employees
Time production	Available production time (hours)
Time production fraction	Fraction of available time towards production
Time not production	Allocated time beyond production (hours)
Available YL to production	Available yearly labour towards production
Queue	The number of analyses in queue
Incoming forecast	Expected number of analyses to come next week

3.1.3 New data from May 2020

To simulate how the machine learning would predict on new data, the latest data available when starting the project – data from May 2020 – was extracted and put aside until the end of the project as an extra holdout data set. This was done for both the case data set and for the fingerprint data set.

3.1.4 The data set for process mining

For the process mining part of the project, the selected data set had 613,920 rows and seven columns. Descriptions and names of the columns are shown in Table 6

Table 6: Names, descriptions, and amount of missing values for the data columns used for process mining.

Feature name	Description	Missing (%)
Analysis ID	Unique ID for each analysis	0
Case created	Date for case creation	0
Material received	Date when last material for analysis came in	1.9
Analysis created	Date for analysis creation	2.7
Analysis started	Date analysis starting	3.2
Analysis completed	Date for analysis completion	2.8
Case completed	Date for case completion	0

3.2 Examining data

Before moving the data from NFC's databases to an external computer the columns with employee names and registration numbers had to be anonymised. This was done in Microsoft Excel. The data sets were then examined in Python (version 3.8) with the Python packages *pandas* (McKinney 2010), *matplotlib* (Hunter 2007), and *pandas-profiling* (Brugman 2020), to assess the quality of the data, to see the distributions of features and to find potential correlations between features.

3.3 Visualising the process with MPM

The software packages Qlik Sense (QlikTech International AB 2020) and MPM (Mehrwerk GmbH 2020) were used to visualise the case administration process based on the case data set. The case data set in CSV format was imported to Qlik Sense and a script was written in Qlik Sense that defined how the data set should be used to create the MPM model. Since *analysis started* and *analysis completed* could be seen as two timestamps of the same activity, they were used to create the activity *analysing* with a defined length. All other activities had only one timestamp and therefore no length (or length 0 seconds). By MPM algorithms the data set was transformed into the required shape and a MPM model was created.

Before importing the data set into Qlik Sense some filtering was done in Microsoft Excel. The instances with negative analysis length or negative case length were removed.

3.4 Pre-processing for machine learning

To fit for machine learning prediction of analysis length, the two raw data sets were filtered and pre-processed and additional features were created. The software for this was Python, and the Python packages used were *pandas* (McKinney 2010), *numpy* (Harris *et al.* 2020), *imblearn* (Lemaitre 2017), *matplotlib* (Hunter 2007), and *scikit-learn* (Pedregosa *et al.* 2011). Most of the pre-processing functions came from *scikit-learn*.

3.4.1 Filtering

Instances with values that were assumed to be incorrect due to human error when manually filling in values or assumed to be unrepresentative of the cases were removed, after discussion with NFC. The instances removed had at least one of the following properties: negative case length, negative analysis length, a missing value for analysis length (the target value). Instances with analysis length, number of order codes, or number of analyses longer than a defined 95%-shortest cutoff were also removed (*i.e.* the longest 5% of each feature was removed).

The 95%-shortest cutoff on the three numerical features *analysis length*, *no of order codes*, and *no of analyses* was motivated by the fact that the 5% highest values were very high and seen as unrepresentative of the data (extreme values). These three numerical features mentioned all have mostly small values; large values are rare. In addition, the model would probably have easier to learn to predict the more common values than the very uncommon, and predict better if the most rare values were removed. Normally in a machine learning project, the model is designed to be able to predict all data, but in this project it was assumed that it would be more useful for NFC to have a model that can predict the common cases well, than a model that predicts all cases with lower accuracy. For the case data set, the 95%-shortest cutoff values were calculated to 65 days for analysis length, 6 order codes and 7 analyses. For the fingerprint data set the cutoff values were 108, 8 and 13.

3.4.2 Pre-processing

Before the pre-processing, the prediction target (*analysis length*) was extracted from the data sets and put in a separate variable, since it did not need to be pre-processed. The following pre-processing steps were made on the case data set:

- Instances with a missing category for *analysis type* were given the new category "missing".
- Instances with a missing category for *priority* were given the new category "missing".
- The 290 different categories of *analysis type* were grouped into 17 groups, based on similarity of the analyses. Values that started with the same letter represented similar analyses and were therefore grouped together, *i.e.* "S21" and "S22" were grouped into category "S".
- The categorical features were converted into numbers by one-hot encoding. For each original feature, one of the resulting features was removed, to reduce correlation between features.
- The data sets were randomly split into two sets, a training set (80% of all instances) for training models and a test set (20%) for evaluating the models.
- The numerical features of the data set were normalised (min-max scaled), based on the training set's maximum and minimum values. The same values were used to normalise the test set. Normalisation was not necessary when using a Random Forest model.

For the fingerprint data set the same steps were made except for the grouping, which was not needed since the fingerprint data set only contained one analysis type category. The training set of the fingerprint data set was resampled by an oversampling technique, to make the distribution of the analysis length more even; instances with less common analysis lengths were sampled again to be more represented in the data. This could not be done for the case data set, due to lack of computer memory and time. Two example instances of the fingerprint data set after pre-processing and feature selection are visible in Table 7. The training instances had this format when fed to the machine learning models.

Table 7: Example of instances of the fingerprint data set, after filtering, pre-processing, and feature selection. Not all features are shown.

INDEX	FEATURES				TARGET
ID	Time production	...	Priority Yes	Priority No	Analysis length
1	0.80	...	1	0	3
2	0.65	...	0	1	7

3.4.3 Feature selection

It was assumed that not all features of the data sets would be informative for predicting analysis length. To select the most important features, a feature importance analysis was made. This was done both for the case data set and the fingerprint data set, using Python and the package *scikit-learn* (Pedregosa *et al.* 2011).

The filtering and pre-processing steps described in the subsections above were used, but no normalisation of numerical features was needed. A Random Forest model called ExtraTreesRegressor was built and fit to the training set. The model consisted of 250 estimators (*n estimators* = 250). Only 50% of the case data set was used; the whole data set would take too much time. The relative importance for each feature was extracted from the model, the standard deviation was calculated, and the features were ordered based on their impurity-based importance. In addition, a permutation importance analysis based on the test set was made using the function *permutation importance* from *scikit-learn*. The function compares the performance on the test set of a trained model that uses all features with the performance on the test set where a feature column has been permuted. The difference between the performances defines the permutation importance of the feature (Pedregosa *et al.* 2011).

