Pontifícia Universidade Católica
do Rio de Janeiro

André Ricardo Ducca Fernandes

Reservoir classification using well-testing pressure derivative data

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Hélio Côrtes Vieira Lopes

Rio de Janeiro
March 2021

Pontifícia Universidade Católica
do Rio de Janeiro

André Ricardo Ducca Fernandes

Reservoir classification using well-testing pressure derivative data

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the Examination Committee.

Prof. Hélio Côrtes Vieira Lopes
Advisor
Departamento de Informática – PUC-Rio

Dr. Guilherme Gonçalves Schardong
Departamento de Informática – PUC-Rio

Prof. Abelardo Borges Barreto Júnior
Departamento de Matemática – PUC-Rio

Drª. Priscila Magalhães Ribeiro
Cenpes/Petrobras

Rio de Janeiro, March 26^{th}, 2021

André Ricardo Ducca Fernandes

Bachelor's in Economic Sciences (2006) at Instituto Vianna Júnior. Master of Business Administration (MBA) in Finance (2011) at Universidade Federal de Juiz de Fora (UFJF). Started his master's at PUC-Rio in 2019, focusing his research on Machine Learning, Time Series Classification, and Reservoir Model Classification.

Bibliographic data

Fernandes, André Ricardo Ducca

Reservoir classification using well-testing pressure derivative data / André Ricardo Ducca Fernandes; advisor: Hélio Côrtes Vieira Lopes. – Rio de Janeiro: PUC-Rio, Departamento de Informática, 2021.

v., 47 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Aprendizado de máquina. 2. Séries temporais. 3. Teste de poços. 4. Derivada de pressão. 5. Shapelets. I. Lopes, Hélio Côrtes Vieira. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

Firstly, I would like to thank God for guiding me through this 2-year journey. It has been a long way. Only by His grace, I'm here finishing this step, with my heart full of gratitude. I thank to my wife, Rebeca, for her love and support in the good and bad moments. It is our achievement. Also I thank my parents, Ricardo and Delza, for being so present in my life. Thank you for all dedication to your children. This moment is a result of such dedication. I thank my brother, William, for all the support and complicity. I wouldn't have done it without you.

I thank my family – Jesana, Tiago, Isabella, and Manuella – for all the moments together. With you, my life is happier. I want to say thank you to Hélio, my advisor, for counselling me along these years, and for teaching me many lessons. Also I thank Abelardo Barreto and Guilherme Schardong who worked with me in this project, from theoretical discussions to code implementation. I thank my laboratory friends for the discussions, the friendship, and for making the laboratory a place I loved to be. Also I thank Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) for the scholarship. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

Abstract

Fernandes, André Ricardo Ducca; Lopes, Hélio Côrtes Vieira (Advisor). **Reservoir classification using well-testing pressure derivative data**. Rio de Janeiro, 2021. 47p. Dissertação de mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Identifying a reservoir model is the first step to correctly interpret the data generated in a well-test and hence to estimate the related parameters to this model. The goal of this work is inversely to use the pressure curves, obtained in a well-test, to identify a reservoir model. Since the data obtained in a well-test can be ordered over time, we reduce this task to a problem of time series classification, where every reservoir model represents a class. For that purpose, we used a technique called shapelets, which are times series' subsequences that represent a class. From that, a new feature space was built, where we measured the distance between every time series and the shapelets of every class. Then we created an ensemble using the models k-nearest neighbors, decision tree, random forest, support vector machines, perceptron, multi-layer perceptron, and adaboost. The preprocessings standard scaler, normalizer, robust scaler, power transformer, and quantile transformer were tested. Then the classification was performed on the new preprocessed feature space. We generated 10 analytical multiclass reservoir models for validation. The results reveal that the use of classical machine learning models with shapelets, using the normalizer and quantile transformer preprocessing, reaches solid results on the identification of reservoir models.

Keywords

Machine learning; Time series; Well-testing; Pressure derivative; Shapelets.

Resumo

Fernandes, André Ricardo Ducca; Lopes, Hélio Côrtes Vieira. **Classificação de reservatório utilizando dados da derivada de pressão de teste de poços**. Rio de Janeiro, 2021. 47p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Identificar o modelo de um reservatório é o primeiro passo para interpretar corretamente os dados gerados em um teste de poços e desta forma estimar os parâmetros relacionados a esse modelo. O objetivo deste trabalho é de forma inversa, utilizar as curvas de pressão obtidas em um teste de poços, para identificar o modelo de um reservatório. Como os dados obtidos em um teste de poços podem ser ordenados ao longo do tempo, nossa abordagem será reduzir essa tarefa a um problema de classificação de séries temporais, onde cada modelo de reservatório representa uma classe. Para tanto, foi utilizada uma técnica chamada shapelet, que são subsequências de uma série temporal que representam uma classe. A partir disso, foi construído um novo feature space, onde foi medida a distância entre cada série temporal e as shapelets de cada classe. Então foi criado um comitê de votação utilizando os modelos k-nearest neighbors, decision tree, random forest, support vector machines, perceptron, multi layer perceptron e adaboost. Foram testados os pré-processamentos standard scaler, normalizer, robust scaler, power transformer and quantile transformer. Então a classificação foi feita no novo feature space pré-processado. Geramos 10 modelos de reservatório multiclass analíticos para validação. Os resultados revelam que o uso de modelos clássicos de aprendizado de máquina com shapelets, usando os pré-processamentos normalizer e quantile trasformer alcança resultados sólidos na identificação dos modelos de reservatório.

Palavras-chave

Aprendizado de máquina; Séries temporais; Teste de poços; Derivada de pressão; Shapelets.

Table of Contents

List of Figures

List of Tables

I have observed something else under the sun. The fastest runner doesn't always win the race, and the strongest warrior doesn't always win the battle. The wise sometimes go hungry, and the skillful are not necessarily wealthy. And those who are educated don't always lead successful lives. It is all decided by chance, by being in the right place at the right time.

Solomon, *New Living Translation Bible, 2015, Ecclesiastes 9:11.*

1
Introduction

Well-test interpretation is an important step in inferring the properties and parameters of a reservoir model. One of the most important information obtained during a well-test is the pressure measured in the bottom-hole of a well. Bourdet et al. found that the derivative of this pressure has a signature accordingly to the type of reservoir (Bourdet et al., 1989). Since the pressure derivative can be ordered over time, we can reduce the task of identifying a reservoir model to a time series classification problem, using the pressure derivative plots obtained in a well-test.

In recent years, the research on the time series classification area has grown. Ye and Keogh proposed in 2009 a data mining primitive to identify a signature on time series curves (Ye and Keogh, 2009). The name of this technique is Shapelet. It has been successfully applied in the literature (Xing et al., 2011, Lines et al., 2012, Patri et al., 2014, Hameurlain et al., 2017, Ahmadi et al., 2017b).

In this work, we propose a system to classify the pressure derivative plots obtained in a well-test. The proposed system can help specialists to correctly interpret the well-test, and thus classify the reservoir model.

Our proposal differs from previous work (Athichanagorn and Horne, 1995, Allain and Horne, 1990, Ahmadi et al., 2017b) since we use shapelets with classical machine learning models to perform multiclass classification.

Since we use supervised Machine Learning algorithms, we need an annotated dataset to train and evaluate the system. We have built an analytical multiclass dataset composed of two groups of classes. The first group indicates whether the reservoir is homogeneous, fracture, fault, or linear flow. The second one indicates whether the reservoir is infinite radial, pseudo-steady-state, or steady-state.

The remainder of this work is organized as follows. Chapter 2 presents related work in the literature. We describe the methodology and the dataset used in the experiments in Chapter 3. Chapter 4 shows the results. Finally, we present the conclusions, examining the contributions and suggesting future work, in Chapter 5.

2
Related work

In this chapter, we present an overview of the existing work in the literature related to well-testing model identification in the context of reservoir classification.

When performing a well-testing interpretation, the first step is to identify the underlying reservoir model and then infer the parameters related to the model (Lee, 1982).

