Federal University of Espírito Santo

Faculty of Technology

Postgraduate Program in Computer Science

Bruno Guilherme Carvalho

Evaluating machine learning techniques for detection of flow instability events in offshore oil wells

Vitória

2021

Bruno Guilherme Carvalho

Evaluating machine learning techniques for detection of flow instability events in offshore oil wells

Submitted in partial fulfillment of the requirements for the degree of Master in Computer Science from the Postgraduate Program in Computer Science of the Federal University of Espírito Santo.

Supervisor: Prof. Dr. Flávio M. Varejão

Co-supervisor: Dr. Ricardo E. V. Vargas

Vitória

2021

Bruno Guilherme Carvalho

Evaluating machine learning techniques for detection of flow instability events in offshore oil wells

Submitted in partial fulfillment of the requirements for the degree of Master in Computer Science from the Postgraduate Program in Computer Science of the Federal University of Espírito Santo.

Dissertation approved. Vitória, October 4th, 2021.

Prof. Dr. Flávio M. Varejão
Supervisor

Dr. Ricardo E. V. Vargas
Co-supervisor

Prof. Dr. Celso J. Munaro
Internal member

Prof. Dr. Ricardo M. Salgado
External member

Vitória
2021

Acknowledgements

When I applied for the master's degree program at UFES I knew what I wanted. To dive deep into the artificial intelligence field. However, as in many occasions in life, I couldn't imagine what was coming. Halfway through, the world went into a virus pandemic that questioned all aspects of modern life. Fortunately, I had found great people to help me.

Firstly, I want to thank my supervisor, professor Flávio Varejão, who accepted me as a student right from the beginning of my participation in this program. Beyond that, his dedication and support have kept me on track and guided the way to a successful finish.

I would like to thank Dr. Ricardo Vargas, a fellow workmate from Petrobras and then co-supervisor, who has always been a resource of fomenting and insight, since we met in March 2019.

To our research associates, professors Celso Munaro and Ricardo Salgado, as well. For the fruitful discussions and their distinguished remarks and contributions at every meeting.

I also want to thank my manager at Petrobras, Laís Esquassante, who has inspired motivation to move on, and provided me with flexible working hours even before the pandemic times.

To all my friends that are always great references.

To my parents, who paved my way with opportunities that brought me here.

To my big brother Carlos, whom I wish I could live closer, for being a role model for all my steps.

And finally, my family, Isabela, the joy of my life, and Fabiana, my true love forever.

Thank you.

"AI is neither artificial nor intelligent." — Kate Crawford

Resumo

Instabilidade de fluxo é uma evento operacional indesejado em poços de petróleo. Para a insdústria de óleo e gás, métodos eficientes de detecção e classificação de falhas é essencial para reduzir os tempos de parada e aumentar a produtividade. A aplicação de algoritmos de aprendizado de máquina tem sido amplamente utilizada no contexto industrial, provando ser uma forma viável para resolver este tipo de problema. Neste estudo, é feita uma avaliação de técnicas de aprendizado de máquina aplicadas a detecção e classificação da instabilidade de fluxo baseadas em leituras de sensores de pressão e temperatura instalados em sistemas submarinos de produção. Primeiramente é definida uma nova estratégia de validação cruzada que elimina o viés de similaridade. Resultados mostram que esta abordagem é mais realista que a divisão tradicional utilizada em trabalhos recentes. Em seguida, *grid search* é utilizado na busca pela otimização de hiperparâmetros. Os resultados não foram satisfatórios. Então, foi aplicado a técnica de seleção de características para reduzir a dimensão do problema e evitar o fenômeno de Hughes. Três métodos foram usados: seleção sequencial de características, um algoritmo hibrido *ranking-wrapper*, e algoritmo genético. Nem todos os métodos conseguiram reduzir o número de características e melhorar simultaneamente a classificação. A aplicação de um método baseado em algoritmo genético foi um que conseguiu ambos os avanços, mostrando-se como um método robusto até mesmo nas abordagens em que o viés foi eliminado, alcançando valores de F_1 acima de $0,7$ em todos os casos. Por fim, uma análise dos resultados de todos os experimentos foi conduzida para determinar quais das características estatísticas são mais relevantes, e de quais sensores foram extraídas. Desvio padrão e variância do sensor P-MON-CKP foram as mais selecionadas.

Palavras-chaves: instabilidade de fluxo; aprendizado de máquina; validação cruzada; seleção de características; algoritmo genético.

Abstract

Flow instability is an abnormal operational state in offshore oil wells. For the oil and gas industry, methods to detect and classify faults as soon as possible are crucial to reduce downtime and increase efficiency. The application of machine learning algorithms has been extensively applied in an industrial context, proven to be a viable way to tackle this kind of problem. In this study, an evaluation is performed on the application of machine learning techniques for the detection and classification of pressure and temperature sensor readings related to flow instability. Firstly, a custom cross-validation splitting strategy is defined and compared to the classical equal split. Results are shown to be much more realistic when checked on previous publications. Next, grid search is chosen to evaluate whether hyperparameter tuning could increase the classifier's performance. Results were not satisfactory. Then, feature selection is applied to reduce problem dimension and circumvent the curse of dimensionality. Three different methods were used: sequential feature selection, hybrid ranking wrapper, and genetic algorithm. Only a few methods have shown a decrease in the number of features selected while improving classification performance measured with F_1. The genetic algorithm was one of those, proving to be a robust selector even when the similarity bias is removed. Finally, an analysis of the results from all experiments is performed to find which of the statistical features are more relevant and from what sensor they come from. Standard deviation and variance from the P-MON-CKP sensor are found much frequently than the others.

Keywords: flow instability; machine learning; cross-validation; feature selection; genetic algorithm.

List of Figures

List of Tables

List of abbreviations and acronyms

AdaBoost	Adaptive boosting
API	Application programming interface
CSV	Comma-separated values
CV	Cross validation
DHSV	Downhole safety valve
EHU	Electro-hydraulic umbilical
ELM	Extreme learning machine
GA	Genetic algorithm
GNB	Gaussian naive Bayes
HRW	Hybrid ranking wrapper
KNN	k-Nearest neighbor
LDA	Linear discriminant analysis
ML	Machine learning
MLP	Multilayer perceptron
NoF	Number of features
PCK	Production choke
PDG	Pressure downhole gauge
QDA	Quadratic discriminant analysis
RF	Random forest
SFS	Sequential feature selection
SVM	Support vector machines
TPT	Temperature pressure transducer
XTREE	Christmas tree
ZR	Zero rule

Contents

1 Introduction

In most industrial systems sensors are applied to process control and monitoring. Detection and classification of operational failure in such systems are crucial to profitability and business success. The development and application of modern tools to protect industrial equipment are longstanding challenges to both enterprises and research communities (ISERMANN, 2005).

Data collected in these industrial processes serve real-time human monitoring and also later diagnosis procedures. One expert operational technician may spot an equipment failure only by watching a sudden change in a sensor measurement, for instance with a drop in some pressure reading. Unfortunately, that's true only for a few extreme cases. For all other cases, more sophisticated tools are required.

Information technology has shifted the global economy from a traditional industry-based to a digital one. Now the digital transformation makes its come back into the industries. According to Schwab (SCHWAB, 2016) "*It is characterized by a fusion of technologies that is blurring the lines between the physical, digital, and biological spheres*".

The amount of sensor data collected on a daily basis grows exponentially in a phenomenon known as *big data*. The term may also refer to techniques, tools, and systems. Here, it is meant as the actual big amount of data. Manually inspecting, treating, and analyzing large datasets is beyond the feasible limits for quite some time now (CHEN; MAO; LIU, 2014).

In recent decades, there have been great advances in computational methods of machine learning (ML) designed especially for this kind of task. More efficient algorithms allied to faster and more powerful hardware components now allow data mining, e. g., pattern recognition tasks, to be performed in much complex problems related to fraud detection (DHANKHAD; MOHAMMED; FAR, 2018), health care (SHAILAJA; SEETHARAMULU; JABBAR, 2018), natural language processing (KHAN et al., 2010), stock market (USMANI et al., 2016), energy distribution (BERRIEL et al., 2017), and in oil and gas industry (MOHAMED; HAMDI; TAHAR, 2015).

Machine learning, in contrast to traditional statistical methods, uses general-purpose algorithms to recognize patterns and perform regression, classification, or other tasks, on a large number of records. These algorithms make none or very few assumptions about how the underlying systems behave (BZDOK; ALTMAN; KRZYWINSKI, 2018).

The oil and gas industry is eager for new ways to reduce costs and increase efficiency. Foremost, there are the operational safety and well integrity requirements (JACKSON,

2014). Facilities, the environment, and workers must be protected from any kind of incident. Secondly, there are physical requirements. Subsea offshore installations are often isolated, offer limited access to their interfaces, and more importantly, they cannot be reached easily, which means they must be remotely controlled and supervised. Lastly, it is a matter of reducing downtime and provide plant optimizations (BROCKWELL; DAVIS, 2016). Consequently, there's a growing trend for novel tools to forecast faulty or abnormal operations, reduce risks, and maximize profitability in oil and gas wells (BATALDEN; LEIKANGER; WIDE, 2017; CADEI et al., 2018).

In offshore exploration, the subsea layout often faces diverse challenges. Among them are the water depth, horizontal distance between the well and floating unit, great temperature gradient, and seabed slope. All these conditions make the hydrocarbon flow susceptible to phase change, leading to **flow instability**.

The flow instability is characterized by a smooth transition from normal operation to a steady-state fault. Sensor signals suffer non-periodic oscillations, but they remain within the tolerable range. Therefore, detecting the flow instability in real-time by an operational technician is a tough job. Machine learning offers numerous advantages: it is fast, flexible, scalable, and it can achieve high performance.

The 3W dataset (VARGAS et al., 2019), a collection of real abnormal events in oil wells, made it possible for researchers around the globe to work on ML models in this field. It contains a collection of Multivariate Time Series (MTS) from wells in the offshore Brazilian region. Data covers different classes of undesirable events, including flow instability. Each MTS may contain the full signal from normal operation, a transient period, and finally measurements from its steady-state faulty.

This dissertation focuses exclusively on the flow instability event. A handful of machine learning techniques are selected to compose a workflow to obtain the highest possible classification performance, without using compromised experimental setups.

1.1 Motivation

In Abril 2021, the well 6-BRSA-1222A-ESS scored the highest producing volume in the Brazilian Jubarte field. It has produced on average 15,453.7 bbl/d[1] to the P-58 platform in the Campos basin. On Abril 30th 2021, the price of Brent Crude Oil was USD 67.13[2], summing up a total of over USD 1.04 million in revenue every day.

In addition to revenue and profitability, operational safety is equally vital for the oil and gas industry. Catastrophic accidents as the explosion and fire during well activities

[1] Data is publicly available from Brazilian Oil and Gas Agency (ANP) on https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-estatisticos/de/ppgn/pp/producao-por-pocos.zip

[2] Quote for XBR/USD symbol from https://www.investing.com/currencies/xbr-usd-historical-data

on the Deep Water Horizon rig call the attention of big audiences. It reminds us to keep vigilant at all times and take safety seriously.

Estimated daily rates of oil rigs range from USD 136,000 to USD 160,000 (PALMIGIANI, 2021). These are special ships that can recover and repair oil wells in failure. Supposing a week-long intervention, costs may exceed USD 1 million, if all goes well.

It is now clear that even the shortest operational failure has a tremendous impact on the field's financial return, and failures can escalate to disasters. Revenues are high, but service costs can be prohibitive, especially for lower productivity wells.

1.2 Research methodology and goals

The research methodology applied in this dissertation is quantitative experimental. An already existing dataset will be explored in a data-driven approach with machine learning algorithms.

The main goal of this study is to evaluate the application of multiple machine learning techniques for detection and classification of the flow instability, focusing on grid search, feature selection, and an experimental design with minimum compromise, i.e. reducing or eliminating the similarity bias. For that, intermediate milestones are the definition of a working pipeline (a workflow) to guide the composition and combination of methods, establishment of an ideal approach, comparison of algorithms, and finally the selection of best models.

1.3 Contributions from this dissertation

This dissertation brings the following contributions:

- an experimental setup design that focused on reducing the similarity bias. A new form of cross-validation, splitting data assuring that the testing fold contains samples from a single oil well, was defined and applied and showed more realistic results than previously published papers;
- the establishment of a workflow in order to evaluate several machine learning techniques applied to the flow instability abnormal event. This workflow showed that grid search was not effective for this problem. However, feature selection was able to simultaneously reduce the number of features and increase classification performance, especially for algorithms robust to local minima as the genetic algorithm;
- a study on feature importance and sensor redundancy showed that sample entropy was the most used by the feature selectors, but it was not associated with the best

results. Standard deviation and variance, on the other hand, were found to lead to higher classification performance. Besides that, the sensor P-MON-CKP seems to be the most relevant for the flow instability detection, as it was present consistently in the best results.

1.4 Dissertation structure

The course of this study is presented in this dissertation starting here in Chapter 1 with a overall introduction. In Chapter 2 there's the problem definition with a quick presentation of the 3W dataset. Chapter 3 contains the foundation theory behind machine learning techniques that will be employed, and then in Chapter 4 those same techniques are customized to this particular problem. Results lie in Chapter 5 and Chapter 6 wraps all up with final conclusions.

