UNIVERSITY OF OULU

FACULTY OF TECHNOLOGY

SUPERVISED NEAREST NEIGHBOR SEARCH IN FINDING SIMILAR OPERATING CONDITIONS – CASE: MINERAL PROCESSING

Juho Anttila

DEGREE PROGRAMME OF PROCESS ENGINEERING

Master’s thesis

April 2022

ABSTRACT

Supervised nearest neighbor search in finding similar operating conditions – case: mineral processing

Juho Anttila

University of Oulu, Degree Programme of Process Engineering

Master's thesis 2022, 57 pp. + 13 Appendices

Supervisors at the university: Jari Ruuska, Markku Ohenoja and Antti Koistinen

The main objective in this thesis was to identify and evaluate the applicability of machine learning methods that could be used in mineral processing as a tool for forecasting the direction in which where the current process values are going. The supervised machine learning method k-nearest neighbors (kNN) was selected for finding the closest correspondences from history data for the current process conditions.

The literature review in this thesis describes mineral processing plant processes, mineral separation using flotation, mine-to-mill optimization, the CRISP-DM (Cross-industry Standard Process for Data Mining) procedure for data handling and machine learning methods in general. In addition, a few case studies on the usage of machine learning in mineral processing are presented. The experimental part of this thesis concentrates on data pre-processing and the development of the k-nearest neighbors function. MATLAB® was used for all the calculations and results presented in this thesis.

The kNN algorithm presented in this study proved to be sufficient in finding the closest correspondence from history data for a current process conditions. At the time of completing this thesis, the kNN function was in a state where it could be used as part of a mine-to-mill expert system to extract a prediction of trajectories for the process's key performance indicators. However, the function could still be improved by the addition of clustering, such as k-means, as a preliminary classification for kNN and ranking of nearest neighbors based on the area between the trajectories of neighbors and the query point.

Keywords: Mineral processing, k-nearest neighbors, froth flotation

TIIVISTELMÄ

Samankaltaisten prosessitilojen etsiminen rikastusprosessissa lähimmän naapurin menetelmällä

Juho Anttila

Oulun yliopisto, Prosessitekniikan tutkinto-ohjelma

Diplomityö 2022, 57 s. + 13 liitettä

Työn ohjaajat yliopistolla: Jari Ruuska, Markku Ohenoja ja Antti Koistinen

Työn päätavoitteena oli sellaisten koneoppimismenetelmien soveltuvuuden tunnistaminen ja arviointi, joita voitaisiin käyttää työkaluna rikastusprosessissa prosessin tilan ennusteissa. Ohjattua koneoppimista edustava k:n lähimmän naapurin menetelmä (kNN) valittiin keinoksi etsiä historiadatasta lähimmät vastaavuudet prosessin nykytilanteelle.

Työn kirjallisuuskatsaus sisältää esittelyn rikastusprosessin yksikköprosesseihin, mineraalien erottamiseen vaahdotuksen avulla, mine-to-mill optimointiin, datan käsittelyyn suunniteltuun toimintamalliin (CRISP-DM) sekä koneoppimisen peruskäsitteisiin. Kirjallisuuskatsauksessa esitellään myös muutama tapaustutkimus koneoppimisen käyttämisestä rikastusprosessissa. Työn kokeellinen osa keskittyy datan esikäsittelyyn sekä k:n lähimmän naapurin menetelmän sovittamiseen työssä käytettävälle datasetille. Esikäsittely sekä työssä esitetyt tulokset tehtiin MATLAB® ohjelmiston avulla.

Työssä esitelty k:n lähimmän naapurin algoritmi todettiin sopivaksi menetelmäksi löytää historiadatasta lähimmät vastaavuudet nykyiselle prosessitilalle. Työn valmistumishetkellä funktio oli siinä tilassa, että sitä voitaisiin käyttää osana mine-to-mill optimointiasiantuntijajärjestelmää prosessin tärkeimpien suorituskyvystä kertovien muuttujien (KPI) arvojen trajektorien ennustamisessa. Funktiota voitaisiin kehittää lisäämällä klusterointi, esimerkiksi k-means esiluokitteluksi k:n lähimmän naapurin menetelmälle, sekä järjestämällä menetelmän löytämät naapurit paremmuusjärjestykseen testipisteen ja niiden välisen pinta-alan perusteella.

Asiasanat: Rikastusprosessi, k-nearest neighbors, Vaahdotus

FOREWORD

This thesis was done in cooperation between Metso Outotec and the University of Oulu, and took place between October 2021 and April 2022. The objective of this research was formed from an intriguing idea to create a "weather forecast" model for current process conditions.

I thank docent Jari Ruuska, postdoctoral researcher Markku Ohenoja and doctoral researcher Antti Koistinen from the University of Oulu for supervising the thesis. I also thank Antti Remes, Jani Kaartinen and others from Metso Outotec for the topic of the thesis, and giving me an opportunity to work with real plant data. Finally, I thank professor Mika Ruusunen from the University of Oulu for guidance and comments on the thesis.

Oulu, 14.4.2022

Juho Anttila

Juho Anttila

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

AG	autogenous mill
AI	artificial intelligence
CNN	convolutional neural network
kNN	k-nearest neighbors
KPI	key performance indicator
ML	machine learning
MWD	monitoring while drilling
PSI	particle size indicator
RNN	recurrent neural networks
SAG	semi-autogenous mill
SVM	support vector machines

1 INTRODUCTION

Mining and mineral processing can be thought as a set of interconnected processes, where each process stage has an impact on later process stages. Roughly speaking, these processes can be divided into drilling and blasting, crushing, grinding and mineral separation. When the objective is to optimise the performance of the whole process, the optimisation of each process stage individually can lead to suboptimal results, since changes made in one process stage may influence downstream processes. For example, this can lead to growth in expenses in the form of an increase in the raw materials used. For this reason, it is essential to understand the effects of each process stage on the final product. One possible way to gain knowledge of these effects is to study measurement data collected from the different processing stages of a mineral processing plant.

In this thesis, the tools for gaining such information from process data were investigated and applied. A general description of a mineral processing plant for gold flotation was given. The calculations presented were done using real plant data, which was acquired through Metso Outotec from a mineral processing plant. The key objective in this study was to find the best correspondences from history data for the current process conditions by using machine learning (ML) methods. The aim was to make predictions about the process key performance indicator (KPI) trajectories and examine whether the built model could be used as part of an expert system. However, development of this kind of model or expert system was beyond the scope of this thesis.

The literature work in this study includes an introduction to the mineral processing process stages, mine-to-mill concept, data-based modelling and machine learning. For data handling, an open standard CRISP-DM framework was applied. Data preparation and all calculations with the selected ML method were done in the MATLAB® environment, and some MATLAB® functions needed for the built model required MATLAB® statistics and machine learning toolbox. Implementation of the chosen ML method required programming of multiple functions and scripts in MATLAB®, which are described in the experimental part of this thesis. The applicability of the selected machine learning method is demonstrated and discussed in the results and discussion chapters.

2 THEORY

2.1 Process description

A mineral processing plant includes several interconnected processes: crushing and grinding, which together make up comminution, and mineral separation. Feed for crushing comes from drilling and blasting processes, which are done at the mining site to remove ores from the bedrock. The most versatile mineral separation process is froth flotation, after which the product and tailings go to a dewatering process. Figure 1 illustrates a typical flowsheet of a mineral processing plant from crushing to flotation products.

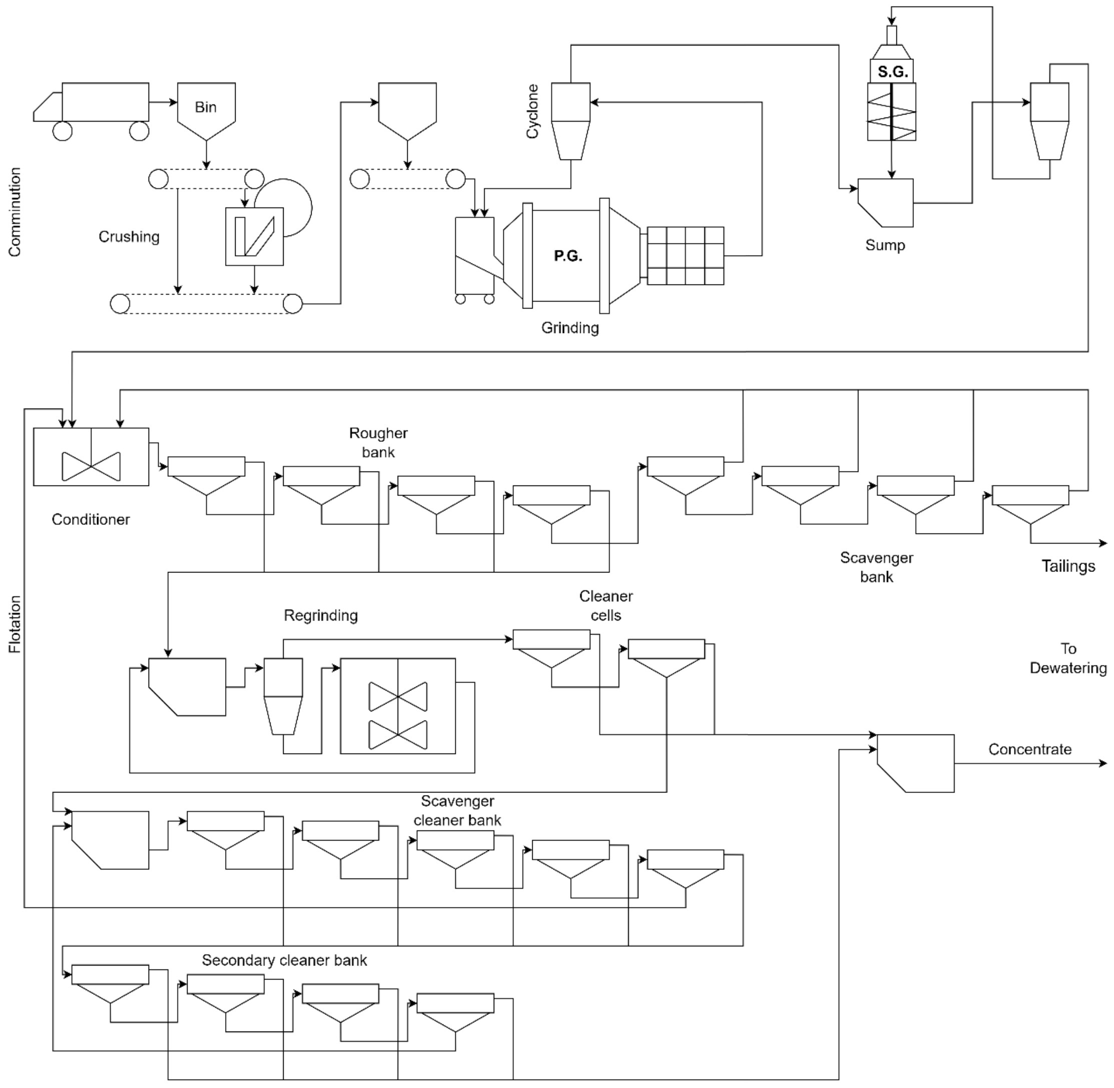

Figure 1. Typical mineral processing flowsheet.

Comminution processes include crushing and grinding. In comminution, the particle size of the ore is reduced step by step to enable separation of valuable particles in the downstream processes. The feed for crushing comes from drilling and blasting by truck and it is fed into a bin before a crusher. The crusher reduces the particle size of the ore to such a level that the mineral and gangue can be separated as particles in grinding. Crushing can be performed with different equipment, such as jaw, gyratory, cone or impact crushers. (Wills 2006)

After crushing, the ore goes to primary grinding (marked P.G. in Figure 1), where usually water is also introduced for reasons of lower power consumption per metric ton of product, elimination of dust and higher capacity per unit mill volume. Primary grinding is typically done using a semi-autogenous grinding (SAG) mill, which utilizes steel balls as grinding media. An autogenous mill (AG) is a tumbling mill, which uses the ore itself as grinding media. In the flowsheet, a SAG mill is assumed. The principal purpose of grinding is to achieve the correct degree of liberation in the ore. Liberation refers to the separation of the valuable minerals from the attached gangue minerals at the coarsest possible particle size. The product from the SAG goes to classification by a hydrocyclone or cyclone. A cyclone operates utilizing a centrifugal action, and it separates fine particles and liquids (overflow) from coarse particles (underflow). The underflow of a cyclone, which is also known as a circulation load, goes back to the SAG, whereas the overflow continues to secondary grinding (S.G. in Figure 1). (Wills 2006)

In secondary grinding, the overflow from the SAG cyclone goes first to a sump, from which it continues to a secondary grinding cyclone. As in primary grinding, the underflow from a cyclone goes back to grinding, and the overflow continues to the next process stage, which is flotation. In the flowsheet, the secondary grinding is assumed to be done using a tower mill, which is more effective in fine and ultra-fine grinding than AG or SAG mills. In a tower mill, steel balls or pebbles are present in a vertical grinding chamber, where an internal screw flight provides medium agitation. The feed enters the tower mill from the top, together with water, and the particle size of the ore is reduced by abrasion and attrition while descending to the bottom of mill. The fine ground particles are carried upward by pumped water and enter a built-in classifier. Coarse particles are circulated back to the bottom of the mill, while the fine particles in the classifier return to a secondary grinding sump as an underflow. (Wills 2006)

The first step in a flotation circuit is agitation, which is performed in a conditioner. In a conditioner, reagents are added to the slurry, and a feasible mixing time is ensured to allow mineral particles to react with chemicals which change the surface properties of selected minerals and allow them to react with air bubbles in flotation cells. The flotation circuit shown in Figure 1 has five flotation banks, all of which have the same working principle. The feed enters the first cell of the bank and some of the valuable minerals are collected in the froth as the overflow. The underflow goes to the second cell in the bank, and more valuable minerals are collected, until the last cell in the bank. From the last cell in the bank, the concentrate continues to a different stage in the flotation circuit than the tailings (underflow). The flotation circuit in Figure 1 includes a rougher bank, a scavenger bank, the first cleaner, a scavenger cleaner and a secondary cleaner. In addition to this, the circuit has a regrinding module. (Wills 2006)

The initial flotation takes place in the rougher bank, where the objective is to collect as much of the valuable minerals as possible. The concentrate from the rougher can contain reasonably coarse particle sizes, which is why regrinding is placed after the rougher bank. From the rougher bank, the concentrate continues to regrinding, and the tailings are directed to the scavenger bank. In the scavenger bank, the aim is to achieve maximum recovery of the valuable mineral particles to minimize losses to tailings. From the scavenger bank, the tailings leave the flotation circuit and enter dewatering, and the concentrate is routed back to the conditioner. The concentrate from the rougher bank goes to regrinding, where the middlings are treated. Middlings are products that are neither concentrate nor tailings. The rougher concentrate may still contain coarse particles, which consist mainly of unliberated valuable minerals, and the objective of regrinding is to reduce the particle size of the feed to achieve better liberation. In regrinding, the fine particles are separated from the coarse particles in a cyclone, from where the underflow goes to the actual regrinding, and the overflow continues to the first cleaner. In the cleaner banks, the level of slurry is kept low to achieve a deep froth and produce a high-grade concentrate. From the first cleaner, the concentrate is routed to the final concentrate, and the tailings enter the scavenger cleaner. The concentrate from the scavenger cleaner enters the secondary cleaner bank, and the tailings are directed back to the conditioner. The concentrate from secondary cleaning enters the final concentrate of the circuit, and the tailings are directed back to the scavenger cleaner bank. (Wills 2006)

The tailings from the scavenger bank and the final concentrate from the cleaner banks leave the flotation circuit and continue to individual dewatering stages. As the name implies, most of the liquids are separated from the solids in dewatering to produce a relatively dry concentrate as the final product of the whole process and the final by-product from tailings. (Wills 2006)

2.2 Mineral separation using froth flotation

Froth flotation is a versatile and important concentration method, which can be applied to many ores and particle sizes by selecting the correct chemicals for flotation. At the beginning of flotation, mineral slurry is formed by grinding the ore with water. Selected flotation chemicals are then added to the slurry according to the ore being processed. The flotation process is based on the differences in the ability of air bubbles to attach selectively to specific mineral surfaces in a slurry. The attached particles are carried to the surface froth phase and separated, while other particles stay in the liquid phase. Valuable minerals are usually attached to the bubbles while gangue falls to the bottom of the flotation cell. If the gangue attaches to bubbles and the valuable mineral falls to bottom, the process is called reverse flotation. (Haavisto 2009)

There are a few different flotation mechanisms, namely bubble attachment, entrainment and aggregation. Valuable minerals are usually recovered using bubble attachment, as described above, while mechanical entrainment and aggregation are mainly used to recover gangue from slurry. In entrainment, fine particles become trapped in a film between the bubbles, where they may be recovered into the concentrate or stay in the slurry. Entrainment is a non-selective mechanism and it happens with all minerals. Rinsing the froth phase with water can decrease recovery by entrainment, but it also has an impact on the stability of the froth. In aggregation, physical entrapment occurs between the particles in the froth attached to the air bubbles. (Wills 2006; Kortelainen 2019)

Grade and recovery are the two most important key performance indicators (KPIs) of flotation circuits. Grade is used to describe the weight percentage of the valuable mineral in the selected stream, for example concentrate. Recovery describes the proportion of the mineral collected from the feed to the concentrate and is given according to Equation (1):

$$R = 100\frac{c(f-t)}{f(c-t)}, \tag{1}$$

where *R* is recovery [%], *c* is the final concentrate grade [%], *f* is the flotation feed grade [%] and *t* is the grade in the tailings that leave the flotation circuit [%]. Grade and recovery can be seen as inversely proportional, since when recovery is increased, the grade decreases. This is due to the fact that, when more valuable minerals carried in the concentrate, the amount of gangue in the concentrate also tends to increase, which leads to a decrease in percentage of the valuable minerals in the concentrate. The main goal of flotation control is to maintain a high grade without loss in recovery. (Kortelainen 2019)

The flotation process is comprised of three interrelated main areas: These main areas are chemical components, operation components and equipment components. Changes in one area will have an effect in other areas. The chemical components include collectors, frothers, activators, depressants and pH modifiers. Operation components are for example feed rate, mineralogy, particle size, slurry density and temperature. Equipment components include cell design, agitation, air flow, cell bank configuration and cell bank control. (Kawatra 2011)

The role of collectors is to make selected minerals hydrophobic to enable flotation. Collectors, which are surfactants, are added to the slurry and are absorbed during a conditioning period. Collectors can be ionizing compounds, which split into ions in water, or non-ionizing compounds, which are indissoluble and make the mineral hydrophobic by covering its surface with a film. (Wills 2006)

Frothers are used to form and stabilize the froth layer. Frothers may mix with water, but some of them do not. Frothers that mix poorly with water are usually added in the grinding circuit, so that agitation can promote dispersion into the slurry. Frothers that are soluble with water are usually added along the flotation process to maintain a stable froth phase and reduce consumption of the frothers themselves. (Crozier 1992)

Modifiers, depressants and activators are added to the process in order to balance the flotation environment by modifying the action of the collector, either by enhancing or reducing its hydrophobic properties on the mineral surface. They can be used to make the collector more selective towards certain minerals. For example, lime can be used in flotation to increase the alkalinity of the slurry, which increases the volume of the froth. (Wills 2006)

The flotation cell is the basic unit in the froth flotation process. It is a vessel which has one inflow and two outflows, the inflow being slurry, and the outflows being concentrate froth and tailings. The cell has an impeller which is used both to introduce the air to the system and to mix the slurry with air and form small bubbles. The hydrophobic minerals can then attach to the bubbles and be lifted to the surface in the froth phase. The froth with minerals is collected from the top as overflow, while tailings are gathered from the bottom and routed to the next stage in a flotation bank. In order for flotation to occur, the particle size of the mineral should be small enough so that bubbles can lift up, the bubbles must form the froth phase on top and the mineral particles should be hydrophilic. (Ur Rehman 2011)

Flotation cells are usually connected with each other in series and referred to as a flotation bank. Slurry goes as an input to the first cell and continues through the whole series. Concentrate is collected from each cell, and the tailings from the last cell. Flotation banks connected in series form a flotation circuit. Slurry goes first to the first cell in the rougher bank, and tailings from the rougher bank are fed to the scavenger bank. Valuable concentrate from the rougher and scavenger banks is fed to the cleaner bank, which is used to further decrease the non-valuable minerals in the concentrate. Regrinding may take place between these circuits.