Allain and Horne proposed a technique to identify the well-testing model in 1990 (Allain and Horne, 1990). They used syntactic information combined with a rule-based approach applied to pressure derivative plots to perform the well-testing model identification.

Al-Kaabi and Lee, in 1993, were the first to eliminate the use of complex rules to identify the well-testing model (Al-Kaabi and Lee, 1993). Their approach was based on the use of Artificial Neural Networks (ANN) to identify the models. The dataset was composed of seven classes. The results of their research showed that the use of artificial neural networks is effective in identifying the well-test interpretation model. They also noted that the use of high-quality data (without noise) to train the ANN reaches better results.

After this work, many others flourished in the field of neural networks. In 1996, Sung et al. proposed the use of Hough Transform in conjunction with a Backpropagation Neural Network to extract simple patterns including noise from the full dataset and to find the correct interpretation model from the well-test (Sung et al., 1996). The dataset used to validate their work had four classes. They concluded that their method could be applied to any type of data.

Deng et al. in 2000 proposed a new method to transform the data points in binary numbers to use as input vectors to train an ANN (Deng et al., 2000). They guarantee that the test curves are in the same model area as its training samples, and as a result, the interpretation model can be correctly identified. The dataset used was composed of three classes. The conclusion was that the proposed method is more efficient than the data series approach.

Vaferi et al. presented an attempt to use a Recurrent Neural Network (RNN) for the interpretation model classification in 2015 (Vaferi et al., 2015).

They used a dataset with eight classes to validate the study. As a result, they compared the accuracy of their model with a multi-layer perceptron. The RNN reached better results, showing better generalization capacity.

In 2016, Vaferi et al. proposed another approach where he used a discrete wavelet transform (DWT) for reducing the amount of data that is used to train a multi-layer perceptron to predict the interpretation model (Vaferi et al., 2016). The dataset classes are the same as his work of 2015. This approach reached good results for the interpretation model identification task.

Ahmadi et al. in 2017 proposed a methodology for the use of ANN to determine the conceptual reservoir model (Ahmadi et al., 2017a). They tested different types of ANNs including multi-layer perceptrons, probabilistic neural networks, and regression neural networks, using a dataset with seven different classes. The result of this research was a methodology to reservoir model identification from pressure derivative plots using ANNs.

In 2017, Ahmadi et al. did another work in the area of reservoir classification, but at this time using an ensemble of classical machine learning models with shapelets (Ahmadi et al., 2017b). Shapelets were used to transform the original data into a new feature space to identify the reservoir models. The machine learning algorithms used were random forest, support vector machines, logistic regression, and probabilistic neural networks. The dataset used to validate their work had five different classes. They used the one-versus-all method to perform the classification task, creating one ensemble to classify each class. This study concluded that shapelets are an effective technique to extract representative features of the pressure derivative plots and are useful for well-testing model classification.

Chu et al. in 2019 were the first to consider the use of the convolutional neural networks (CNN) to predict the reservoir model based on pressure derivative plots (Chu et al., 2019). They used a dataset with five different classes to validate their research and concluded that CNNs are more suitable for the classification of well-test derivative plots than fully connected neural networks (FCNN).

As we observed, a large part of the literature on well-test model classification uses a type of neural network to approach this problem. In 2017, Ahmadi changed this strategy using shapelets to capture the different signatures in the pressure derivative curves and create a new feature space to train an ensemble of machine learning models (Ahmadi et al., 2017b).

The first difference between our work to the literature is the use of only classical machine learning models in a voting fashion to make the final

classification. This proposal is based on the scarcity of annotated data in real life to be used to train machine learning models, and as known, neural network algorithms rely on plenty of annotated data for training.

The second difference is that, in our work, we tested several data preprocessing to find the ones that best fit reservoir classification's context. As in real-life data is noisier and can be more dispersed, we did extensive tests, with different data preprocessing to serve as a starting point to use when working with real-life data.

Finally, we generated ten different multiclass scenarios and classified them into two steps. The intuition here is to train our models to identify a specific class even it is suffering actions of diverse flow regimes.

3
Methodology

In this chapter, we first explain the approach used to generate the analytical scenarios used to train, validate and test our learning models as well as the division of classes to perform the multiclass prediction in section 3.1. Then, we show the data preprocessing used in this work in section 3.2. In addition, in section 3.3, we present a time-series technique called shapelets, which was used to extract representative features of pressure derivative curves, creating a new feature space to train our classifiers. Next, in section 3.4, we describe the machine learning models used to compose the voting classifier and accomplish the final classification. Lastly, we specify the metrics used to evaluate our work in section 3.5.

3.1
Dataset Creation

We generated the set of scenarios by solving 10 differential equations of well-test analytical models and generating 200 scenarios of each class. Table 3.1 contains a summary of the generated scenarios by class. Figure 3.1 shows these 2000 scenarios. All the columns represent respectively the scenarios with INFINITE RADIAL, PSEUDO-STEADY-STATE, or STEADY-STATE flow regimes. In contrast, each row represents respectively the scenarios with HOMOGENEOUS, FRACTURE, FAULT, or LINEAR FLOW regimes.

The infinite radial class represents the scenarios where after the wellbore storage effects are over, and we don't see reservoir boundaries, the reservoir acts as if it were infinite in extent. The pseudo steady state class describes a reservoir closed on all sides, while the steady state class represents a reservoir enclosed for a constant pressure boundary (Horne, 1995).

The fault class represents scenarios where we have an impermeable barrier close to the well. The fracture class describes reservoirs with a double porosity behavior. In contrast, the homogeneous class represents reservoirs with homogeneous properties. The linear flow class represents scenarios with a well placed between two parallel sealing faults (Horne, 1995, Bourdet, 2002).

In Figure 3.2 we can see an example of each scenario by class.

To generate the scenarios, we varied the permeability $\kappa \in [500, 10000]$ with a step of 500mD. We still varied the well-flow rate $q \in [100, 1000]$ with steps of 100m^3/day. All scenarios were generated with a skin factor of 2 and a wellbore storage coefficient of $10\text{m}^3/(\text{kgf/cm}^2)$. All other parameters used to generate the scenarios were kept constant and are specified in Table 3.2.

The derivative is obtained by the derivation of the pressure measured during a drawdown period in respect of the natural logarithm of time using the well-known Bourdet's algorithm (Bourdet, 2002).

This is a multiclass dataset and to perform the reservoir classification task, we divided the labels into two groups. In the first group, we gather the classes FAULT, FRACTURE, HOMOGENEOUS, or LINEAR FLOW, shown in Figure 3.3a. The second group contains the rest of the classes, which are INFINITE RADIAL, PSEUDO STEADY STATE, or STEADY STATE, represented in Figure 3.3b.

Thus, we modeled our task to have two classification steps, in the first one is predicted the class for the first group of labels and in the second one is predicted the class for the second group of labels.

We divided our dataset into two parts, 80 percent of the scenarios to train and 20 percent to test our machine learning models.

Table 3.1: Summary of the generated scenarios by class.