2 Flow instability in offshore oil wells

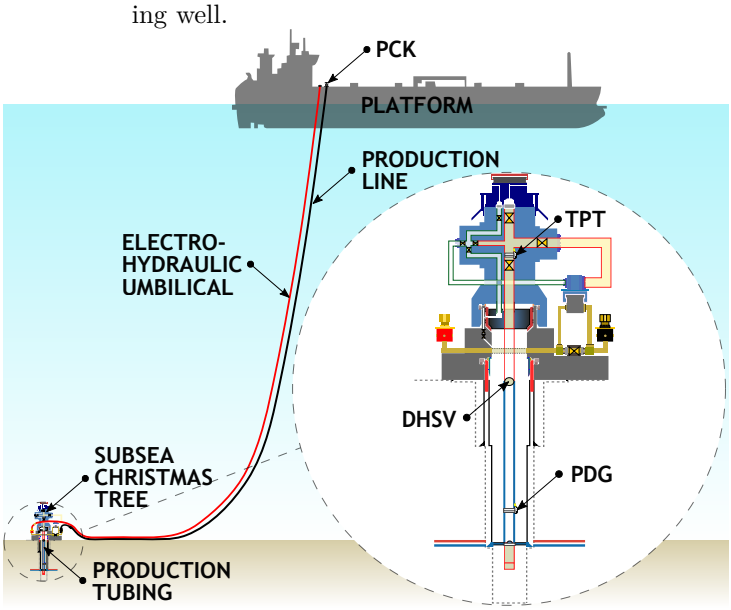
In this chapter, there's a gentle introduction to the offshore production system, a discussion on the flow instability problem, and a brief presentation of the 3W dataset.

2.1 Offshore oil and gas production systems

Offshore oil and gas production systems are equipped with different types of sensors and actuators. Actuators are usually hydraulic driven. Sensors, analog or digital, communicate in low voltage with a master control station through an umbilical cable (VAZ et al., 1998). This cable supplies hydraulic power and copper or fiber optics communication channels.

A possible setup of such system includes pressure measurement with the Pressure Downhole Gauge (PDG), located inside the well; temperature and pressure with the Temperature and Pressure Transducer (TPT), located inside the Subsea Christmas Tree (XTREE); pressure and temperature at the gas lift choke (GLCK); and pressure and temperature in the production choke (PCK), located on the platform. Figure 1 shows the composition of all these elements.

Figure 1 – Simplified scheme of a typical offshore naturally flowing well.



Source: (VARGAS et al., 2019).

The XTREE is an equipment installed on the wellhead and is responsible for controlling inlet and outlet well streams. Oil and gas flow from the reservoir through the production tubing up to the XTREE, where it is diverted into the production line, which is connected to the platform.

In many situations, artificial elevation is employed. That is, a method of elevation that adds energy to the fluid to improve well productivity. Natural gas injection through a service pipeline, a method called *gas lift*, is frequently used (SOUZA et al., 2010). Electric pumps in the downhole or on the seabed are also common solutions.

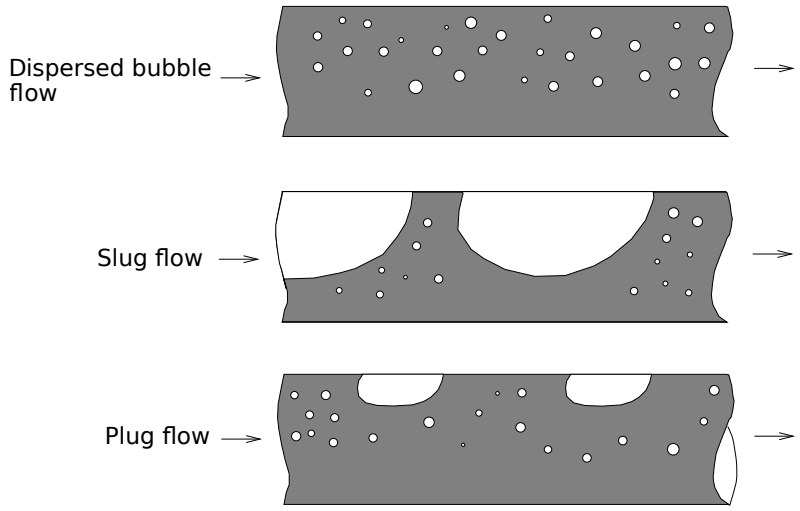
Operations of these systems often face unexpected and undesirable events. An *undesirable event* is an anomalous behavior causing the system to deviate unacceptably from its normal operating conditions (ALDRICH; AURET, 2013).

2.2 Flow instability

In offshore oil and gas production, wells are typically constructed in an arrangement that results in horizontal distance to the floating unit, where primary separation is conducted. Water depth adds a vertical component to that arrangement. In order to connect the dots, the subsea infrastructure is constructed using steel pipelines, manifolds and templates, and umbilical cables. Hydrocarbon flux along those pipelines is subjected to head loss, which causes a phase change in the mixture, from liquid to gas. Hydrodynamic instabilities may arise from this increase in the gas-liquid ratio. Different flow patterns depending on the gas-liquid ratio are shown in Figure 2.

Another critical aspect of the flow instability is its tendency to evolve to a slug flow, which is a more severe condition and more dangerous to the installations.

Figure 2 – Different multiphase flow patterns in pipelines. Plug flow is the equivalent representation of flow instability.



Source: (ZAKARIAN, 2000)

Degraded flow conditions may also arise from constructions on descending bathymetry levels. Avoiding such circumstances is not always possible due to several other subsea layout restrictions.

Flow instability, as any other industrial anomalous operation, is associated with downtime, increased maintenance costs, and eventually losses. As already stated, this study focuses primarily on this event. We take that flow instability occurs naturally and it is a recurrent source of operational problems (TAKEI et al., 2010).

2.3 The 3W dataset

The 3W dataset is a collection of sensor measurements recorded from real operations in offshore oil wells, plus software simulated and hand-drawn synthetic data. It includes eight types of abnormal events: 1) Abrupt increase of BSW (Basic Sediment and Water); 2) Spurious closure of DHSV (Downhole Safety Valve); 3) Severe slugging; 4) Flow instability; 5) Rapid productivity loss; 6) Quick restriction in PCK; 7) Scaling in PCK, and 8) Hydrate in production line.

Data is organized in one folder per event type and a comma-separated value (CSV) file for each session of data acquisition. File naming convention is composed of the source, a sequential numbering, and a timestamp for instances from real events. These files are instances of the respective fault and they may contain the full signal starting from the normal operation, passing through the transient period, ending with the steady-state anomaly. Table 1 shows the number of instances available in this dataset. In Table 2 there's a summary of observations in the instances captured from real events.

Table 1 – Number of instances for each abnormal event and its respective source.

Event	**Real**	**Simulated**	**Hand-drawn**	**Total**
0 - Normal operation	597	-	-	597
1 - Abrupt increase of BSW	5	114	10	129
2 - Spurious closure of DHSV	22	16	-	38
3 - Severe slugging	32	74	-	106
4 - Flow instability	344	-	-	344
5 - Rapid productivity loss	12	439	-	451
6 - Quick restriction in PCK	6	215	-	221
7 - Scaling in PCK	4	-	10	14
8 - Hydrate in production line	3	81	-	84

Each acquisition contains an uninterrupted ordered sequence of measurements taken every second. Eight process variables were collected: pressure at PDG (P-PDG), pressure at TPT (P-TPT), temperature at TPT (T-TPT), pressure upstream of PCK (P-MON-CKP), temperature downstream of CKP (T-JUS-CKP), pressure downstream of

gas-lift choke (P-JUS-CKGL), temperature downstream of GLCK (T-JUS-CKGL), and gas-lift flow rate (QGL). Pressure values are measured in Pascal (Pa), the temperature in degree Celsius (^{o}C), and flow rate in cubic meters per second (m^3/s). Records also include a timestamp and a class label.

Table 2 – Number of observations in the instances captured from real events.

Event	**Normal**	**Transient**	**Steady-state**	**Total**
0 - Normal operation	9956791	-	-	9956791
1 - Abrupt increase of BSW	33591	77814	5870	118294
2 - Spurious closure of DHSV	52651	88388	16615	158680
3 - Severe slugging	-	-	569152	569152
4 - Flow instability	-	-	2462076	2462076
5 - Rapid productivity loss	32232	318158	10147	361998
6 - Quick restriction in PCK	34387	6252	12951	54212
7 - Scaling in PCK	12526	257728	821	271708
8 - Hydrate in production line	8567	79255	2900	91091

In order to give the reader a sense of what a failure looks like, Figure 3 presents two different signals from a flow instability instance. On the left, P-MON-CKP and T-JUS-CKP clearly show oscillations around the same value. Then, for the same sensor and the same well, after the flow instability has been identified, plots show much greater oscillations without a common point. Abscissa and ordinate were kept with the same scale.

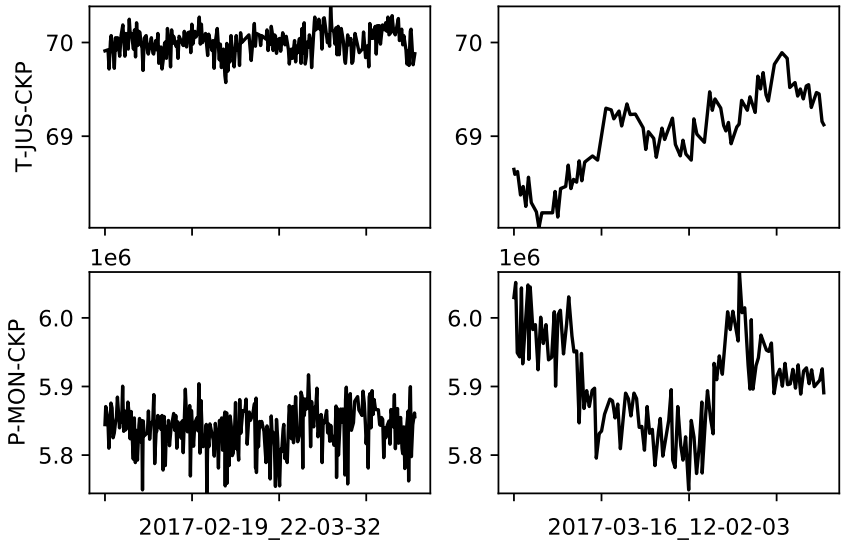
It should be noted from Tables 1 and 2 that the flow instability event is restricted to real instances and only available as a steady-state signal. As already stated, the event's smooth degradation and oscillations inside the tolerable range make it hard to define its boundaries. Consequently, software simulation or hand-charting are burdensome tasks.

Another relevant aspect of the dataset is its natural class imbalance for all events. This poses a major challenge to the application of machine learning methods, because it implies additional measures in model assessment, especially in how to choose the evaluation metric.

2.4 Related research

The original 3W dataset publication (VARGAS et al., 2019) presents the dataset itself and two suggested benchmarks for machine learning practitioners and researchers. Next, Vargas (VARGAS, 2019) Ph.D. thesis presents the first results for those benchmarks. Both benchmarks don't cover the flow instability event.

Figure 3 – Well 0001 - P-MON-CKP and T-JUS-CKP signals showing different patterns for normal operation in the left and abnormal in the right. Samples were taken from files identified as timestamp 20170219220332 and 20170316120203, respectively.



Source: Created by the author.

Marins et al. (MARINS et al., 2021) have studied different approaches to the fault events in 3W dataset. In *experiment 2* they investigate the flow instability and obtain 99% accuracy in classification with random forest. However, they split data for cross-validation in a traditional fashion (equal splits).

Brønstad (BRØNSTAD, 2020) applies a very similar approach, with the extraction of nine statistical features, and usage of random forest. According to the author, an effort was made to avoid data contamination by using group k-fold cross-validation and leave-one-out cross-validation (LOOCV) at the instance level. Reported results of accuracy are above 0.967 for the flow instability event.

Turan and Jaschke (TURAN; JASCHKE, 2021) use a similar approach performing sliding window with feature extraction, followed by standardization, grid search, feature selection, and traditional k-fold cross-validation. Their study applies multiclass classification (except for scaling in PCK) to real and simulated data. They found that feature selection with PCA was helpful, and found the decision tree to be the best classifier, reaching $F_1 = 0.95$ for flow instability.

Fernandes Júnior et al. (FERNANDES JÚNIOR et al., 2020) apply a slightly different approach with one-class classification. Results are not segregated by event type, hence it will not be comparable to the results in this dissertation.

Lastly, it is worth mentioning the paper from Figueirêdo et al. (FIGUEIREDO et al., 2021) that goes through a different method by feeding unsupervised algorithms with raw time series data.

From the aforementioned studies, one can see similarities in the approaches to the problem. However, they share the same shortcomings that is failing to avoid the similarity bias. Data from a single well may appear in both training and testing sets at the same time. The classifier's performance is likely to be overoptimistic.

3 Machine learning and fault detection

Machine learning (ML) is a sub-field of artificial intelligence (AI). While AI is concerned with wider goals like reasoning, planning, and perception, machine learning is focused on identifying patterns in past data to predict the future on new data (MITCHELL, 1997; BARAL; GELFOND, 2000; RUSSELL; NORVIG, 2009).

The history of ML and AI together traces back to mid 20th century. The Turing Test from Alan Turing (TURING, 1950) is a seminal work and a pioneer publication in this field from 1950. The term **machine learning** is attributed to Arthur Samuel, who developed a computer program for playing checkers in 1959 (SAMUEL, 1959).

Fast forward to the late 2000s, a series of developments have led to a surge in machine learning interest and growth. The ImageNet (DENG et al., 2009) is a database that unlocked and allowed advances in computer vision and neural network architectures. In 2011 and 2012 a Convolutional Neural Network (CNN) called DanNet became famous by winning a series of contests starting from the Chinese handwritten recognition at ICDAR 2011 (CIREŞAN; MEIER; SCHMIDHUBER, 2012). A year later, the Google Brain team won the AlphaGo competition.

Following this trend, companies around the world are adopting machine learning in their processes (JORDAN; MITCHELL, 2015), which include health care, manufacturing, education, finance, and the oil and gas industry.

The term *industry 4.0* (LASI et al., 2014) is being used to describe the concepts and technologies that are driving the transformations from our current state to a new paradigm in the industrial environment. Blockchain, 3d printing, smart devices, and machine learning are among those technologies.

Machine learning can be applied in many different ways in the industry sector, as in key performance index (KPI) prediction, manufacturing anomalies identification, maintenance planning, optimization in the supply chain, and fault detection. A *fault* is an abnormal event, or malfunction, in any technical system.