Mass pull is another important indicator for flotation performance. Mass pull is the flow rate of solid content reporting to the concentrate, and it is affected by making changes in the froth structure and stability. Froth structure and stability can be controlled using the air flow rate and froth depth (Hadler et al. 2010). The two main control objectives for mass pull control are to optimise and stabilize the performance of the system. Mass pull can be regulated by having a controller on each flotation unit, where the manipulated variables are the reagent dosing rate, froth depth and air flow rate. As in mineral processing in general, the biggest challenge in mass pull control is the constant variation in ore characteristics, strict final product requirements and the need to maximize the recovery of a finite resource. (Muller et al. 2010)

2.3 Mine-to-mill concept

As mentioned in the introduction, the process stages in mineral processing are interconnected and changes made in one process stage may influence later stages. For example, changes made in drilling and blasting can have an undesirable impact on crushing and grinding processes, which can then lead to further changes in downstream processes. Consequently, each process stage should be optimised and analysed in the context of the whole operation and every change made to independent processes should be considered since it will probably affect the bigger picture. The goal of mine-to-mill optimization is to develop and apply integrated mining and processing strategies customized for the target plant to minimize the overall cost per metric ton and maximize operating profit in a sustainable way. (Kawatra and Young 2019)

In most mineral processing operations, drilling and blasting can be seen as the first process stage. It is an energy-efficient preliminary step for crushing and grinding processes. Crushing and grinding are extremely energy-intensive processes. Where drilling and blasting consume about 0.1–0.25 kWh/t of energy, crushing consumes 0.5–8 kWh/t, and grinding 10–35 kWh/t of energy. In mineral processing, comminution processes make up approximately 30–60% of the total energy consumption. Since crushing and especially grinding processes consume an immense amount of energy, and also have an impact on the overall productivity of plant operations, it is important to optimise drilling and blasting processes to make rock-breakage efficiencies as good as possible during crushing and grinding. (Park and Kim 2020)

Hence, one of the main goals of mine-to-mill optimisation in the early process stages is to improve rock-breakage efficiency for crushing and grinding. This can be done by optimising blast fragmentation, since blasting produces microcracks that decrease the required comminution energy for crushing and grinding. The optimization process for blast fragmentation is complex, and it requires consideration of rock mass characterization, blasting energy and downstream comminution processes (Park and Kim, 2020). The optimisation must also consider the circuit flowsheet, types and sizes of comminution and classification equipment, power capacity and final product specifications. The additional cost of increasing blasting energy is well compensated by the increase in throughput and recovery and decrease in energy consumption in downstream processes. (Kawatra and Young 2019)

Determining rock mass characteristics in real time is important but also difficult, since gathering the data is labour-intensive and time-consuming. Rock mass characteristics, or ore characteristics, are often determined by means of laboratory and field tests. Another challenge with rock mass characteristics is the limited approach options for applying the data to blast energy design. Representing the whole rock mass in real time is very challenging because of the frequency and scale of blasting and the widely varying rock properties found at mines. (Park and Kim 2020)

One method for rock mass characterization is evaluation of drilling performance data (Park and Kim, 2020). The data acquired from blasthole drilling is especially useful, since it is gathered systematically, regularly and in real time. This practice is called monitoring while drilling (MWD), and it is used to gather data on penetration rate, flushing pressure, rotational speed and global positioning system (GPS) drillhole positioning data. MWD data can be used to determine the rock hardness, which is a useful measurement for blast design. Lately, MWD data has been used with machine learning to distinguish rock types (Park and Kim 2020).

Mine-to-mill optimisation should also consider mineral separation, for example the flotation circuit. The performance of the flotation circuit can be affected by an increased grinding circuit throughput. Due to the limited capacity of grinding circuits, increased throughput can result in coarsening of the product size, which can then have an undesirable impact on mineral recovery in the flotation circuit. Flotation recovery is heavily dependent on the particle size distribution of the valuable mineral in the flotation feed, and the highest recoveries are achieved for intermediate-sized particles. The recovery is lower for ultra-fine particles due to poor flotation kinetics, and for coarse particles due to poor liberation and settling. For these reasons, a fine particle size should be achieved in grinding to optimise the recovery and grade in flotation. However, achieving a fine particle size is an aim that must consider the higher cost of grinding finer due to increased power usage and the income that results from increased throughput in grinding, which in turn results in a coarser particle size. The flotation circuit itself can also be adjusted for a coarser feed, often by using regrinding capacity. Comminution and flotation operations can be optimised with respect to the overall operation by finding the optimum trade-off between target grind size (and throughput) and flotation recovery. For example, in some cases increased throughput can outweigh the lower recovery by providing an overall increase in production and profitability. (Kawatra and Young 2019)

2.4 Data-based modelling utilizing machine learning

When dealing with machine learning (ML), the term artificial intelligence (AI) must first be addressed. AI is an old concept which was first introduced in the late 1950s. However, due to the lack of data availability and calculational power, practical usage of AI did not begin until the 1980s. Since then, due to the immense improvements in these features, the opportunities provided by AI and ML have also increased. AI, as the name states, refers to computational methods that imitate human intelligence, for example learning or problem solving. AI can be approached from two different directions, namely computationalism and connectionism. In computationalism, the aim is to imitate logic and formal reasoning. An example of computationalism is an expert system. (Barragán-Montero et al. 2021)

Machine learning methods, on the other hand, are an example of connectionism. Machine learning is a set of methods where the input data is fed into a machine which can learn from the data without being expressly programmed. There are usually two stages in ML: training and deduction. In training, previously gathered data is put into the algorithm and it forms patterns from the training data. In the deduction stage, new data is fed into the algorithm, where the machine compares patterns formed in the training stage to the new data. An example of the usage of machine learning is image recognition. Other tasks that a machine learning algorithm can perform are for example decision making and predictions. Machine learning approaches can be roughly divided into three categories, i.e. supervised, unsupervised and reinforcement learning. Other learning approaches are semi-supervised learning and self-supervised learning. (Barragán-Montero et al. 2021)

In this thesis, a supervised machine learning method, k-nearest neighbors (kNN) is covered in depth. In addition to training and deduction, machine learning methods require that the input data is prepared sufficiently. One way to do this is to follow the CRISP-DM procedure, which is described in the section below.

2.4.1 CRISP-DM

CRISP-DM (cross-industry standard process for data mining) is an open standard approach to data mining. It was developed in the late 1990s in co-operation with Daimler-Benz, NCR Corp., OHRA and SPSS Inc. CRISP-DM was developed so that it could be applied to any kind of data without any specific software requirements. The CRISP-DM approach consists of six phases: business understanding, data understanding, data preparation, modelling, evaluation and deployment. These phases are visualized in Figure 2. (North 2012)

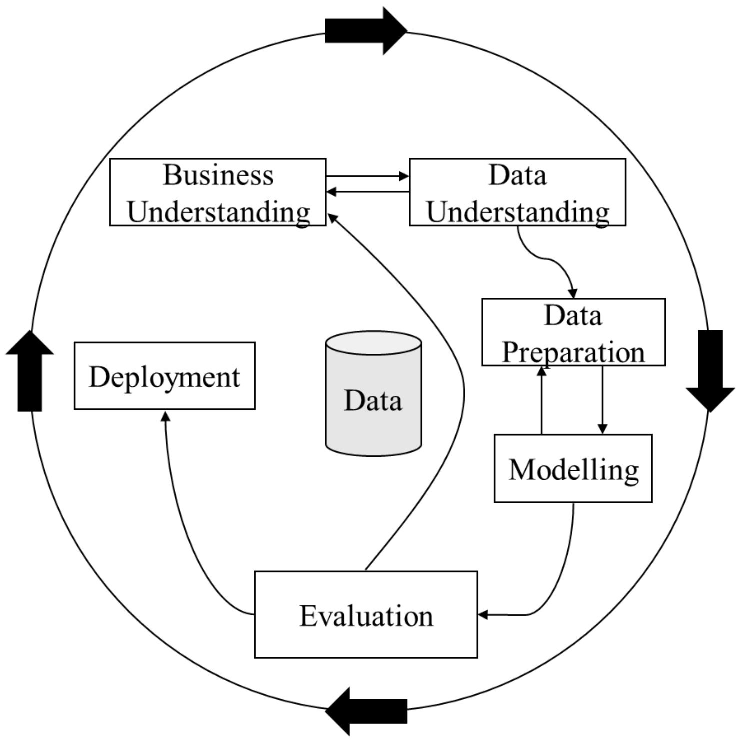

Figure 2. CRISP-DM phases (retelling Jaggia et al. 2020).

Business understanding is the first step in the CRISP-DM approach. It is an essential step at the beginning of the process to fully understand the problem, and the motivation for data mining in the first place. The second step in the process is data understanding. This includes initial data collection, familiarization with the data, identifying possible quality problems with the data, gaining insights into the data and dividing data into subsets. (Jaggia et al. 2020)

The third step is data preparation, which includes many tasks. These might be data reduction, data cleansing or data transformation for testing purposes. The fourth step is modelling. This includes the selection and development of analytics techniques and models. The fifth step, which is evaluation, involves readdressing and clarifying the results of the analysis in the context of the objectives and success criteria determined in the first step of the process, business understanding. In the final step, which is deployment, the results from the data analysis are rephrased into a set of recommendations and actions. This step also includes communication of the results among the analysts and other business members, as the analysts might still improve the model performance and accuracy. (Jaggia et al. 2020)

2.4.2 Data preparation

In any data analysis, data preparation, or data pre-processing, plays a significant role in the reliability and availability of the data. It is important to do the pre-processing carefully, since it can also have a negative impact on the analysis. It is also important to have knowledge of the analysed data, so that the results and their utilization can be understood. Raw industrial data is usually stored in large databases, which include both direct measurements from process instruments and calculated variables. Unmeasured disturbances with process and instrument/machine failures contaminate the data and they can be seen as different operating points. These distortions in data can be challenging to trace, especially afterwards unless they are logged as soon as they happen (Posio et al. 2008). A system can also contain many separate data collection systems, which need to be combined, introducing a problem with synchronization. Since the problems with raw data are very specific to the process, process knowledge is essential in data pre-processing. (Posio et al. 2008)

Outliers are a common phenomenon in raw data. They are significant deviations from the majority of measurements. Outliers have an impact on data statistics, such as standard deviation, mean and median, if they are not removed before the calculation. Data pre-processing often includes outlier detection, removal and replacement, which can be done for example by interpolation. It is important that this is done so that the original data structure is preserved, and no actual measurements are marked as outliers. A common method for outlier detection is to search for measurements which deviate by more than three times the standard deviation from the mean. This method is called the 3-sigma method. One challenge with this method, however, is that the outliers present have an impact on both the mean and standard deviation. When the number of outliers grows, the more distorted the mean and standard deviations become. (Posio et al. 2008)

One important step of data pre-processing is data normalization. This is used to reduce the effects of outliers and dominant features by making the data numerically uniform. Normalization is done for example by scaling the values to a common range which makes sure that each feature in the data will have an equal contribution in the classifier. The scaling is done so that the data preserves the original data distribution. Normalization methods include data-wise normalization, and feature-wise normalization. (Singh and Singh 2022)

Other steps in data pre-processing can include for example re-sampling of the data, removing NaNs (not a number -values), replacing missing values, calculating the mean, median and standard deviation, and summing individual measurements.

2.4.3 Machine learning and k-nearest neighbor method

Supervised and unsupervised learning methods imitate human learning. Supervised learning is the simplest of the machine learning approaches. In the supervised approach, the machine is introduced to inputs and desired outputs by an operator, or “teacher”. In the training stage, well-known input-output pairs are used, and the goal for the machine is to learn a pattern in which a desired output is reached with a given input. Typical supervised learning methods include linear regression, support vector machines (SVM), convolutional neural networks (CNN), recurrent neural networks (RNN), decision trees, random forests and k-nearest neighbors (kNN). In unsupervised learning, the training data does not have well-known input-output pairs, and the goal is to find patterns in the data. Classic unsupervised learning methods are auto encoders, clustering and dimensionality reduction. (Barragán-Montero et al. 2021)

k-nearest neighbor (kNN) is a supervised ML method used for classification and regression which has a simple operating principle. When the algorithm is given a test sample, it finds *k* nearest training samples based on the pre-defined distance metric, after which the *k* neighbors are used to make predictions. The testing sample can be predicted by voting or averaging. For example, voting can be used in classification applications to predict the test sample as the most frequent class label in *k* neighbors, and averaging can be used in regression to predict the test sample as the average of *k* real-value outputs (Zhou 2021). The k-nearest neighbors method differs from many other ML methods in the sense that it does not have an obvious training process. Instead, kNN represents a learning procedure, where samples are stored in the training phase, and only used once the algorithm receives test samples. In kNN, choosing the right value for parameter *k* is important, especially when multiple variables are introduced to the algorithm, since different values of *k* can lead to a difference in the classification results. Another important aspect which must be considered in kNN is distance calculation, since usage of different distance metrics can lead to significantly different neighbors, which in consequence leads to different results. A visual illustration of kNN classifier example is shown in Figure 3. (Zhou 2021)

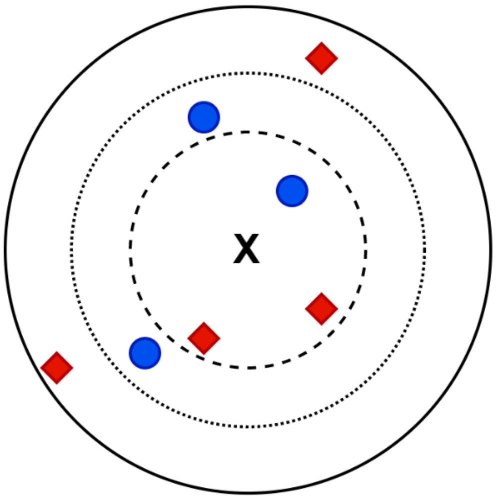

Figure 3. Visual illustration of k-nearest neighbors classifier.

In the example above, the test sample X is classified either as blue circles, or red diamonds. When *k* = 3, which is illustrated by the sparse dashed line, the test sample is classified as a red diamond, since the area holds two red diamonds and only one blue circle. When *k* = 5, marked by the densely dashed line, the test sample is classified as a blue circle, because the area includes three blue circles and only two red diamonds. Lastly, when *k* = 7, the test sample is classified as a red diamond.

In case of distance function (d (.,.)), following axioms hold:

- non-negativity: $d(x_i, x_j) \geq 0$,
- identity of indiscernibles: $d\left(x_i, x_j\right) = 0$ if and only if $x_i = x_j$,
- symmetry: $d\left(x_i, x_j\right) = d(x_j, x_i)$,
- and triangle inequality: $d\left(x_i, x_j\right) \leq d(x_i, x_k) + d(x_k, x_j)$,

where the samples are: $x_i = (x_{i1}, x_{i2}, \dots, x_{in})$ and $x_j = (x_{j1}, x_{j2}, \dots, x_{jn})$. The Minkowski distance, which can be modified to city block, Euclidean and Chebyshev distances, is written in Equation (2):

$$d_{minkowski}\left(x_i, x_j\right) = \left(\sum_{u=1}^{n} |x_{iu} - x_{ju}|^p\right)^{\frac{1}{p}}, \tag{2}$$

where the axioms listed above are satisfied when $p \geq 1$. The city block distance is a special case of the Minkowski distance where $p = 1$. The city block distance is described in Equation (3):

$$d_{city\ block}(x_i, x_j) = \sum_{u=1}^{n} |x_{iu} - x_{ju}|. \tag{3}$$

The Euclidean distance is a special case of the Minkowski distance where $p = 2$. The Euclidean distance can be written as follows:

$$d_{euclidean}(x_i, x_j) = \sqrt{\sum_{u=1}^{n} |x_{iu} - x_{ju}|^2}. \tag{4}$$

For the special case of $p = \infty$, the Minkowski distance gives the Chebyshev distance, which can be written (Zhou 2021; Mathworks 2022):

$$d_{chebyshev}(x_i, x_j) = max_u(|x_{iu} - x_{ju}|). \tag{5}$$

2.4.4 Machine learning in mineral processing

Applying AI and ML methods for industrial use has become popular in recent years. Mineral processing, among other fields of industry, is searching for ways to utilize AI and ML methods in practice to improve process productivity and reduce human error. AI and ML methods are also used to solve problems that are unique in mineral processing. The demand for ethical mining has increased together with decreasing ore grades. (Mishra 2021)

Since the ore bodies being mined and processed tend to have more complex mineralogy and lower grades, a few challenges have emerged. There might be a need to extract multiple minerals, or minerals of varying concentrations from the same ore. The quality of the deposit might be high, but the size of accumulation is low and vice versa. Also, due to the demand for sustainable and green development, the importance of green mining with minimal climate impact and material footprint of the process chain has increased. This has led to a shortage of ores. These problems, green mining and reduction in ore grade exacerbate each other, which is problematic. (Mishra 2021)

The reduction in ore grades has created one possible application of AI and ML methods in mineral processing. Whereas mineral processing machines and techniques have been quite similar across all mining operations due to the similarity in ore properties, it is now almost a requirement to adjust the process parameters in real time, due to fluctuations in ore properties. This real-time adjustment of process parameters is anticipated to be a major application of AI and ML in mineral processing. (Mishra 2021)

Unsupervised machine learning methods, for example k-means clustering or partitioning around medoids, could be used in block classification. Block classification has a direct impact on the profitability of the mine. The impact on the performance of processing stages when having inputs with significant inconsistencies in grades is often neglected. (Li et al. 2020). Since deviations from the target grade of a process stream lead to unwanted losses in recovery, a consistent input for the process is necessary but hard to achieve. More consistent input also leads to increased profit and more even recovery and output. K-means clustering could be utilized to make clusters from blocks, with each cluster having a unique target grade. In a case study conducted by Li et al. (2020), Clustering LARge Applications (CLARA) and K-means-based Approximate SPectral clustering (KASP) methods were used to generate clusters of selective mining units with similar grades that corresponded to different process destinations, while minimizing differences in mineral grades within clusters.

In addition to block classification, ML methods have also been used in froth flotation. In a case study by Horn et al. (2017), Convolutional Neural Networks (CNN) were used in a feature extraction of platinum froth flotation images. Classical froth texture feature extraction procedures are prone to variation in imaging conditions, such as changes in environmental lightning, they consider limited feature types and require expert knowledge in selection. In the case study, CNN was selected since it does not have the drawbacks listed above, and it mitigates the curse of dimensionality that is a characteristic in fully connected networks. Deployment of CNN enables extraction of both textural and spectral features from images. To summarize, the quality of extracted CNN features was compared to features that were extracted with classical methods. The CNN feature extraction method was found to be competitive with other extraction methods in terms of performance, but the comparison of methods was complicated due to the strong spectral features present in the dataset. The results from the case study do not provide sufficient data to differentiate the types of features extracted by CNN, and further studies on this method are required. (Horn et al. 2017)

Another application of ML methods in froth flotation has been implemented in a case study by Pu et al. (2020). In the case study, deep learning, encompassing long short-term memory (LSTM) architecture, was used in flotation process modelling, for forecasting the concentrate purities for iron and waste silica. For the LSTM model, 23 variables from the flotation plant were monitored and hourly data was collected for six months; the model was trained and tested with prepared data. The model built during the case study was capable of predicting real-time concentrate purities for iron and waste silica and was significantly more proficient in modelling a froth flotation process when compared to a random forest model used in the study. The study states that, despite the accuracy with which the purities could be predicted with the LSTM, a few aspects of the model require further examination. One problem with the model is the black box feature of the deep learning method, meaning that there is a lack of knowledge concerning how the output was formed, and therefore an explicit function of the trained LSTM model could not be extracted. However, the study encourages the utilization of deep learning methods in mineral processing, especially in flotation process automation control. (Pu et al. 2020)

3 DATA PREPARATION AND METHOD IMPLEMENTATION

The process data which was used for calculations and ML methods in this thesis was acquired from a mineral processing plant through Metso Outotec. All the calculations and ML implementations were done in MATLAB®. Before any ML implementations, the process was studied using SCADA (supervisory control and data acquisition) pictures of the plant, and the raw data at 5-minute frequency sent by the plant. The initial objective of this research was to use history data to recognize different process situations with clustering or classification. The second objective was to compare the current process values to the history data to find out the direction the process is taking and how the process should be controlled, and to draw trajectories for the closest correspondences from history to make a prediction for the current process situation. The supervised machine learning method k-nearest neighbors was applied to find the nearest matches from history data for current process KPI values. The selected KPI variables were the flotation circuit dry feed, flotation feed gold grade, final concentrate gold grade and flotation recovery % for gold. It should be noted that the KPI variables correlate with each other, and that the overall recovery % is a calculated variable (see Equation 1). The KPI variables were selected based on expert knowledge. The use of kNN functions required the MATLAB® statistics and machine learning toolbox.