Examples	**Reservoir model**
200	Homogeneous Infinite Radial
200	Homogeneous Pseudo Steady State
200	Homogeneous Steady State
200	Fracture Infinite Radial
200	Fracture Pseudo Steady State
200	Fracture Steady State
200	Fault Infinite Radial
200	Fault Pseudo Steady State
200	Fault Steady State
200	Linear flow Infinite Radial

3.2 Data Preprocessing

Kotsiantis et al., in their work about data preprocessing in 2006, showed that many elements affect the success of a Machine learning model, and the quality of the dataset is one of the main factors. We can have noisy data, redundant data, missing values, magnitude difference data etc., affecting the training step. Thus, data preprocessing becomes an essential

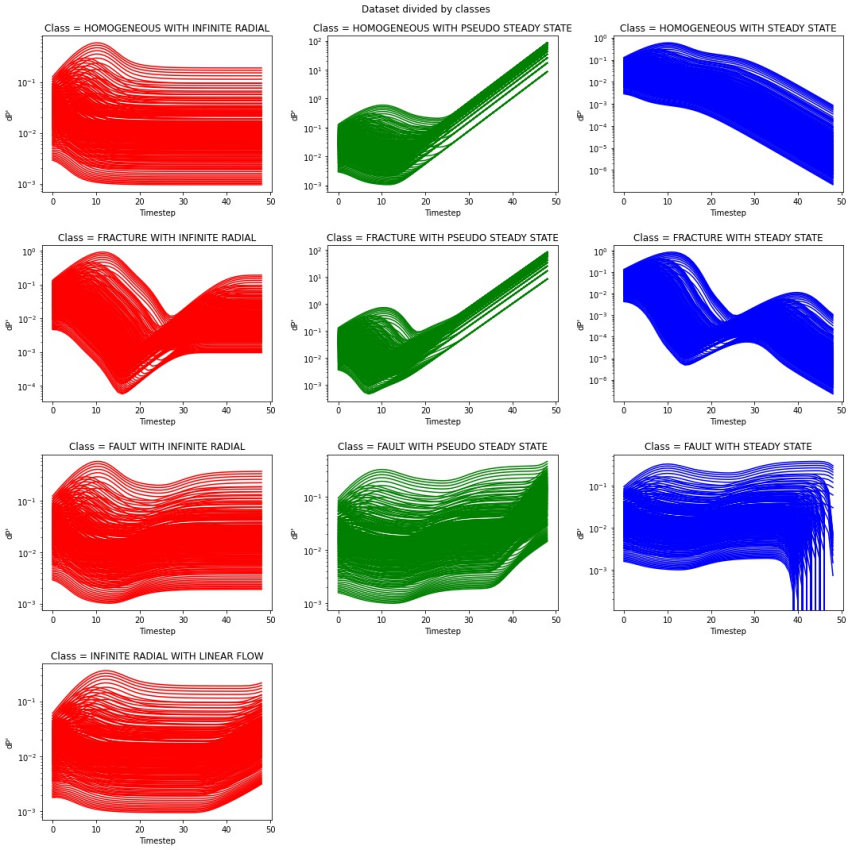


Figure 3.1: Logarithmic pressure derivatives of the scenarios generated solving the 10 differential equations. All the columns represent respectively the scenarios with INFINITE RADIAL, PSEUDO-STEADY-STATE, and STEADY-STATE flow regimes. In contrast, each row represents respectively the scenarios with HOMOGENEOUS, FRACTURE, FAULT, and LINEAR FLOW regimes.

Table 3.2: Nomenclature and standard values for the variables used in the dataset creation.

Variable	**Value(s)**	**Description**
q	Specified by model	Well flow rate (m^3/day)
μ	1.0	Fluid viscosity (cP)
B	1.0	Formation Factor Volume (m^3/m^3)
ϕ	0.3	Porosity
c_t	200×10^{-6}	Total system compressibility $((kgf/cm^2)^{-1})$
p_i	300.0	Reservoir initial pressure (kgf/cm^2)
r_w	0.1	Wellbore radius (m)
κ	Specified by model	Reservoir permeability (mD)
h	100	Layer width (m)
α_p	19.03	Constant
α_t	3.484×10^{-4}	Constant

step to create a final training dataset that correctly generalizes the initial dataset (Kotsiantis et al., 2006).

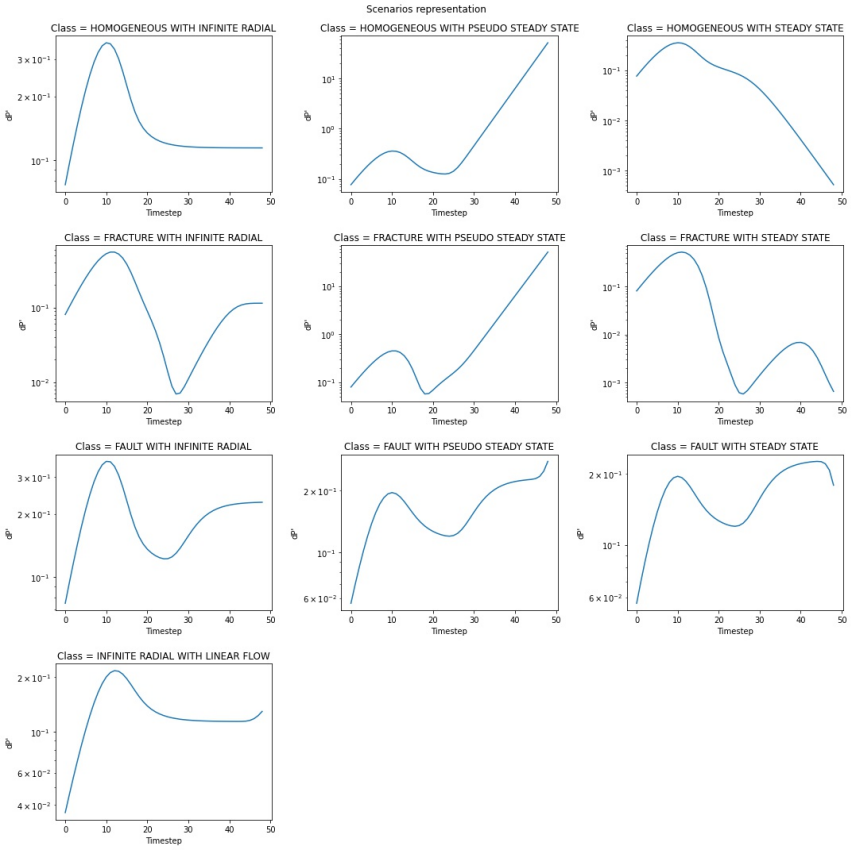


Figure 3.2: An example of each analytically generated scenario.

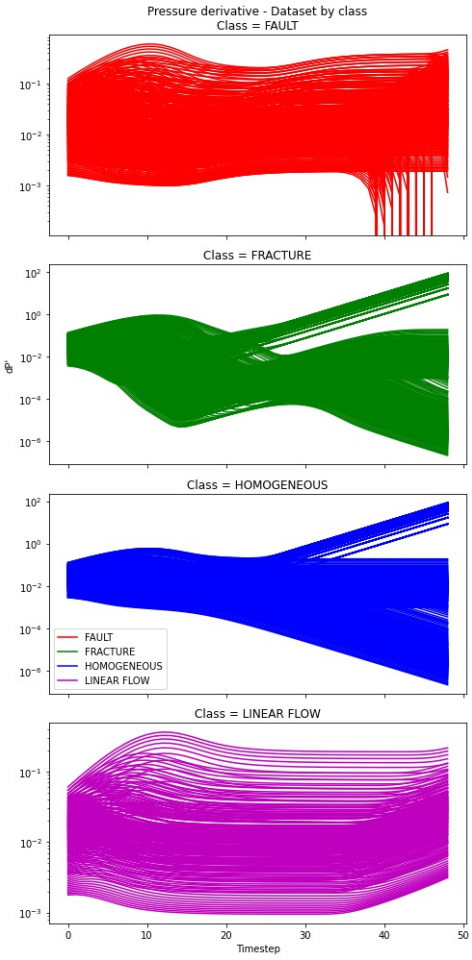
In this work, we tested several data preprocessing to evaluate which one reaches better results with this data. We evaluated the following data preprocessing: standard scaler, normalizer, robust scaler, power transformer, and quantile transformer. For each preprocessing, we present visualizations using the training sets.

Let $\mathbf{X}$ be a feature vector $\mathbf{X} = \{x_1, x_2, ..., x_n\}$ for the following data preprocessing formulations.