3.5 Training and evaluation of machine learning models

This subsection explains how the training and evaluation of machine learning regression models was made. The Python packages used were *pandas* (McKinney 2010), *numpy* (Harris *et al.* 2020), *joblib* (Joblib Development Team 2020), and *scikit-learn* (Pedregosa *et al.* 2011). Training was done directly after the pre-processing. First, different models were trained and evaluated on the training set using CV. The following models were tested: Linear Regression (with the ordinary least squares method),

the regularised linear regression models Lasso and Ridge, SGD Regressor (linear regression with the stochastic gradient descent method), Decision Tree, Random Forest, Support Vector Regressor (SVR), Multilayer Perceptron, and Linear SVR. The default parameters of the models were used.

The best performing models for the case data set and for the fingerprint data set were selected and fine-tuned. The performance was quite similar for all models, but for both data sets the Random Forest performed a bit better. The two Random Forest models were fine-tuned, first by a Randomised Search using CV on a 50 different combinations of hyperparameters, and then with a following Grid Search using CV based on the results from the Randomised Search. The set of hyperparameters used included *n estimators*, *max depth*, *min samples split*, *min samples leaf*, *max features*, and *bootstrap*. These hyperparameters decide the size of the trees of the Random Forest and also constrain the model. The *bootstrap* parameter decides if different subsamples of the training data set should be used to train the trees or all training data. The best hyperparameter values according to the searches did not increase the performance very much, but these hyperparameter values were still chosen to continue with.

The two models with the selected hyperparameter values were evaluated, both on the training set by CV and on the test set by predicting the test set and calculating the error using RMSE and MAE as error measures. Finally, the two models were evaluated on the small new data sets from May 2020.

4 Results

Here, the results from the different parts of the project are presented. First, the process model from the process discovery part is described, and then the results from the feature importance analysis and the machine learning modelling are presented.

4.1 Process model

The software MPM calculated a process model from the data sets' timestamps, which showed the activities, the average lag time between activities and the different process variants. In the Qlik Sense environment, different subsets of the data set could be selected, and the MPM algorithms recalculated the model in real-time for the selected

subset. Figure 3 shows the three most common process variants for the full case data set in two different views.

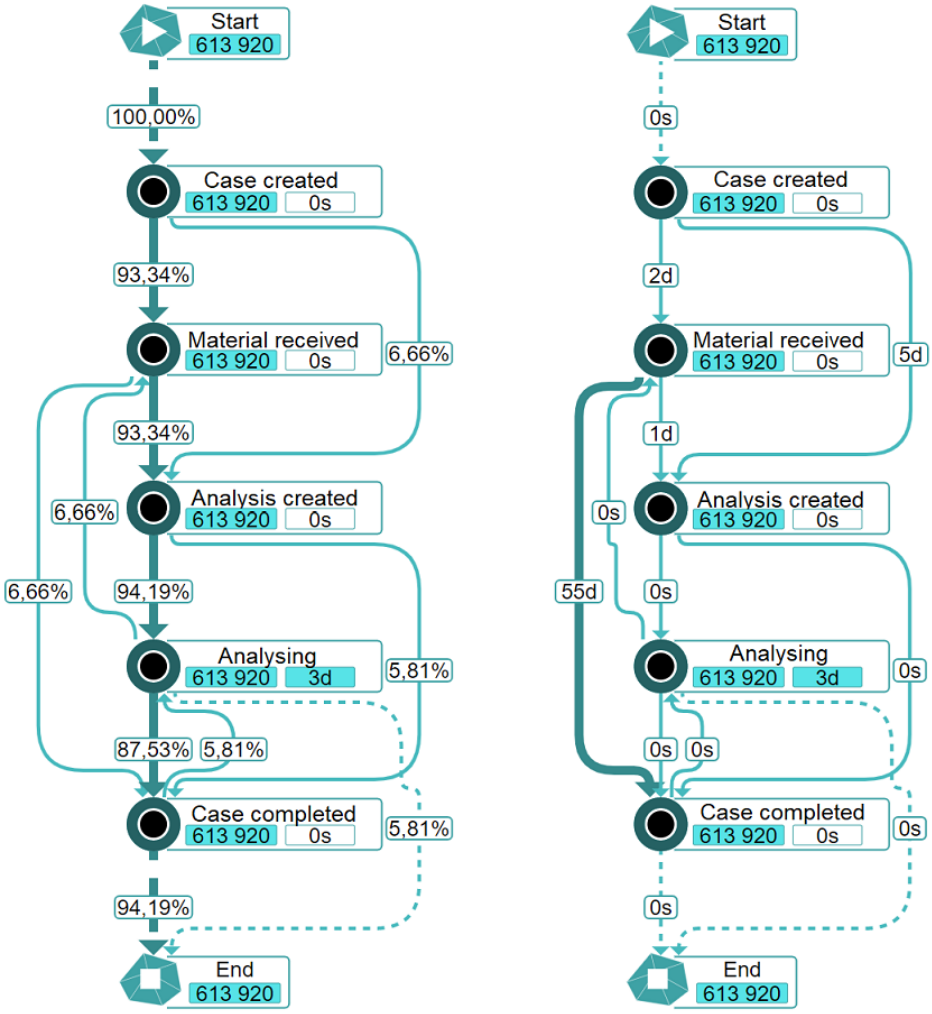

Figure 3: Process model of the three most common process variants, generated by Mehrwerk Process Mining. 95.4% of all analyses (rows) in the case data set follow these process variants. The paths between nodes are called edges. The nodes represent activities, and the numbers at each node are the number of analyses passing that activity and the median time for the activity. Left: the edges are labelled with the fraction of analyses that cover each edge. Right: the edges are labelled with median time between activities. Since all activities except for *Analysing* only have one timestamp, the activity length is 0 seconds. s: seconds, d: days.

Much information could be gathered from different subsets and views of the process model. To summarise the results from the process discovery, the process models showed that there were 25 different process variants discernible in the data, and 83.5% of all

instances followed the main variant. The main variant is the process variant that NFC expects. Some rare process variants are used by only one instance. The number of instances that pass at all activities is not constant (not visible in the view in Figure 3), which means some timestamps are missing in this data.

Of all instances (analyses) in the data set, about 76% belong to cases that consist of only one analysis. The remaining 24% are instances belonging to cases with two or more analyses. When comparing the most common process variants for these two groups of instances (not shown here) it is visible that a higher number of instances pass *Analysing* before *Material received* for the cases with two or more analyses. Another difference between the two groups is that the median time before the analysis is started and after it is completed is longer for the instances belonging to cases with many analyses. This is natural, since these instances can have another analysis being performed during the case, before the case can be completed.