3.1 Data pre-processing

The first step in pre-processing, as also mentioned in section 2.4.1, was business and data understanding. The raw data from the mineral processing plant was received in eight different datasets, according to different process areas, namely crushing, grinding, flotation and dewatering. In addition to these process areas, particle size indicator (PSI), froth camera, reagents and Courier data were received in their own datasets. Courier is a trademark of Metso Outotec for an online X-ray fluorescence analyser, and the dataset includes chemical element contents from certain process stages. All datasets included process measurement data at a 5-minute sample rate from a one-year time period, November 2020 – November 2021. Some datasets had a difference of five minutes in time and were synchronized. The total amount of timestamps for each variable was 105120 across all datasets. The number of individual process variables differed greatly between the process areas. Flotation data had the greatest number of variables by far with

187, followed by grinding data with 58 variables, dewatering with 46 variables, reagents with 41 variables, Courier data with 29 variables, PSI data with 13 variables, and crushing and froth camera data with 7 variables. In total, the number of individual variables was 388.

The very first step in data handling was to import the data from Excel to MATLAB®. In the raw data, the measurements in each process area were identified with tag numbers, which corresponded to certain process variables. These tag numbers were renamed based on the requested tags list sent by the mineral processing plant and SCADA pictures from the process areas to make the data more understandable and make it simpler to find wanted variables in future. Also, the measurement units for each variable were added to the new variable names. While studying the SCADA pictures of each process area for variable names, it was also noted that a few requested variable measurements were missing. For example, the circulation loads for grinding mills were not received.

The first practical step in the data pre-processing, or data preparation, was to sort all the data from oldest to newest, since the raw data was sorted starting from the newest measurements. This was done simply to ease the interpretation of data. The datasets were also synchronized in relation to each other, because of the differences in timestamps between the datasets. This was done to make calculational operations more reliable between datasets. Also, in the raw data, valve positions were logged in on/off format, so they were converted to binary 1/0 format, to make calculations with these tags possible.

With the valve positions in binary format, the sum vector for each valve set was calculated and added to the corresponding dataset. These valve sets were related to grinding and regrinding mill classification cyclones. Other calculational variables included the recovery % for gold, iron and arsenic in the Courier dataset, and the plant dry feed, which is the dry feed for the flotation circuit in the crushing data set. The recovery vectors for gold, iron and arsenic were calculated according to Equation 1, presented in section 2.2. The plant dry feed was calculated with the following Equation (6):

$$X = y - ((\frac{p}{100} * y), \tag{6}$$

where X is the dry feed [t/h], y is the wet plant feed from the belt scale [t/h] and p is the plant feed ore moisture [%].

After calculated variables were added to the datasets, the data was examined visually. The raw data was examined using timeseries, histogram and boxplot graphs. Also, the mean and standard deviation was calculated for each variable. The purpose of this examination was to identify any possible problems with the variables, and to identify the ranges for each variable. Each variable was examined individually for noise, data breakages, negative values and possibly missing data. The ranges were stored for later use in data pre-processing. At this stage, it was noted which variables contained a large number of data breakages, negative values or long timespans where no measurements were gathered. Also, it was observed that, within the same dataset, data breakages could occur at the same time for many different variables.

After the data had been visually examined and ranges gathered for each variable, outliers were removed from the data. This was done by using the 3-sigma method, replacing outliers with linearly interpolated values. For the Courier dataset, minimum and maximum ranges for values were manually pre-determined, since 3-sigma was unable to remove the outliers there because they were very large negative values. After all the datasets had been pre-processed in this manner, the five-minute data was resampled to one-hour and one-day average sample rates. Linearly interpolated values were kept in the data at this stage.

The one-day averaged data was again visually examined with time-value, histogram and boxplot graphs. Mean and standard deviation were calculated again for each variable, and any remaining problems in the pre-processed variables were noted. A few variables were marked as unusable, since they contained mainly unhealthy data, meaning that the data was almost completely missing. It was also noticed that, when a longer time period with outliers occurred in a variable, the linear interpolation of outliers distorted the measurements significantly. Because of this, pre-processing was conducted later in a different manner. For now, however, this pre-processed data was normalized in order to make test runs with the k-nearest neighbors algorithm.

Due to the problems that occurred with interpolated outliers, it was decided to remove outliers with 3-sigma by determining a maximum time period after which outliers would not be interpolated any more. A time period of one hour was used for all the data, and pre-processing was performed again. The outliers were removed while keeping all the timestamps present in the data and making the outliers NaN values. In addition, minimum and maximum ranges from visual data examination were used for all the variables in the data in outlier removal, and normalization was done before resampling for the final normalized data.

3.2 Implementation of k-nearest neighbors

The k-nearest neighbors algorithm was selected for finding process points from the history data that closely matched the current process values (KPIs). The kNN function was constructed so that it could take data of any sampling rate as an input. However, for the kNN function, one-hour averaged data was used as an input for the results, since the target was to draw trajectories for current process values, as well as history data. The problem with a more frequent sample rate would have been substantial noise, which was observed in the visual examination of the original data (five-minute sample rate). On the other hand, a longer sample rate or filtering the data, which would have more efficiently negated the noise, would potentially have lacked the process dynamics of interest; there was a special need to examine eight-hour time periods, in other words shifts, which was quite simple to do with one-hour averaged data.

The custom kNN function was built around the *ExhaustiveSearcher* and *knnsearch* functions, which are part of the MATLAB® statistics and machine learning toolbox. The exhaustive searcher model object stores the training data, distance metric and parameter values for the distance metric for an exhaustive nearest neighbor search. This algorithm finds the distance from each query observation to all *n* observations in the training data, which is an *n*-by-*K* numeric matrix. Once the model object is created with *ExhaustiveSearcher*, neighbouring points to the query data can be found from the training data by performing a nearest neighbor search, in this case using the *knnsearch* function. It is also worth mentioning that the algorithm searches nearest neighbors only from the past, meaning that nearest neighbors can only be timestamps that occurred before the current process point (query point).

The kNN functions were built separately for a 2-, 3-, and 4-variable nearest neighbor search, and the KPIs to be considered in the kNN search were specified inside the function. All the functions shared the same inputs and outputs, with the natural exception being the visual illustration of the results across functions due to changes in dimensionality. The function has the following outputs:

- t: the timestamps of the nearest neighbors found are collected into a 1 by *k* matrix t.
- D: with a 1 by *k* matrix D, we can examine the distance indexes for the nearest neighbors.
- Mdl: an optional output, which holds the parameters of the *ExhaustiveSearcher* function.
- Number of nearest neighbors excluded due to th input.

The input parameters for the functions were:

- t_test,
- k,
- deadtime,
- th,
- n_step,
- datasets with and without normalization, and
- draw.

The query point, which is a timestamp for which nearest neighbors are sought, is defined by the *'t_test'* and the number of requested nearest neighbors presented as a result is designated by *'k'*. The algorithm searches for a larger number of nearest neighbors but returns only *k* valid neighbors. The minimum timespan between the query point and nearest neighbors that were found was specified with the input *'deadtime'*, with a notation that the input value for deadtime was in the same time resolution as the input dataset. This was done to ensure that nearest neighbors were sought from history, and not from timestamps that had happened recently. For example, an input value of eight for deadtime means that if the query point (*'t_test'*) is 100, 98 cannot be found as a nearest neighbor, but values from 92 and lower can.

In addition to deadtime, another threshold, *'th'*, was assigned to avoid consecutive timestamps being found as neighbors. For example, if the *'t_test'* is 100, the *'k'* value is 3 and *'th'* is assigned to be 2, then if the nearest neighbors are 91, 67 and 90 in order of distance index, timestamp 90 is excluded from the results, since it is less than *'th'* timestamps away from the previous nearest neighbor with a lower distance index. This excluded neighbor would then be replaced with the next valid nearest neighbor.

The *'n_step'* parameter can be used to specify the timespan for which the trajectories for timestamps are plotted. For example, if *'n_step'* is assigned to be 8, and one nearest neighbor is 50, variable values from timestamp 42 to 58 are plotted. This is important in order to see how the selected variables behave around the timestamp that is found as the nearest neighbor. The *'dataset'* inputs can be used to specify the sampling rate at which the search for nearest neighbors is done. The *'draw'* parameter is used to determine whether the results are plotted as normalized values, real process values or not at all. With a parameter value of 0, no results are plotted. With a value of 1, results are plotted in normalized values, and with 2 in real process values.

The selection of the applied distance metrics in the kNN function also had to be made. As mentioned in section 2.4.3, different distance metrics can have a significant impact on the results, depending on the application of the algorithm. Various distance metrics, which are pre-built into MATLAB® *ExhaustiveSearcher* function, were tested within the function, such as Euclidean, city block, Chebyshev, Hamming, standardized Euclidean and Mahalanobis distances. The distance indexes and timestamps for found neighbors were compared between the tested distance metrics with different values of *k*, and with a different number of dimensions. Thus, a custom script, for observing the *k*=1 distance index for a time span of 1200 timestamps, was coded and the results were plotted. This allows the visualization of differences in distance indexes between the distance metrics, and to see if the metrics found the same timestamps as the nearest neighbors.

3.3 Selection of query points

Once the custom functions for 2-, 3-, and 4-variable examination were ready, the query points for testing had to be decided. This was done by visually examining a time-value plot of overall gold recovery and choosing timestamps as query points where the gold recovery was about 85–90%, and reasonably stable for a few days. The fact that neighbors could only be found from past timestamps was also considered, and query points were selected so that there was plenty of training data for the *ExhaustiveSearcher* function to work with. Finally, when the query points had been selected, the KPI variable values from the query points were visualized and compared with a MATLAB® community-made radar chart function (Matlabcentral 2022) to verify that the KPI values had some variance across the query points. The query points selected for final testing were 7018, 5584, 5158, 7435 and 6980, and they are plotted on the radar chart shown in Figure 4 below. Please note that the values in Figure 4 are normalized.

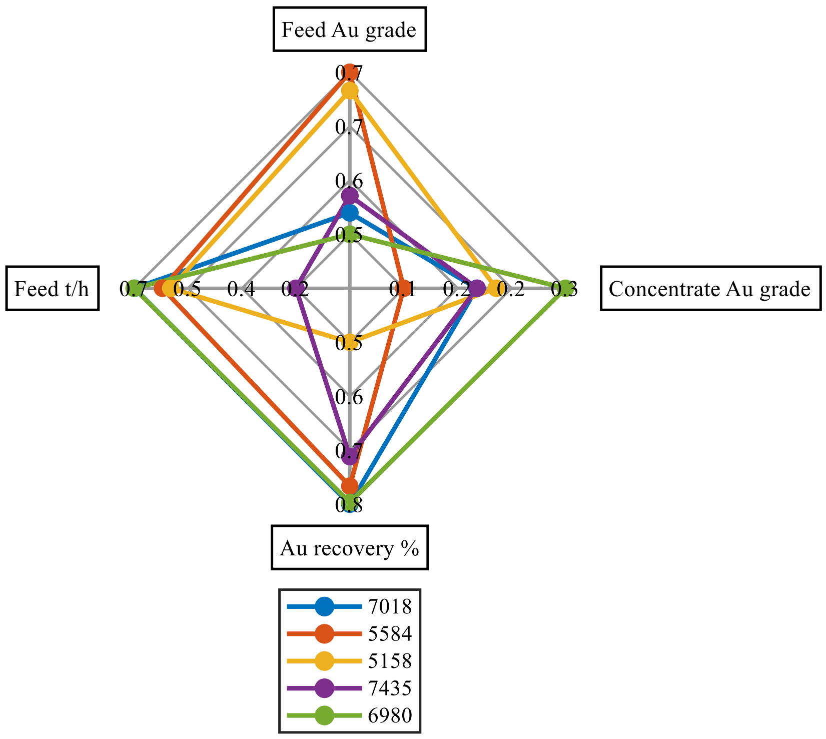

Figure 4. Radar chart of normalized KPI variable values at query points.

In Figure 4, the feed Au grade values are similar at query points 7018, 7435 and 6980, whereas at 5584 and 5158 the value is higher. However, for concentrate Au grade, query points 7018, 7435 and 5158 show similar values, while 5584 has lower values and 6980 higher values. For Au recovery, all query points have similarly high values except for 5158, which has a relatively low value. For flotation feed, all query points except for 7435 have similarly high values. As the figure shows, all query points are in different process situations, which was the aim.

4 NEIGHBOR SEARCH RESULTS

As a conclusion for the comparison of distance metrics, Euclidean (which is the default distance metric set by MATLAB®), Chebyshev and city block distances had the best performance and offered very similar results, finding the same timestamps as nearest neighbors when the value of *k* was smaller or equal to three. Differences between these metrics emerged when searching for more than three nearest neighbors, with a *k* value of 6 for example, the neighbors 4-6 being different across the metrics. Figure 5 below illustrates the differences between the selected metrics.

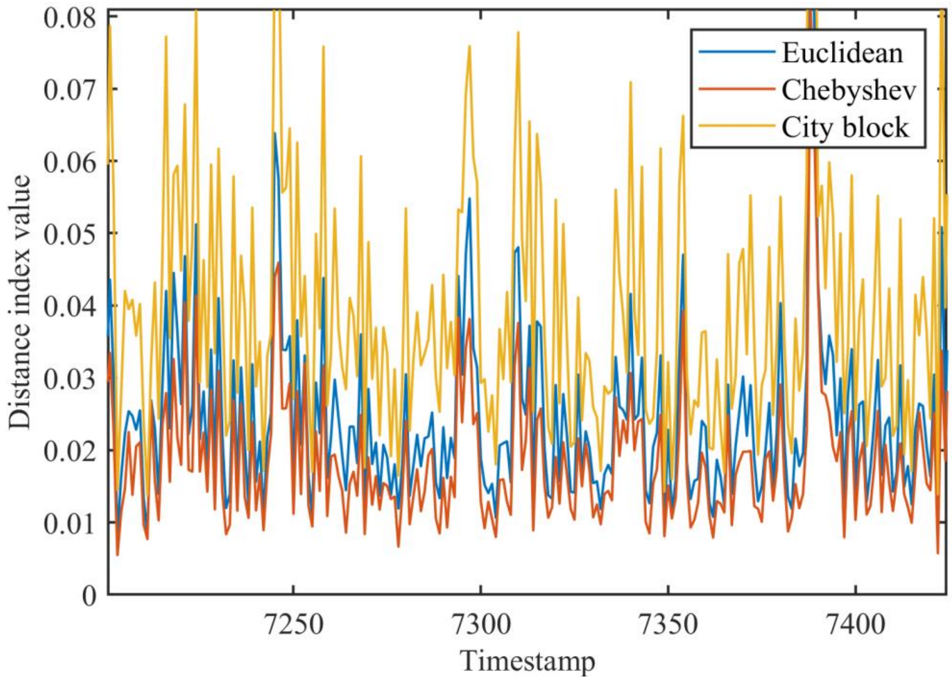

Figure 5. Distance metric comparison with *k*=1.

From Figure 5, distance index values can be seen for duration of 1200 timestamps with *k* value of 1 for Euclidean, Chebyshev and city block distance metrics. In the figure, individual distance index values should not be compared between the metrics since some distance metrics give lower distance index value even if they find the same timestamp as a neighbor. Differences between the metrics can be seen from the trends for each metric. If there is a sudden change in trends across the metrics, it indicates that the metrics found different timestamp as a nearest neighbor. The Chebyshev distance metric was ultimately selected to be used in the *ExhaustiveSearcher* function because of its good performance and since the other metrics tended to find neighbors, or timestamps, which were right next to each other.

In the following figures which display trajectories for the query point and nearest neighbors, the data has been filtered with a four-hour moving average with the aim of making the trajectories easier to read, without distorting the values in the kNN search. In MATLAB®, the *smoothdata* function was used with additional arguments of *movmean* as a method, and 4 as the length of the window. For all the figures in this Chapter, one-hour averaged and normalized data was used, consisting of 8759 data points for each variable. For a 2-variable search, the flotation dry feed, and the final concentrate Au grade were the KPIs used. With a 3- and 4-variable search, the overall Au recovery and the feed Au grade, respectively, were considered as additional KPIs.

Figures 6-12 below share the same query point, which was 7018 presented in Figure 4, and the same algorithm input parameters. These input parameters were *k*=6, *deadtime*=8, *th*=12 and *n_step*=12. However, Figures 6-8 present results for a 2-variable kNN search, while Figures 8-9 are results for a 3-variable search, and Figures 10-11 for a 4-variable search at the same query point. Table 1 collects the results for query point 7018.

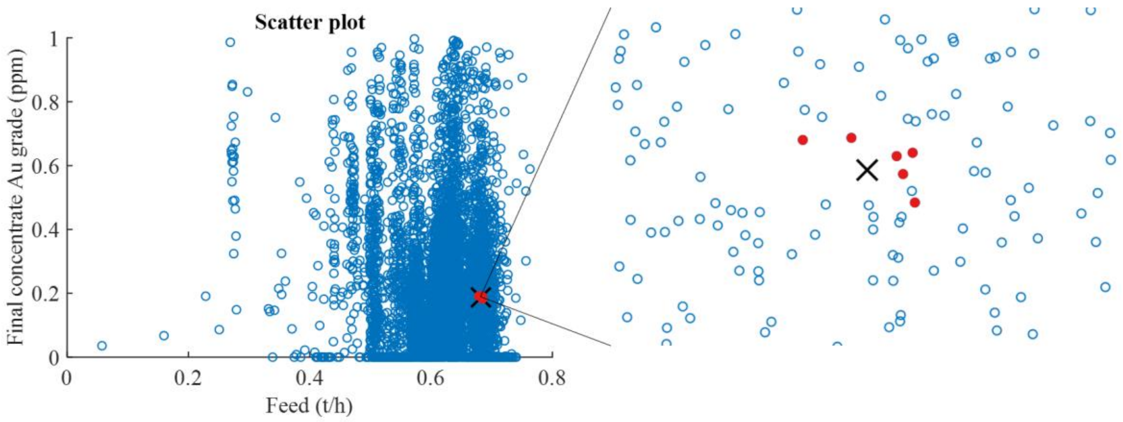

Figure 6. Scatter plot for query point 7018 (2 var.).

Figure 6 presents a scatter plot with values from 2 KPI variables (flotation dry feed (t/h) and final concentration Au grade (ppm)). In the figure, the black cross marks the query point, and red dots are the *k* nearest neighbors found with Chebyshev distance and the threshold criteria described. The nearest neighbors found in order from nearest to farthest were: 6877, 6952, 6826, 2378, 2357 and 6985. It should be noted that in this case one neighbor was excluded from the results due to the threshold argument. The distances are visualized in Figure 7 below.

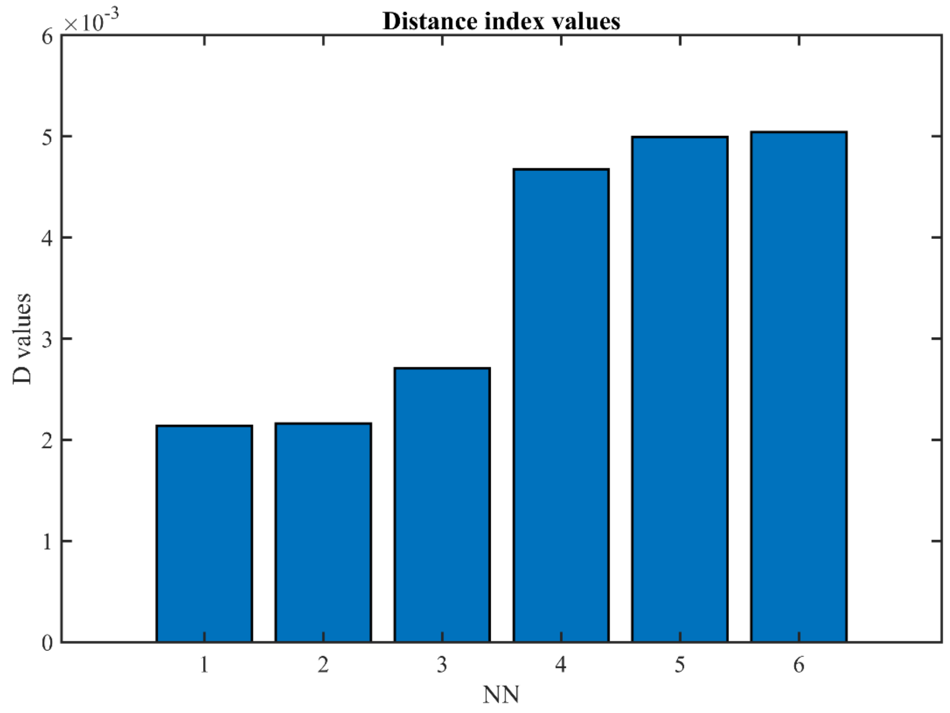

Figure 7. Bar chart for distance indexes of nearest neighbors (2 var.).