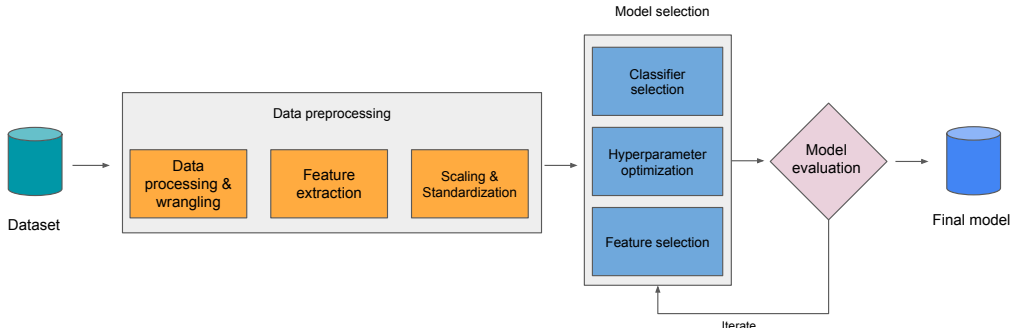
Fault detection is performed a posteriori (*offline*) when data is analyzed after the event has happened, or with active monitoring of signals, called *online*. Thus, an online detection system requires the availability of these signals, like images or sensors. Fault detection contrasts with fault diagnosis in that the latter focuses on finding specific details about the event, and also with fault identification that aims to identify which fault it is.

In the next sections, there is a short presentation of a handful of machine learning techniques used for fault detection in this dissertation.

3.1 Machine learning workflow

The application of machine learning techniques to a particular problem requires a sequence o multiple steps or tasks. Figure 4 shows a proposed workflow for this dissertation consisting of the main phases: pre-processing, model selection, and model evaluation.

Figure 4 – Machine learning workflow composed of a sequence of multiple tasks.



Source: created by the author.

3.1.1 Pre-processing

It all starts with data. Some authors (WANG et al., 2017; ROH; HEO; WHANG, 2019) even consider data collection or retrieving as an initial step. For the sake of brevity, let's assume data is already (easily) available. Then, the first phase is called *pre-processing*. It consists of tasks related to transform and adjust the data set to meet a specific application.

The first task in pre-processing is data wrangling (or data preparation). This step often translates to data cleaning, structuring, and converting. This is where one removes records with missing values, deletes or modifies variables. Some classifiers are tolerant to empty values, others simply won't take it, however.

The second task is to transform the original prepared data into algorithm-ready data. For example, with time-series input one may use a sliding window to calculate statistical properties for each variable. A sliding window is an iterative technique applied to time series that in each step a sub-sequence is selected and statistics are calculated upon. These statistics if used as input for machine learning algorithms are called **extracted features**. Feature extraction serves multiple purposes. It can be used to expand the search space (NARGESIAN et al., 2017), improve accuracy (SARKER, 2021), and reduce the problem dimension.

The last step in pre-processing is to transform feature scales. Some algorithms are sensitive to features in different magnitudes (see details in Section 3.2). Standardization is one of those transformations. It translates the mean of each feature to zero and its variance to the unit (ESTéVEZ et al., 2009).

3.1.2 Model selection

According to (DUDA; HART; STORK, 2000) learning is the task when an algorithm reduces output error on a set of training data. Learning is the core of the *model selection* process, which consists of classifier selection, hyperparameter tuning, and feature selection.

Supervised learning is the application of machine learning algorithms to problems where a dependent variable is known (CUNNINGHAM; CORD; DELANY, 2008). In the 3W dataset, the dependent variable is categorical, the class label. Thus, the detection of flow instability is a classification problem.

The starting point of model selection is to choose a list of algorithms that can be used as classifiers. It is started by selecting a few simple ones. Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis, and Gaussian Naive Bayes (GNB). These algorithms don't require hyperparameter tuning and can scale easily. Adaptive Boosting (AdaBoost) and Support Vector Machines (SVM) are completely different from each other but offer interesting strengths as less susceptibility to overfitting. Extreme Learning Machine (ELM) and Multilayer perception (MLP) are neural networks that are suitable for non-linear problems. k-Nearest Neighbors is a non-parametric algorithm with few hyperparameters that performs reasonably well in many cases. Random Forest is flexible and stable. More details of these algorithms are presented in Section 3.2.

Caruana and Niculescu-Mitzil (CARUANA; NICULESCU-MIZIL, 2006) suggest that it's essential to compare as many methods as possible to approach a given problem when optimal performance is desired. Then, to determine which is the best one may employ an iterative search combined with hyperparameter tuning and model evaluation.

Hyperparameters are internal algorithm settings not related to training data. Parameters, on the other hand, are values learned from data that will be used for predictions. Hyperparameter optimization or *hyperparameter tuning* is the process of finding the best attributes for each algorithm to reach maximum classification performance. To fulfill this goal, grid search is a long-time frequently used method that evaluates the model in all points of a chosen subspace (the *grid*), helping to find the best fit for a model (LERMAN, 1980).

Another step in model selection is feature selection. Feature selection has been found to enable the classifier to perform better on a subset of the original features (KIRA; RENDELL, 1992). It also improves the algorithm's speed on trained models. With feature extraction, the problem dimension normally increases and the available data becomes more sparse. Thus, it becomes harder to find the best model's parameters. Feature selection helps to reduce this side effect.

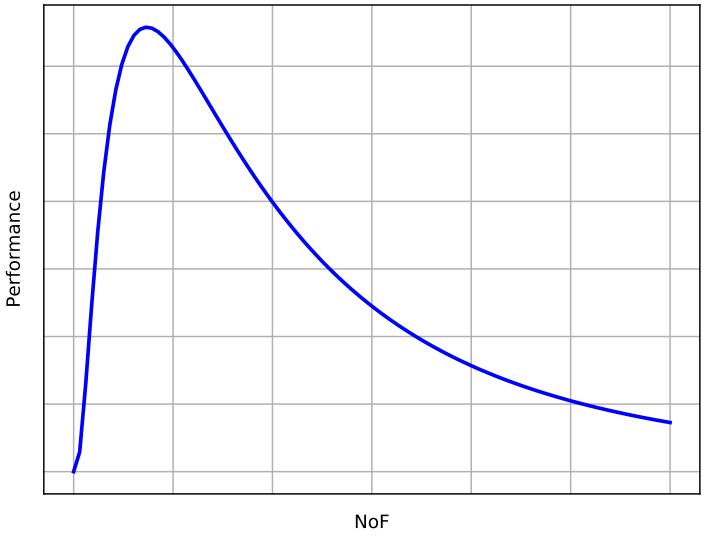
There are three groups of algorithms for feature selection: i) filter (*ranking*), ii) wrapper, and iii) embedded (KUMAR; MINZ, 2014). The filter approach tries to explore

the data regardless of the classifier, for instance analyzing the variance among the features. The wrapper methods take advantage of the relationship between data and classifier, i. e., the wrapper uses the classifier to find the best features. Last, an embedded method performs feature selection while training.

Feature extraction and feature selection, in a more general concept of feature engineering, are vulnerable to the curse of dimensionality. The term used interchangeably with *Hughes' phenomenon*, states that the performance of an algorithm will increase with a higher number of features until a certain point then it will deteriorate (HUGHES, 1968; INDYK; MOTWANI, 1998). An illustration of this is given in Figure 5.

Consequently, applying feature selection is not only a matter of optimizing the model, as it may be a requirement to mitigate the Hughes' phenomenon.

Figure 5 – Illustration of the curse of dimensionality. Performance increases rapidly with features up to its maximum, then decays.



Source: created by the author.

Dimensionality reduction and feature selection are slightly different, and sometimes get mixed up. In dimensionality reduction often a few new features are created but they don't retain their original meaning. This is one of the applications of Principal Component Analysis (PCA). Feature extraction and selection maintain a link to the source data while reducing the problem dimension.

Sequential feature selection (SFS) can be used as a wrapper algorithm. It adds (forward) or removes (backward) one feature at a time to the selected group until there's no improvement in overall performance (KIRA; RENDELL, 1992). Ranking methods can be combined with wrappers as proposed in (BOLDT et al., 2015). There is also the possibility to use a Genetic Algorithm (GA) (LEARDI; BOGGIA; TERRILE, 1992) since it is naturally binary.

3.1.3 Evaluation

Evaluation or *model assessment* is the task to calculate the model's performance metrics and compare to each other. It guides the practitioner in the best direction.

A more fundamental definition of model evaluation is the necessity to estimate how will be the model's error on unseen data. The ability of an algorithm to produce low output error is called *generalization*. That is, it is capable of generalizing what it has learned from past data to apply to different observations.

This concept brings us to the *Bayes error rate*. The Bayes error is the minimum error that an algorithm may achieve in a pattern recognition problem (TUMER; GHOSH, 1996). Fukunaga (FUKUNAGA, 1990) discusses multiple forms of calculation. In practice, however, obtaining the Bayes limit is often not possible. Therefore, estimation of Bayes error rate is not addressed in this dissertation.

Although choosing a measure of performance plays a central role in machine learning research, there is no consensus on how to do it (CHICCO; JURMAN, 2020). Some metrics became very popular because they tend to accommodate most of the information needed to interpret the results. F-score is one of those. F_1, which is a special case of the F-score, is the harmonic mean of precision p and recall r, defined as in (SOKOLOVA; LAPALME, 2009):

$$F_1 = 2 \times \frac{p \times r}{p + r} \tag{3.1}$$

$$p = \frac{TP}{TP + FP} \tag{3.2}$$

$$r = \frac{TP}{TP + FN} \tag{3.3}$$

where TP accounts for the number of true positives, FP for false positives, and FN for false negatives. True negatives are not taken into account. The accuracy metric is defined as:

$$accuracy = \frac{TP + TN}{n} \tag{3.4}$$

where n is the total number of samples.

The F-score can be calculated in multiple ways (GRANDINI; BAGLI; VISANI, 2020). As a macro average for each class, as a micro average, or weighted by the class balance. The **macro** F_1, shown in Equation 3.5 as the mean from the positive class (F_{1P}) and the negative class (F_{1N}) has an interesting behavior. It treats all classes with the same weight. Thus, it mitigates the effects of class imbalance. If macro F_1 results are high (close to 1), one can be confident of the good performance of the classifier. F_{1P} and F_{1N} are calculated using equations 3.1, 3.2 and 3.3

alternating fault as positive and normal operation as positive, respectively.

$$F_1 = \frac{F_{1P} + F_{1N}}{2} \tag{3.5}$$

Besides choosing the metric to use in a problem, how that metric is calculated is crucial to the correct interpretation of results. Usually, model selection is performed on 70% of the cleaned data, and the remaining 30% is set aside for testing, e.g., model evaluation. However, there are many alternatives. Cross-validation, or more specifically, *k-fold cross-validation* (CV) is a more suited technique for most problems because it makes use of all data for both training and testing and it helps to remove the bias from a unlucky 70-30 split (BURMAN, 1989; RODRIGUEZ; PEREZ; LOZANO, 2009). In CV, data is split in k folds and then combined in a series of k cycles in which $k-1$ folds for training and one for testing are evaluated. Figure 6 shows the an example with five cycles if data were to be split equally.

Figure 6 – k-fold cross-validation split scheme for $k = 5$. At each cycle a different set is used for testing, and the remaining $k-1$ splits are used for training.

Source: created by the author.

One of the variants of cross-validation is *nested cross-validation*. It is useful when combined with model selection techniques, as grid search or feature selection. Consider cycle 1 in Figure 6. The first fold is reserved for testing, with all other folds for training. In nested cross-validation the splitting is performed again on the training set, so the first fold of training becomes the validation fold for the inner cycle. The remaining three folds are used for training. In the case of the grid search, for instance, the hyperparameter tuning would be done repeatedly in the inner cycles, and then the best result could be transferred to be used in the outer fold for testing.

Cross-validation is an effective method to expose the model to different patterns from the same dataset. The entire dataset is used for training and testing, separately. However, it can also lead to optimistic results. Applying nested cross-validation mitigates this problem because the decisions (hyperparameter, features, etc) are made based on restricted portions of the dataset. This implies that the model will have to show a stronger generalization capability. Therefore, nested cross-validation is the favored method choice for model evaluation.

3.2 Algorithms for classification

In this section, a short introduction to the selected algorithms for classification is given.

3.2.1 Adaptive Boosting

Adaptive Boosting (FREUND; SCHAPIRE, 1996) is an algorithm that uses base estimators to minimize the error rate iteratively. These so-called *weak* classifiers refer to the fact that they just have to be better than random guessing (HASTIE et al., 2009). The algorithm starts by assigning a uniform weight to all samples. At each iteration it focuses on the misclassifications, rising (*boosting*) the weight only for those samples (*adaptive*).

Due to its emphasis on the samples that originate misclassifications, AdaBoost is known to be noise-sensitive.

3.2.2 Extreme Learning Machine

An xtreme learning machine (ELM) is a feed-forward neural network in which parameters can be calculated in a single pass (HUANG; ZHU; SIEW, 2006). Weights for the first layer are initialized randomly and kept fixed, and instead of optimizing the output errors with back-propagation, ELM finds the minimum error using the Moore–Penrose generalized inverse of activations to calculate weights for the hidden layer.

ELM is somewhat subject to controversy because its core ideas are similar to radial basis functions (BROOMHEAD; LOWE, 1988) and also to random vector functional link (PAO; PARK; SOBAJIC, 1994). Nevertheless, it's worth finding how it performs on the flow instability problem.

3.2.3 Gaussian Naive Bayes

Naive Bayes algorithms are a family of parametric algorithms resulting from the application of Bayes' theorem in machine learning problems. The term *naive* refers to the theorem's assumption that input features are independent of each other. In this study, features are also assumed to follow a Gaussian distribution. Thus, it's called Gaussian Naive Bayes (GNB).

Due to its simplicity, GNB offers some interesting properties: it is scalable, that is, it is applicable in problems with a large number of features; calculation of parameters comes from solving a closed-form expression, so it does not require iterations; it requires few training examples to estimate its parameters. According to Rish (RISH, 2001), GNB can achieve good classification results in practice even when its internal probability estimates are imprecise.