4.2 Feature importance analysis

The feature importance analysis was done to decide which features to use for prediction, but also for understanding what factors that affect the analysis length. It showed that for both data sets, the number of order codes and the number of analyses for the instances were important features. For the fingerprint data set, the features covering occupancy rate of the analysis departments were important, as expected. Full results from the feature importance analysis are found in the Appendix.

4.3 Machine learning models

Evaluation of the Random Forest models trained and fine-tuned on the case data set and on the fingerprint data set is presented in this section. Because of stochasticity in machine learning performances the models may differ between trainings, since they use randomness in learning.

4.3.1 Random Forest model trained on the case data set

First, the fine-tuned Random Forest model's performance was evaluated by using CV on the training data, *i.e.* to iteratively train the model on one part of the data and then evaluate on the other part. This gave a stable estimate of the performance without using the test set. The training set contained 441,958 instances and the test set 110,490

instances. To judge if the model was overfitting or underfitting the data set, a learning curve for the model was calculated when using CV on different sizes of the training set, see Figure 4. To compare with, the learning curve for a SGD Regressor was calculated also. When comparing the training score and the validation score, it becomes clear that the Random Forest model is affected by the number of training examples to learn from. The difference between the training score and validation score decreases when more training data is added, which indicates less overfitting of the model. Adding more data makes the model better at generalising and then it predicts better at the validation data.

Figure 4: Learning curves for a Random Forest model (left) and a SGD Regressor model (right), trained on the case data set using 3-fold cross-validation on the training data for different data set sizes. The red curve is the training error and the green curve the validation error. The Python code for generating the figure was inspired by an example called "Plotting Learning Curves" from the *scikit-learn* package webpage (Pedregosa *et al.* 2011). MAE: mean absolute error.

To understand how large errors the Random Forest made, a histogram of the errors on the test set was created, see Figure 5. Once again the Random Forest was compared with an SGD Regressor. Both error distributions are reasonably good, the errors are centred around zero. The average error by the Random Forest on the test set was –0.03 days, the standard deviation was 7.8 days, and the errors were in the range of –35 and 62 days. 75% of all errors were within ± 4.2 days.

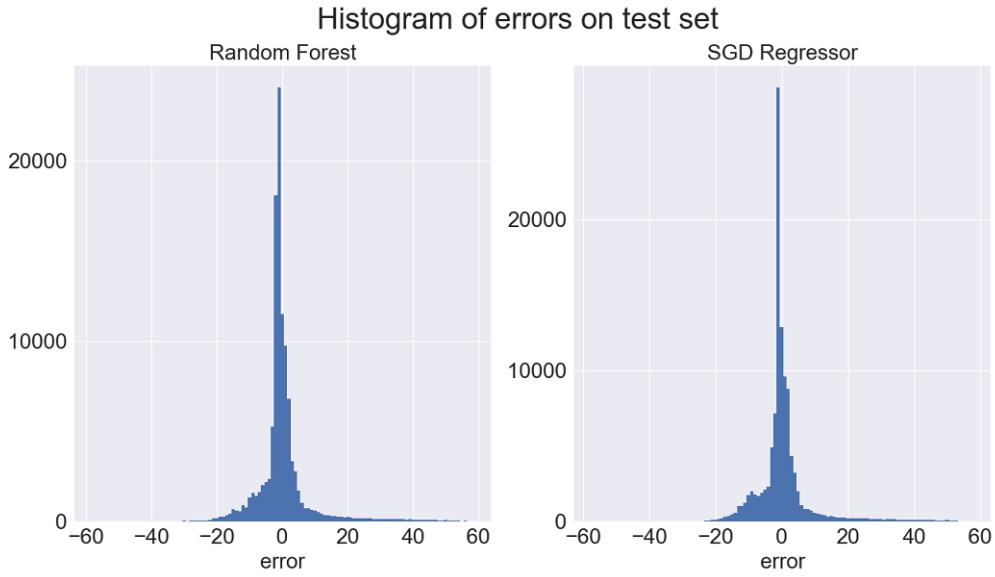

Figure 5: Histograms showing the distribution of the errors made on the test set by Random Forest (left), and SGD Regressor (right). The error is measured in days and calculated as the actual value subtracted with the predicted value.

The Random Forest was then evaluated on the test set. In Table 8, different performance measures are shown. The average RMSE after 3-fold CV shows that the standard deviation is quite low; the performance is stable across different folds. The reason to calculate the performance on both the test set and the training set is to see over- or underfitting. If the training error is much better than the test error, the model is too well fitted to the training data. Here, the model is not overfitting.

Are the RMSE and MAE errors on the test set high or low? As a baseline model to compare with, the mean and median values for each analysis type were calculated on the training set, shown in Table 9. When using these means on the test set, the errors were as follows: RMSE 8.7 days, and MAE 5.0 days. And when using the median values instead, the errors were: RMSE 9.3 days, and MAE 4.4 days. So, the Random Forest model gave somewhat better predictions than using the mean or median values for each analysis type.

Table 8: Performance measures for a Random Forest model trained on case data set. The average root mean square error (RMSE) after 3-fold cross-validation on the training set is shown, together with the RMSE and the mean absolute error (MAE) on the training and test sets.

Training set			Test set	
avg RMSE CV (std)	RMSE	MAE	RMSE	MAE
7.79 (0.014)	7.70	4.20	7.80	4.24

RMSE: root mean square error, MAE: mean absolute error, CV: cross-validation, avg: average, std: standard deviation

Table 9: Calculated mean and median values from the training set for each analysis type.

Type	Mean	Median	Std
A	12	1	18
B	8.9	6	10
D	10	3	13
E	15	9	15
F	21	16	16
G	19	13	16
I	20	14	18
K	6.6	3	9.4
L	6.1	2	9.1

Type	Mean	Median	Std
M	18	12	17
N	4.4	3	4.4
O	25	22	18
P	34	35	16
S	17	12	16
U	6.0	2	9.2
V	16	8	17
X	18	11	18

std: standard deviation

4.3.2 Models trained on the fingerprint data set

A Random Forest model was selected and fine-tuned for this part of the project. The model was then evaluated with CV on the training data and on the test data. The training set contained 3,569 instances and the test set 594 instances. The model was overfitting badly, therefore I chose to compare with other, more simple models. I chose a Linear Regression and a SVR model with a radial basis function (not explained here). The learning curves for different training sizes for the three models are shown in Figure 6. The Random Forest model was overfitting according to the figure, even when using all training data available; the training curve was much better than the validation curve. The model was overfitting despite the regularisation with the chosen hyperparameter values.