Distance index values for the above-mentioned nearest neighbors are plotted in a bar chart in Figure 7. The values were 0.0021 for NN (nearest neighbor) 1, 0.0022 for NN 2, 0.0027 for NN 3, 0.0047 for NN 4, and 0.0050 for NN 5 and NN 6. It can be observed that neighbors 1-3 are quite close to each other in terms of distance index, and there is a jump in distance index between neighbors 3 and 4. Neighbors 4 to 6 are again close to each other. The trajectories for these neighbors and the query point are plotted in Figure 8.

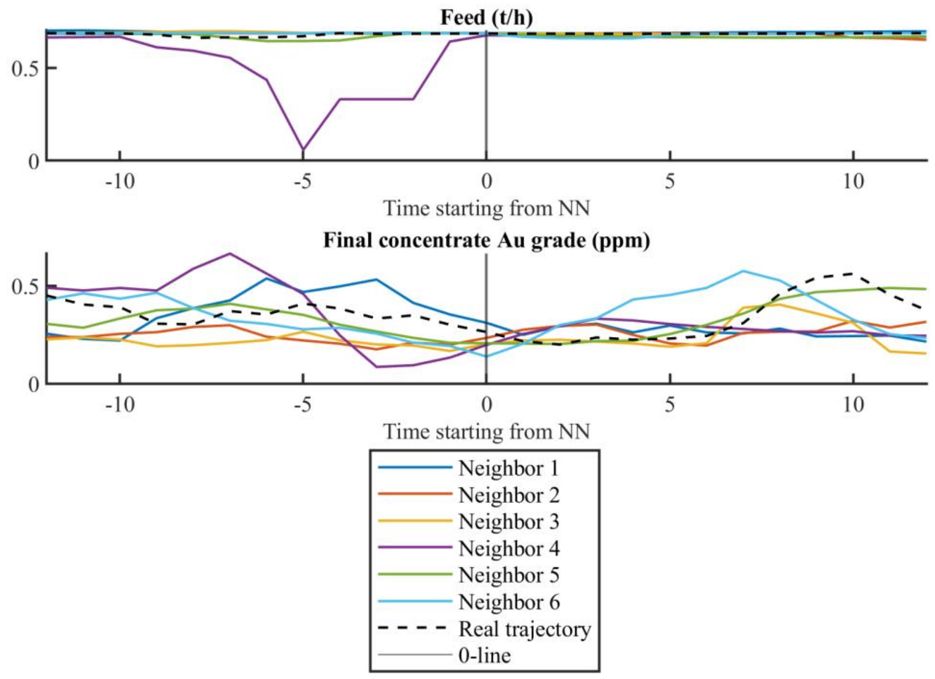

Figure 8. Trajectories for query point 7018 and nearest neighbors (2 var.).

Figure 8 displays the trajectories of the variable values for the query point, which is the dashed line, real trajectory, and the nearest neighbors which are solid-coloured lines. The vertical 0-line in the figure marks the timestamps for the query point and nearest neighbors which were found as a result of the kNN search. The y-axis shows the normalized values for these variables, and by reading the x-axis we can see how the query point and nearest neighbors changed in relation with time. Note that the x-axis is dependent on the dataset used. Because one-hour averaged data was used, meaning that the time interval between each timestamp is one hour, and *n_step*=12, each trajectory can be observed 12 hours to the past and 12 hours to the future from the 0-line. The figure shows that the query point and NN variable values are quite close to each other at the 0-line.

Below, in Figures 9 and 10, the results are shown for the 3-variable kNN search at the same query point 7018 and same input parameters, as in the results displayed above for 2-variable kNN search. Now, with the 3-variable search, the nearest neighbors found in order from nearest to farthest were: 6877, 6967, 6309, 6840, 6125 and 6062. Again, one neighbor was excluded from the results due to the threshold argument. The distances are visualized in the Figure 9 below.

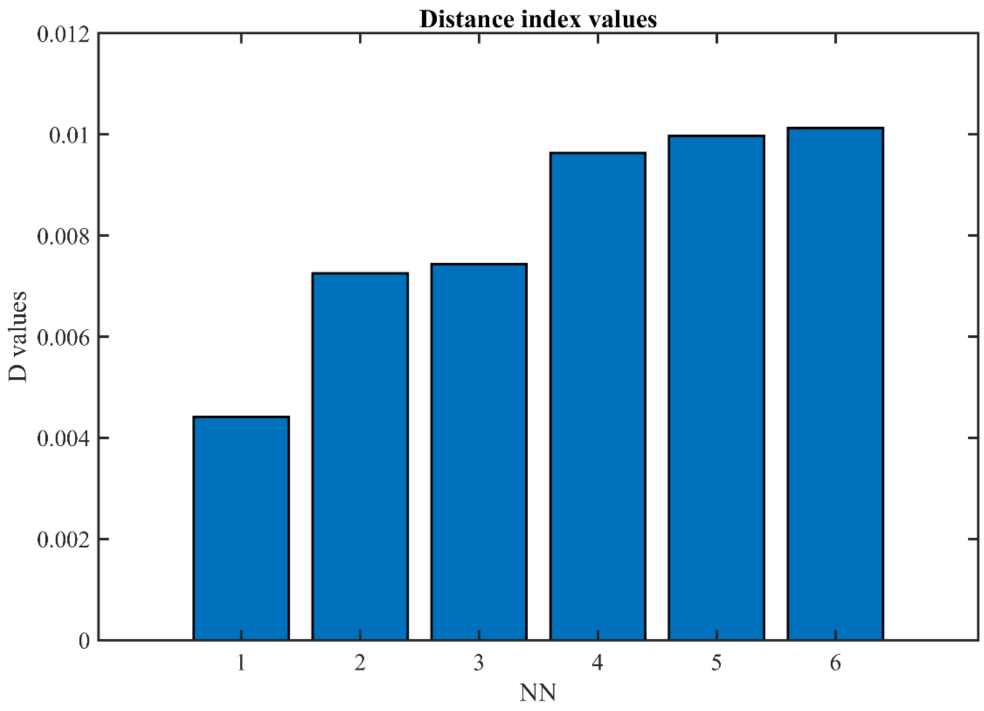

Figure 9. Bar chart for distance indexes of nearest neighbors (3 var.).

With the 3-variable search, the distance indexes for nearest neighbors were as follows: 0.0044 for NN 1, 0.0073 for NN 2, 0.0074 for NN 3, 0.0096 for NN 4, 0.0100 for NN 5 and 0.0101 for NN 6. We can see that despite NN 1 being the same as in the 2-variable search, the distance indexes are larger with the 3-variable search when compared to the 2-variable search. The trajectories for the found neighbors and query point are drawn in Figure 10.

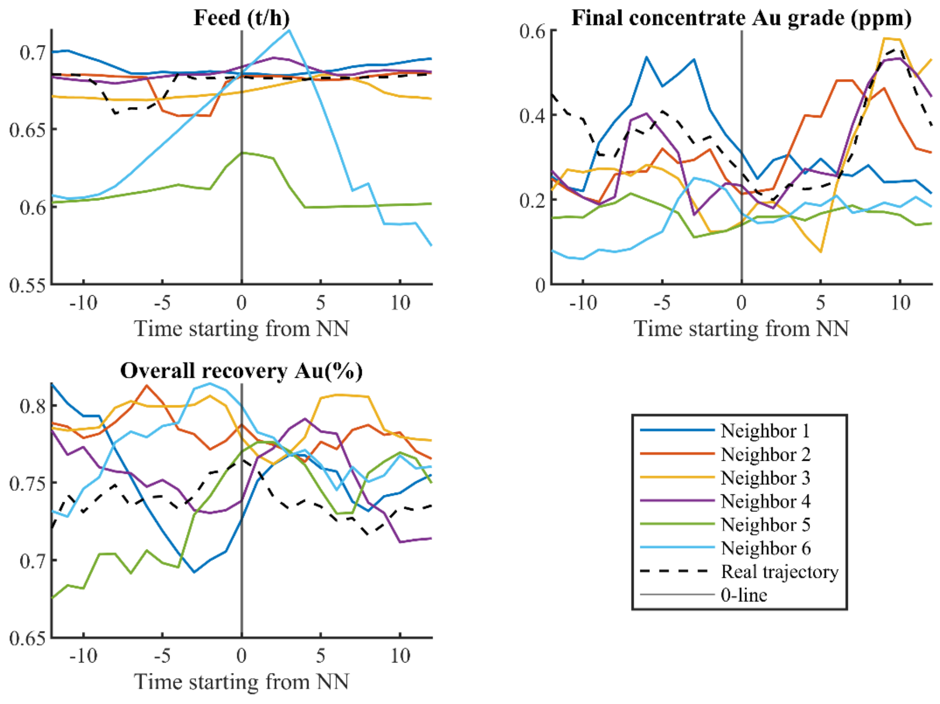

Figure 10. Trajectories for query point 7018 and nearest neighbors (3 var.).

From the trajectories above in Figure 10, it can be observed that the variance between the query point and NN variable values is greater than in the 2-variable kNN search. The real trajectory and NN trajectories show similar behaviour. It can also be observed that for NN 5 there is a large difference to query point's value in Feed (t/h) variable, but the values for Au recovery % are very similar.

Figures 11 and 12 below show the results for the 4-variable kNN search for the same query point 7018 and input parameters as for the 2- and 3-variable searches. With the 4-variable search, the nearest neighbors found in order from nearest to farthest were 6967, 6868, 6840, 6935, 6694 and 6720. Again, one neighbor was excluded from the results due to the threshold argument. Interestingly, the 4-variable search found 2 NNs that also appeared in the results of the 3-variable search. The distances are visualized in Figure 11 below.

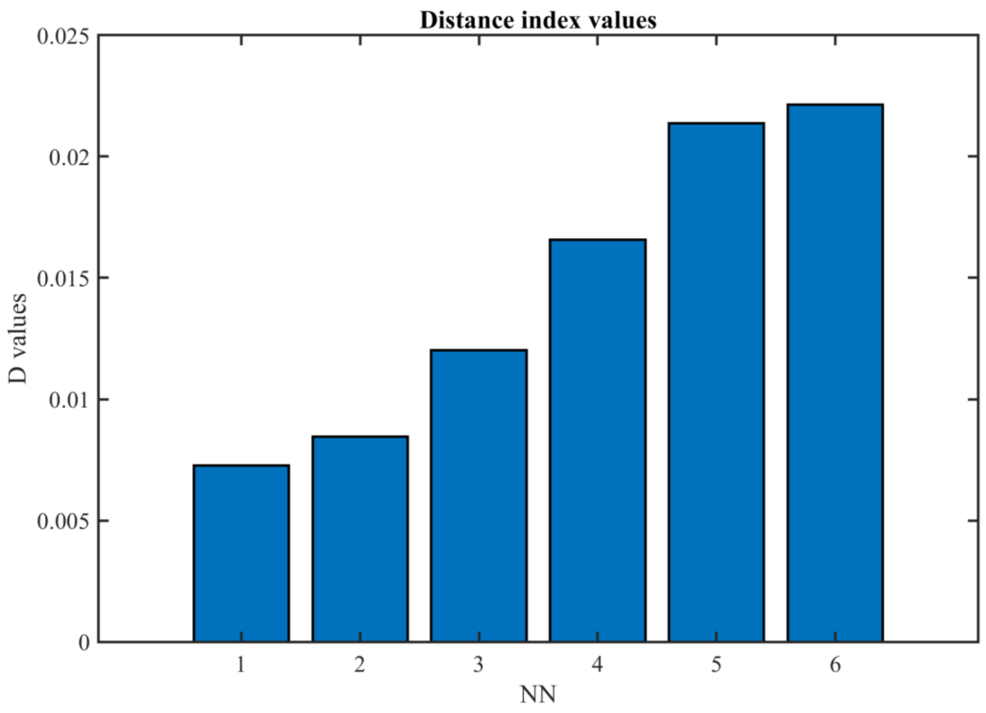

Figure 11. Bar chart for distance indexes of nearest neighbors (4 var.).

The distance indexes for the 4-variable kNN search were 0.0073 for NN 1, 0.0085 for NN 2, 0.0120 for NN 3, 0.0166 for NN 4, 0.0214 for NN 5 and 0.0221 for NN 6. From the distance index values it can be concluded that the distances in the 4-variable results are greater than in the 2- and 3-variable results. The result trajectories for the 4-variable search are shown in Figure 12 below.

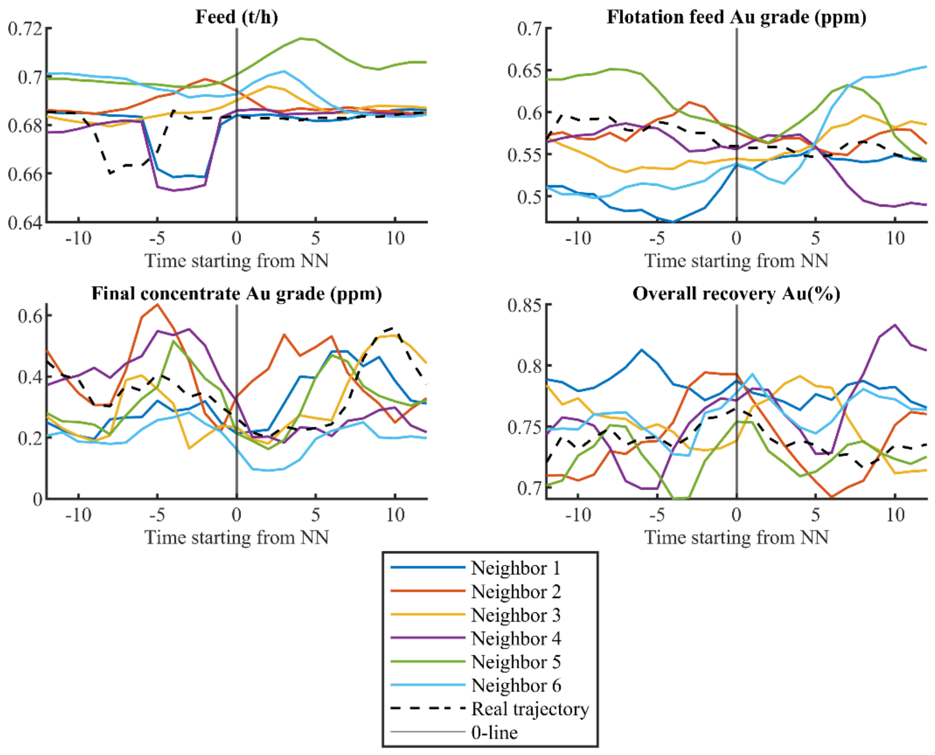

Figure 12. Trajectories for query point 7018 and nearest neighbors (4 var.).

In Figure 12 above, the variance between the query point and NNs on the 0-line is still relatively small. Some similar behaviour between the real trajectory and NN trajectories is also visible. The summarized results for the 2-, 3-, and 4-variable kNN searches at query point 7018 are listed in Table 1 below. Nearest neighbors are sorted from nearest to farthest, NN 1 being the nearest. Timestamps that occurred across 2-, 3- or 4-variable search are written in bold. From the results, it can be seen that when only 2 variables were involved, the search found two timestamps as neighbors that begin with numbers 23-, but when the number of variables was increased, these neighbors disappeared from the results.

Table 1. Results for query point 7018 kNN search.

Query point (7018)	2 variables	3 variables	4 variables
NN 1	**6877**	**6877**	**6967**
NN 2	6952	**6967**	6868
NN 3	6826	6309	**6840**
NN 4	2378	**6840**	6935
NN 5	2357	6125	6694
NN 6	6985	6062	6720
Distance index for NN 1	0.0021	0.0044	0.0073
Distance index for NN 2	0.0022	0.0073	0.0085
Distance index for NN 3	0.0027	0.0074	0.0120
Distance index for NN 4	0.0047	0.0096	0.0166
Distance index for NN 5	0.0050	0.0100	0.0214
Distance index for NN 6	0.0050	0.0101	0.0221
NNs removed due to threshold	1	1	1

The second query point from Figure 4 was 5584. For this query point, the same input parameters and variables were used as for query point 7018. The results are collected in Table 2. For the 2-variable search, the nearest neighbors in order from nearest to farthest were 2228, 5048, 5222, 3427, 5519 and 4019. One neighbor was excluded from the results due to the threshold argument. The distance indexes for these neighbors were 0.0027 for NN 1, 0.0029 for NN 2, 0.0038 for NN 3, 0.0039 for NN 4, 0.0043 for NN 5 and 0.0049 for NN 6. A scatter plot and bar chart for the distance indexes can be found in Appendix 1, Figures 16 and 17. Below, the trajectories for the query point and nearest neighbors are shown in Figure 13.

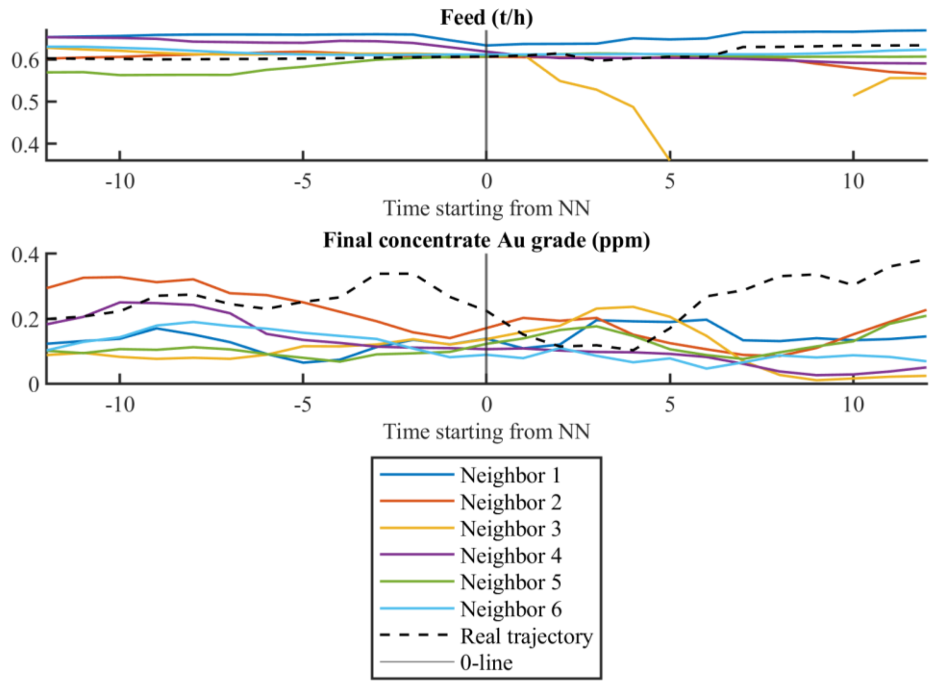

Figure 13. Trajectories for query point 5584 and nearest neighbors (2 var.).

In the figure above displaying trajectories, neighbor 3 for the feed (t/h) variable includes five NaN values, in other words five hours of data are missing from its trajectory.

For the 3-variable kNN search in query point 5584, the nearest neighbors in order from nearest to farthest were 5519, 5559, 3303, 3118, 5572 and 3166. In this case, five neighbors were excluded from the results due to the threshold argument, which means that the initial search found many timestamps as neighbors which were closer than 12 hours to each other. The distance indexes for these neighbors were 0.0095 for NN 1, 0.0114 for NN 2, 0.0133 for NN 3, 0.0163 for NN 4, 0.0188 for NN 5 and 0.0215 for NN 6. The bar chart of the distance indexes in this case can be found in Appendix 2, Figure 18. The trajectories for the query point and nearest neighbors can be found below in Figure 14.

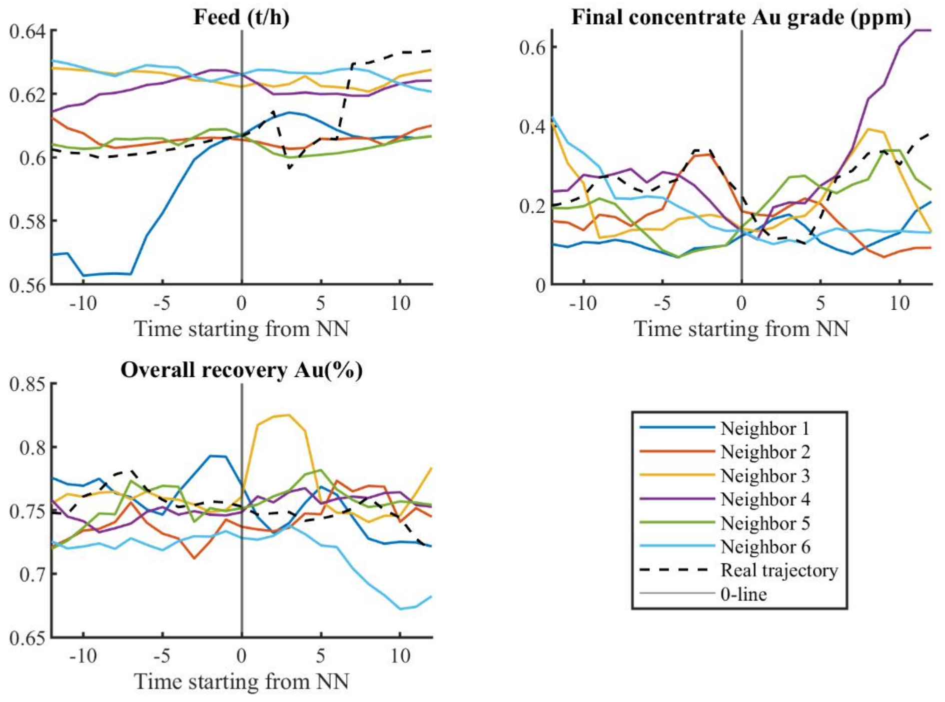

Figure 14. Trajectories for query point 5584 and nearest neighbors (3 var.).