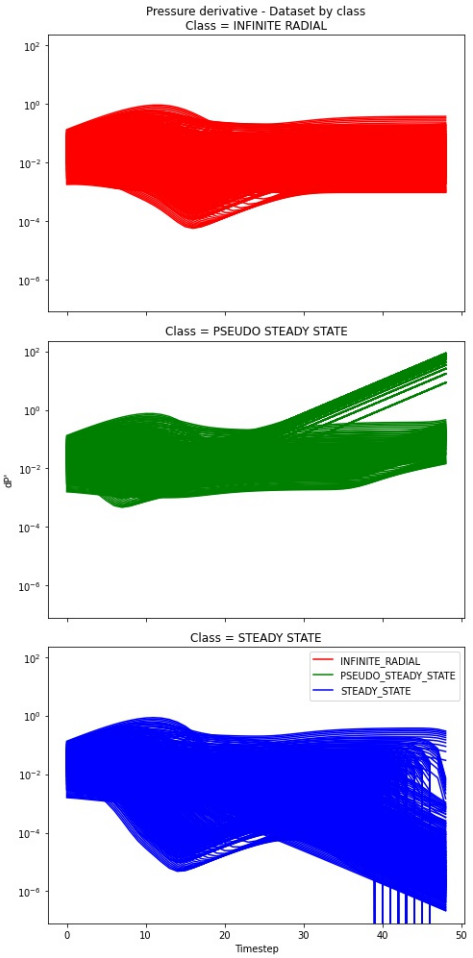
3.2.1 Standard Scaler

In this preprocessing, the first step is to center the data. For that, we remove the mean μ of each feature x_i of $\mathbf{X}$. After that, we divide this feature by the standard deviation σ to scale them. Figure 3.4 shows the scaled training set. The standard score z_i for a sample x_i of $\mathbf{X}$ is:

$$z_i = \frac{x_i - \mu}{\sigma}$$



(a) Logarithmic pressure derivative of the scenarios divided by the classes FAULT, FRACTURE, HOMOGENEOUS, and LINEAR FLOW.



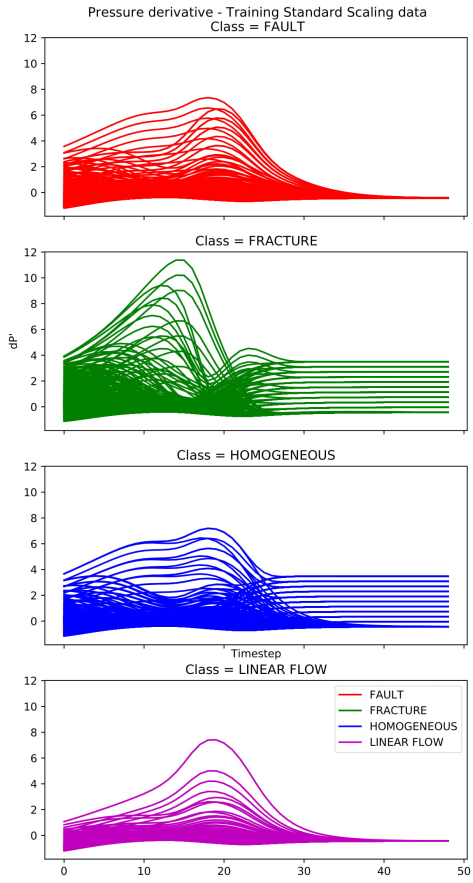
(b) Logarithmic pressure derivative of the scenarios divided by the classes INFINITE RADIAL, PSEUDO STEADY STATE, and STEADY STATE.

Figure 3.3: Logarithmic pressure derivatives of the scenarios grouped by the classes that were predicted in each step by the machine learning models.

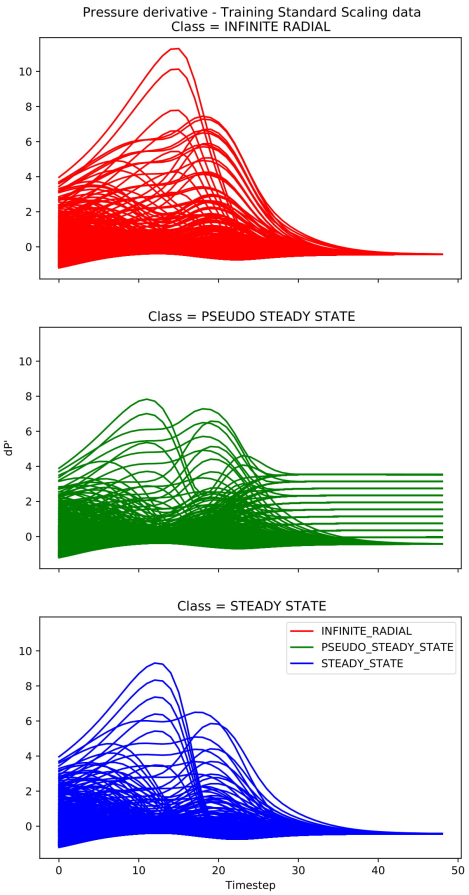
3.2.2 Normalizer

Here we choose the maximum norm to scale the dataset. It is expressed by each feature divided by the maximum feature value of the dataset. Thus all the feature values lie on between $[-1, 1]$. Figure 3.5 presents the normalized training set. Let m be the maximum value of the vector $\mathbf{X}$, then $m = \max\{x_1, x_2, ..., x_n\}$. The normalized value z_i of an element x_i of the vector $\mathbf{X}$ is:

$$z_i = \frac{x_i}{m}$$



(a) Training set divided by the classes FAULT, FRACTURE, HOMOGENEOUS, and LINEAR FLOW.



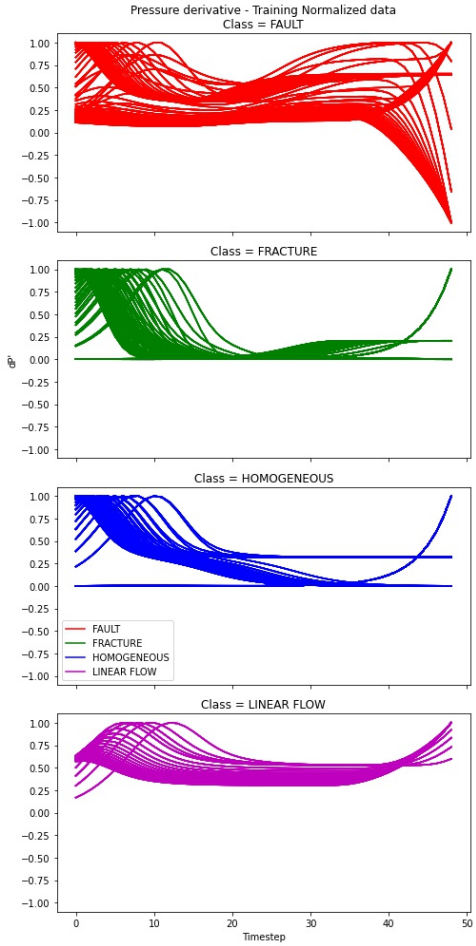
(b) Training set divided by the classes INFINITE RADIAL, PSEUDO STEADY STATE, and STEADY STATE.

Figure 3.4: Standard scaling of the training set.