With more training data, the model overfits less and the validation score was higher. The SVR model and the Linear Regression model were not overfitting the training data; they were less complex models and would not benefit from more training data.

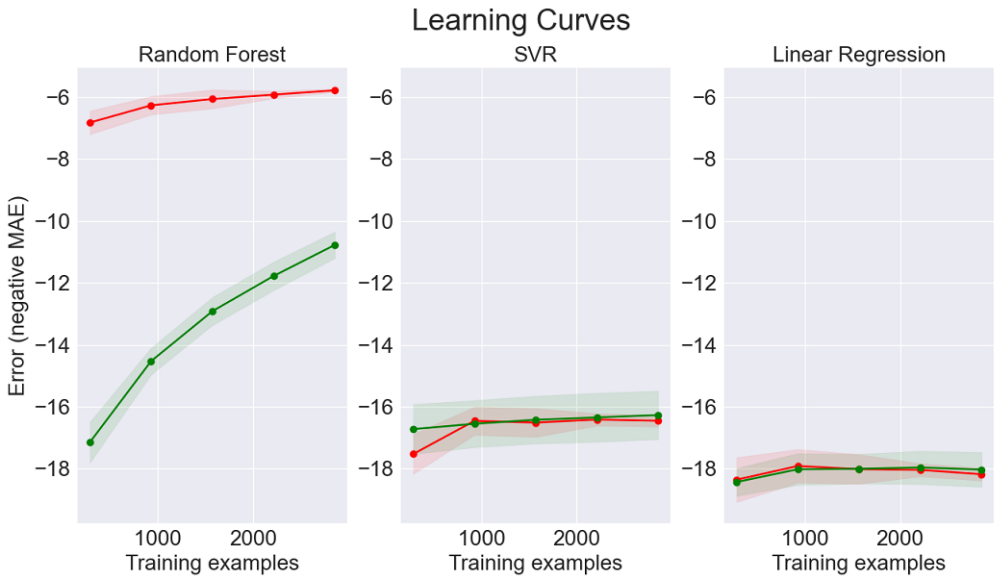

Figure 6: Learning curves for Random Forest (left), SVR (middle), and Linear Regression (right) models trained on the fingerprint data set, for different training sizes using 5-fold cross-validation on the training data. The score is negative mean absolute error (MAE). The green line is the validation curve and the red line the training curve. The Python code for generating the figure was inspired by an example called "Plotting Learning Curves" from the scikit-learn package webpage (Pedregosa *et al.* 2011).

Figure 7 shows that the different models made different errors on the test set. The Random Forest centred its errors closer to zero than the other two models; the SVR and the Linear Regression models often overestimated their predictions. 50% of all errors by the Random Forest are within ± 10 days, and the errors range from –85 to 105 days. The average error is –1.4 days and the standard deviation is 24 days for the Random Forest.

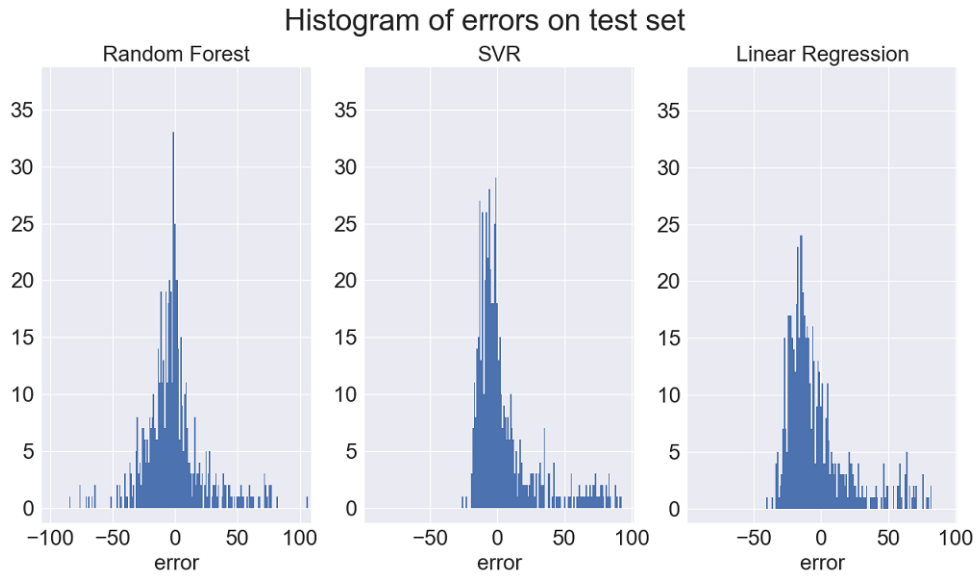

Figure 7: Histograms showing the distribution of the errors made on the test set by Random Forest (left), SVR (middle), and Linear Regression (right). The error is measured in days and calculated as the actual value minus the predicted.

A summary of the performance measures is shown in Table 10. The average RMSE after CV shows that the variation is large between iterations. The Random Forest is overfitting very much, while SVR is overfitting slightly and Linear Regression not at all. SVR has in general lower MAE than Linear Regression, but the opposite is true for RMSE. This indicates that the models make different errors, since RMSE is more sensitive to extreme errors that MAE.

The mean analysis length for S21-analyses is 24.3 days and the median analysis length is 15 days, based on the training data. When using only the mean value to "predict" all instances in the test set, the RMSE was calculated to 25.3 days, and the MAE to 19.7. And with only the median value, the RMSE was 25.8 days and MAE was 16.6 days. This implies that the model predictions are about as good as using the mean and median to estimate the length.

Table 10: Performance measures for a Random Forest model trained on case data set. The average root mean square error (RMSE) after 3-fold cross-validation on the training set is shown, together with the RMSE and the mean absolute error (MAE) on the training and test sets.

Model	Training set			Test set	
	avg RMSE CV (std)	RMSE	MAE	RMSE	MAE
Random Forest	17.5 (2.8)	9.99	5.56	23.5	15.9
SVR	26.3 (5.2)	25.6	16.3	24.6	15.3
Linear Regression	25.0 (2.0)	24.0	18.1	23.9	18.5

RMSE: root mean square error, MAE: mean absolute error, CV: cross-validation, avg: average, std: standard deviation

4.4 Predictions of new data

To see how the trained models performed on new data, a small case data set and a small fingerprint data set with cases from May 2020 were used. The first had 5,972 instances and the second 31 instances. The data was filtered using the same maximum values (cutoffs), normalised using the same maximum and minimum values as when normalising the training set, and one-hot encoded using the categories that were present in the training set.