In this case, although five neighbors were excluded from the results and replaced with new ones, we can see that the replacement neighbors are quite close to the query point in terms of value.

The 4-variable kNN search in query point 5584 found the following neighbors in order from nearest to farthest 5519, 5569, 5535, 5475, 5547 and 5452. Three neighbors were excluded from the results due to the threshold argument. The distance indexes for these neighbors were 0.0095 for NN 1, 0.0203 for NN 2, 0.0268 for NN 3, 0.0285 for NN 4, 0.0321 for NN 5 and 0.0330 for NN 6. A bar chart for these distance indexes can be found in Appendix 2, Figure 19. The trajectories for the query point and nearest neighbors can be seen in Figure 15 below.

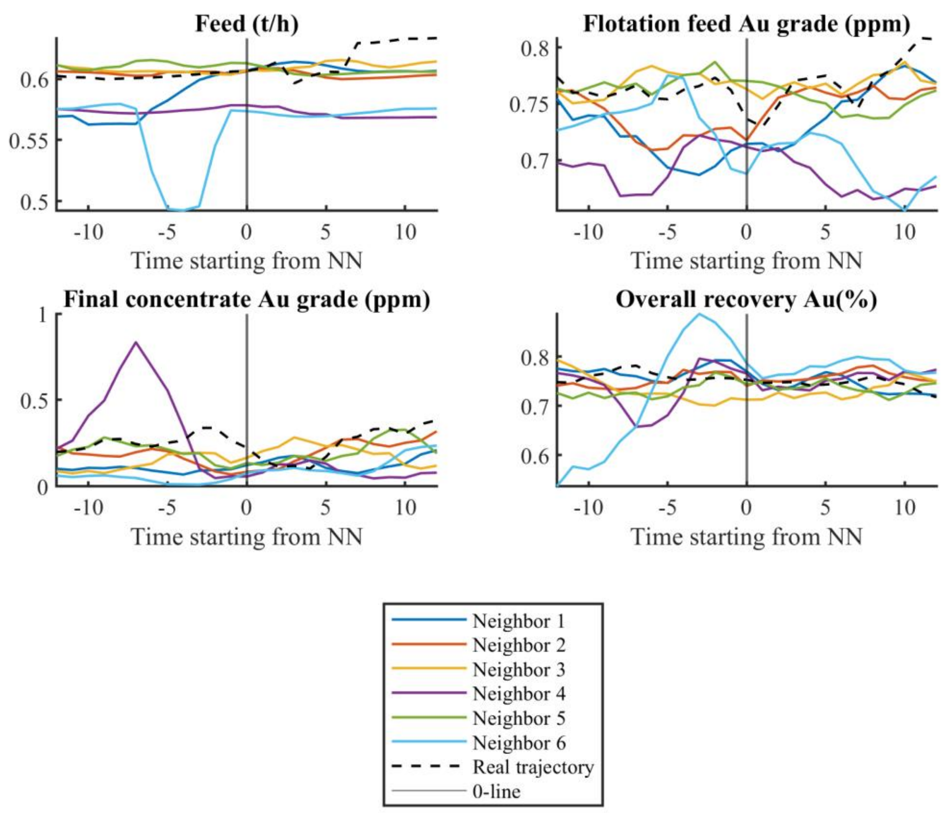

Figure 15. Trajectories for query point 5584 and nearest neighbors (4 var.).

From the trajectories above, it can be seen that neighbors 4 and 6 differ from the other neighbors and real trajectory in terms of behaviour and values. For example, neighbor 4 shows significantly different behaviour for the final concentrate Au grade when compared to other neighbors and the real trajectory. Table 2 below shows the summarized results for 2-, 3-, and 4-variable kNN searches at query point 5584. Nearest neighbors are sorted from nearest to farthest, NN 1 being the nearest. Timestamps that occurred across 2-, 3- or 4-variable searches are written in bold.

Table 2. Results for query point 5584 kNN search.

Query point (5584)	2 variables	3 variables	4 variables
NN 1	2228	**5519**	**5519**
NN 2	5048	5559	5569
NN 3	5222	3303	5535
NN 4	3427	3118	5475
NN 5	**5519**	5572	5547
NN 6	4019	3166	5452
Distance index for NN 1	0.0027	0.0095	0.0095
Distance index for NN 2	0.0029	0.0114	0.0203
Distance index for NN 3	0.0038	0.0133	0.0268
Distance index for NN 4	0.0039	0.0163	0.0285
Distance index for NN 5	0.0043	0.0188	0.0321
Distance index for NN 6	0.0049	0.0215	0.0330
NNs removed due to threshold	1	5	3

It can be seen in Table 2 that timestamp 5519 was found as a neighbor in all kNN search results for this query point. It can also be observed that the distance index grows significantly in the 4-variable search between NN 1 and NN 2 when compared to the 2- and 3-variable results. As mentioned earlier, five neighbors were replaced in the 3-variable search due to the threshold argument.

The results for the third query point presented in Figure 4, 5158, are summarized in Table 3 below. Again, timestamps that occurred across the 2-, 3- or 4-variable searches are written in bold. Figures for the results, consisting of a scatter plot, distance indexes and trajectories can be found in Appendices 3–6, in Figures 20–26. Interestingly, the timestamp 5001 occurs in the 2-, 3- and 4-variable kNN searches, but not as the nearest or even second nearest neighbor.

Table 3. Results for query point 5158 kNN search.

Query point (5158)	2 variables	3 variables	4 variables
NN 1	5107	**5108**	**5108**
NN 2	4102	5064	5120
NN 3	5030	**5001**	5061
NN 4	3906	5127	4962
NN 5	5017	5090	**5001**
NN 6	**5001**	1658	4907
Distance index for NN 1	0.0024	0.0124	0.0124
Distance index for NN 2	0.0039	0.0154	0.0242
Distance index for NN 3	0.0040	0.0155	0.0358
Distance index for NN 4	0.0056	0.0163	0.0365
Distance index for NN 5	0.0071	0.0169	0.0378
Distance index for NN 6	0.0084	0.0191	0.0380
NNs removed due to threshold	0	2	2

The results for the fourth query point 7435 are summarized below in Table 4. Figures for the results, containing a scatter plot, distance indexes and trajectories can be found in Appendices 6-9, in Figures 27-33. It can immediately be observed that the distance indexes are significantly greater in this case when compared to the query points presented earlier. It can also be seen that many timestamps occur again, despite the fact that the number of variables is increased in the kNN search. Also, there is quite a large gap in time between the query point and the nearest neighbors.

Table 4. Results for query point 7435 kNN search.

Query point (7435)	2 variables	3 variables	4 variables
NN 1	**2066**	**2066**	2064
NN 2	**4439**	**4439**	**4297**
NN 3	4300	**4297**	4439
NN 4	4588	3914	**4267**
NN 5	3960	**4267**	1678
NN 6	**5225**	**5225**	**5225**
Distance index for NN 1	0.0194	0.0197	0.0844
Distance index for NN 2	0.0389	0.0581	0.0940
Distance index for NN 3	0.0863	0.0940	0.1351
Distance index for NN 4	0.1010	0.1229	0.1564
Distance index for NN 5	0.1012	0.1564	0.1754
Distance index for NN 6	0.1119	0.1596	0.1767
NNs removed due to threshold	3	4	5

The results for the fifth and final query point 6980 are summarized in Table 5 below. Figures for the results, comprising a scatter plot, distance indexes and trajectories can be found in Appendices 10–13, in Figures 34–40. Based on Table 5, these results offer some similarities to query point 7018 (Table 1). Also, it can be observed that one timestamp (6955) appears in the results for the 2-, 3- and 4-variable searches.

Table 5. Results for query point 6980 kNN search.

Query point (6980)	2 variables	3 variables	4 variables
NN 1	2179	6292	**6955**
NN 2	2336	6757	6843
NN 3	6812	6208	6705
NN 4	**6955**	6744	6814
NN 5	2276	6970	6718
NN 6	6851	**6955**	6969
Distance index for NN 1	0.0013	0.0090	0.0165
Distance index for NN 2	0.0014	0.0097	0.0276
Distance index for NN 3	0.0019	0.0104	0.0279
Distance index for NN 4	0.0025	0.0109	0.0301
Distance index for NN 5	0.0035	0.0115	0.0313
Distance index for NN 6	0.0041	0.0119	0.0323
NNs removed due to threshold	3	0	4

5 DISCUSSION

It should be mentioned that, due to the way the kNN algorithm was emphasised in this work, no characteristic output of the algorithm can be used to validate the reliability of the method. However, this thesis does not concentrate on the model itself, but rather on the implementation of the selected methods and data-based modelling.

In this thesis, the scatter plot was employed as a useful visualization tool for 2-variable kNN search cases. A scatter plot can be used to confirm that the nearest neighbors found by the algorithm surround the query point, thus being the closest matches from the history data for the test sample. In Figure 6, it can be observed that query point 7018 and the nearest neighbors are in a dense cluster that contains a large number of timestamps. This indicates that at least with the two variables (flotation feed and final concentrate Au grade) involved, this query point represents a common process condition. Scatter plots for query points 5584 and 6980 which can be found in Appendices 1 and 10, Figures 16 and 34 also show that these query points are in a dense cluster of timestamps. In contrast, scatter plots for query points 5158 (Appendix 3 Figure 20) and especially 7435 (Appendix 6 Figure 27) indicate uncommon process situations. From the scatter plot for query point 5158, the query point and nearest neighbors are in a sparse cluster next to the densest cluster with the variables, which indicates a slight deviance from the normal operating point. From the scatter plot for query point 7435, it can be observed that the query point is not located in any prominent cluster and neighbors are distributed very sparsely, which indicates a very rare process situation. It should be mentioned that the kNN algorithm requires a lot of training data to function properly. When the size of the training dataset is increased, meaning in this work more timestamps that can be found as nearest neighbors for the query point, the probability to find neighbors that correspond to the query point is also increased. However, the amount of the training data should not be a problem if the kNN algorithm is applied to an online process with comprehensive history data that can be used as training data.

With distance index values, performance of each neighbor search can be evaluated. The distance index values presented in Chapter 4 indicate the distance between the query point and its nearest neighbors. The smaller the distance index value, the closer the neighbors are to the query point. The neighbors found also correspond better to the query point when the distance index is smaller. From the results it can be concluded that, when the number of variables is increased in the kNN search algorithm, the values of the distance indexes also increase. This is expected and does not necessarily mean that the results become worse in terms of correspondence, as can be seen when comparing the trajectory figures for the 2-, 3- and 4-variable searches presented in the previous chapter. However, this means that only results with the same number of variables involved should be compared against each other. The case with query point 7435 stands out, with its significantly large distance index values when comparing the 2-, 3- and 4-variable search results across the other query points. This indicates that query point 7435 represents a unique process situation, or a transition state in the process. In Figure 29 in Appendix 7, which is the trajectory figure for the 2-variable kNN search for query point 7435, the trajectories for the nearest neighbors are also far from the trajectory for the query point. In contrast, with query points 7018 and 6980, the distance index values are relatively small when compared to other query point results.

It should be mentioned that the kNN function used in this study did not have a defined threshold for the maximum distance index value. One simple improvement related to large distance index values *e.g.,* in the case of query point 7435, could be a limit for the maximum distance index, or a threshold after which the kNN function gives an alarm about a significantly large distance index value. This threshold limit should be defined separately for 2-, 3- and 4-variable kNN searches, as the distance index values will inevitably grow when the number of variables increases.

The trajectory figures in Chapter 4 and the appendices serve as a visual confirmation of the kNN search results. The trajectories show whether the nearest neighbors are close to the query point or not in the variable values. They also show, depending on the *n_step* parameter, how the values for the query point and nearest neighbors change over time, and whether the trajectories of the nearest neighbors share similar behaviour with the query point trajectory or not. As the objective of this study was to find the closest correspondences from history data for the query point/current process situation, ideally all the nearest neighbors found by the algorithm would have similar values to the query point and follow the query point trajectory closely in relation to time. A trajectory inspection of the results becomes especially important if the training dataset includes NaN values, as seen in Figure 13. The problem with NaN values is that, hypothetically with the kNN function used in this work, it is possible that the algorithm finds a neighbor with multiple NaN values before and after the timestamp. This can happen because the algorithm only searches for individual timestamps. It should be noted that, in an online situation, where nearest neighbors are sought for a current process situation, the trajectory for the query point cannot be plotted forwards, since measurements from the future situation will not exist. However, due to the nearest neighbors, a future trajectory for the current process situation can be estimated and even manipulated based on an input analysis of the nearest neighbors.

Tables 1–5 provide a summary of the results of the kNN searches for the query points presented in Figure 4. With the tables, it is easy to see the kNN algorithm’s performance at each query point with distance index values and compare the results to each other. In the tables, timestamps that occur with different number of variables involved in search are written in bold font for ease of interpretation. With all the query points presented in the results, at least one timestamp reappeared across the results when the number of KPI variables involved in search was increased. One explanation for this phenomenon is that the KPI variables involved in the search are not independent from each other. Both query points 5584 and 6980 had exactly one timestamp that occurred in the 2-, 3- and 4-variable cases: 5519 for query point 5584 and 6955 for 6980. For query point 7435, which was not near any cluster based on the Appendix 6 Figure 27 scatter plot and offered the worst results in terms of distance index values, five timestamps reappeared when the number of variables was increased. These facts indicate even more strongly that query point 7435 represents a highly unique process situation.

The number of timestamps excluded from the results due to the threshold argument can also be seen for each case in Tables 1–5. Without the threshold argument, for most query points the results would contain neighbors very close to each other, which would not be ideal. This happens because in time series data, subsequent observations are dependent on each other. Test runs without the threshold argument proved that, if the algorithm found two subsequent neighbors as a result, the trajectories of these neighbors would naturally behave in the same manner, and they would just be time-shifted copies of each other. The addition of the threshold argument adds quality to the results, as it ensures that the neighbors found are independent from each other. For the query points presented in the results, 5584 with 3-variable search and 7435 with 4-variable search both had five neighbors excluded from their results due to the threshold argument. In the case of query point 5584, which is in a dense cluster (see Figure 16 in Appendix 1), the exclusion of five neighbors does not significantly raise the distance index values, or the fact that the trajectories of the nearest neighbors in Figure 13 are relatively close to the trajectory of the query point. The results for the 2-variable search for query point 5158 and the 3-variable search for query point 6980 had zero neighbors excluded, which is unusual, based on the other results and test runs with the algorithm.

According to Figure 4, query points 7018 and 6980 have very similar KPI variable values, with the greatest difference being in the concentrate Au grade value. Similarities between these two query points can also be observed in the scatter plots (Figure 6 and Figure 34 in Appendix 10) and Tables 1 and 5, which contain the results of kNN searches for these query points. From the scatter plots it can be concluded that both query points belong to the same dense cluster. Tables 1 and 5 show that, for both query points, when only 2 variables were involved, the search found timestamps from 2100-2350 as neighbors, but when the number of variables was increased, these timestamps disappeared from the results. One explanation for the similarities could be that the time difference between these query points is only 38 hours, and as seen from the scatter plots, the process stayed in the same state during this time period. One possible reason for the disappearance could be that when the algorithm is introduced with a new variable, this new variable steers the search towards a slightly different process situation.

At the stage in which it was used in this study, the kNN function was not sophisticated enough to be used in process control or optimisation independently; the function only searches for individual timestamps for nearest neighbors and does not take the behaviour or trend of neighbors into account. In an ideal situation, the trajectories of nearest neighbors would follow the query point’s trajectory as closely as possible until there were no more measurements for the current process situation, in order to make more accurate predictions.

With the addition of clustering, such as k-means as a preliminary classification for kNN, more information about the query point could be gained, especially when more than two variables were involved in the kNN search. As a simple example, with k-means clustering, the flotation feed KPI could have been divided into clusters with high feed days and low feed days.

If the kNN function built in this study is used as an expert system, the dataset used by the function as training data should constantly be updated with new data from the process, which would require some adjustments to the function. To make the kNN function search for closest or desired trajectories, a few improvements to the function are possible, which are discussed below.

The simplest improvement to the kNN function in order to search for desired trajectories would be to rank the neighbors' trajectories based on their trend with respect to the query point's trajectory. This can be done for example by calculating the area between the query point's trajectory and the neighbors' trajectories with an integral and ranking the neighbor with the smallest area as best. One other option would be that, in addition to searching for individual timestamps for nearest neighbors, the kNN function would search for neighbors with similar gradients and values. A search based on a time-based feature calculation for the query point (current situation) and history data is also a possibility, for example in four-hour time periods.

6 CONCLUSIONS

The main objective in this thesis was to identify and evaluate the applicability of machine learning methods that could be used in mineral processing as a tool for forecasting the direction in which where the current process values are going. The supervised machine learning method k-nearest neighbors (kNN) was selected for finding the closest correspondences from history data for the current process situation.

The results showed that, when neighbors are sought for a query point where KPI variables have uncommon values, the distance index values can become very large and the nearest neighbors can be quite far from the query point in values. Based on the testing and results, further testing with the function is needed to determine accurate limits for the distance index values.

The kNN implementation built in this study proved to be sufficient in finding the closest correspondence from history data for a current process situation. As it stands, the function could be used as part of a mine-to-mill optimisation expert system to predict the direction in which the process KPI variable values are going. From the trajectories of nearest neighbors, a neighbor with the desired trend could be selected and used to analyse the process inputs to determine the process variable values for the current process situation.

In future, it would also be possible to try a more advanced ML method instead of kNN in the search for nearest matches for the current process situation from history data. In this study, kNN was found to be a good starting point as a method for use in this kind of application. In order to extend the approach to forecasting, for example kNN regression, which is briefly mentioned in section 2.4.3, could be a noteworthy contender. Another possible ML method could be the LSTM model mentioned in section 2.4.4, which has been used in forecasting concentrate purities in the flotation process.

7 SUMMARY

The objective of this thesis was identification of ML methods that could be used in mineral processing as a tool for forecasting, how the current process values are going to be changing. Chapter 2 contains the theory related to the topics of this thesis. A generalized process flowsheet for the target plant is presented, followed by a more in-depth review of flotation process phenomena. The basics of mine-to-mill optimisation, an important aspect of today's mineral processing, are introduced. In section 2.3, after a brief introduction to AI and ML in general, the open standard data mining approach CRISP-DM is introduced. The importance of data preparation is also considered, in addition to a more in-depth review of ML methods and current ML applications in mineral processing.

The chapter on data preparation and ML implementation describes the data that was used in this thesis, steps for data pre-processing and the building of the k-nearest neighbors function. The pre-processing, or data preparation, in this thesis included synchronization between the datasets, renaming of measurement tags, converting data in on/off format to binary data, addition of calculational variables (valve sum vectors, recovery % for gold, iron and arsenic, and flotation dry feed) to the datasets, definition of data ranges for 3-sigma outlier removal with visual examination, data normalization and resampling. Data pre-processing and writing of the kNN function were done in MATLAB®. The kNN function was built around MATLAB's pre-built *ExhaustiveSearcher* and *knnsearcher* functions, which required the statistics and machine learning toolbox. One-hour averaged process data was used for the calculations with the kNN function. With query point and k-value being the main inputs, the function gives timestamps of the nearest neighbors and distance index values as an output. In addition to these outputs, the function draws a scatter plot for the variables, emphasizing the query point and nearest neighbors, and the trajectories of the nearest neighbors for a defined time period as an output. In this thesis, kNN search was done separately with two, three and four variables, which were the process KPI variables, selected based on expert knowledge. In this case the KPIs were the flotation dry feed, flotation feed Au grade, final concentrate Au grade and Au recovery %.