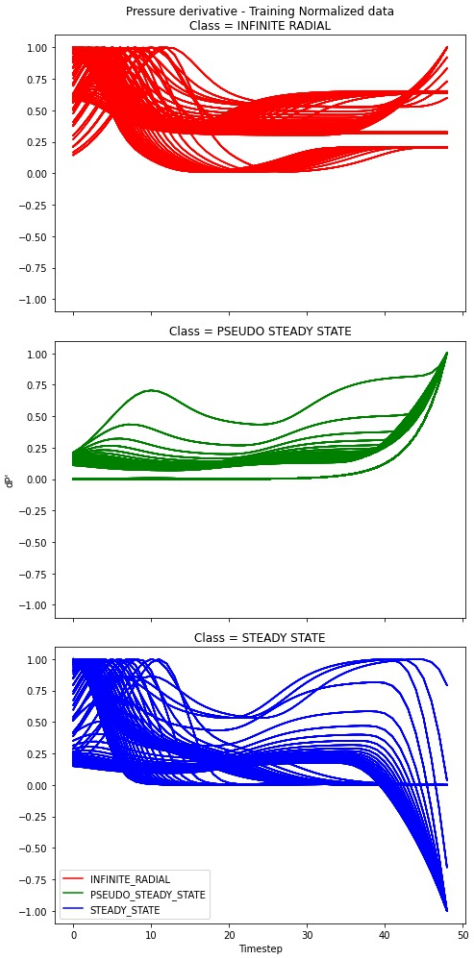
3.2.3 Robust Scaler

In some cases, outliers can negatively influence a machine learning model. So, the use of this preprocessing is a good option because it is robust to outliers. First, we center the features of $\mathbf{X}$ by removing the median md of each element. Then, we scale $\mathbf{X}$, dividing their features by the interquartile range (IQR) as shown in Figure 3.6. The IQR is the first quartile q_1 subtracted from the third quartile q_3. Thus, the robust score z_i for a sample x_i of $\mathbf{X}$ is:

$$z_i = \frac{x_i - md}{q_3 - q_1}$$



(a) Training set divided by the classes FAULT, FRACTURE, HOMOGENEOUS, and LINEAR FLOW.



(b) Training set divided by the classes INFINITE RADIAL, PSEUDO STEADY STATE, and STEADY STATE.

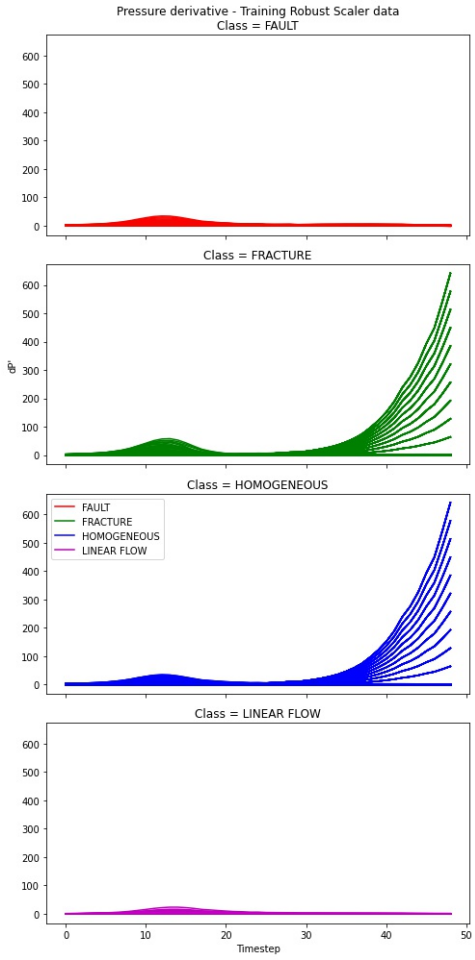
Figure 3.5: Normalization of the training set.

3.2.4 Power Transformer

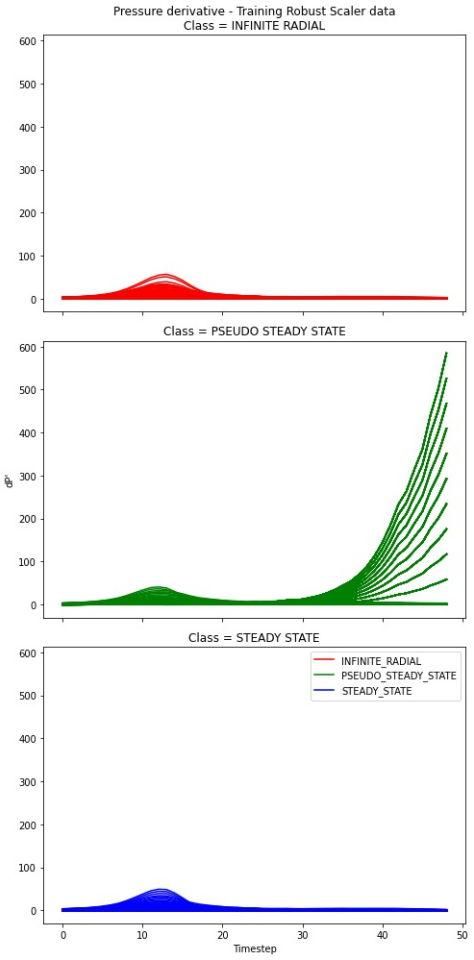
It is a monotonic transformation applied to make the data as close as possible to a Gaussian distribution. As we have positive and negative numbers in our dataset, we used the Yeo-Johnson transformation given by:

$$x_i^{(\lambda)} = \begin{cases} [(x_i+1)^{\lambda}-1]/\lambda & \text{if } \lambda \neq 0, x_i \geq 0, \\ \ln(x_i+1) & \text{if } \lambda = 0, x_i \geq 0 \\ -[(-x_i+1)^{2-\lambda}-1]/(2-\lambda) & \text{if } \lambda \neq 2, x_i < 0, \\ -\ln(-x_i+1) & \text{if } \lambda = 2, x_i < 0 \end{cases},$$

where λ is determined by the maximum likelihood estimation and x_i are the features of $\mathbf{X}$. Figure 3.7 presents the power transformed training set.



(a) Training set divided by the classes FAULT, FRACTURE, HOMOGENEOUS, and LINEAR FLOW.

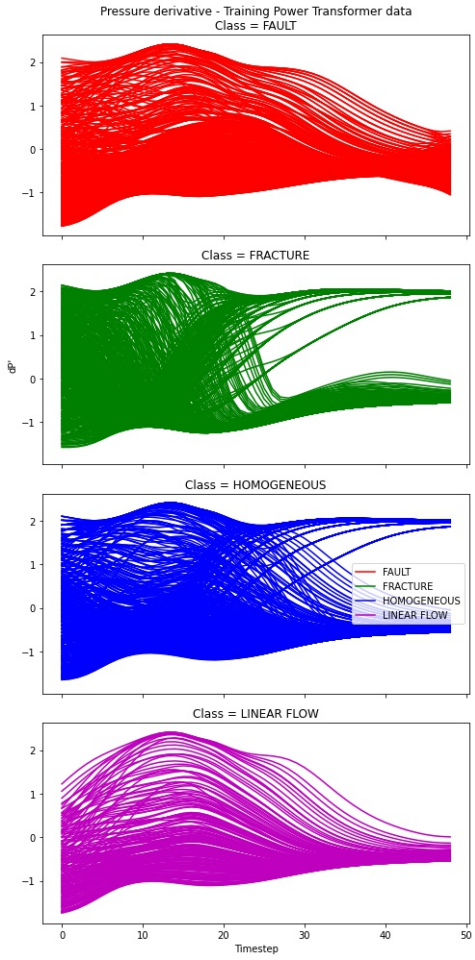


(b) Training set divided by the classes INFINITE RADIAL, PSEUDO STEADY STATE, and STEADY STATE.

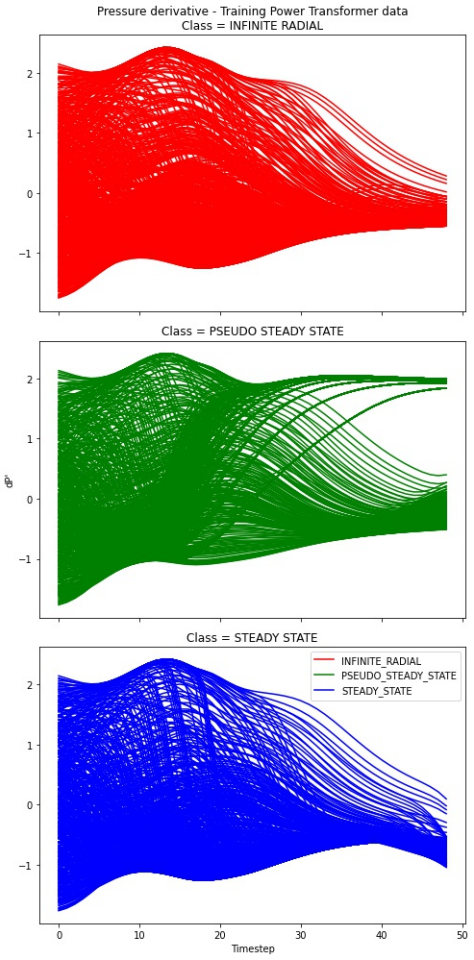
Figure 3.6: Robust scaling of the training set.