3.2.4 k-Nearest Neighbors

The k-nearest neighbors is an algorithm with origins tracing back to Evelyn Fix and Joseph Hodges, 1951. But Thomas Cover and Peter Hart (COVER; HART, 1967) were those

who actually developed the algorithm.

Class membership is attributed to the sample by the voting of its k neighbors, with $k >= 1$. In order to identify which samples are neighbors, the Euclidean distance is frequently used if the variable is continuous, and Hamming distance if values are categorical.

This algorithm is non-parametric, i.e., it is not based on any probability distribution, and it has only two main hyperparameters already mentioned: the number of neighbors **k** and the distance function.

A major concern with kNN is that it tends to suffer from overfitting (HAWKINS, 2004). Furthermore, it is sensitive to feature scales, like any other algorithm based on some measure of distance. Thus, a normalization is advised.

3.2.5 Linear and Quadratic Discriminant Analysis

Linear Discriminant Analysis (LDA) is a method used to determine a linear combination of input variables to segregate the samples into different groups, or classes. Based on the same idea, Quadratic Discriminant Analysis (QDA) is a classifier that combines input variables in a quadratic form.

Apart from the linear combination, LDA also can take different decision rules. One used in this study is based on Bayes' theorem. The classifier will assign a sample to a class that maximizes the product of prior probability (which in turn is based on class balance) with population density. LDA has also the assumption that output classes have the same covariance.

The ultimate goal of LDA is to minimize the variation within a class and maximize the distance between classes. Then, projecting the samples on a new axis, data will be separable by a linear decision boundary.

Another characteristic of LDA worth mentioning is that the model remains interpretable after the linear combination. This might be appealing for some applications.

3.2.6 Random Forest

Random Forests refers to the algorithm from (BREIMAN, 2001), which is based on an ensemble of decision trees. As of 2021, *Random Forests* is also a trademark owned by Minitab[1]. To be clear in this dissertation, the name in the plural is reserved for the original publication and the trademark, and the term in the singular to the actual algorithm and any of its implementations.

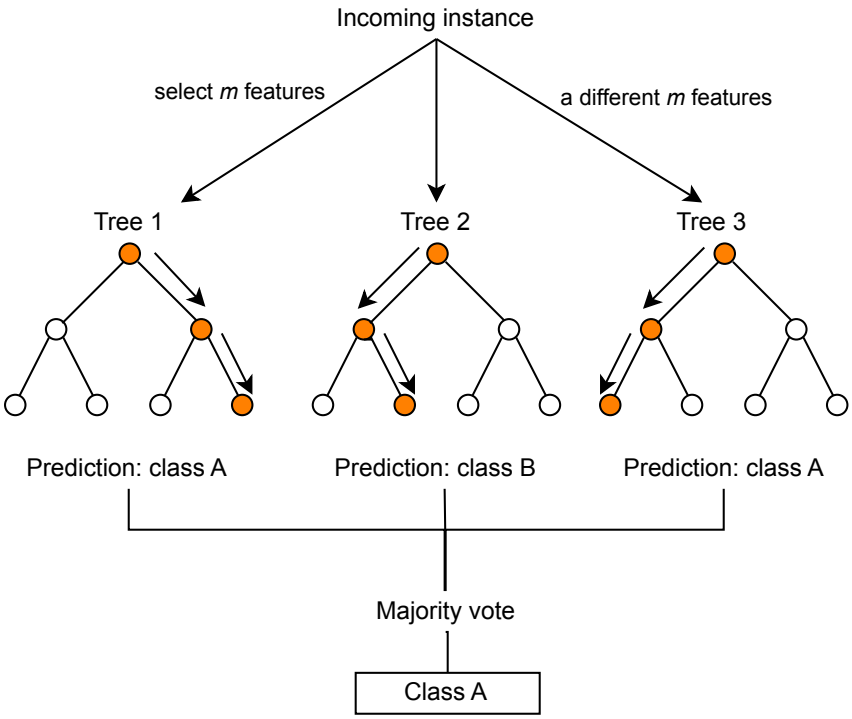
The random forest (RF) algorithm offers flexibility regarding the input data. It can handle missing values naturally. It is also versatile, reaching good performance in many different domains and problem types (OSHIRO; PEREZ; BARANAUSKAS, 2012; ALI et al., 2012). Another characteristic is that it assigns an importance value (similar to weight) to each input feature (PAL, 2005). That may be useful in some cases. There are two more advantages. RF is robust to overfitting due to the bagging mechanism (GHOJOGH; CROWLEY, 2019), and it is

[1] Minitab LLC - State College, Pensilvania - USA

not sensitive to feature scale. Thus, standardization for RF is optional.

In the disadvantage side, RF can be slow for prediction when the size of the tree is big. Despite having its origins in decision trees, RF lacks interpretability due to its final ensemble voting step (shown in Figure 7).

Figure 7 – Representation of the tree ensemble making a decision in the random forest classifier.



Source: (WOOD, 2020)

3.2.7 Support Vector Machines

The support-vector networks (CORTES; VAPNIK, 1995) algorithm was originally developed for classification problems of two classes. Later, it became widely known as support vector machines (SVM), and it was extended for regression and multi-class classification.

SVM is a statistical learning algorithm based on the transformation of input variables to a feature space, where a special dot product is defined and named *kernel function*. In this new space, the separation of classes becomes a quadratic optimization problem. A quadratic problem is subject to substantial mathematical formulation, proven to be convex. Thus, finding the global minima is guaranteed and the solution is unique. Another property of this transformation is that no assumption about the functional form is made.

Some disadvantages of SVM are the requirement of feature normalization and its impossibility of interpretation because calculations are done on a transformed space.

3.3 Algorithms for feature selection

In this section, there's an introduction to the feature selection algorithms that were used in this study.

As explained earlier, there are a tremendous amount of possible algorithms proposed in the literature. Three stand out and will be used here: i) sequential feature selection (PUDIL; NOVOVICOVÁ; KITTLER, 1994), which is intuitive and may highlight relevant aspects of the problem; ii) hybrid ranking-wrapper (BOLDT et al., 2015) offers a promising combination; and iii) genetic algorithm which is a powerful heuristic used extensively in this manner (SIEDLECKI; SKLANSKY, 1993).

3.3.1 Sequential feature selection

Sequential feature selection (PUDIL; NOVOVICOVÁ; KITTLER, 1994; CHANDRASHEKAR; SAHIN, 2014) are algorithms that run iteratively adding or removing features. The SFS forward (SFS-F) adds one random feature at a time starting from an empty set. The SFS backward (SFS-B) removes features individually from the full set. SFS-F and SFS-B belong to the wrapper class of feature selection algorithms. They take advantage of a black box classifier to measure the performance of the selected features.

3.3.2 Hybrid ranking wrapper

Many variations and improved feature selection methods have been proposed over the past years. One interesting combines a feature ranking in a classifier wrapper, hybrid ranking-wrapper (HRW) (BOLDT et al., 2015). Hybrid approaches to feature selection are seen as promising in the literature (JOVIĆ; BRKIĆ; BOGUNOVIĆ, 2015). In this study, there's a comparison of three hybrid combinations with the following ranking methods: Analysis of Variance (ANOVA), Mutual Information (MI) (ROSS, 2014), and random forest (RF) itself as a univariate estimator.

Analysis of Variance (ANOVA) is a set of statistical tools that can be used to feature selection (MEHMOOD; DU; LEE, 2017). In ANOVA, the F-test is the ratio of the variance between classes over variance within each class.

Mutual Information (MI) is a measure of how much one variable depends on the other. The higher is the measure, the more the two variables share the same information. MI can calculate the importance of features effectively, and then that measure is used for feature selection (YANG; MOODY, 1999; HOQUE; BHATTACHARYYA; KALITA, 2014).

For random forest, instead of using its internal Gini estimate or out-of-bag permutation (MENZE et al., 2009), the choice was to apply it as a univariate estimator. The output classification metric is used to rank features.

For the wrapper step, experimentation is to exploit SVM and RF. Results and discussion in Section 5 explain why this path was followed. The algorithm was implemented exactly as proposed by Bolt et. al. (BOLDT et al., 2015). It consists of two basic steps: get a feature ranking,

i.e. a list of features in decreasing order by some metric. Then, the algorithm adds one feature at a time taking from the ordered list. After each new feature is added the problem is evaluated with 5-fold cross-validation and checks if there is improvement in the final measure.

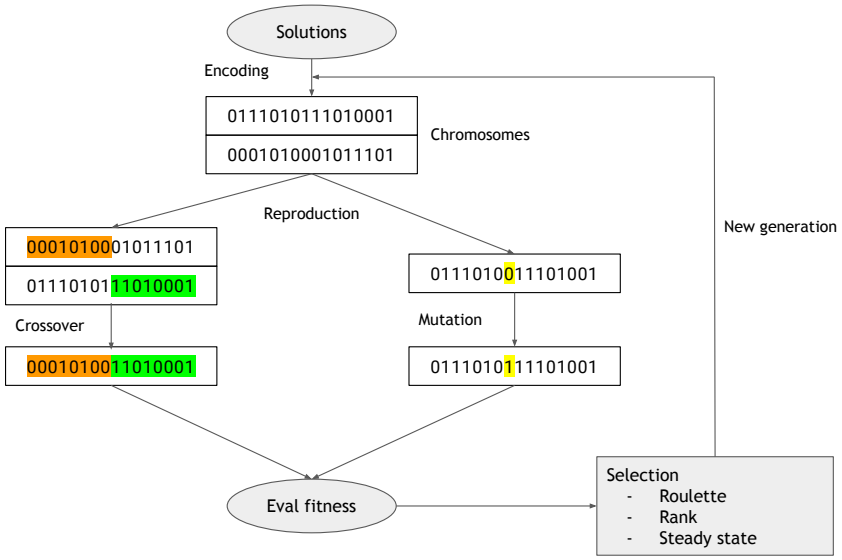
3.3.3 Genetic algorithm

Genetic algorithm (GA) (HOLLAND, 1992) is a suitable choice for feature selection because of its underlying binary nature and robustness against local minima. A feature is used in the model when the individual (also called chromosome) has the respective gene value equal to one. Otherwise, the feature is discarded. An individual is a vector of binary integers of length equal to the total number of features.

The evolving mechanism in the GA algorithm consists basically of initializing a random population, evaluate the objective function (also called fitness) for each individual, and then combining the best individuals to form an improved new generation. An elitism criterion is chosen to maintain the best individuals without change for the next generation. A pair-wise merge called *crossover* is performed to complete the new generation's population. Crossover operation resembles reproduction in a way that the offspring gets a combination of parents' genomes. These steps are illustrated in Figure 8.

After the crossover, the *mutation* is also applied. A mutation is a random change in one of the individual's genes. It helps the algorithm to explore the search space, and eventually escape a local minimum.

Figure 8 – Schematic of the genetic algorithm showing its main components: encoding solutions in binary format (the chromosomes); application of operators crossover and mutation; evaluation of the fitness function, and finally the selection process to form a new population.



Source: Based on (GEN; CHENG; WANG, 1997).

3.4 Fault detection

Fault detection is a broad term. Its specific meaning depends on the context, especially in a form of application.

From a statistical point of view, fault detection can be seen as outlier detection. An outlier is a measurement, or a sample, different from the majority of its counterparts. The notion of "different" often comes from the application of some density estimate (HODGE; AUSTIN, 2004). Since outliers are, by definition, a minority, fault detection holds inherently imbalanced classes.

A control engineering approach splits fault detection into two categories. Based on system models or based on signal processing (VENKATASUBRAMANIAN et al., 2003). Signal processing, for instance, combines linear algebra, time-frequency analysis, and calculus to remove noise, amplify, transform, and filter signals.

A modern definition groups fault detection methods into three groups: model-based, knowledge-based, and data-driven (CHEBEL-MORELLO; MALINOWSKI; SENOUSSI, 2016). Machine learning brings a data-driven solution to the problem without relying on any particular assumption.

Leveraging the 3W, in this study binary classification is applied to fault detection, i.e. a data-driven approach with machine learning classifiers are used to state whether the group of sensor readings indicate oil operation in a normal or abnormal condition. Therefore, our two classes are the normal operation, represented by the integer **0** (zero), also called the *negative* class, and the abnormal operation, represented by the number **1** (one), called the *positive* class.

As already discussed in Section 2.2, the flow instability event transitions smoothly from normal operation to a full steady-state fault. Thus, it doesn't match the outlier or rare event definition. Nonetheless, the class imbalance can challenge the classifier's abilities.

4 Flow instability detection based on machine learning techniques

In this chapter, the techniques presented in Chapter 3 are customized for application specifically on the flow instability problem.

4.1 Customizations in the workflow steps

Firstly, all gas-lift related data are dropped. These variables presented a high rate of missing values. After that, any record with missing values is also dropped (row-wise). There are a few handling mechanisms for missing data as propagating the last known number forward or using the mean from the last known values, but they all may represent additional noise to the signal. Dropping rows with missing values removed about 14.8% (1472177 observations) from the normal operation. From the flow instability original 2462076 observations, only 1806 (0.07%) were dropped. Apart from that, data from class 2 is heavily affected, losing 86.7% of its observations. The hydrate in the production line (class 8) was completely removed because all its 91091 records contain missing values for the T-JUS-CKP sensor. Note that, in the case of class 8, all values were missing, so it wouldn't be possible to use any filling strategy. And for class 2, it might be a concern to waste data, but it is important to remember that the focus of this study is on the flow instability event. As for the normal operation (class 0), removing the missing values helped to reduce the class imbalance. All records from classes 1, 5, and 6 were kept, and from class 3 and 7 only 719 and 362 were removed, respectively.