Chapters 4 and 5 contain the results and discussion. Results for five query points are presented, which were selected based on visual examination of a gold recovery graph. The query points were also verified as having some difference in KPI values with the

MATLAB® community-made radar chart function. In general, the kNN function performed the best when the KPI values at the query point had common values for the process. In these situations, the values for the nearest neighbors were relatively close to the KPI values of the query point, considering that the kNN search took four variables into account, and offered a good prediction of the direction in which the KPI values for the query point were going. However, in situations where the query point's KPI values were uncommon based on the scatter plot, the values for the nearest neighbors were far from the query point's values, and the prediction accuracy was poor, which is to be expected due to the lack of such situations in the history data.

The kNN function built in this thesis proved to be sufficient in finding the closest correspondence for the query point from the history data and could be used as part of a mine-to-mill optimisation expert system to predict where the process KPI variable values are going. From the trajectories of the nearest neighbors, a neighbor with the desired trend could be selected and included in the input analysis to determine process variable values for the current process situation. However, the function could be improved with the addition of k-means clustering as a preliminary classification for kNN, to obtain more information on the query point and nearest neighbors, especially when more than two variables are included in the kNN search. The function could also be improved by ranking the nearest neighbors based on the area between a neighbor and the query point for a selected time period, instead of the distance index in a single timestamp, to take into account the trend of trajectories for the neighbors and query point. More advanced ML methods could also be used instead of kNN, for example the LSTM model used in a case study by Pu *et al.* to predict concentrate purities in the flotation process.

REFERENCES

Barragán-Montero, A. et al., 2021. Artificial intelligence and machine learning for medical imaging: A technology review. Physica Medica, 83, pp. 242–256. doi:10.1016/j.ejmp.2021.04.016.

Crozier, R.D., 1992. Flotation: theory, reagents and testing. Available at: https://1lib.sk/book/2971609/9828b5?dsource=recommend [Accessed 11 January 2022].

Erkayaoğlu, M., 2015. A Data Driven Mine-To-Mill Framework For Modern Mines. Available at: https://repository.arizona.edu/handle/10150/579114 [Accessed 2 November 2021].

Haavisto, O., 2009. Reflectance Spectrum Analysis of Mineral Flotation Froths and Slurries. Available at: http://lib.tkk.fi/Diss/2009/isbn9789522482372/ [Accessed 11 January 2022].

Hadler, K., Smith, C.D. and Cilliers, J.J., 2010. Recovery vs. mass pull: The link to air recovery. Minerals Engineering, 23(11), pp. 994–1002. doi:10.1016/j.mineng.2010.04.007.

Horn, Z.C. et al., 2017. Performance of Convolutional Neural Networks for Feature Extraction in Froth Flotation Sensing. IFAC-PapersOnLine, 50(2), pp. 13–18. doi:10.1016/j.ifacol.2017.12.003.

Jaggia, S. et al., 2020. Applying the CRISP-DM Framework for Teaching Business Analytics. Decision Sciences Journal of Innovative Education, 18(4), pp. 612–634. doi:10.1111/dsji.12222.

Kawatra, S., 2011. Fundamental principles of froth flotation, SME mining engineering handbook, pp. 1517–1532.

Kawatra, S.K. and Young, C., 2019. SME Mineral Processing and Extractive Metallurgy Handbook. Englewood, Colorado: Society for Mining, Metallurgy, and Exploration, Inc. Available at: http://pc124152.oulu.fi:8080/login?url= [Accessed 2 November 2021].

Kortelainen, J., 2019. Utilization of digital twin in model-based control of flotation cells. Available at: https://lutpub.lut.fi/handle/10024/160275 [Accessed 13 December 2021].

Li, S., Sari, Y.A. and Kumral, M., 2020. Optimization of Mining–Mineral Processing Integration Using Unsupervised Machine Learning Algorithms. Natural Resources Research, 29(5), pp. 3035–3046. doi:10.1007/s11053-020-09628-0.

Mathworks 2022. Classification Using Nearest Neighbors - MATLAB & Simulink - MathWorks Nordic. Available at: https://se.mathworks.com/help/stats/classification-using-nearest-neighbors.html [Accessed: 7 April 2022].

Matlabcentral, F. 2022. spider_plot. Available at: https://se.mathworks.com/matlabcentral/fileexchange/59561-spider_plot [Accessed 7 April 2022].

Mishra, A.K., 2021. AI4R2R (AI for Rock to Revenue): A Review of the Applications of AI in Mineral Processing. Minerals, 11(10), p. 1118. doi:10.3390/min11101118.

Muller, D., de Villiers, P.G.R. and Humphries, G., 2010. A Holistic Approach to Flotation Mass Pull and Grade Control. IFAC Proceedings Volumes, 43(9), pp. 133–136. doi:10.3182/20100802-3-ZA-2014.00031.

North, M., 2012. Data mining for the masses. S.I.: Global Text Project.

Park, J. and Kim, K., 2020. Use of drilling performance to improve rock-breakage efficiencies: A part of mine-to-mill optimization studies in a hard-rock mine. International Journal of Mining Science and Technology, 30(2), pp. 179–188. doi:10.1016/j.ijmst.2019.12.021.

Posio, J. et al., 2008. Outlier Detection for 2D Temperature Data. IFAC Proceedings Volumes, 41(2), pp. 1958–1963. doi:10.3182/20080706-5-KR-1001.00333.

Pu, Y., Szmigiel, A. and Apel, D.B., 2020. Purities prediction in a manufacturing froth flotation plant: the deep learning techniques. Neural Computing and Applications, 32(17), pp. 13639–13649. doi:10.1007/s00521-020-04773-2.

Singh, D. and Singh, B., 2022. Feature wise normalization: An effective way of normalizing data. Pattern Recognition, 122, p. 108307. doi:10.1016/j.patcog.2021.108307.

Ur Rehman, B., 2011. Modelling a Mineral Froth Flotation Process : Case Study: Minerals process at Boliden AB. Available at: http://urn.kb.se/resolve?urn=urn:nbn:se:umu:diva-51600 [Accessed 11 January 2022].

Wills, B.A., 2006. Wills' Mineral Processing Technology - 7th Edition. Available at: https://www.elsevier.com/books/wills-mineral-processing-technology/wills/978-0-7506-4450-1 [Accessed 2 November 2021].

Zhou, Z.-H., 2021. Machine Learning. Singapore: Springer Singapore. doi:10.1007/978-981-15-1967-3.

Appendix 1. Scatter plot and distances for 2-variable kNN search at query point 5584

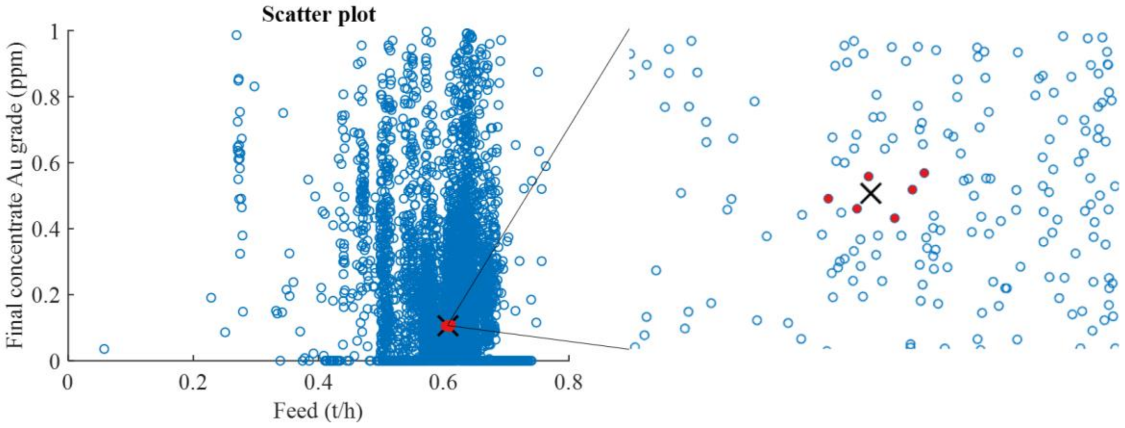

Figure 16. Scatter plot for query point 5584 (2 var.).

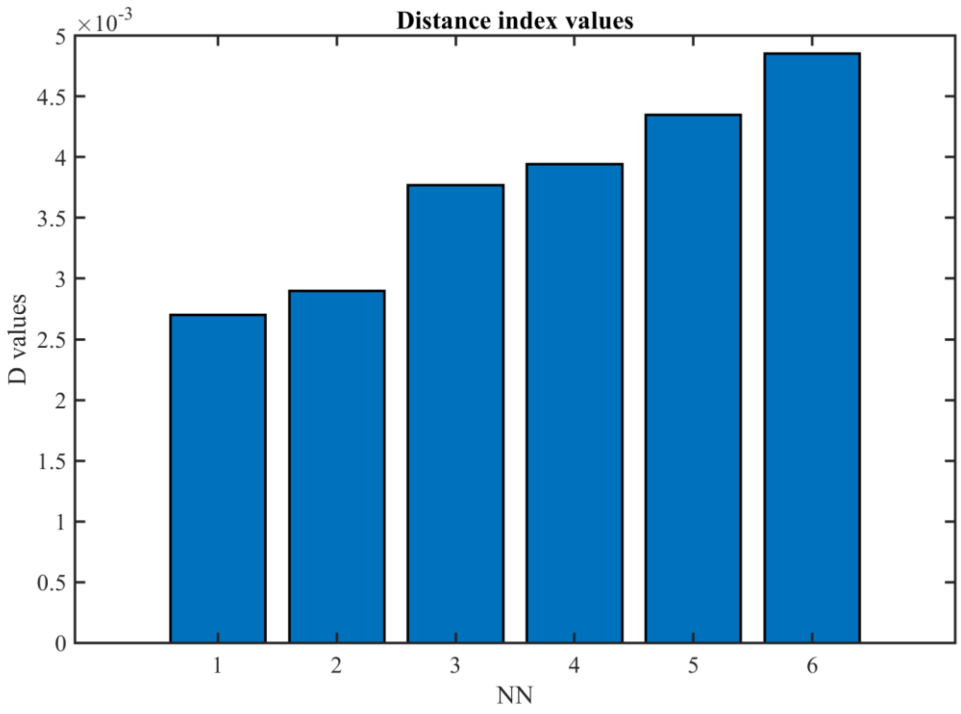

Figure 17. Distance indexes for query point 5584 (2 var.).

Appendix 2. 3- and 4-variable kNN search distance indexes for query point 5584

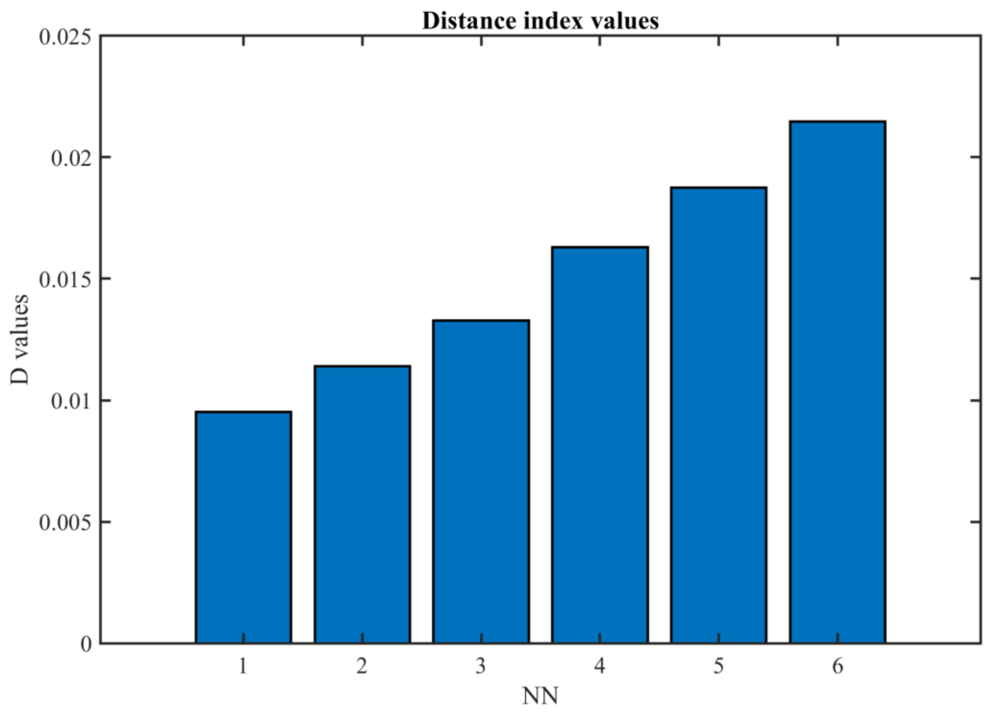

Figure 18. Distance indexes for query point 5584 (3 var.).

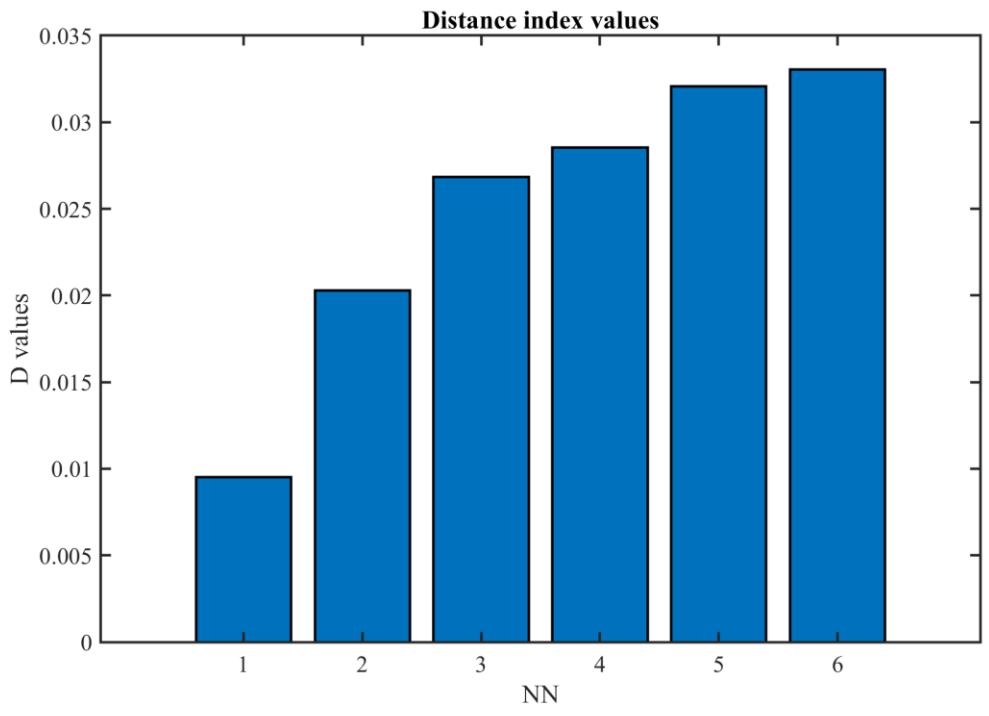

Figure 19. Distance indexes for query point 5584 (4 var.).

Appendix 3. Scatter plot and distances for 2-variable kNN search at query point 5158

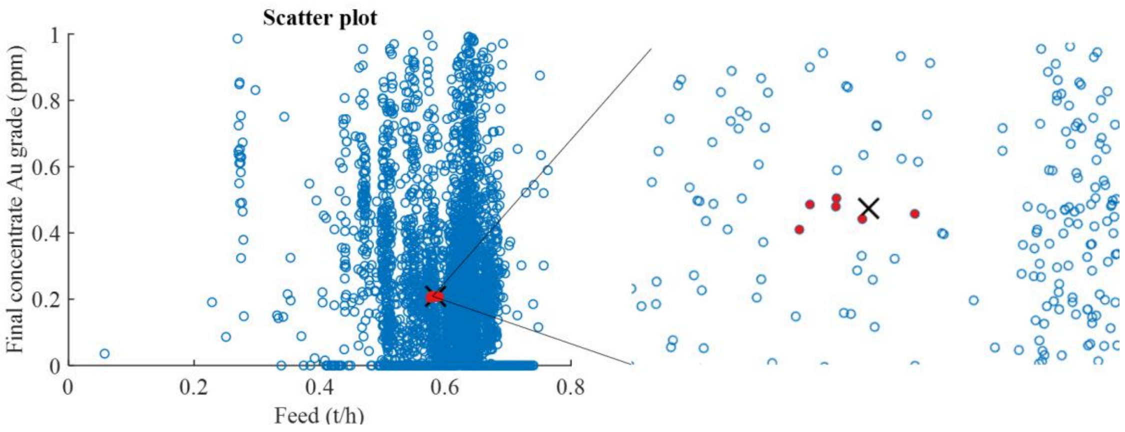

Figure 20. Scatter plot for query point 5158 (2 var.).

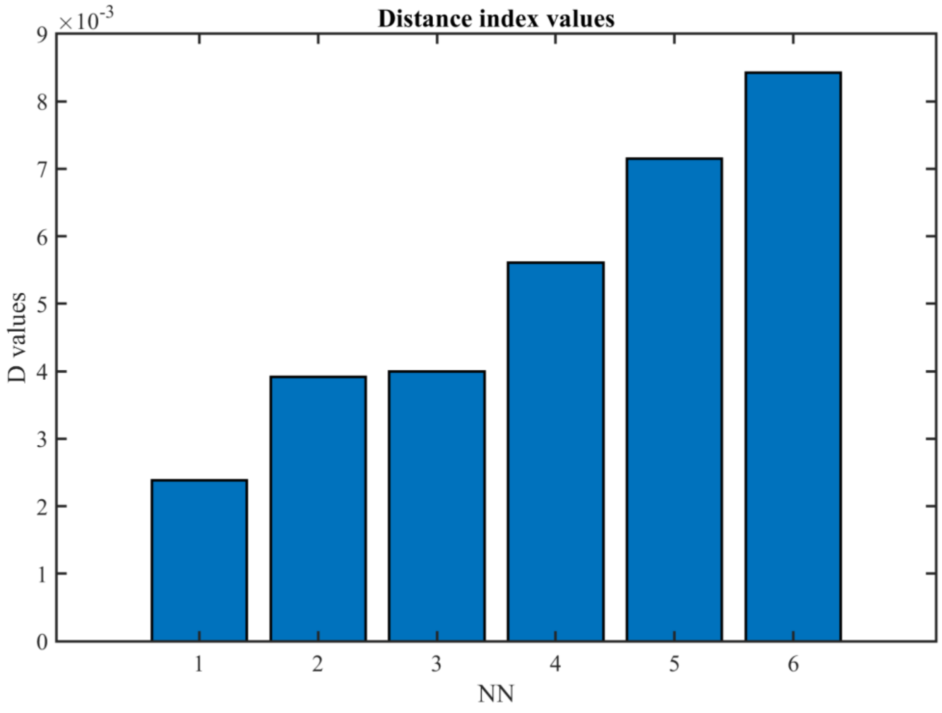

Figure 21. Distance indexes for query point 5158 (2 var.).

Appendix 4. Trajectories for 2-var. search in 5158 and distances for 3-var. search

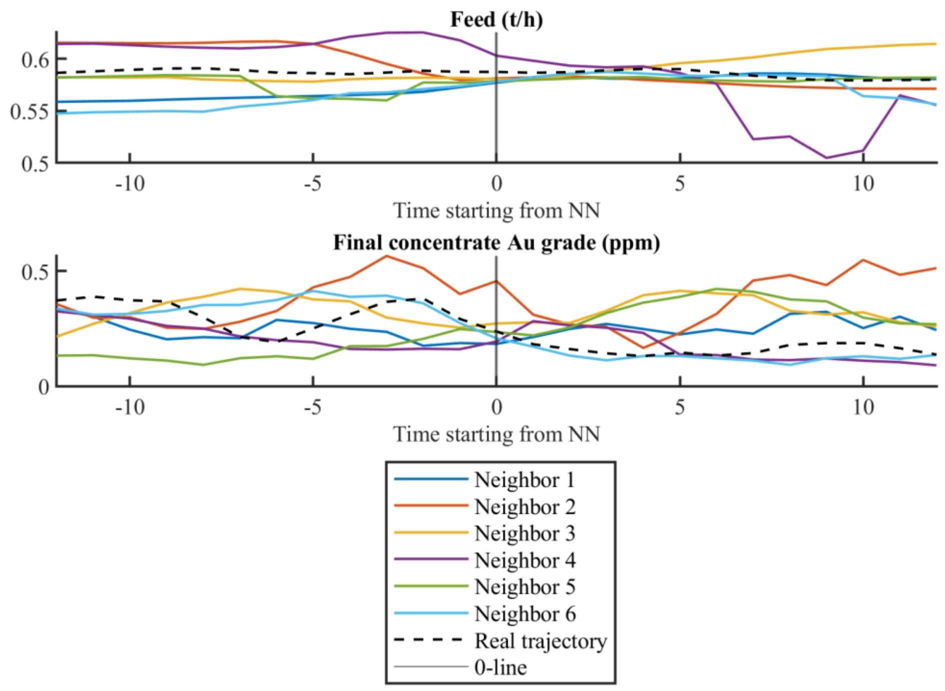

Figure 22. Trajectories for query point 5158 and nearest neighbors (2 var.).