3.2.5 Quantile Transformer

This is also a monotonic transformation, but in this case, it makes the data transformed to follow a uniform or a normal distribution. This preprocessing conduces to separate the most frequent values. Firstly, we map the original values to a uniform distribution using an estimation of the cumulative distribution function. Then, the obtained values are mapped to the desired output distribution, applying the associated quantile function. In Figure 3.8, we have the uniform quantile transformation, while in Figure 3.9, we have the normal.



(a) Training set divided by the classes FAULT, FRACTURE, HOMOGENEOUS, and LINEAR FLOW.



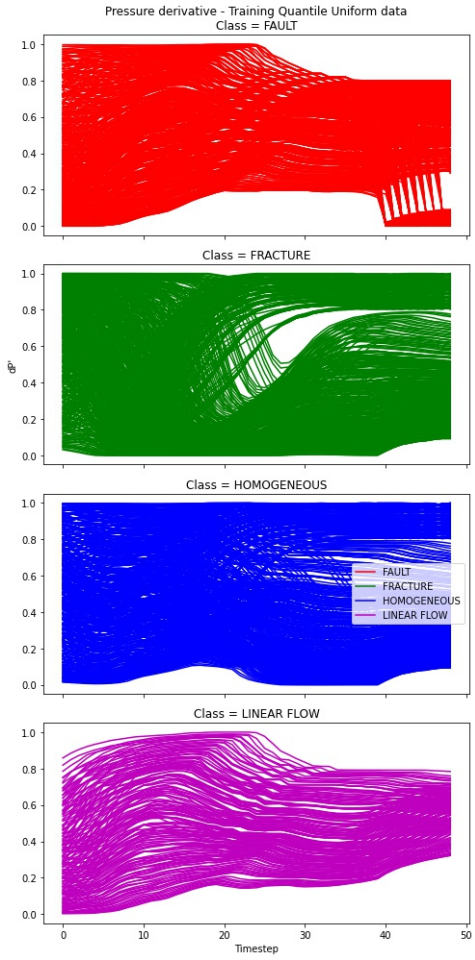
(b) Training set divided by the classes INFINITE RADIAL, PSEUDO STEADY STATE, and STEADY STATE.

Figure 3.7: Power transforming of the training set.

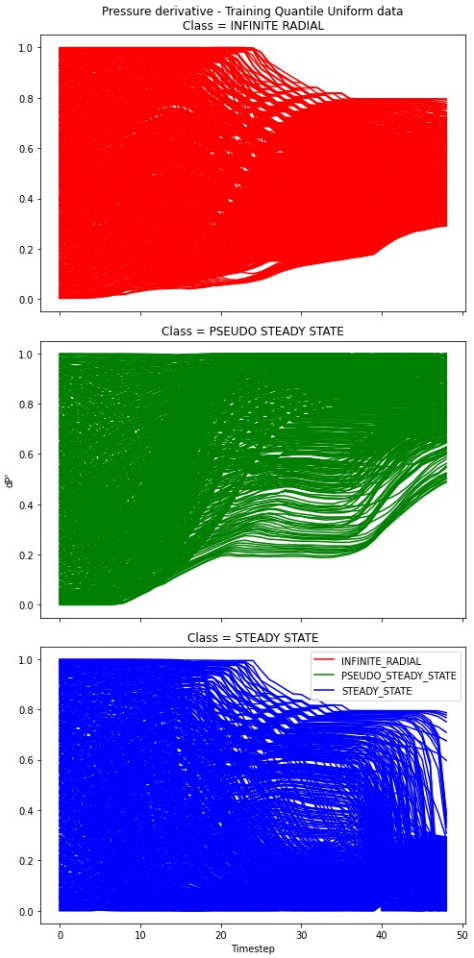
3.3 Shapelets

Shapelets are time series' subsequences that represent a class. In 2009, Ye and Keogh proposed this data mining primitive to identify a signature on time series curves (Ye and Keogh, 2009). In our context, we construct a new feature space using the distance of the time series from the shapelets. Then we use that as a feature to perform a time series classification using classical machine learning models.

Shapelets has been successfully applied in the literature. Xing et al. in 2011 did early classification in time series data using shapelets (Xing et al., 2011). In 2012, Lines et al. proposed to create a new data space transforming data by calculating the distances from a time series to each shapelet and then use this to perform the clas-



(a) Training set divided by the classes FAULT, FRACTURE, HOMOGENEOUS, and LINEAR FLOW.



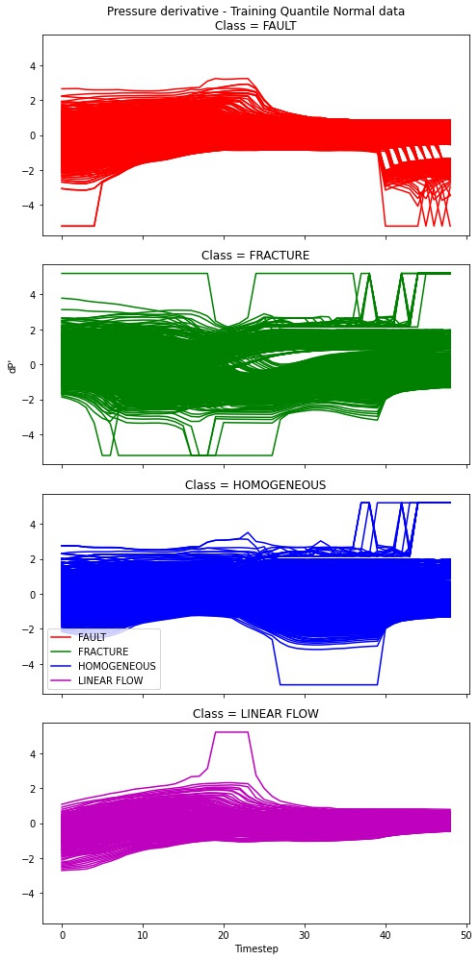
(b) Training set divided by the classes INFINITE RADIAL, PSEUDO STEADY STATE, and STEADY STATE.

Figure 3.8: Quantile uniform transforming of the training set.

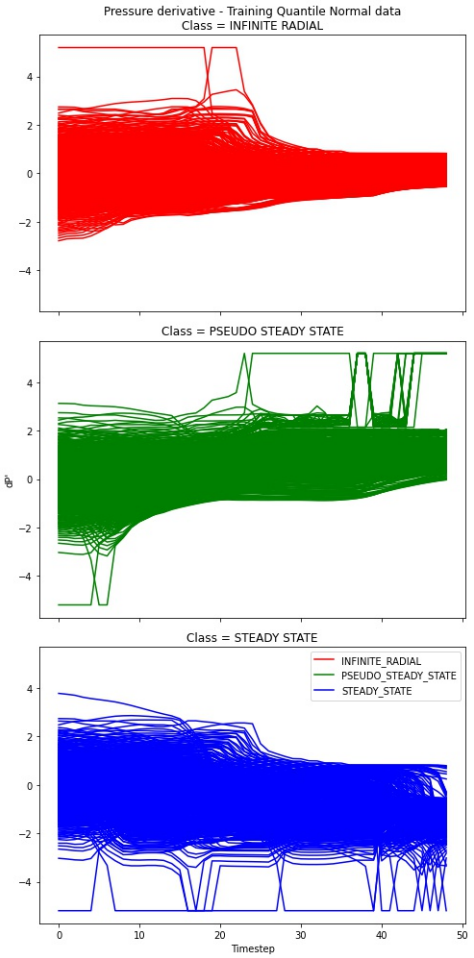
sification (Lines et al., 2012). Patri et al. in 2014 and 2016 employed shapelets for failure detection and failure prediction from oilfield sensor data (Patri et al., 2014, Patri et al., 2016). In 2017, Ahmadi et al. applied shapelets to perform well-testing model identification from pressure derivative plots (Ahmadi et al., 2017b).