The original number of samples, before cleaning, is presented in Table 3. The focus of this study is on flow instability, so the table groups all fault except flow instability. Data comes only from the real events recorded in the 3W. The problem is tackled in two binary cases: A) normal operation vs. flow instability, and B) all events plus normal operation vs. flow instability, i.e. in this case the data representing the normal operation contains data from all other events apart from flow instability, regardless of period (normal, transient or steady-state). While the case B could be more challenging to all classifiers and perhaps a stronger approach, the case A remains an important matter if one seeks for multiclass classification.

Regarding the sliding window technique, a fixed value of 900 samples (15 minutes) is used. This value was presented by Vargas et al. (VARGAS et al., 2019) as an empirical reference for the flow instability event.

During the sliding window processing, statistical features were extracted from each sensor. Firstly, six features were considered: maximum, minimum, mean, median, standard deviation, and variance. This also follows the (VARGAS et al., 2019). Then, an extended list was used to investigate whether more features would be required, composed of: maximum, minimum, mean,

median, standard deviation, variance, 25% percentile, 75% percentile, skewness, kurtosis, and sample entropy. This sums up a total of 55 features. Shuffling was done after feature extraction.

Next, the z-score transformation was applied (Equation 4.1). Original feature values x_{ij} were normalized so that their new mean becomes zero, and their variance becomes one (*unit*). This transformation was applied to both training and testing, but its parameters (μ e σ) are estimated only with the training data.

$$x_{ij}^{z} = \frac{x_{ij} - \mu_j}{\sigma_j} \tag{4.1}$$

Implementation of all experiments was done in the Python computer language, making use of the *scikit-learn* package (PEDREGOSA et al., 2011). Linear algebra is performed with *numpy* (HARRIS et al., 2020), and Cython (BEHNEL et al., 2011) was also used to speed up feature extraction. Code is available on Github[1].

4.2 Experimental design

The core of the experiments in this dissertation is the cross-validation approach. As explained in Section 3.1.3, CV is one of the alternatives to estimate the model's performance on unseen data, that is, how it will behave for data likely different from what it has seen during training. Additionally to the traditional splitting strategy, that is, create k equally sized sets, in this study, it's proposed to split data for training and testing based on which well it comes from, the *well split*. Then, the well split is extended to be used also in the nested cross-validation for hyperparameter tuning and feature selection.

The well split is believed to impose a bigger challenge on the classifier's abilities because it removes the similarity bias. Furthermore, beyond the natural class imbalance, the well split creates imbalanced testing sets, as shown in Table 4. The numbers presented in Table 4 represent the feature vectors, i.e. the amount of data after pre-processing (cleaning, sliding window, and feature extraction). Apart from the imbalance in the data, one can also see how much noise is added by incorporating all events as the normal operation in case B.

The F_1-macro is the default metric unless otherwise noted.

In all experiments only a single round was performed, i.e. only one seed was used for stochastic functions.

4.2.1 Initial experiment

The initial experiment is to establish the ground reference where the next experiments are built upon. It uses the first group of 30 features, traditional split, and the newly proposed method of well split. For this experiment, performed on cases A and B, a list of simple classifiers is considered. By doing this, we can exclude the effects of hyperparameter tuning.

[1] https://github.com/bgcarvalho/ufes-masters-degree-experiments

Table 3 – 3W dataset sample count for normal, transient and steady-state fault periods grouped by well and class.

Well ID	Fault	Normal	Transient	Steady-state
00001	0	1688171	-	-
	4	-	-	261457
	others	6104	47696	20578
00002	0	3659066	-	-
	4	-	-	807601
	others	35392	16127	15566
00003	0	463543	-	-
	others	5320	15020	931
00004	0	85505	-	-
	4	-	-	307674
	other	3889	1791	524
00005	0	979611	-	-
	4	-	-	271525
00006	0	2058403		
	others	38875	249109	1160
00007	0	14370	-	-
	4	-	-	71651
00008	0	1008122	-	-
00009	others	3523	2076	1099
00010	others	2305	2135	830
	4	-	-	592220
00011	others	31509	55823	13063
00012	others	2601	2442	-
00013	others	3637	5796	500
00014	others	-	-	551176
	4	-	-	149948
00015	others	7555	8891	2386
00016	others	3177	8714	7761
00017	others	21500	300553	-
00018	others	-	32167	-
00019	others	4877	44827	1327
00020	others	173	14350	1109
00021	others	3517	20078	464

The list of classifiers includes 1NN, LDA, QDA, GNB, and RF. Random forest does not meet the requirements described above. However, it allows the very first results to be comparable to previous publications, as in (MARINS et al., 2021).

Table 4 – Number of patterns in each one of the 5-fold cross-validation for approaches A and B obtained from the well split strategy.

Set	**CV Cycle**	**A**		**B**	
		Negative	**Positive**	**Negative**	**Positive**
Train					
	1	7550	2442	9018	2442
	2	5361	1835	6836	1835
	3	9331	2391	10874	2391
	4	8337	2430	9888	2430
	5	9410	2653	10960	2653
Test					
	1	1874	289	1957	289
	2	4064	896	4139	896
	3	94	339	101	339
	4	1087	300	1087	300
	5	14	78	14	78

4.2.2 Grid search

In the grid search experiment, setup follows a similar approach as the split-per-well strategy in the initial experiment, except for the classifiers and depth of cross-validation. Classifiers used in the earlier setup required few or no hyperparameters, leaving not much room for improvement. Now, a list of more sophisticated algorithms is selected: ADA, ELM, KNN, MLP, RF, and SVM. Their respective hyperparameters and grid values are presented in Table 5. The selection of those values is arbitrary and loosely based on multiple published papers. Note, however, that only hyperparameters related to the classifiers were analysed. Window size or step, for instance, were kept fixed.

4.2.3 Feature selection

This step is again an improvement from the initial experiment and grid search. Now, the feature set extracted from the time series is extended to 11 features for each sensor, which include: maximum, minimum, mean, median, standard deviation, variance, 25% percentile, 75% percentile, skewness, kurtosis, and sample entropy.

Three types of feature selection algorithms will be applied.

Sequential feature selection will be applied in the forward (SFS-F) and backward approaches (SFS-B). The scikit-learn package provides an implementation for the two. However,

Table 5 – Subsets of hyperparameter values to be used in grid search with nested cross-validation.

Classifier	**Hyperparameter**	**Values**
ADA	number of estimators	5, 25, 50, 75, 100, 250, 500
	algorithm	SAMME, SAMME.R
ELM	number of neurons	null, 50, 100, 250, 500 1000, 2500
	activation	tanh, sigmoid
KNN	number of neighbors	1, 3, 5, 7, 10, 15
MLP	hidden layer size	64, 100, 256, 512
	maximum of iterations	100, 200, 500
RF	maximum features	auto, 1, 2, 4, 6, 8
	number of estimators	100, 25, 50, 250
SVM	gamma	0.001, 0.01, 0.1
	C	0.001, 0.01, 0.1, 1.0

instead of resulting in the best feature set, this specific implementation returns a given number of features. That is represented in line 6 of Listing 4.1. Note that the order of loops won't change the results or affect the computational burden, as there's no stopping criteria.

Line 5 represents the distinct splitting strategies used in this study, either an equal split or split per oil well. They were also implemented in Python, derived from the `KFold` class in the scikit-learn API.

Listing 4.1 – Pseudo code for sequential feature selection (SFS) used in this dissertation. Line 6 refers to implementation in the scikit-learn package.

```
1 load sklearn
2 set n to the number of features
3 for i from 1 to n
4     for fold from 1 to 5
5         set cv to customfold(fold)
6         call sklearn.SFS(i, cv)
7         evaluate model with best features for fold
```

The hybrid ranking wrapper algorithm was implemented exactly as proposed in (BOLDT et al., 2015), and shown in Listing 4.2. Variable `factor` controls the exploration. It was kept to 0.99. The ranking method is used right at the second line, sorting the features in descending order.

After variables initialization, the loop runs sequentially over the ordered features. The algorithm takes one feature from the list, adds this feature to a temporary list, and evaluates the classification performance based on that temporary list (lines 12, 13, and 14). If results improve from the current value times the exploration factor, then that feature is added to the current list. After that, the current value is compared to the overall best value. If it has improved, then

updates both the best value and best subset. Otherwise, it checks if the number of features is in the range of `maxfeatures`. This last if-else control structure allows the algorithm to search for a combination of features without obtaining direct improvement from each one of them. That is, suppose the algorithm tries feature 52, it reaches line 16, but it does not reach line 22. If the number of features is still less than `maxfeatures`, then the `currentvalue` is kept for the next iteration but `bestvalue` was not updated.

After the main loop, if the number of features is higher than the limit, it falls back into SFS backward. The limit is 11 (20% of 55), and it is somewhat arbitrary.

A custom implementation of the genetic algorithm was developed dedicated to this study targeting a single objective. The fitness function was defined as the $f(s) = 1 - F_1^s$, where F_1^s is the mean after k-fold cross-validation with the s subset of features.

Listing 4.2 – Pseudo code for hybrid ranking wrapper algorithm.

```
1  load sklearn
2  set features to get_features_ranked(originallist)
3  set factor to 0.99
4  set maxfeatures 20% of total number of features
5  set subset as empty list
6  set bestvalue to -1
7  set bestsubset as empty list
8  set currentvalue to -1
9  set currentsubset as empty list
10
11 while length of features is greater than zero
12     take a feature from features
13     add feature to subset
14     evaluate model with the subset
15     if result is greater than currentvalue * factor
16         set currentvalue to result
17         set currentsubset to subset
18     else
19         remove feature from subset
20
21     if currentvalue is greater than bestvalue
22         set bestsubset to currentsubset
23         set bestvalue to currentvalue
24     else
25         if length of currentsubset is greater then maxfeatures
26             set currentvalue to bestvalue
27             set currentsubset to currentsubset
28 if length of bestsubset is greater than maxfeatures
29     call sklearn.SFS(maxfeatures, "backward")
```

The genetic algorithm is naturally binary as presented in Section 3.3.3. The encoding consists of transforming the list of features into a binary array that is fed into the GA. The Equation 4.2 shows a representation of the binary vector (the chromosome) that indicates whether a feature is selected (one) or not (zero).

$$\mathbf{x_{enc}} = \begin{bmatrix} 0 & 1 & 1 & 0 & \dots & 0 \end{bmatrix} \tag{4.2}$$

The crossover operator merges two random parents to form two new descendants. Thus, line 8 of Listing 4.3 modifies the search space with the new individuals. The combination is performed by *cutting* each parent's chromosomes in a random position and joining their alternate parts (shown in Figure 8). This operation was applied to 99% of the population.

The mutation is a random flip in a single gene of an individual. This operation was applied to 1% of the population at each generation (line 9 in Listing 4.3).

For the selection process, the *elitism selection* was used. At each generation, the two best individuals were kept unchanged. These two individuals could, however, generate offspring with the crossover operator.

Finally, there was no stopping criterium. The algorithm was run for 200 generations on purpose to watch the behavior of the classification metric and the number of features selected. Thus, lines 13 and 14 in the pseudo-code (Listing 4.3) were not actually implemented. They were represented because it is a common practice, and improves the pseudo-code readability.

The source code for GA developed for this experiment is available on GitHub[2].

Listing 4.3 – Pseudo code for the genetic algorithm.

```
1 set population to 100
2 set dimension to 55
3 initialize search space
4 initialize scores
5
6 for iteration from 1 to 200
7     call selection(space, scores)
8     call crossover(space)
9     call mutation(space, iteration)
10
11    scores = eval_objective(space)
12
13    if stopcriteria
14        break
```

[2] https://github.com/bgcarvalho/ufes-masters-degree-experiments

5 Experimental results

The following sections detail results from the direct application of techniques presented in Chapter 3 and customized in Chapter 4. The rationale behind the experiments is this: firstly, one needs to establish a base problem upon which the next will be built on. Secondly, a grid search technique was used to investigate whether hyperparameters have a significant influence on the results. Then finally, feature selection was applied as the ultimate path to increase classification performance.

5.1 Initial experiment

Results are presented in Table 6. In the basic strategy of splitting data evenly in 5 folds, Random Forest was best in both cases A and B, with accuracy of 0.9950 and 0.9954, respectively. These results are comparable to those from Marins et. al. (MARINS et al., 2021). It is important to note that results so far are high for this scenario, with values above 0.99, which poses a limitation for further improvements.

Meanwhile, when cross-validation is performed per well, results present great variance. Case A had the best result of F_1 from QDA and the best accuracy from RF. For case B, LDA had the highest F_1 and 1NN the highest accuracy. In all these results, classifiers were superior to Zero Rule.

The poor results from GNB are explained by the mismatch of its main assumption that variables are not completely independent. In the oil flow, sensors are related to each other due to the energy transfer.

Having found LDA and QDA with the highest results in the split-per-well strategy was a bit surprising. These classifiers assume a linear and quadratic relation of input variables and the output class, respectively. One nearest neighbor (1NN) was third-best for case A, and second for case B. It is an emblematic result for a single neighbor. With this list of simple classifiers, one might have expected RF to stand out. It did for traditional folds, but it was not the case for split per well strategy. In this approach, RF may perform better after hyperparameter tuning.

5.2 Grid search

Grid search is performed with nested cross-validation, i.e. the hyperparameters are selected based on the average results from an inner loop of 4 folds. Different from the initial experiment, the traditional split is no longer applied.

Table 7 presents results for both cases A and B, individual results for each cycle of the cross-validation, and the final mean F_1.

Table 6 – Mean and standard deviation for F_1 and accuracy after 5-fold in two different approaches of cross-validation. The bold-faced values were the highest for each experiment.