In Figure 8, 20 predictions on new data made by the Random Forest trained on the case data set are shown. Since the data contained analyses started and completed in May, the lengths were expected to be quite short. The actual values were all between 1 and 10 days in this subset of the new data, but the model predicted all values to around 25 days. The RMSE after predicting the new case data was 16.9 days for the Random Forest model, and MAE was 14.6 days – much higher than on the test set.

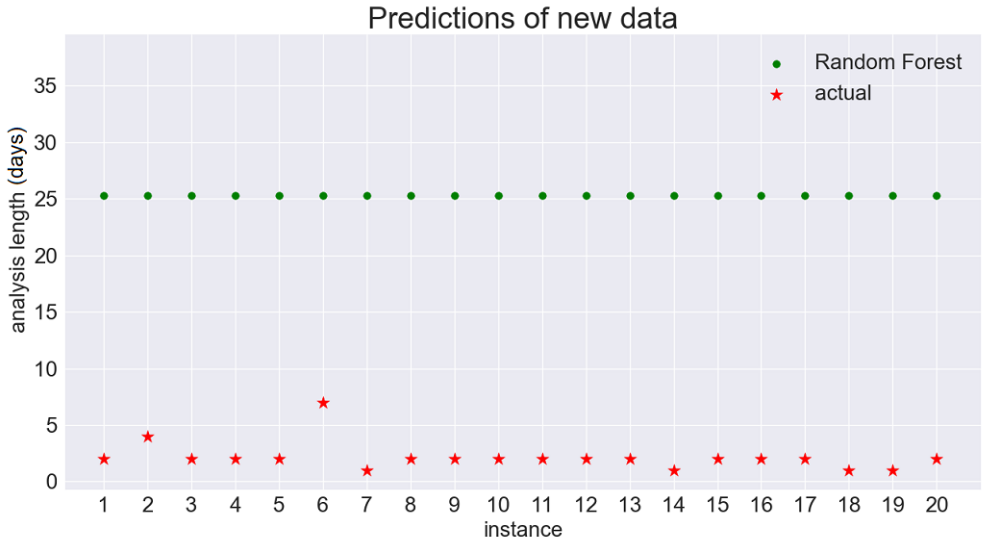

Figure 8: Predictions of the 20 first instances of the case data set from May 2020, made by the Random Forest model.

When the models trained on the fingerprint data set were predicting the fingerprint data set from May 2020, it resulted in the predictions shown in Figure 9. The actual values are between 1 and 10 for all those instances, but the models have predicted much higher values. Random Forest and Linear Regression predicts higher values than the SVR model, but in general all three models predict badly.

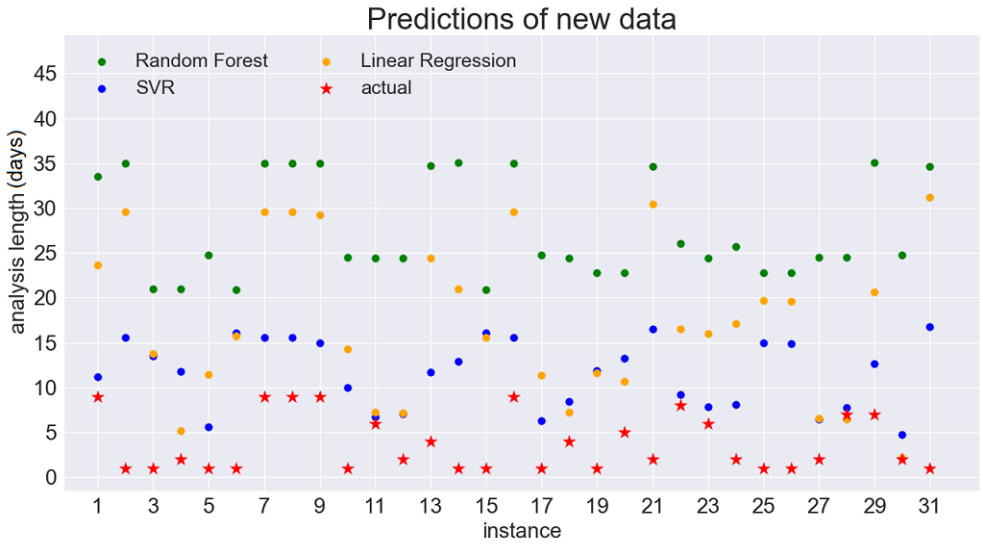

Figure 9: Predictions of the small fingerprint data set from May 2020, with 31 instances, by the three models Random Forest, SVR, and Linear Regression.

5 Discussion and conclusions

In this section I discuss the interpretation and usefulness of the results, the decisions made and alternative methods, and at last, I draw conclusions from this project and discuss the future work in this field.

5.1 Interpretation and usefulness of results

The project goals were to visualise the process using process mining and to see if a machine learning model could predict analysis length. Conclusions from the process mining part of the project are that the data set was in a good format for MPM and that Qlik Sense and MPM were quite easy to use. The MPM model was easy to adjust and showed interesting process variants. I expect this to be very useful for NFC, but it will take time to interpret and fully understand the different variants. An advantage with the MPM model is that it is easy to choose different subsets of the data, and NFC can for example choose to look only at short cases (the most common ones), only at cases longer than desired, or only at certain order codes or analysis categories. In general, I think that

process discovery tools like MPM can be very helpful for organisations with process-related data. Another conclusion is that it may be difficult to understand what are true variants and what are errors and bad data quality; rare process variants that only one or a few cases follow could occur due to human-made mistakes in the data registration, for data where the registration is done manually.

In several of the process variants there is a backflow – the instances are not passing the activities in the expected order. But it is important to know that when for example instances go to *Analysis created* and *Analysing* before *Material received* (as for 6.66% in Figure 3, it may be due to the fact that *Material received* is the timestamp for the *last* material that has been received. The analysis can start before all materials have reached NFC. Some instances go in ways that are impossible in NFC's workflow, for example to *Case completed* before *Analysing* for 5.81% of the instances in Figure 3. The reason for this is unknown, but some of the confusing paths could probably be explained by human mistakes in the manual registration of timestamps.