Suppose we have a set of n time series, $\mathbf{T} = \{T_1, T_2, ..., T_n\}$, and each time series T_i has a class value c_i, where c_i is a value from the label set $\mathbf{l} = \{l_1, l_2, ..., l_k\}$. First, we need to generate the candidates to find a shapelet. A subsequence S of length q of a time series T_i of length m is a contiguous sequence of q points in T_i, where $m \geq q$. The set of all subsequences of length q for series T_i is $W_{i,q}$ and all subsequences of length q for $\mathbf{T}$ are $W_q = \{W_{1,q}, W_{2,q}, ..., W_{n,q}\}$.

The Euclidian distance between two subsequences S and R of length q is



(a) Training set divided by the classes FAULT, FRACTURE, HOMOGENEOUS, and LINEAR FLOW.



(b) Training set divided by the classes INFINITE RADIAL, PSEUDO STEADY STATE, and STEADY STATE.

Figure 3.9: Quantile normal transforming of the training set.

$$dist(S, R) = \sqrt{\left(\sum_{i=1}^{q}(s_i - r_i)^2\right)}.$$

We need to define a distance measure between a subsequence S and a time series T_i. We define that as the minimum distance between S and all subsequences of T_i of length q ($W_{i,q}$)

$$d_{i,S} = \min_{R \in W_{i,q}} dist(S, R).$$

We specify the Shanon Entropy of a subset $\mathbf{T}'$ of the set of time series $\mathbf{T}$ as

$$I(\mathbf{T}') = -p_0 \log p_0 - p_1 \log p_1$$

where p_1 is the fraction of the subset $\mathbf{T}'$ that has the class l_1 and p_0 is the

fraction of the subset that has a different class. It is also performed for l_2, ..., l_k. A smaller entropy reflects a more homogeneous set concerning the categories of its time series.

Given a subsequence S and a positive value p, let

$$\mathbf{T}_{<p} = \{i | dist(S, T_i) < p\} \text{ and } \mathbf{T}_{>p} = \{i | dist(S, T_i) > p\}$$

be, respectively, the time series that have smaller and larger distance to S than p. Now, we define a measure of the *discriminatory power* for a subsequence S and the set of time series $\mathbf{T}$ as

$$\max_{p>0} \left\{ I(\mathbf{T}) - \frac{|\mathbf{T}_{<p}|}{|\mathbf{T}|} \cdot I(\mathbf{T}_{<p}) - \frac{|\mathbf{T}_{>p}|}{|\mathbf{T}|} \cdot I(\mathbf{T}_{>p}) \right\}.$$

The *discriminatory power* increases when we have a p that splits $\mathbf{T}$ into subsets that are more homogeneous concerning the dataset's categories.

With that proposed, we can define a shapelet for $\mathbf{T}$ as a subsequence S with maximum discriminatory power.

We utilize in this research the classical shapelet algorithm that is described in Algorithm 1.

Algorithm 1 Shapelet Algorithm, adapted from Lines et al. (Lines et al., 2012).

Require: $\mathbf{T}$ as the time-series dataset, *min* is the minimum subsequence length and *max* is the maximum subsequence length

```
1: best ← 0
2: bestShapelet ← ∅
3: W ← genCandidates(T)
4: for l = min to max do
5:     for all subsequences S in W_l do
6:         quality, dist ← calcDistances(S,W_l)
7:         if quality > best then
8:             best ← quality
9:             bestShapelet ← S
10:        end if
11:    end for
12: end for
13: label ← getLabel(bestShapelet)
14: return bestShapelet, label
```

One of the advantages of shapelets is the gain of interpretability (Ye and Keogh, 2009, Lines et al., 2012). When using the new feature space

based on the distance between a pressure derivative curve and a shapelet, as input to train a machine learning model, the practitioner can interpret the predictions of the model, comparing the shapelets with the curves and observe if the extracted shapelet for a class is representative.

In Figure 3.10, we have an example of a shapelet extracted from our train dataset to represent the linear flow class. Firstly, we recognize that the shapelet captures the effect at the end of the pressure derivative that describes this class. Secondly, we observe that the shapelet is closer to the linear flow curve than other classes' curves. Due to that, we see that this shapelet correctly represents the linear flow class.

With a specialist in our workflow, he can identify if a shapelet is not detecting, for example, the desired flow regime, and based on that, choose not to use it. After the prediction, the specialist can also verify the shapelets used to form the feature space used to train the machine learning model, giving graphic insights into the problem domain.

Another advantage is that the use of shapelets makes the classification faster because the new feature space used to train our model is more compact (Ye and Keogh, 2009, Lines et al., 2012).

Despite the use of shapelets gives more accuracy to some datasets in literature, we hypothesize that for our domain, shapelets may be more robust to outliers and noise during the tests, at the cost of slightly lower classification scores. The increased robustness would allow the trained models to generalize better on new data.

3.4 Classification Models

One of the goals of this work is the use of classical machine learning models to perform the pressure derivative plots classification, since, in the real world, we do not always have an abundance of scenarios to train them. We restricted our experiments to the following models: k-Nearest Neighbors (k-NN), Decision Tree, Random Forest, Support Vector Machines (SVM), Perceptron, Multi-layer Perceptron, and Adaptive Boosting. We also used the ensemble learning method Voting Classifier to perform the final classification.

3.4.1 k-Nearest Neighbors

This algorithm considers all dataset examples, points in the n-dimensional space. The nearest neighbors of a point are defined in terms of distance, and the most common measure is the Euclidean distance. To deter-

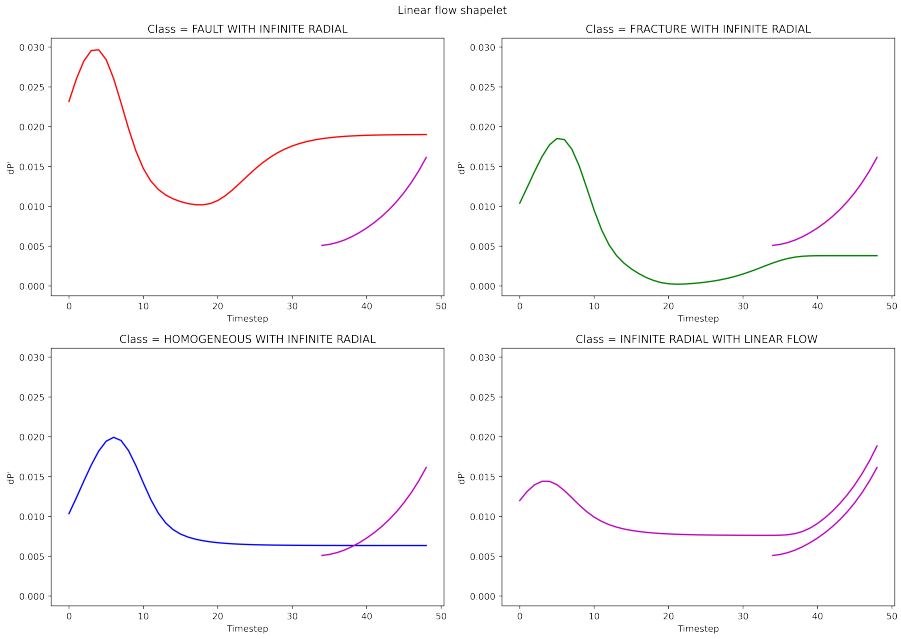


Figure 3.10: Linear flow shapelet plotted with one curve of FAULT, FRACTURE, HOMOGENEOUS and LINEAR FLOW, all with INFINITE RADIAL flow regime, for the unprocessed training dataset.

mine a class of one example, the algorithm takes the k closest samples from the training set. The most frequent class of the neighbors is then assigned to the example that is being evaluated. Usually, the distance between the elements is weighted, in such a way that the nearer neighbors contribute more to the class determination than the more distant ones (Mansbridge et al., 2018).