CV Strategy	**Classifier**	**Case A**		**Case B**	
		F_1	*accuracy*	F_1	*accuracy*
5 folds split equally	1NN	0.9864 ± 0.0026	0.9905 ± 0.0018	0.9848 ± 0.0047	0.9902 ± 0.0030
	GNB	0.8518 ± 0.0293	0.9001 ± 0.0372	0.8033 ± 0.0721	0.8536 ± 0.0814
	LDA	0.9351 ± 0.0048	0.9568 ± 0.0031	0.8657 ± 0.0134	0.9226 ± 0.0069
	QDA	0.9480 ± 0.0463	0.9608 ± 0.0384	0.8680 ± 0.0554	0.9000 ± 0.0483
	RF	**0.9928** ± 0.0017	**0.9950** ± 0.0012	**0.9929** ± 0.0024	**0.9954** ± 0.0015
	ZR	0.4367 ± 0.0000	0.7753 ± 0.0002	0.4437 ± 0.0002	0.7975 ± 0.0005
5 folds split per well	1NN	0.5669 ± 0.2533	0.7728 ± 0.2300	0.5747 ± 0.2368	**0.8186** ± 0.1601
	GNB	0.5045 ± 0.2903	0.7407 ± 0.3006	0.3207 ± 0.1701	0.5425 ± 0.3552
	LDA	0.5069 ± 0.2966	0.7399 ± 0.3020	**0.6806** ± 0.3385	0.7909 ± 0.3152
	QDA	**0.6716** ± 0.2644	0.8665 ± 0.1266	0.2896 ± 0.1957	0.4771 ± 0.3567
	RF	0.6434 ± 0.2326	**0.8742** ± 0.0785	0.5219 ± 0.3108	0.7290 ± 0.3312
	ZR	0.3329 ± 0.1632	0.5677 ± 0.3517	0.3350 ± 0.1617	0.5718 ± 0.3502

The ADA algorithm together with ELM were the less performant. With values of $F_1 = 0.3575$ e $F_1 = 0.5086$ for ELM indicates that the number of neurons, for example, in this problem does not affect the classification performance.

SVM, RF, and MLP were the top three for case A, and SVM, MLP, and KNN were the top three for case B.

Grid search is often applied in many machine learning problems with immediate improvements, but it was not the case for the flow instability in this experiment. Note that there is a high oscillation in the results among folds. For instance, when testing against well #5, many classifiers have reached $F_1 > 0.97$. However, for well #7, most results are stuck with $F_1 = 0.4588$.

Table 7 – F_1 results for the grid search experiment. k-Fold cross-validation with $k = 5$ is shown with results for each individual cycle labeled with the respective oil well identifier and the final mean.

Classifier	**Case**	**Well #1**	**Well #2**	**Well #4**	**Well #5**	**Well #7**	F_1
ADA	A	0.5553	0.2641	0.4822	0.4394	0.4588	0.4400
	B	0.4571	0.4512	0.5325	0.4394	0.1321	0.4024
ELM	A	0.6262	0.4002	0.1784	0.4426	0.1402	0.3575
	B	0.4655	0.4336	0.1885	0.9968	0.4588	0.5086
KNN	A	0.3512	0.4504	0.4688	0.9936	0.4588	0.5446
	B	0.4642	0.4512	0.5387	0.9968	0.4588	0.5811
MLP	A	0.4676	0.4249	0.7789	0.9882	0.4588	0.6237
	B	0.4656	0.4512	0.7701	0.9794	0.4588	0.6250
RF	A	0.8745	0.4738	0.5410	0.9936	0.4588	0.6516
	B	0.4656	0.4512	0.6816	0.9871	0.1321	0.5435
SVM	A	0.4947	0.4503	0.9628	0.9936	0.4588	**0.6720**
	B	0.4654	0.4509	0.9674	0.9893	0.4588	**0.6664**

The next step in the workflow is to apply feature selection. This technique is very compute-intensive, making the use o MLP prohibitive. Thus, SVM and RF are used in further experiments.

So far it is important to acknowledge the high variability among all results for both the initial experiment and the grid search experiment. One could speculate that results lie in the same range without statistical significant difference. This could be an indication of greater heterogeneity in data across all 5 oil wells. It remains uncertain whether those wells are geologically similar or could be a mix of post-salt and pre-salt. Running statistical tests was not in the scope of this study.

5.3 Feature selection

Following the experiments with grid search, the next step is to perform feature selection. The goal is to find an optimal subset of the original features which leads to better classification and results in a less complex model, saving computational power.

Feature selection was applied in two different ways. First, using the proposed per well split strategy, but taking the whole dataset to select the features. Take for instance the SFS algorithm. The number of features is set before the 5-fold execution. That is, the five cycles use the same features. Thus, the resulting list of selected features is based on the entire dataset. Then, in the second strategy, feature selection was performed using nested cross-validation. In this strategy, the feature selection occurs in the inner loops, and it may result in a different number of features (or features themselves) for each outer cycle. While the latter is regarded as the ideal approach, it was important to run the former to compare results and confirm that their

contrasting settings lead to distinct results. Another aspect is that in practice having a diverse number of features per fold makes the direct application of the model cumbersome.

Despite producing the best results in grid search, SVM in initial trials for feature selection showed some irregular and inconsistent results, especially for the hybrid ranking that will be presented next. So, the random forest was set as the only algorithm for the following experiments.

5.3.1 Feature selection with cross-validation

Results for SFS-F (solid line) and SFS-B (dashed line) are plotted in Figure 9 with approach A in green and approach B in blue.

For approach A, SFS-B was able to select 6 features and reached $F_1 = 0.8644$. SFS-F selected 15 features with a little higher F_1 of 0.8724. In approach B, both SFS algorithms have selected even fewer features, 5 for SFS-F and only 4 for SFS-B. However, in this case, the final metric for SFS-F was better with $F_1 = 0.7886$ against $F_1 = 0.6932$ from SFS-B. An early stop criterium, checking if F_1 improved from the previous round, would have been successful if combined with SFS, except for SFS-F in approach A, where it would have prevented the algorithm to reach the optimum value of 15 features. In that case, the result would be $NoF = 8$ with $F_1 = 0.8171$. To be clear, stop criteria are common practice in experiments like these, but were not used here.

In all four experiments, one can observe that classification improves in the first third of features and then decreases.

With these findings, one can directly compare Figure 9 with Figure 5. More than resemblance, the plots are quite similar. That is a demonstration of the curse of dimensionality.

GA was the best selection method for case A, reaching $F_1 = 0.9117$ with 8 features. For case B, the SFS forward won with $F_1 = 0.7886$ selecting 5 features. Tracking the GA over the generations is presented in Figure 10. In terms of the number of features, SFS backward was the most efficient, greatly reducing the problem dimension and also improving the output metric. The HRW methods also have done a good job in decreasing the number of feature while raising the classification. In fact, HRW RF-RF for case B has reached F_1 higherthan SFS-B, but in other cases, they stood behind. Complete results are presented in Table 10.

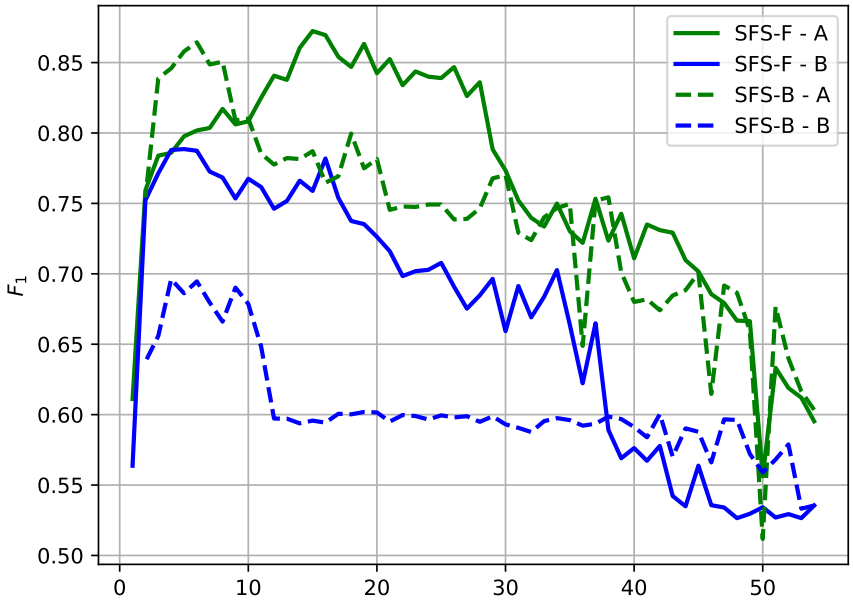
5.3.2 Feature selection with nested cross-validation

The application of feature selection with nested cross-validation results in a different number of features for each fold. The number of features is presented in Table 8, F_1 in Table 9, and the compiled results in Table 11.

For the case A, the three combinations of HRW resulted in fewer features, but lower F_1 if compared to previous values. For case B, the metric drops even more, but the number of features varies.

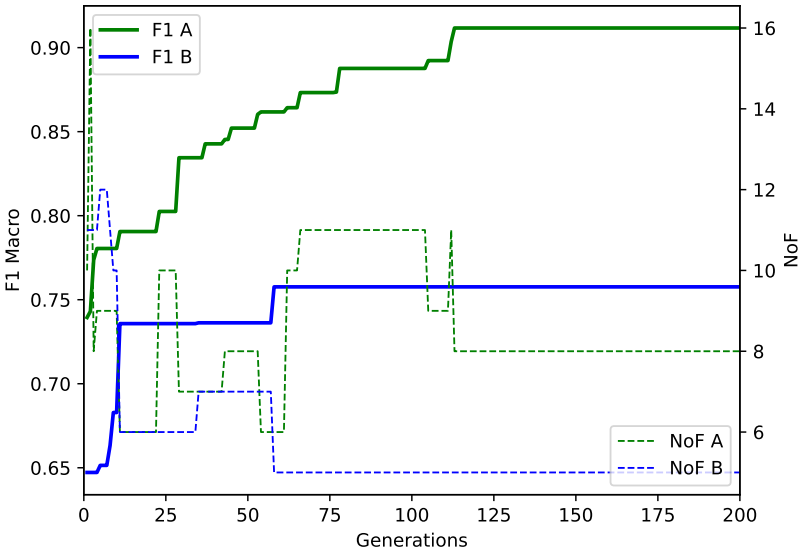
The results from SFS-F and SFS-B regarding the number of features is similar to what it

Figure 9 – Sequential Feature Selection Forward (SFS-F) and Backward (SFS-B). Results of F_1 for each of the number of features (NoF) selected with random forest (RF) classifier.



Source: Created by the author.

Figure 10 – GA evolution over generations.



Source: created by the author.

was with cross-validation (non-nested). However, the values of F_1 are lower, especially for case B with $F_1 \approx 0.39$. Nevertheless, there was an improvement. SFS-F decreased the features from 55 to an average of 15.2 (across all 5 folds), and increased F_1 from 0.5878 to 0.7070 in case A. In case B, results from both SFS methods are disappointing.

GA was the most efficient method in this approach. On average, the problem dimension was reduced to an average of 14.8 features, resulting in $F_1 = 0.7860$ for case A, which is an interesting value considering that bias has been removed. For case B, GA was quite similar,

Table 8 – Number of features (NoF) for each fold numbered with the respective oil well identifier, and the final F_1. Values in parentheses are NoF before SFS-B.

Algorithm	**Case**	**#1**	**#2**	**#4**	**#5**	**#7**	F_1
SFS Forward	A	20	15	10	8	23	0.7070
	B	11	12	3	11	4	0.3947
SFS Backward	A	7	9	3	4	5	0.6411
	B	6	4	4	6	5	0.3930
HRW-ANOVA	A	11 (21)	11 (24)	11 (26)	11 (26)	11 (21)	0.4901
	B	11 (16)	6	9	11	11 (18)	0.3850
HRW-MI	A	11 (15)	8	11 (20)	11	11 (12)	0.5139
	B	11 (15)	8	8	7	11 (12)	0.4073
HRW-RF	A	11	9	9	5	11 (12)	0.6627
	B	6	11	3	6	7	0.5519
GA	A	11	15	16	17	15	**0.7860**
	B	13	12	11	12	13	**0.7429**

with $NoF = 12.2$ and $F_1 = 0.7429$, a significant improvement over the reference value of 0.5204. It is important to remember that the fitness function has focused only on the classification performance. Hence, feature selection is an indirect reward.

The GA has another important advantage over both SFS. It can be tuned. Although it was not part of this study, it is possible to conjecture that results may be subject to further improvement with the same algorithm.

Table 9 – Nested cross-validation experiment results, with intermediate F_1 values numbered with the respective well identifier. Final F_1 is the mean from the 5-folds.

Algorithm	**Case**	**#1**	**#2**	**#4**	**#5**	**#7**	F_1
SFS Forward	A	0.7269	0.7268	0.6988	0.9268	0.4556	0.7070
	B	0.4782	0.3859	0.4992	0.4782	0.1321	0.3947
SFS Backward	A	0.6770	0.7683	0.7645	0.9957	0.0000	0.6411
	B	0.4638	0.4527	0.4527	0.4638	0.1321	0.3930
HRW-ANOVA	A	0.5946	0.4358	0.5221	0.4394	0.4588	0.4901
	B	0.4654	0.6707	0.6172	0.4394	0.1321	0.4650
HRW-MI	A	0.4198	0.8541	0.4822	0.4394	0.3741	0.5139
	B	0.4724	0.4512	0.5416	0.4394	0.1321	0.4073
HRW-RF	A	0.7159	0.5578	0.5854	0.9957	0.4588	0.6627
	B	0.5158	0.4558	0.6621	0.9936	0.1321	0.5519
GA	A	0.8210	0.7118	0.8195	0.7345	0.8433	0.7860
	B	0.7696	0.7654	0.6873	0.6391	0.8530	0.7429

Table 10 – Number of features (NoF) and F_1 results for experiments with cross-validation. *Reference* refers to the base problem with RF classifier with default hyperparameters and a total of 30 features.

Case	**Reference**	**Grid Search**	**55 features**	**Ranking Method**	**Wrapper method**		**SFS-F**		**SFS-B**		**GA**	
					NoF	F_1	NoF	F_1	NoF	F_1	NoF	F_1
A	0.6434	0.6516	0.5878	ANOVA	21	0.6294	15	0.8724	*6*	0.8644	8	**0.9117**
				MI	12	0.7127						
				RF	13	0.8485						
B	0.5219	0.5435	0.5204	ANOVA	7	0.6139	5	**0.7886**	*4*	0.6932	11	0.7576
				MI	12	0.5271						
				RF	6	0.7571						

Table 11 – Average number of features (NoF) and average F_1 results for experiments with nested cross-validation.