My hypothesis for the machine learning part was that the case data set would not have enough information or patterns that could describe the analysis length, but that the fingerprint data set with the additional features would have. When summarising the machine learning results, it is clear that the performances of the models were not good, neither on the case data set or on the fingerprint data set. The errors were high, both on the test set and on the new data. For the case data set, the data was probably enough in quantity, but not informative enough. The features did not hold information that the target (analysis length) depended on, at least not to the extent that a model could predict the target. But according to the feature importance analysis, the features did affect the analysis length to some extent. The fingerprint data set, on the other hand, might have had informative features to use, but the data set was too small for a machine learning model to learn from how to predict well. By studying the learning curves it was seen that the Random Forest was overfitting, but if there would have been more data to train on, the model could probably have performed much better. If more data would have been available for the fingerprint data set, the performance on the test set could possibly have been as good as the performance on the training set. The other two models, SVR and Linear Regression, did not overfit the data but they did not predict well either.

The predictions of the new data sets were worse than the predictions of the test data. When examining the predictions of the test sets manually, it is clear that the models predict varying, some high and some low predictions. But the predictions on the new data sets were almost all in the same range. A reason to this could be that the new data sets were quite homogeneous and had little variance, so that the Random Forest models predicted many instances to have the same length (they ended in the same leafs of the

trees). Another reason to the bad predictions of the new data sets could be that the ”old” data was not representative for the new data.

How accurate must results be to be useful for an organisation? It depends on the application. In most prediction tasks the error should be very small, but for NFC, an error margin of for example 2-3 days would probably be okay if the purpose would be to discern short analyses from long analyses. To implement a model in the daily work at NFC or another organisation should be possible. Every time a new case with one or several analyses is ordered by a client, the user could input the values of analysis type, case type, priority, and other values. Then the model would output the expected number of days the analysis will take, with about 1-2 days margin of error. The client would know from the start if the analysis will be fast or slow, and the client could ask the organisation to put in more resources to speed up the processing. Of course, the whole case could take longer time than the analysis itself (or analyses).

Are the results reliable? Both the MPM model and the machine learning models are as reliable as the data is. Both are very dependent on the data quality and the degree of how representative the training data is. Machine learning models are almost always evaluated on a test set of randomly selected instances from the data set, before model training. So the test set characteristics and distributions are approximately the same as for the training data, but will new data have the same properties? The same range of values and the same patterns? This is important for all organisations to consider when developing models. The models trained can only be useful in the future if the new data produced by the organisation still looks the same as the old data. Else, new models should be trained on the new data or at least updated with new training data once in a while. NFC has changed a bit during the years from 2015, which makes the older part of the data less representative than the newer part. It could be interesting to remove the older part and see if different results were achieved.

5.2 Choices made and alternative methods

There are several decisions made in the project that could have been made differently. For example, why was the analysis length predicted and not the whole case length? The first intention was to predict the case length, but when examining the case length and the analysis length it seemed too difficult to predict the case length based on this data set. As we know, the majority of all cases consist of only one analysis, and if using only these one-analysis-cases the case lengths should be similar or a bit longer to the analysis lengths. But sometimes the case lengths were much longer, for an unknown reason. Sometimes the cases took very long time, despite that the analyses were completed. If

there were several analyses and they were performed in serial order, it should take long for a case to be completed. But it took long time even for cases with only one analysis. So I assumed that the case length depended on some factors that the data set did not have information about.

Why was oversampling done of the minority lengths of the fingerprint data set, but not undersampling of the majority length? Resampling of a data set is made to change the distribution of the classes or target values and thereby increasing the model's possibility to learn more from the minority classes or the rare target values. The aim is that the model should become almost as good at predicting rare cases as common cases. But the test set is normally left unchanged, since it should reflect the distribution of new data. In this project, undersampling of the majority class did not improve the performance of the model and did not seem to be necessary. Resampling methods could have been useful for the case data set as well, but the computer memory was not sufficient for that.

Cutoff values were used for filtering out the 5% longest analysis lengths, numbers of analyses and numbers of order codes. The reason for this, as mentioned earlier, is that it was assumed be more valuable for NFC to be able to predict the analysis length accurately for common cases than less accurately for all cases, and to get a good representation of the more common cases. Normally in a machine learning project data should not be removed if there is not a very good reason. The cutoff values are very important for the accuracy in the results. With a hard cutoff, the errors might become lower, but the model cannot be used on all cases. So, depending on how an organisation wants to use the predictions, they could choose to remove much or little of the less common data. I tried some different cutoffs, and with a high cutoff (using only common (short) analysis lengths) the size of the errors become much smaller but the error percentage is not much lower, and the predictions become less useful for NFC. To predict if an analysis will take 2, 3 or 6 days is maybe not so interesting, but it is more valuable to say whether an analysis will take 10 or 20 days. But the analyses that are longer than 10 days are few. When using 80% cutoffs for the fingerprint data set predictions (up to 40 days long analyses), the test set prediction errors are still around 10 days.

A change that could have improved the predictions is to make the grouping of analysis types differently. The analyses that belong to the same group, *i.e.* have the same first letter, are not the same analyses. Almost all types are ”K” (42%) or ”N”(39%). I could have chosen to use only these two type and divided them into other, more specific groups.

When using one-hot encoding on the categorical features, for each original feature one of the new features created was removed from the data set (called dummy encoding), since that feature was dependent on the values of the other new features. If a feature with five categories is encoded into five new features, then if four of the new features have the

value 0, the fifth new feature must have the value 1. This effect is called multicollinearity, *i.e.* correlation between one feature and the other features, which is undesirable. On the other hand, the model could probably be affected by which feature that is removed. One-hot encoding is not the only method to deal with categorical variables, but in this project it suited the purpose of dividing categories without defining an order (*e.g.* red $>$ blue $>$ green).

In this project the only process mining software used was MPM, but there are many others. Probably, other process models could have been achieved by other methods. In the same theme – more different machine learning models, pre-processing steps, and resampling techniques could have been tried. The purpose of this project was not to come up with the very best solution, but to show the possibilities and usefulness in these tools on this type of data and problems.

5.3 Future work and conclusions

In this project, I have investigated if an organisation's case administration data can be used by process mining and machine learning, with the aim to improve the understanding and the efficiency of the organisation's work. And indeed, it is possible, but there are challenges. This conclusion is supported by related work made in this research field. There are many things to consider and many choices to make in a project like this. But I still think that process mining and machine learning are two great tools for analysing administration process data sets.

Time predictions and process discovery are likely to be helpful for other organisations as well, for other authorities in Sweden or organisations with similar process data to NFC's. MPM is definitely well adapted to use at other processes, while machine learning models need to be adjusted to fit new organisations and new tasks. The requirement is high quality data with informative features for machine learning, or data with timestamps of the activities for process mining. To record as much data as possible increases the chance of success.