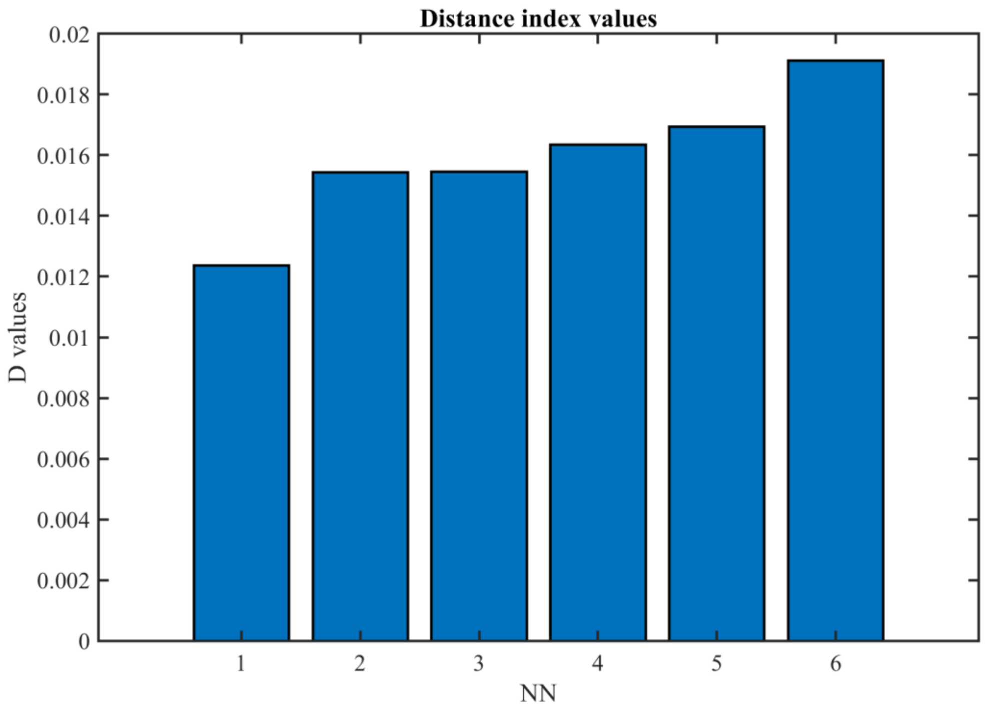

Figure 23. Distance indexes for query point 5158 (3 var.).

Appendix 5. Trajectories for 3-var. search in 5158 and distances for 4-var. search

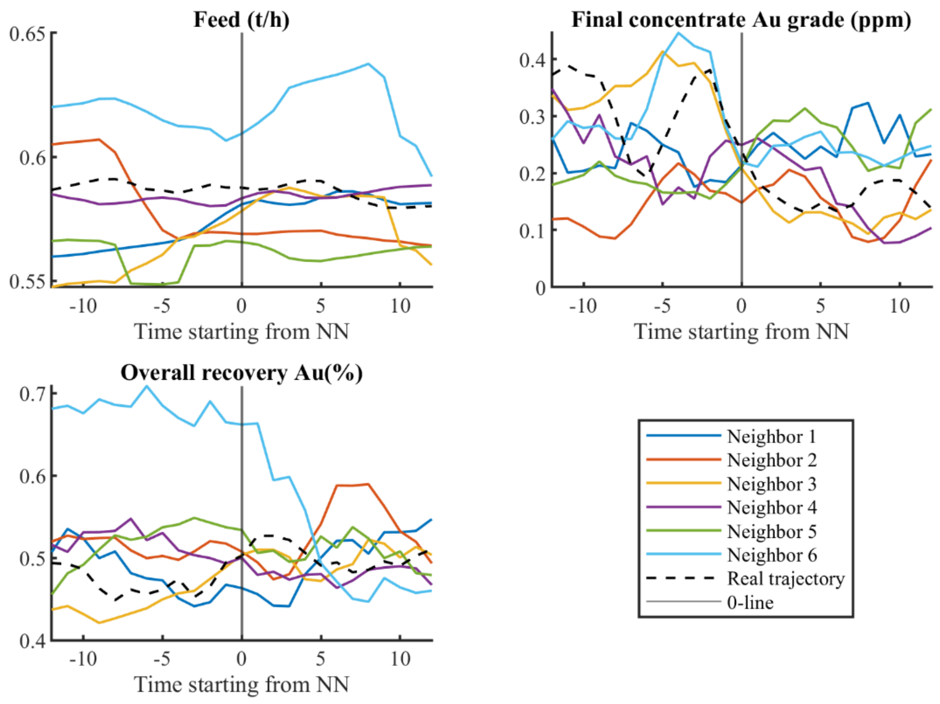

Figure 24. Trajectories for query point 5158 and nearest neighbors (3 var.).

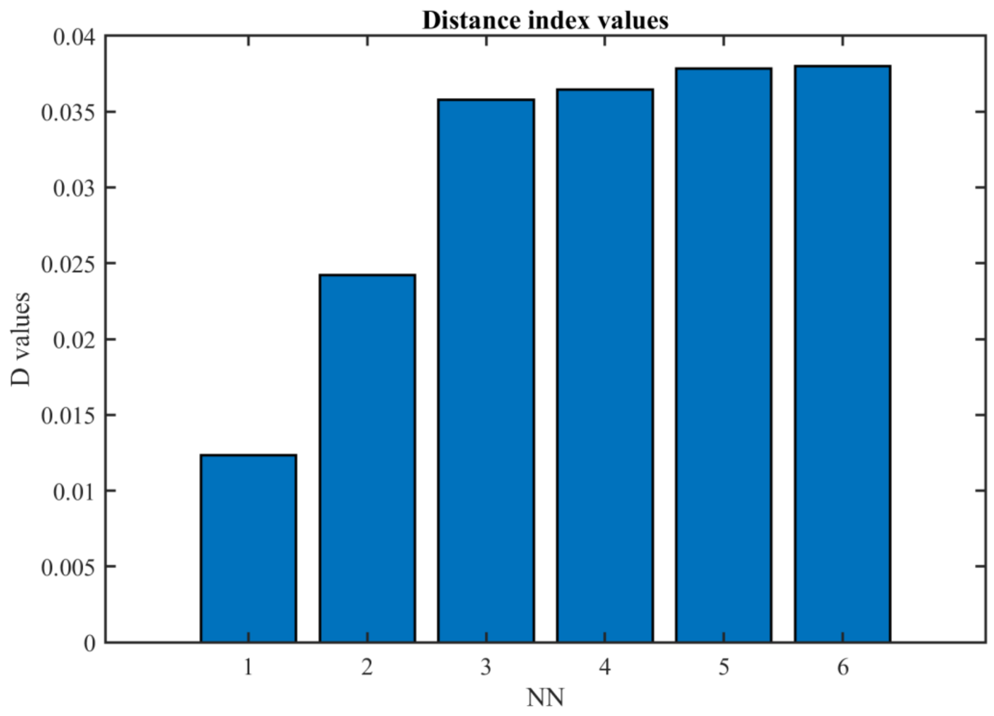

Figure 25. Distance indexes for query point 5158 (4 var.).

Appendix 6. Trajectories for 4-var. search in 5158, and scatter plot for query point 7435

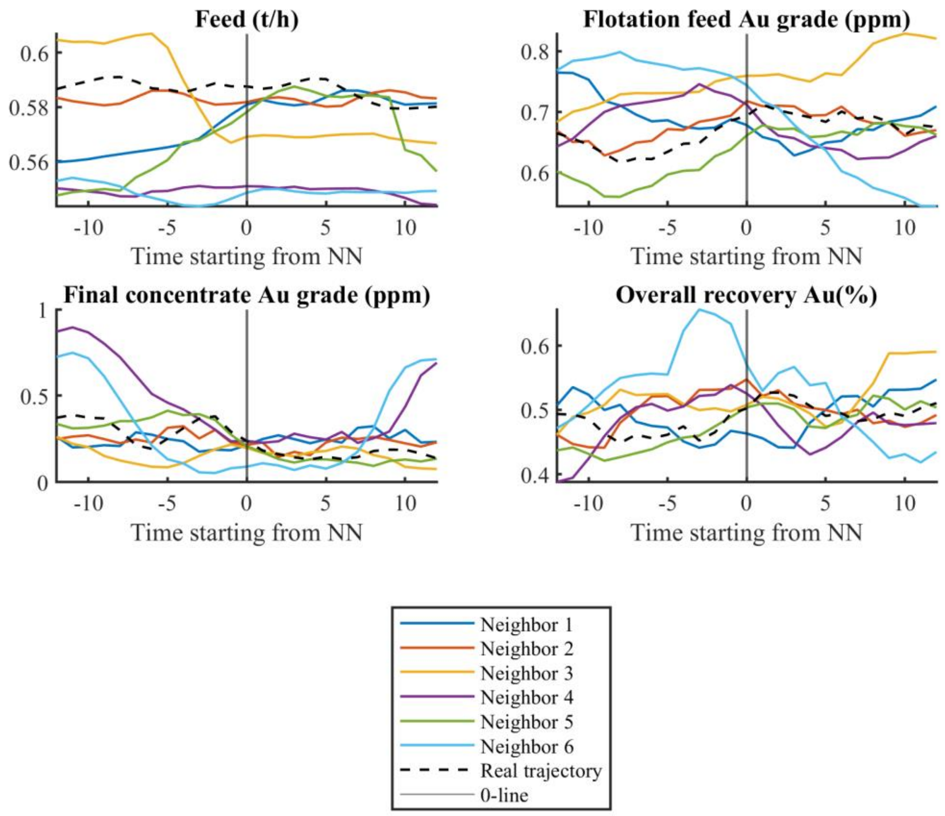

Figure 26. Trajectories for query point 5158 and nearest neighbors (4 var.).

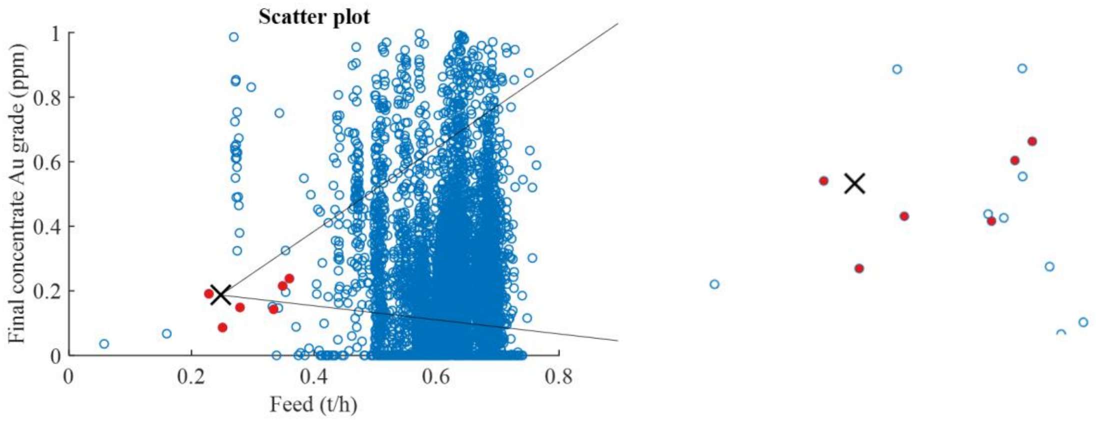

Figure 27. Scatter plot for query point 7435 (2 var.).

Appendix 7. Distances and trajectories for 2-var. search in 7435

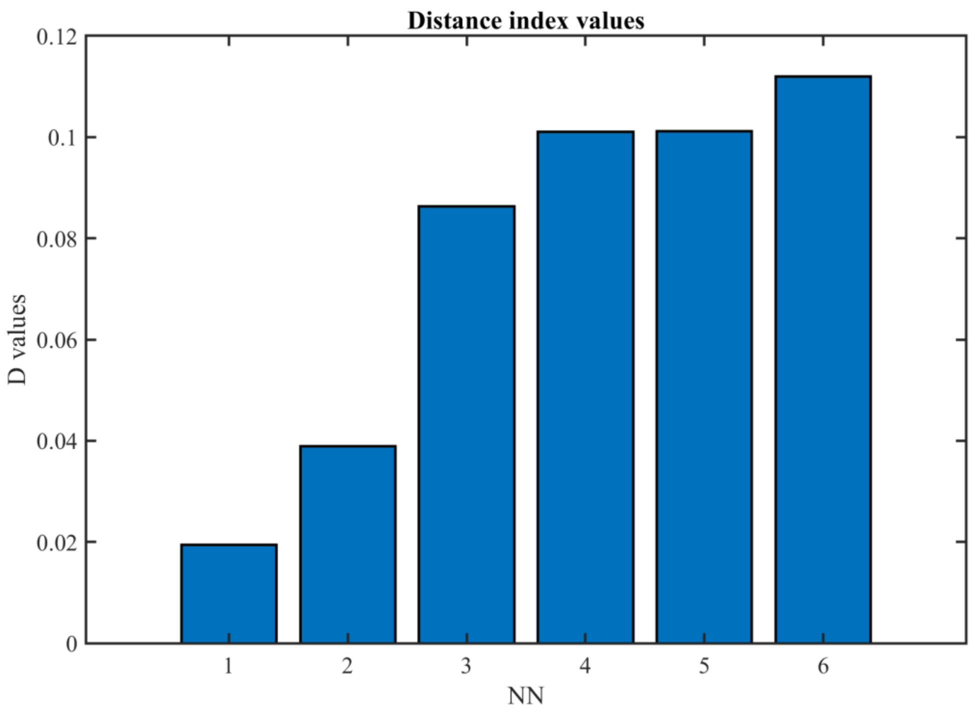

Figure 28. Distance indexes for query point 7435 (2 var.).

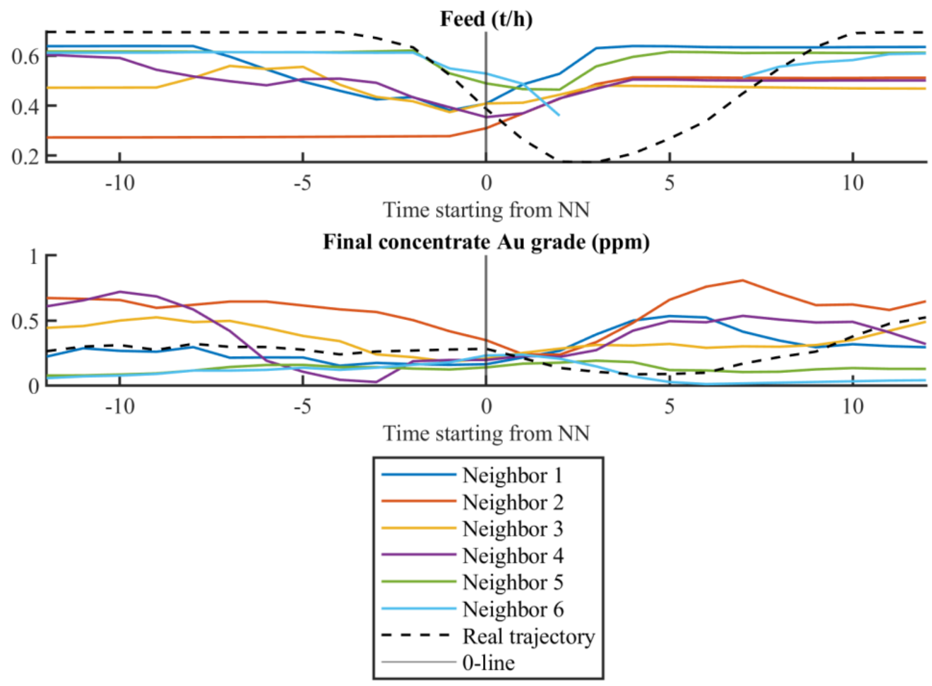

Figure 29. Trajectories for query point 7435 and nearest neighbors (2 var.).

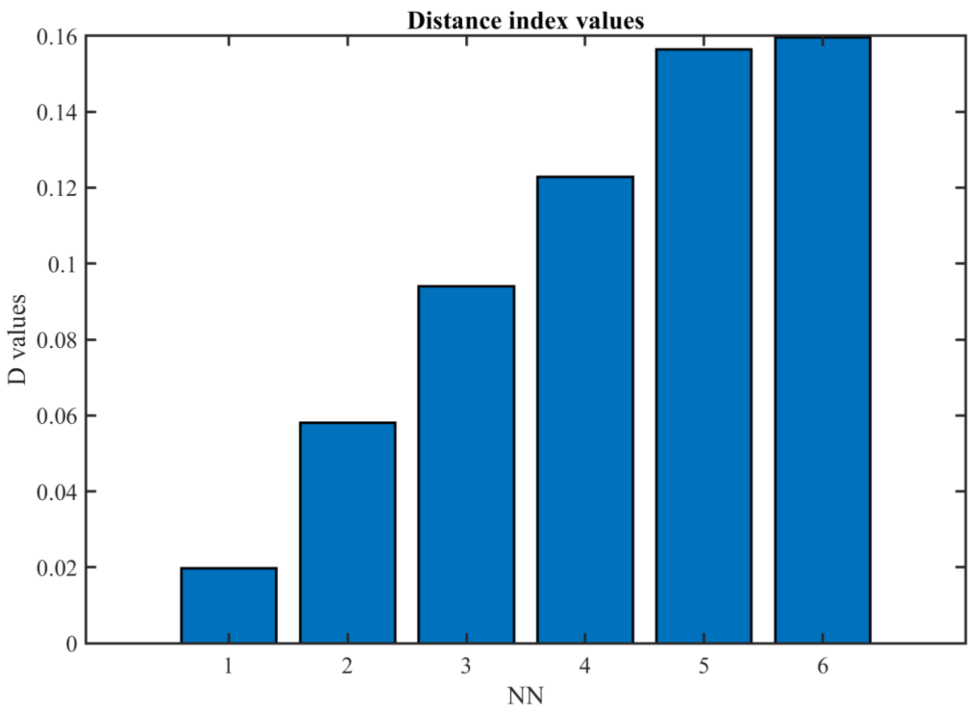

Figure 30. Distance indexes for query point 7435 (3 var.).

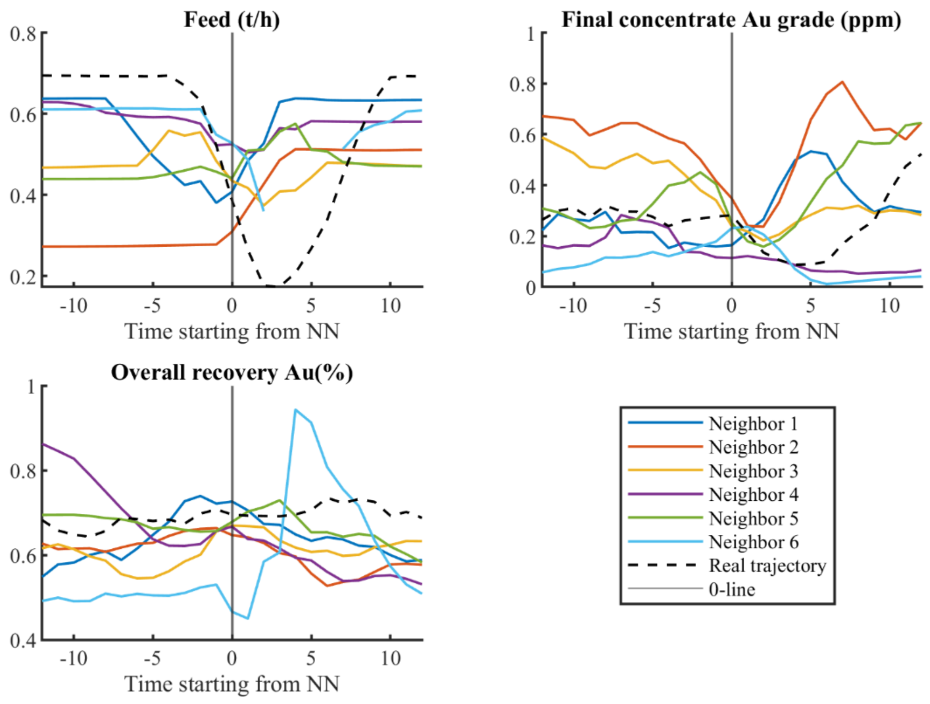

Figure 31. Trajectories for query point 7435 and nearest neighbors (3 var.).