Case	**Reference**	**Grid Search**	**55 features**	**Ranking Method**	**Wrapper method**		**SFS-F**		**SFS-B**		**GA**	
					NoF	F_1	NoF	F_1	NoF	F_1	NoF	F_1
A	0.6434	0.6516	0.5878	ANOVA	11.0	0.4901	15.2	0.7070	5.6	0.6411	14.8	**0.7860**
				MI	10.4	0.5139						
				RF	9.0	0.6627						
B	0.5219	0.5435	0.5204	ANOVA	9.6	0.4650	8.2	0.3947	5.0	0.3930	12.2	**0.7429**
				MI	9.0	0.4073						
				RF	6.6	0.5519						

5.3.2.1 Feature importance

Combining results from all six experiments with nested cross-validation, cases A and B, and five folds each, there were 589 features selected. The total number of features analyzed is 3300. Feature frequency count is presented in Table 12, showing that sample entropy, standard deviation, and variance are the top three. Surprisingly, the *mean* was the least selected.

However, higher values in Table 12 can be a symptom of two causes: i) the feature is relevant to describe the problem, or ii) the feature selection algorithm was inefficient and unable to remove redundancy, masking which are the best features.

Table 12 – Frequency count for selected feature in all six experiments with nested cross-validation.

Feature function	**Frequency**
sample entropy	83
standard deviation	73
variance	67
kurtosis	58
max	51
min	50
percentile 25%	50
median	49
skewness	47
percentile 75%	35
mean	26
Total	589

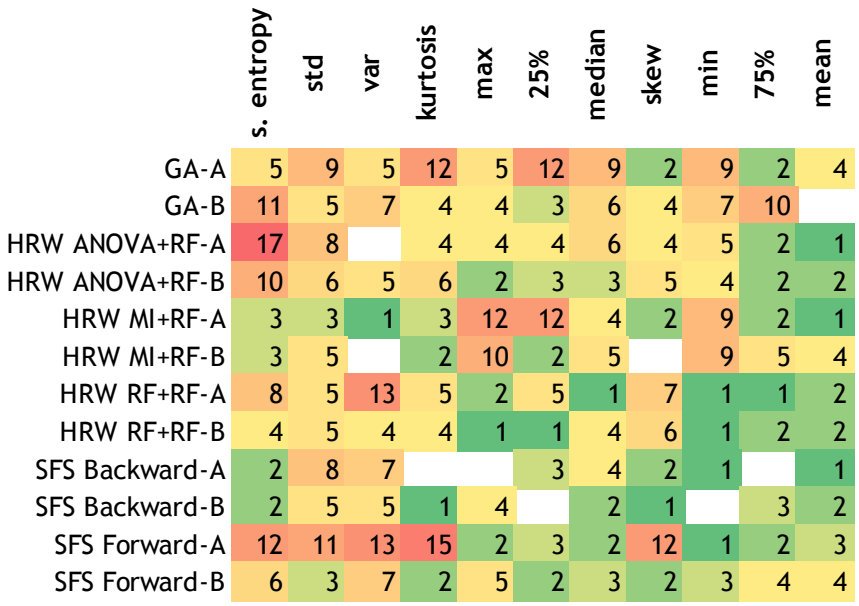
In Figure 11 there's a heat map for the algorithm-case vs. feature function count. It is possible to see the higher utilization of sample entropy concentrated for hybrid ranking wrapper in case A and B, and also for GA in case A, and SFS Forward in case B. Standard deviation, on the other hand, was distributed across the algorithms. The variance was not distributed as standard deviation, but also not concentrated as sample entropy. This indicates that standard deviation and variance could be the most relevant statistical features.

For the mutual information algorithm, there was some concentration for the maximum function. The kurtosis function seems to be useful mostly for GA and SFS forward only in case A.

The SFS backward has selected the lesser number of features if one takes both approaches (simple cross-validation vs. nested cross-validation). For this algorithm, the more important features were standard deviation and variance.

Contrarily, the HRW with ANOVA has selected greater amounts of features, and it relied especially on sample entropy.

Figure 11 – Heat map showing the frequency count for the combination of algorithm-experiment with feature functions.



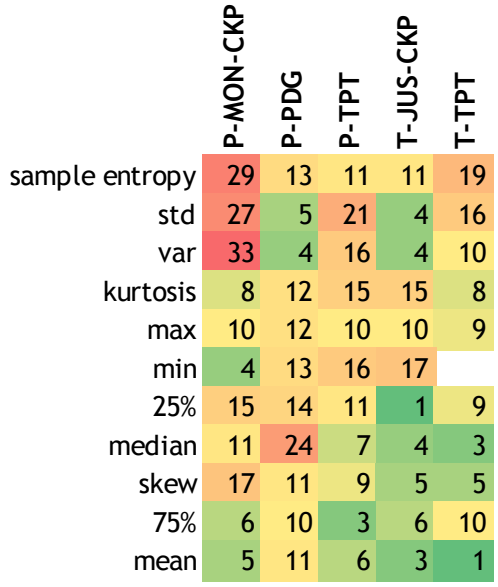
Source: created by the author.

5.3.2.2 Sensor redundancy

The P-MON-CKP sensor was the most frequent in all 12 experiments combined. Its features appeared 165 times.

Following the intuition in the previous section, when the feature selector makes poor choices, the classifier relies on a higher number of features. That might be the case for sample entropy. However, it can be seen clearly in Figure 12, the P-MON-CKP stands out from the other sensors.

Figure 12 – Heat map presenting the frequency count of each sensor and its respective feature functions.



Source: created by the author.

Looking into the details of results from P-MON-CKP, in case A, standard deviation and variance appear together in 11 times out of the 30 folds (5 folds for each one of the 6 algorithms). In 13 times they appear exclusively, either standard deviation or variance. In the remaining 6 times none of them were selected. For case B, in 19 times standard deviation or variance appeared exclusively and in 8 times they were never selected. Only for 3 times they were together. At least for case B, feature selection algorithms seemed to have found that standard deviation and variance are closely related.

Interestingly, the T-JUS-CKP, which is installed close to the P-MON-CKP, was the least used appearing only 80 times. In fact, from the original 55 features, 33 are pressure-related. Yet, after feature selection, 71% of features are calculated on pressure values. This finding indicates that the flow instability is an event better recognizable by pressure dynamics.

Another aspect is that T-JUS-CKP might be expendable. Although, it is one of easy repair because it lies on the platform. For the TPT sensor, the results were in the middle. One may argue that it has contributions in all models. It is not clear so far and it requires further analysis.

Pressure at the PDG was the second most selected. This is the hardest to fix and repair. To get access to this sensor, special offshore oil rigs are required and operations are costly.

6 Conclusion

In this dissertation, the flow instability abnormal event in oil wells was defined and analyzed in a data-driven approach. It was shown that this fault is related to the multiphase hydrocarbon flow with frequent manifestation. Hence, it is an important matter that requires dedicated solutions.

A workflow of multiple machine learning tasks was proposed and successfully applied. A combination of feature extraction, model selection, and evaluation, and feature selection guided the study to an improved solution when compared to initial experimentation.

The initial experiment has shed light on important aspects of the flow instability abnormal event. It was used as starting point and base reference. Firstly, because the traditional splitting strategy resulted in an output metric comparable to prior publications. Secondly, because a novel approach to cross-validation was defined and shown to be crucial to establish the ideal experimental design. A design without compromising essential concepts of the machine learning theory and practice, effectively removing optimistic biases.

The grid search technique was applied in a traditional setting. Different from extensive publications in the literature, it was not able to improve the results satisfactorily.

From the grid search results in hand, it was possible to argue that original features were not able to capture and represent the flow instability event. Then, the reasonable move was to increase the problem dimension with an extended list of statistical features. Unfortunately, results were even lower than before grid search. However, keeping in mind the curse of dimensionality, it was plausible that many of those features were redundant and highly correlated.

To remove dependent and irrelevant features, multiple feature selection algorithms were applied, in two different strategies. Using the entire dataset with cross-validation, results seemed promising. Both SFS-F and SFS-B greatly reduced the number of features while incresing the classification performance. GA was able to go even further, reaching $F_1 = 0.9117$.

However, when bias is removed, through the use of nested cross-validation with well splits, results are more realistic. All algorithms tried in this study removed notable amount of features, but the final classification metric was not satisfactory for case B. In case A, SFS-F with $F_1 = 0.7070$ is a reasonable achievement. Nevertheless, the genetic algorithm was the best in both cases in this approach. With more than 70% of reduction in problem dimension, and F_1 of 0.7860 and 0.7429 in cases A and B, respectively, the proposed workflow in this dissertation achieved its goals.

The SFS backward was the most effective if one considers only the number of features. It fails to generalize, however. For an industrial application, lower NoF is not the top requirement. The industry is likely more interested in correct classifications.

The hybrid ranking wrapper methods were fair alternatives because they run fast (less

computer demand). Yet, the results possibly lead to too many misclassifications.

In all experiments, feature selection was able to reduce the problem dimension and avoid the curse of dimensionality. They add an extra layer of computation in model development, but from the results in this study, it is a promising technique worth taking to further studies.

Having found that less than 30% of features are useful, it was important to answer which are those features and from what sensor they come from. It was shown that, although the sample entropy was the most selected feature in the nested cross-validation experiments, it doesn't mean it is the most important one. Sample entropy was concentrated mostly in a few cases. On the other hand, standard deviation and variance were the most selected right after sample entropy, and they are notably distributed across all experiments. The mean, which is extensively used to describe a dynamic of a signal, was the least selected.

And finally, analyzing the sensors, it was clear the pressure sensors express the patterns associated with the flow instability. The P-MON-CKP was the most used, and with standard deviation and variance, as already discussed. This is great news for offshore oil operations since this sensor is installed on the platform allowing ease of maintenance and repair.

6.1 Future work

In this study, it was possible to bring substantial new information on flow instability fault detection by applying multiple machine learning techniques. That is not to say, however, that was exhaustive research. As a suggestion, some improvements can be followed, as:

- use bayesian optimization instead of arbitrary subspace for grid search. Grid search looks for the best hyperparameters in all pre-configured points. This would eventually waste computational resources in regions where the algorithm has already found evidence contrary to the improvement direction. Then, using a smarter optimization method, one could even combine hyperparameter tuning with feature selection in the same step;
- tweak the genetic algorithm. GA has been extensively studied in the past years, which has lead to the development of several modifications in the operators and the selection process. A GA variation could go even further in the flow instability problem;
- or even apply deep neural networks (deep learning). Given the fair amount of observations in the 3W dataset, the application of deep neural networks could bring improvements from generalization capabilities;
- the present study has concerned with the flow instability. The application of the workflow presented in this here may be a challenge for other types of abnormal events, especially those with fewer collected instances. In any case, further experimentation is needed and it may require changes in the pipeline;
- apply hypothesis tests to evaluate statistical significance among results. This could improve analysis of findings and lead to more robust conclusions.

Bibliography

ALDRICH, C.; AURET, L. *Unsupervised process monitoring and fault diagnosis with machine learning methods*. [S.l.]: Springer, 2013.

ALI, J. et al. Random forests and decision trees. *International Journal of Computer Science Issues (IJCSI)*, v. 9, n. 5, p. 272, 2012. Publisher: Citeseer.

BARAL, C.; GELFOND, M. Reasoning agents in dynamic domains. In: *Logic-based artificial intelligence*. [S.l.]: Springer, 2000. p. 257–279.

BATALDEN, B.-M.; LEIKANGER, P.; WIDE, P. Towards autonomous maritime operations. In: *2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*. [S.l.]: IEEE, 2017. p. 1–6.

BEHNEL, S. et al. Cython: The best of both worlds. *Computing in Science & Engineering*, IEEE, v. 13, n. 2, p. 31–39, 2011.

BERRIEL, R. F. et al. Monthly energy consumption forecast: A deep learning approach. In: *2017 International Joint Conference on Neural Networks (IJCNN)*. [S.l.]: IEEE, 2017. p. 4283–4290.

BOLDT, F. de A. et al. Fast feature selection using hybrid ranking and wrapper approach for automatic fault diagnosis of motorpumps based on vibration signals. In: *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*. [S.l.]: IEEE, 2015. p. 127–132.

BREIMAN, L. Random forests. *Machine learning*, v. 45, n. 1, p. 5–32, 2001. Publisher: Springer.

BROCKWELL, P. J.; DAVIS, R. A. *Introduction to Time Series and Forecasting*. 3 edition. ed. [S.l.]: Springer, 2016.

BRØNSTAD, C. *Data-driven detection and identification of undesirable events in subsea oil wells*. Dissertação (Mestrado) — Universitetet i Sørøst-Norge, 2020.

BROOMHEAD, D. S.; LOWE, D. Radial basis functions, multi-variable functional interpolation and adaptive networks. *Royal Signals and Radar Establishment Malvern (United Kingdom)*, 1988.

BURMAN, P. A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods. v. 76, n. 3, p. 503–514, 1989. Publisher: Oxford University Press.

BZDOK, D.; ALTMAN, N.; KRZYWINSKI, M. *Points of significance: statistics versus machine learning*. [S.l.]: Nature Publishing Group, 2018.

CADEI, L. et al. Big Data Advanced Anlytics to Forecast Operational Upsets in Upstream Production System. In: *Abu Dhabi International Petroleum Exhibition & Conference*. [S.l.]: Society of Petroleum Engineers, 2018.

CARUANA, R.; NICULESCU-MIZIL, A. An empirical comparison of supervised learning algorithms. In: *Proceedings of the 23rd international conference on Machine learning*. [S.l.: s.n.], 2006. p. 161–168.

CHANDRASHEKAR, G.; SAHIN, F. A survey on feature selection methods. *Computers & Electrical Engineering*, v. 40, n. 1, p. 16–28, 2014. Publisher: Elsevier.