For future machine learning predictions for NFC, it might be enough information with the existing features (including the employee occupancy rate and analysis queue) to predict the analysis length accurately. If NFC would like to use machine learning further, occupancy rate and work load of the organisation at the departments should be recorded for all types of analyses, not just for fingerprinting. It could also be good to add more timestamps and data parameters if possible, and to infer logical controls in the registration system to ensure high quality of the data. Generally, more data is better. Of course,

too many features can decrease the performance of a model, but there is always the possibility to use feature selection and reduce the number of features. Before beginning a machine learning project, it can be good to check if the data set seem promising; to check if there are any visible correlations between features and the target. Does it seem like the features could tell something about the target? In addition, a feature importance analysis can be run, to see if any features are important for predicting.

There are many other machine learning applications that would be interesting to try on a process administration data set like this, for example anomaly detection. A model could learn from high quality training data what the process should look like, and then detect in real-time deviations and give notifications to the user when a case is stalling in an activity abnormally long. So that the user can act and maybe add more resources to that case.

This project has been very interesting for me and hopefully for NFC and Drake Analytics too. It has shown that for an organisation with this type of data, machine learning and process mining is definitely worth to try and could give valuable insights in the process. But this project was only a small study; it takes more knowledge about the process at NFC, and more time developing and testing, to create a good prediction model that can be integrated in NFC's systems.

6 Acknowledgements

I would like to thank my supervisor Helene van Ettinger-Veenstra – you are a great supervisor with a lot of valuable experience from machine learning projects and it has been interesting to hear your thoughts. Thank you Ralf Planting, for believing in me when I first took contact and for helping me finding a project, and thank you and the others at Drake Analytics for being very supportive and for taking your time to help me with Qlik Sense and MPM.

Thank you Emil Hjalmarson, Mikael Högfors, Camilla Rehme, Anna Stenfeldt, and Desirée Eriksson at NFC for giving me this project, for trusting and helping me and for taking your time to answer my questions. It has been very interesting to work with your data.

Thank you my subject reader Jan Komorowski for reviewing my report, and thank you, course coordinator Lena Henriksson and examiner Pascal Milesi, for all your help.

References

Brugman, S. 2020. Pandas-profiling. WWW document: https://pandas-profiling.github.io/pandas-profiling/docs/master/rtd/pages/announcements.html. Accessed 6 October 2020

Géron A. 2017. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, Incorporated, Sebastopol, United states

Gruson D, Helleputte T, Rousseau P, Gruson D. 2019. Data science, artificial intelligence, and machine learning: Opportunities for laboratory medicine and the value of positive regulation. Clinical Biochemistry 69: 1–7

Hunter JD. 2007. Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering 9: 90-95

Joblib Development Team. 2020. Joblib: running Python functions as pipeline jobs. WWW document: https://joblib.readthedocs.io. Accessed 6 October 2020

McKinney, W. 2010. Data structures for statistical computing in python. In: Proceedings of the 9th Python in Science Conference, Volume 445, 2010

Larson D, Chang V. 2016. A review and future direction of agile, business intelligence, analytics and data science. International Journal of Information Management 36: 700–710

Lemaitre G, Nogueira F, Aridas CK. 2017. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. Journal of Machine Learning Research 18: 1-5

McKinney W. 2010. Data Structures for Statistical Computing in Python. WWW document: https://conference.scipy.org/proceedings/scipy2010/pdfs/mckinney.pdf. Accessed 6 October 2020

Mehrwerk GmbH. Mehrwerk Process Mining. WWW document: https://mpm-processmining.com/en

Meyer J, Reimold J, Wehmschulte C. An Introduction to MPM - MEHRWERK ProcessMining. WWW document: http://ceur-ws.org/Vol-2374/paper6.pdf. Accessed 7 October 2020

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12: 2825-2830

Qlik Tech International. Data Analytics & Data Integration Solutions. WWW document: https://www.qlik.com/us. Accessed 6 October 2020

Tax N, Verenich I, Rosa ML, Dumas M. 2017. Predictive business process monitoring with LSTM neural networks. Advanced Information Systems Engineering : 29th International Conference, CAiSE 2017, Essen Germany, June 12-16, 2017. Proceedings, pp. 477–492. Springer

van der Aalst W. 2016. Process Mining. 2nd ed. Springer, Berlin Heidelberg

Vartiainen, T. 2017. Analyzing event logs to discover process models and to detect anomalies in real-time. Thesis for the degree of Master of Science in Technology, Aalto University

Appendix

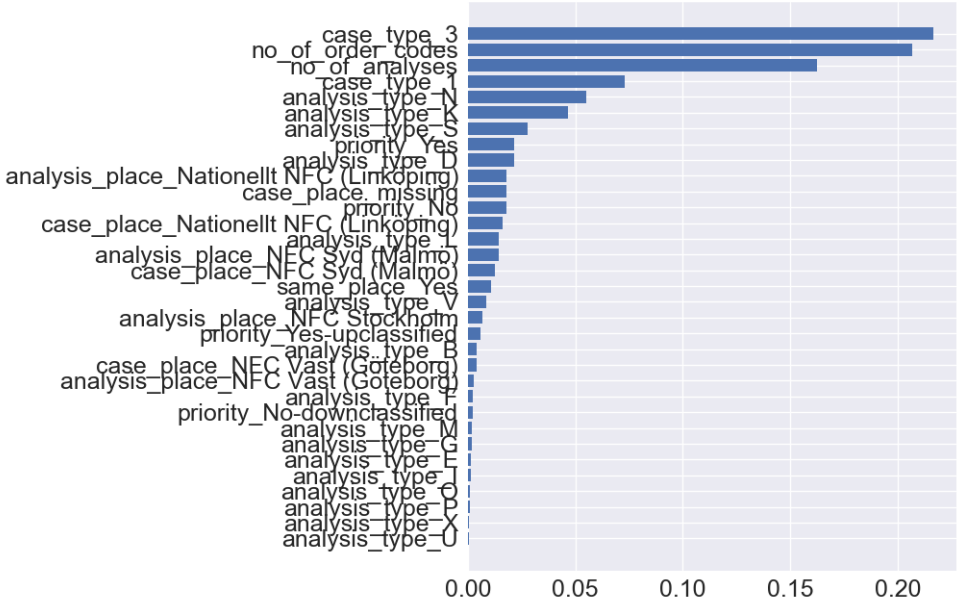

Figure A1: Feature importances of the features in the case data set, based on mean decrease in impurity. The Python code for generating the figure was inspired by an example called "Permutation Importance vs Random Forest Feature Importance (MDI)" from the *scikit-learn* package webpage (Pedregosa *et al.* 2011).