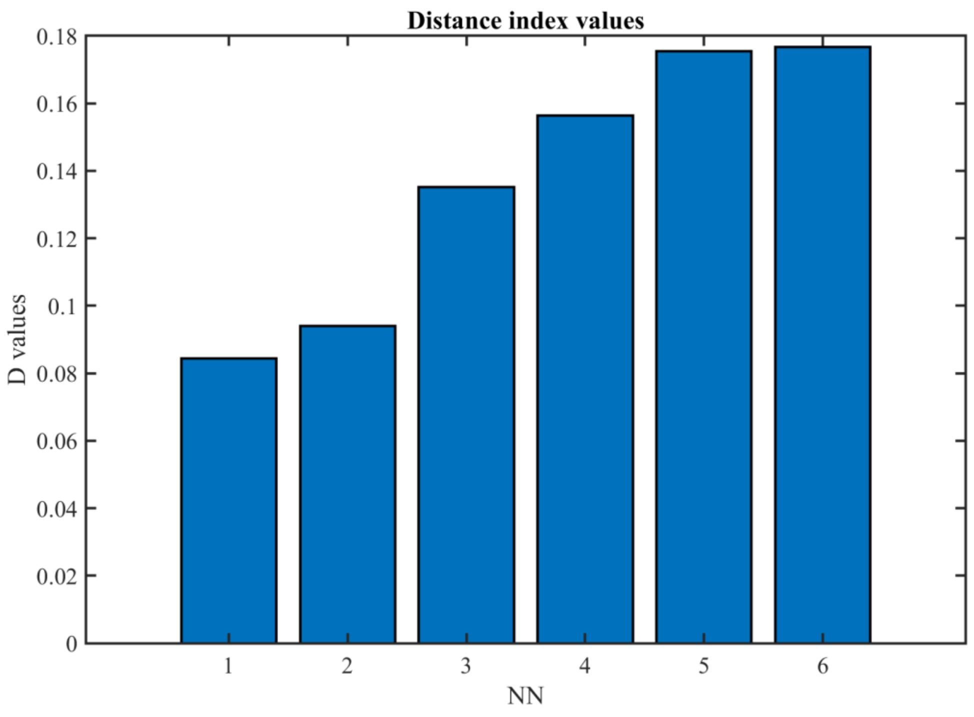

Figure 32. Distance indexes for query point 7435 (4 var.).

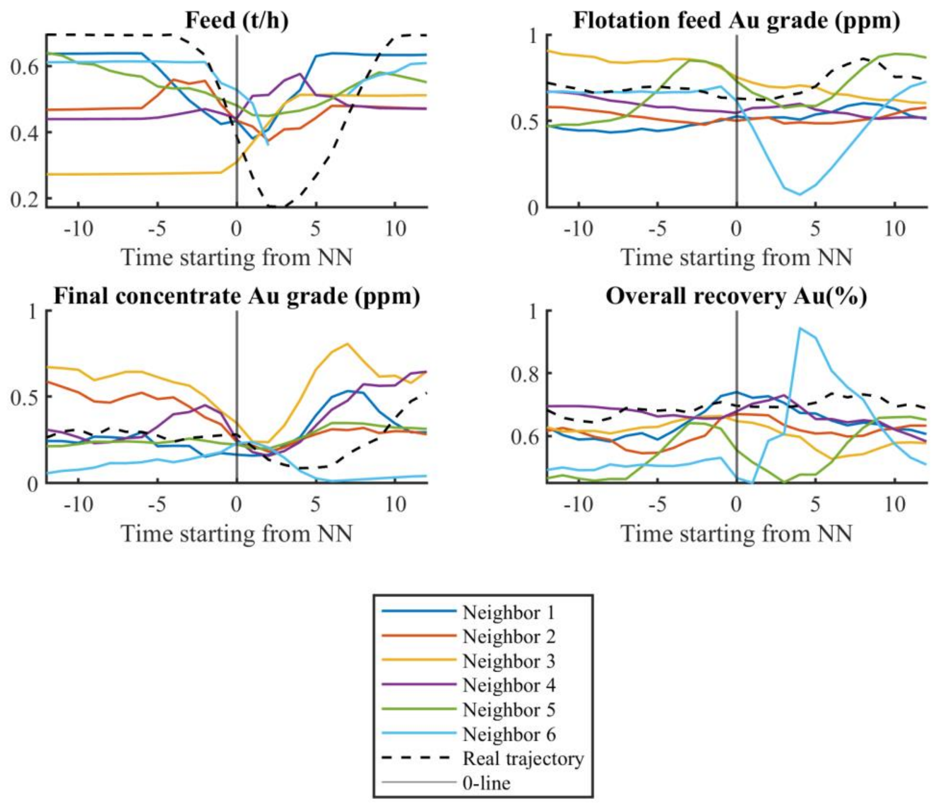

Figure 33. Trajectories for query point 7435 and nearest neighbors (4 var.).

Appendix 10. Scatter plot and distances for 2-var. search in query point 6980

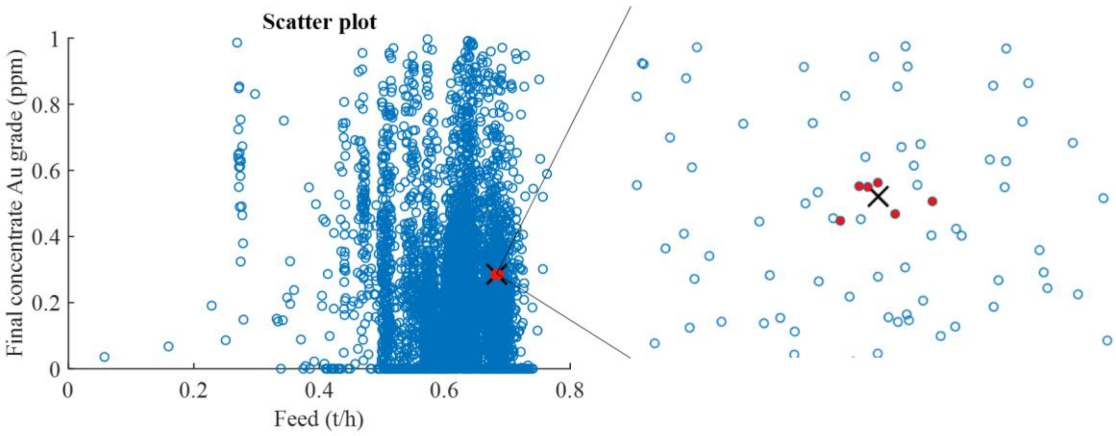

Figure 34. Scatter plot for query point 6980 (2 var.).

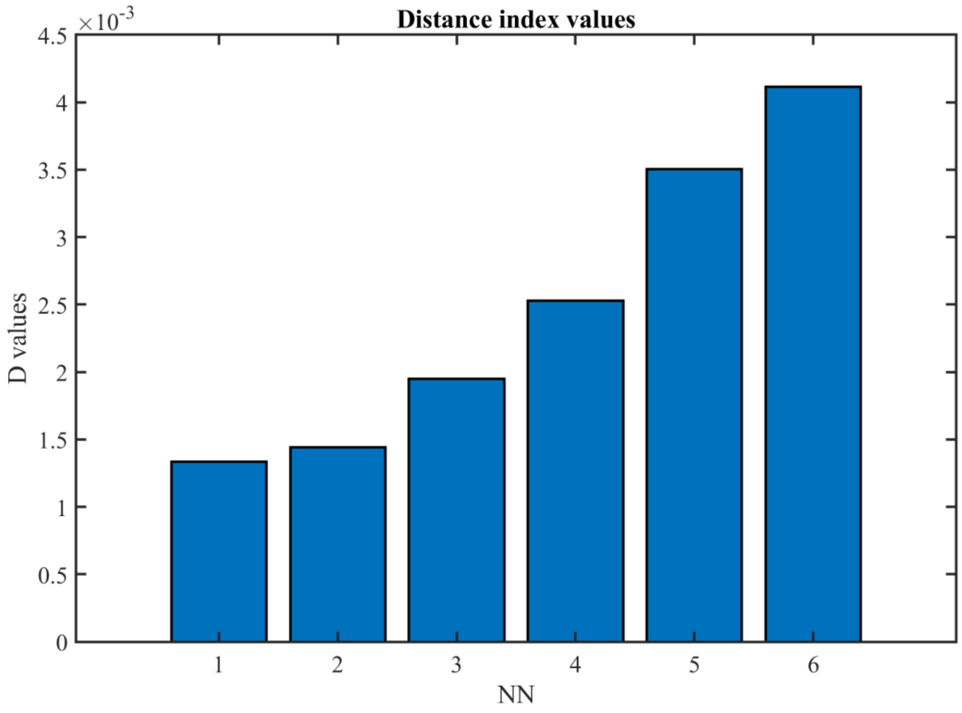

Figure 35. Distance indexes for query point 6980 (2 var.).

Appendix 11. Trajectories for 2-var. search in 6980 and distances for 3-var. search

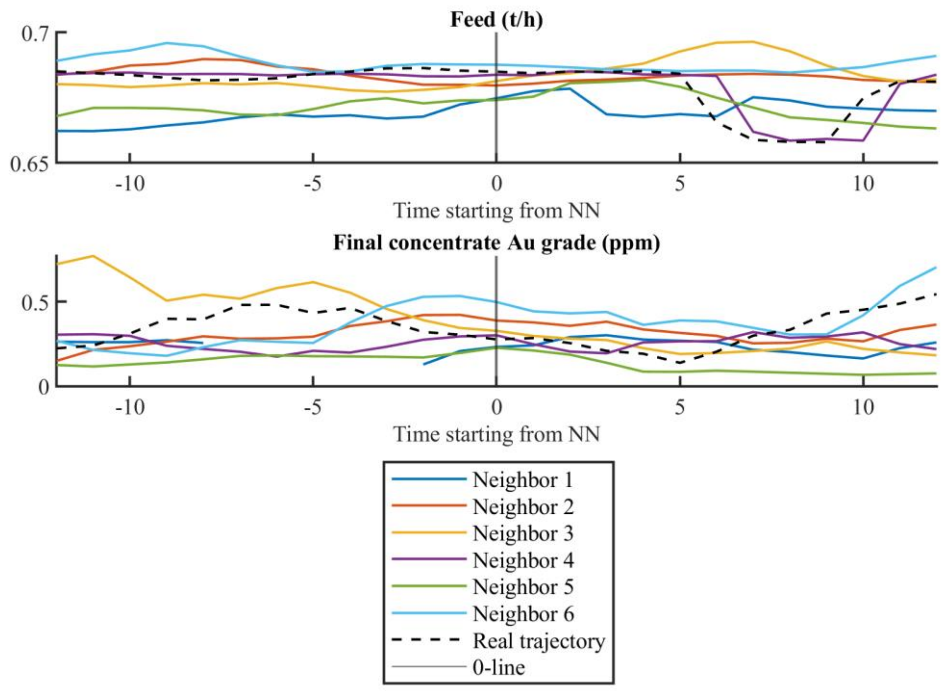

Figure 36. Trajectories for query point 6980 and nearest neighbors (2 var.).

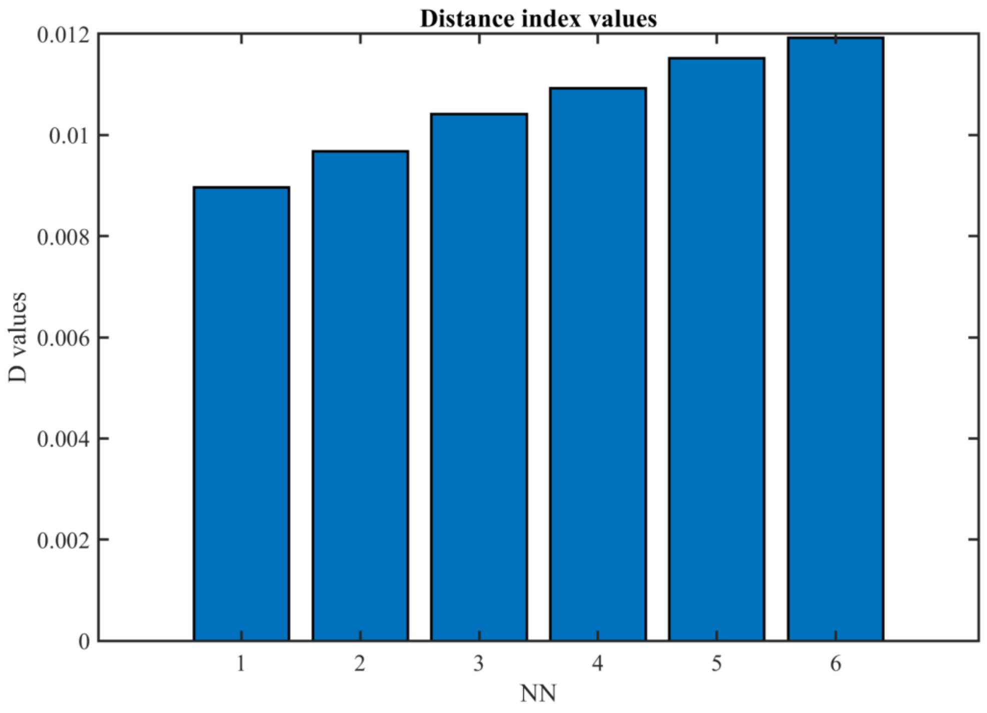

Figure 37. Distance indexes for query point 6980 (3 var.).

Appendix 12. Trajectories for 3-var. search in 6980 and distances for 4-var. search

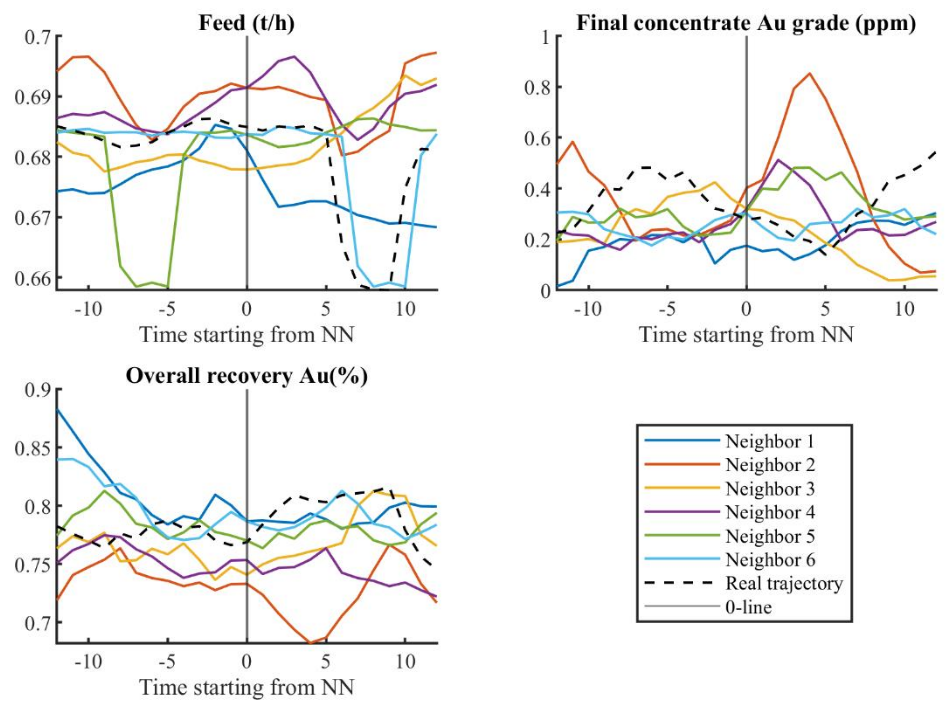

Figure 38. Trajectories for query point 6980 and nearest neighbors (3 var.).

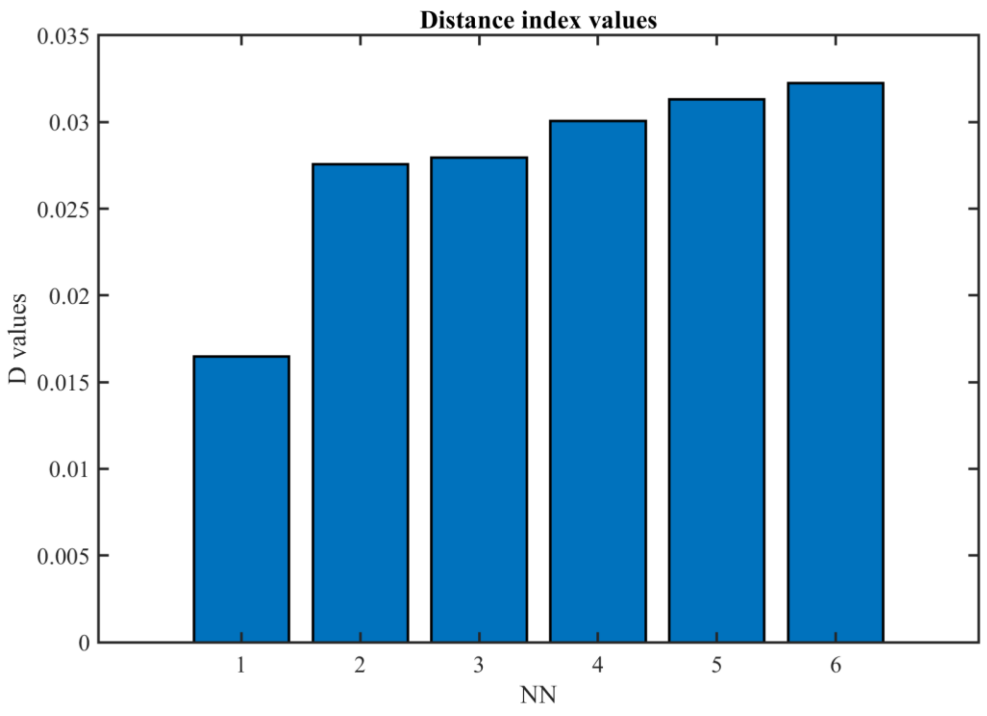

Figure 39. Distance indexes for query point 6980 (4 var.).

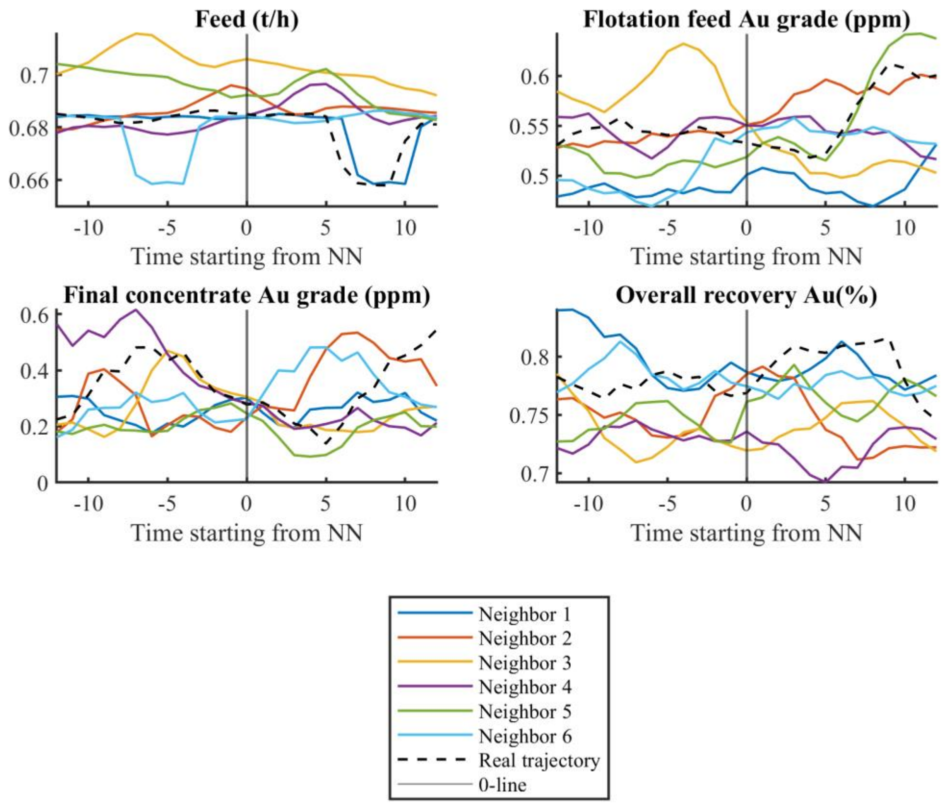

Figure 40. Trajectories for query point 6980 and nearest neighbors (4 var.).