CHEBEL-MORELLO, B.; MALINOWSKI, S.; SENOUSSI, H. Feature selection for fault detection systems: application to the Tennessee Eastman process. *Applied Intelligence*, v. 44, n. 1, p. 111–122, 2016. Publisher: Springer.

CHEN, M.; MAO, S.; LIU, Y. Big data: A survey. *Mobile networks and applications*, v. 19, n. 2, p. 171–209, 2014. Publisher: Springer.

CHICCO, D.; JURMAN, G. The advantages of the matthews correlation coefficient (MCC) over f1 score and accuracy in binary classification evaluation. v. 21, n. 1, p. 1–13, 2020. Publisher: Springer.

CIREŞAN, D.; MEIER, U.; SCHMIDHUBER, J. Multi-column Deep Neural Networks for Image Classification. *arXiv:1202.2745 [cs]*, fev. 2012. ArXiv: 1202.2745. Disponível em: <http://arxiv.org/abs/1202.2745>.

CORTES, C.; VAPNIK, V. Support-vector networks. *Machine Learning*, v. 20, n. 3, p. 273–297, set. 1995. ISSN 1573-0565. Disponível em: <https://doi.org/10.1007/BF00994018>.

COVER, T.; HART, P. Nearest neighbor pattern classification. *IEEE transactions on information theory*, v. 13, n. 1, p. 21–27, 1967. Publisher: IEEE.

CUNNINGHAM, P.; CORD, M.; DELANY, S. J. Supervised learning. In: *Machine learning techniques for multimedia*. [S.l.]: Springer, 2008. p. 21–49.

DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: *2009 IEEE conference on computer vision and pattern recognition*. [S.l.]: Ieee, 2009. p. 248–255.

DHANKHAD, S.; MOHAMMED, E.; FAR, B. Supervised machine learning algorithms for credit card fraudulent transaction detection: a comparative study. In: *2018 IEEE International Conference on Information Reuse and Integration (IRI)*. [S.l.]: IEEE, 2018. p. 122–125.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern Classification*. 2 edition. ed. New York: Wiley-Interscience, 2000. ISBN 978-0-471-05669-0.

ESTéVEZ, P. A. et al. Normalized mutual information feature selection. *IEEE Transactions on neural networks*, v. 20, n. 2, p. 189–201, 2009. Publisher: IEEE.

FERNANDES JÚNIOR, W. et al. Detecção de anomalias em poços produtores de petróleo usando aprendizado de máquina. *Anais da Sociedade Brasileira de Automática*, v. 2, n. 1, 2020.

FIGUEIREDO, I. S. et al. Detecting interesting and anomalous patterns in multivariate time-series data in an offshore platform using unsupervised learning. In: *Offshore Technology Conference*. [S.l.]: OnePetro, 2021.

FREUND, Y.; SCHAPIRE, R. E. Experiments with a new boosting algorithm. In: *icml*. [S.l.]: Citeseer, 1996. v. 96, p. 148–156.

FUKUNAGA, K. *Introduction to Statistical Pattern Recognition*. 2ª edição. ed. [S.l.]: Academic Press, 1990. ISBN 978-1-4933-0048-8.

GEN, M.; CHENG, R.; WANG, D. Genetic algorithms for solving shortest path problems. In: *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)*. [S.l.]: IEEE, 1997. p. 401–406.

GHOJOGH, B.; CROWLEY, M. The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial. *arXiv preprint arXiv:1905.12787*, 2019.

GRANDINI, M.; BAGLI, E.; VISANI, G. Metrics for multi-class classification: an overview. *arXiv preprint*, 2020. Disponível em: <https://arxiv.org/abs/2008.05756>.

HARRIS, C. R. et al. Array programming with NumPy. *Nature*, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: <https://doi.org/10.1038/s41586-020-2649-2>.

HASTIE, T. et al. Multi-class adaboost. *Statistics and its Interface*, v. 2, n. 3, p. 349–360, 2009. Publisher: International Press of Boston.

HAWKINS, D. M. The problem of overfitting. *Journal of chemical information and computer sciences*, v. 44, n. 1, p. 1–12, 2004. Publisher: ACS Publications.

HODGE, V.; AUSTIN, J. A survey of outlier detection methodologies. *Artificial intelligence review*, v. 22, n. 2, p. 85–126, 2004. Publisher: Springer.

HOLLAND, J. H. Genetic algorithms. *Scientific american*, v. 267, n. 1, p. 66–73, 1992. Publisher: JSTOR.

HOQUE, N.; BHATTACHARYYA, D. K.; KALITA, J. K. MIFS-ND: A mutual information-based feature selection method. *Expert Systems with Applications*, v. 41, n. 14, p. 6371–6385, 2014. Publisher: Elsevier.

HUANG, G.-B.; ZHU, Q.-Y.; SIEW, C.-K. Extreme learning machine: theory and applications. *Neurocomputing*, v. 70, n. 1-3, p. 489–501, 2006. Publisher: Elsevier.

HUGHES, G. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, v. 14, n. 1, p. 55–63, 1968.

INDYK, P.; MOTWANI, R. Approximate nearest neighbors: towards removing the curse of dimensionality. In: *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. [S.l.: s.n.], 1998. p. 604–613.

ISERMANN, R. Model-based fault-detection and diagnosis–status and applications. *Annual Reviews in control*, v. 29, n. 1, p. 71–85, 2005. Publisher: Elsevier.

JACKSON, R. B. The integrity of oil and gas wells. *Proceedings of the National Academy of Sciences*, v. 111, n. 30, p. 10902–10903, 2014.

JORDAN, M. I.; MITCHELL, T. M. Machine learning: Trends, perspectives, and prospects. *Science*, v. 349, n. 6245, p. 255–260, 2015. Publisher: American Association for the Advancement of Science.

JOVIĆ, A.; BRKIĆ, K.; BOGUNOVIĆ, N. A review of feature selection methods with applications. In: *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*. [S.l.]: Ieee, 2015. p. 1200–1205.

KHAN, A. et al. A review of machine learning algorithms for text-documents classification. *Journal of advances in information technology*, v. 1, n. 1, p. 4–20, 2010.

KIRA, K.; RENDELL, L. A. A practical approach to feature selection. In: *Machine learning proceedings 1992*. [S.l.]: Elsevier, 1992. p. 249–256.

KUMAR, V.; MINZ, S. Feature selection: a literature review. v. 4, n. 3, p. 211–229, 2014.

LASI, H. et al. Industry 4.0. *Business & information systems engineering*, v. 6, n. 4, p. 239–242, 2014. Publisher: Springer.

LEARDI, R.; BOGGIA, R.; TERRILE, M. Genetic algorithms as a strategy for feature selection. v. 6, n. 5, p. 267–281, 1992. Publisher: Wiley Online Library.

LERMAN, P. M. Fitting segmented regression models by grid search. v. 29, n. 1, p. 77–84, 1980. Publisher: Wiley Online Library.

MARINS, M. A. et al. Fault detection and classification in oil wells and production/service lines using random forest. *Journal of Petroleum Science and Engineering*, v. 197, p. 107879, 2021. Publisher: Elsevier.

MEHMOOD, R. M.; DU, R.; LEE, H. J. Optimal feature selection and deep learning ensembles method for emotion recognition from human brain EEG sensors. *Ieee Access*, v. 5, p. 14797–14806, 2017. Publisher: IEEE.

MENZE, B. H. et al. A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data. *BMC bioinformatics*, v. 10, n. 1, p. 1–16, 2009. Publisher: Springer.

MITCHELL, T. M. *Machine Learning*. Edição: 1. New York: McGraw-Hill Science/Engineering/-Math, 1997. ISBN 978-0-07-042807-2.

MOHAMED, A.; HAMDI, M. S.; TAHAR, S. A machine learning approach for big data in oil and gas pipelines. In: *2015 3rd International Conference on Future Internet of Things and Cloud*. [S.l.]: IEEE, 2015. p. 585–590.

NARGESIAN, F. et al. Learning Feature Engineering for Classification. In: *Ijcai*. [S.l.: s.n.], 2017. p. 2529–2535.

OSHIRO, T. M.; PEREZ, P. S.; BARANAUSKAS, J. A. How many trees in a random forest? In: *International workshop on machine learning and data mining in pattern recognition*. [S.l.]: Springer, 2012. p. 154–168.

PAL, M. Random forest classifier for remote sensing classification. *International journal of remote sensing*, v. 26, n. 1, p. 217–222, 2005. Publisher: Taylor & Francis.

PALMIGIANI, F. *Petrobras launches fresh Brazil deep-water drilling tender - here's how many rigs it is chasing | Upstream Online*. 2021. Section: rigs_and_vessels. Disponível em: <https://www.upstreamonline.com/rigs-and-vessels/petrobras-launches-fresh-brazil-deep-water-drilling-tender-heres-how-many-rigs-it-is-chasing/2-1-968570>.

PAO, Y.-H.; PARK, G.-H.; SOBAJIC, D. J. Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing*, v. 6, n. 2, p. 163–180, 1994. Publisher: Elsevier.

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

PUDIL, P.; NOVOVICOVÁ, J.; KITTLER, J. Floating search methods in feature selection. *Pattern recognition letters*, v. 15, n. 11, p. 1119–1125, 1994. Publisher: Elsevier.

RISH, I. An empirical study of the naive Bayes classifier. In: *IJCAI 2001 workshop on empirical methods in artificial intelligence*. [S.l.: s.n.], 2001. v. 3, p. 41–46. Issue: 22.

RODRIGUEZ, J. D.; PEREZ, A.; LOZANO, J. A. Sensitivity analysis of k-fold cross validation in prediction error estimation. v. 32, n. 3, p. 569–575, 2009. Publisher: IEEE.

ROH, Y.; HEO, G.; WHANG, S. E. A survey on data collection for machine learning: a big data-ai integration perspective. *IEEE Transactions on Knowledge and Data Engineering*, 2019. Publisher: IEEE.

ROSS, B. C. Mutual information between discrete and continuous data sets. *PloS one*, v. 9, n. 2, p. e87357, 2014. Publisher: Public Library of Science.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 3rd ed. edição. ed. Upper Saddle River: Prentice Hall, 2009. ISBN 978-0-13-604259-4.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, v. 3, n. 3, p. 210–229, 1959. Publisher: IBM.

SARKER, I. H. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, v. 2, n. 3, p. 1–21, 2021. Publisher: Springer.

SCHWAB, K. *The Fourth Industrial Revolution: what it means and how to respond*. 2016. Disponível em: <https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond/>.

SHAILAJA, K.; SEETHARAMULU, B.; JABBAR, M. A. Machine Learning in Healthcare: A Review. In: *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. [S.l.: s.n.], 2018. p. 910–914.

SIEDLECKI, W.; SKLANSKY, J. A note on genetic algorithms for large-scale feature selection. In: *Handbook of pattern recognition and computer vision*. [S.l.]: World Scientific, 1993. p. 88–107.

SOKOLOVA, M.; LAPALME, G. A systematic analysis of performance measures for classification tasks. *Information processing & management*, v. 45, n. 4, p. 427–437, 2009. Publisher: Elsevier.

SOUZA, J. N. M. D. et al. Modeling, simulation and optimization of continuous gas lift systems for deepwater offshore petroleum production. *Journal of Petroleum Science and Engineering*, v. 72, n. 3-4, p. 277–289, 2010.

TAKEI, J. et al. Flow Instability In Deepwater Flowlines And Risers-A Case Study Of Subsea Oil Production From Chinguetti Field, Mauritania. In: *SPE Asia Pacific Oil and Gas Conference and Exhibition*. [S.l.]: Society of Petroleum Engineers, 2010.

TUMER, K.; GHOSH, J. Estimating the bayes error rate through classifier combining. In: *Proceedings of 13th international conference on pattern recognition*. [S.l.]: IEEE, 1996. v. 2, p. 695–699.

TURAN, E. M.; JASCHKE, J. Classification of undesirable events in oil well operation. In: *2021 23rd International Conference on Process Control (PC)*. [S.l.: s.n.], 2021. p. 157–162.

TURING, A. M. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX, n. 236, p. 433–460, out. 1950. ISSN 0026-4423. Disponível em: <https://doi.org/10.1093/mind/LIX.236.433>.

USMANI, M. et al. Stock market prediction using machine learning techniques. In: *2016 3rd International Conference on Computer and Information Sciences (ICCOINS)*. [S.l.]: IEEE, 2016. p. 322–327.

VARGAS, R. E. V. *Base de dados e benchmarks para prognóstico de anomalias em sistemas de elevação de petróleo*. Tese (Doutorado) — Universidade Federal do Espírito Santo, 2019.

VARGAS, R. E. V. et al. A realistic and public dataset with rare undesirable real events in oil

wells. *Journal of Petroleum Science and Engineering*, v. 181, p. 106223, out. 2019. ISSN 0920-4105. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0920410519306357>.

VAZ, M. A. et al. Experimental determination of axial, torsional and bending stiffness of umbilical cables. In: *Proceedings of the 17th International Offshore & Arctic Engineering Conference (OMAE'98)*. [S.l.: s.n.], 1998. p. 7.

VENKATASUBRAMANIAN, V. et al. A review of process fault detection and diagnosis: Part i: Quantitative model-based methods. v. 27, n. 3, p. 293–311, 2003. Publisher: Elsevier.

WANG, M. et al. Machine learning for networking: Workflow, advances and opportunities. *IEEE Network*, v. 32, n. 2, p. 92–99, 2017.

WOOD, T. *Random Forests*. 2020. Disponível em: <https://deepai.org/machine-learning-glossary-and-terms/random-forest>.

YANG, H.; MOODY, J. Feature selection based on joint mutual information. In: *Proceedings of international ICSC symposium on advances in intelligent data analysis*. [S.l.]: Citeseer, 1999. v. 23.

ZAKARIAN, E. Analysis of two-phase flow instabilities in pipe-riser systems. *ASME-PUBLICATIONS-PVP*, v. 414, n. 1, p. 187–196, 2000.