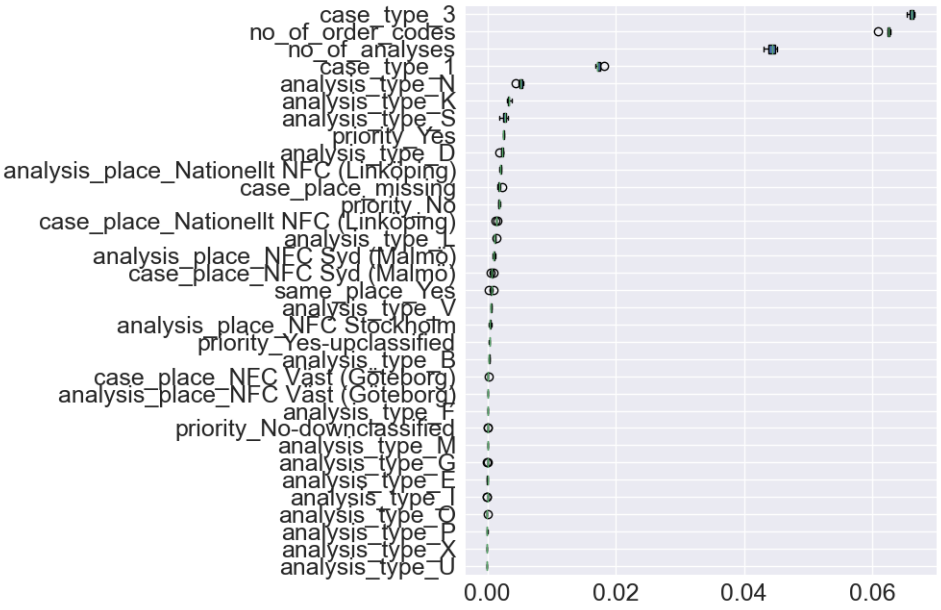

Figure A2: Permutation importances of the features in the case data set based on the test set. The Python code for generating the figure was inspired by an example called "Permutation Importance vs Random Forest Feature Importance (MDI)" from the *scikit-learn* package webpage (Pedregosa *et al.* 2011).

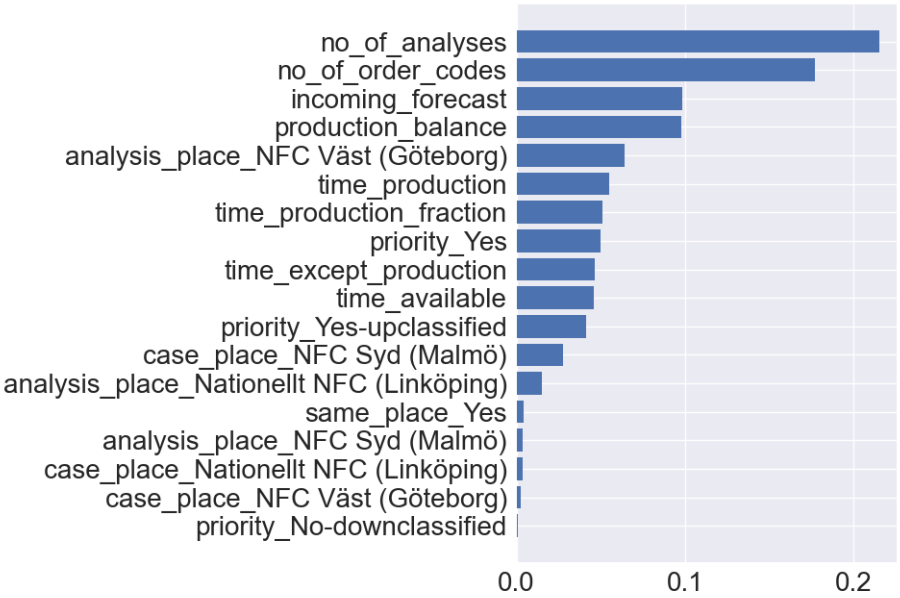

Figure A3: Feature importances of the features in the fingerprint data set, based on mean decrease in impurity. The Python code for generating the figure was inspired by an example called "Permutation Importance vs Random Forest Feature Importance (MDI)" from the *scikit-learn* package webpage (Pedregosa *et al.* 2011).

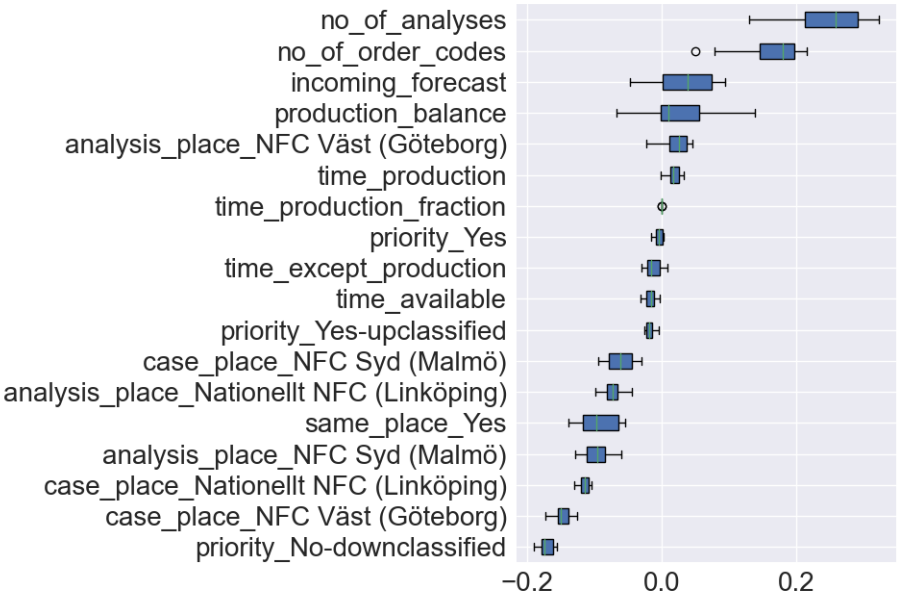

Figure A4: Permutation importances of the features in the fingerprint data set based on the test set. The Python code for generating the figure was inspired by an example called "Permutation Importance vs Random Forest Feature Importance (MDI)" from the *scikit-learn* package webpage (Pedregosa *et al.* 2011).