3.4.2 Decision Tree

Decision Tree is a method where the classification process is performed using a set of hierarchical decisions on the feature space, and it has the arrangement of a tree. The decision at a node of the tree (split criterion) is a condition on one or more features in the training data. Each node in a decision tree represents a subset of the data, and the goal is to identify a split criterion that minimizes the different classes in each branch (Aggarwal, 2015).

3.4.3 Random Forest

This method is a combination of multiple Decision Trees trained on variations of the training set. The result of the prediction is the combination of the prediction of their Decision Trees in a manner of majority voting or the application of weights (Mansbridge et al., 2018).

3.4.4 SVM

SVM is a classifier that creates a hyperplane that separates the samples of the training set in order to produce the best separation between two classes, known as *maximum margin hyperplane*. The points that define the margin are the support vectors. If the data points are linearly separable, a linear hyperplane is constructed to separate the classes. Otherwise, it is used a mapping function, known as kernel, that take the original data points to a much higher-dimensional space where they are linearly separable (Aggarwal, 2015, Mansbridge et al., 2018, Ahmadi et al., 2017b). To deal with n-class datasets, it is built n SVM classifiers using the one-versus-all approach.

3.4.5 Perceptron

Perceptron is a basic neural network architecture. It contains only two layers of nodes, the input nodes, and a unique output node. Each input node is connected by a weighted connection to the output node and receives and transmits a single numerical value to the output node. A mathematical function is applied on the inputs received by the output node to produce the output value. So the learning is achieved by adjusting the weights of the connections between the input nodes and the output node whenever the predicted label is wrong (Aggarwal, 2015).

3.4.6 Multi-layer Perceptron

The main difference from the Perceptron is that between the input and the output layer, we have a hidden layer. The nodes in the hidden layer can be connected with multiple layers, and it is assumed that the nodes in a layer are fully connected with the nodes of the next layer (Aggarwal, 2015).

3.4.7 Adaptive Boosting

In this algorithm, for every training instance, we have an associated weight, and the training is guided by the use of these weights. The purpose is to concentrate on the misclassified instances by increasing their relative weight after each interaction and decreasing the weights of the correctly classified instances. Adaptive Boosting is an ensemble learning that works with base classifiers and combines their outputs in a voting manner to determine the output at each step (Aggarwal, 2015).

3.4.8 Voting Classifier

Accordingly to Polikar, a set of classifiers trained in the same dataset can have different generalizations. In fact, he goes further and shows that even classifiers with comparable generalization may perform differently in the test dataset. To mitigate the risk of a poorly classification, he proposes combining the output of several classifiers (Polikar, 2006).

Our approach using the Voting Classifier combines different machine learning models, where each model's prediction represents one vote. We use the majority of votes to predict the class label.

3.5 Evaluation Metrics

To evaluate a classifier, we generally use a confusion matrix as illustrated in Figure 3.11. It has information about the predicted classification and actual classes. The entries in the confusion matrix are divided in:

- True Positives (TP) are the number of positive samples rightly classified as positive.
- False Positives (FP) are the number of negative samples misclassified as positive.
- True Negatives (TN) are the number of negative samples rightly classified as negative.
- False Negatives (FN) are the number of positive samples misclassified as negative.

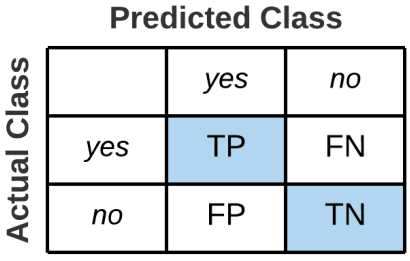


Figure 3.11: Illustration of a confusion matrix, where TP is true positives, TN is true negatives, FP is false positives, and FN is false negatives.

Based on the confusion matrix, we calculate some metrics. The first one is *precision* that can be considered as a measure of exactness. It is the number

of positive samples that were correctly classified divided by the number of samples classified as positive (Han et al., 2011). Let *precision* be

$$precision = \frac{TP}{TP + FP}$$

.

Next, we have *recall* that is a measure of completeness. It is the number of positive samples that were correctly classified divided by the number of positive samples (Han et al., 2011). Based on that, *recall* is defined as

$$recall = \frac{TP}{TP + FN}$$

.

The last measure is the harmonic mean of precision and recall, where we have equal weight for both. It is called *F-measure* – also known as *F1* or *F-score*. It is defined as

$$F_1 = \frac{2 * precision * recall}{precision + recall}$$

.

4
Experiments

We divided the experiments into two parts. In the first one, we use the dataset already described in section 3.1 to perform the reservoir classification, while in the second part, we added a sine noise $S \in [0.1; 0.5]$ to the dataset. In section 4.1, we describe the calibration of our models. In section 4.2, we show the results using the raw dataset and, in section 4.3, we present the outcomes using the dataset with sine noise.

4.1
Models Calibration

In order to calibrate our models, we performed a 10-fold cross-validation over the training set, testing each different hyperparameter combination. In this technique of model calibration, we divide the training set into k-folds – $F_1, F_2, ..., F_k$ – of the same size. Then we train and test k times, ensuring that each time the test subset is from a different fold. For instance, in iteration i, fold F_i is retained as the test set, and the other folds are used to train the model. This is used to avoid overfitting during the training step (Han et al., 2011).

Now we show the evaluated hyperparameters for each model.

- k-NN
  - Number of neighbors: range [4, 31]
- Decision Tree
  - Measure quality of a split: *gini* and *entropy*
  - Split criterion: *best* and *random*
  - Minimum number of samples required at a leaf node: range [2, 31]
- Random Forest
  - Measure quality of a split: *gini* and *entropy*
  - Minimum number of samples required at a leaf node: range [2, 31]
  - Number of trees: range [10, 200] with steps of 10
- SVM - Linear

  - Penalty: l_1 and l_2
  - Solve dual optimization problem: *true* and *false*
  - Regularization parameter: range [1, 10]
  - Maximum number of iterations: 10000
  - Multi-class strategy: *one-versus-rest* and *crammer singer*

- SVM - RBF

  - Regularization parameter: range [1, 10]
  - Maximum number of iterations: 10000

- SVM - Polynomial

  - Polynomial degree: 2, 3, and 4
  - Regularization parameter: range [1, 10]
  - Maximum number of iterations: 10000

- Perceptron

  - Penalty: l_1, l_2 and *elasticnet*
  - Maximum number of iterations: 10000
  - Early stoping: *true*

- Multi-layer Perceptron

  - Activation function for the hidden layer: *logistic*, *tanh*, and *relu*
  - Solver for weight optimization: *lbfgs* and *adam*
  - Early stoping: *true*
  - Proportion of training data to set aside as validation set for early stopping: 0.2

- Adaptive Boosting

  - Maximum number of estimators at which boosting is terminated: 50
  - Learning rate: 1
  - Algorithm: SAMME.R

- Voting Classifier

  - Voting type: *hard*
  - Cross-validation: 10
  - Number of models to compose voting: 3, 5, and 7

4.2 Multiclass Classification Using Multiple Shapelets Without Noise Data

As stated in section 3.1, our classification is performed in two steps. Thus we have two voting classifiers, one for each group of classes. The first group is composed of scenarios that contain FAULT, FRACTURE, HOMOGENEOUS, and LINEAR FLOW regime. The second one contains scenarios with INFINITE RADIAL, PSEUDO STEADY STATE, and STEADY STATE flow regime.

Table 4.1 presents the precision, recall, and F1 scores for the first group of classes. Each row represents the results for the unprocessed or preprocessed data without or with shapelets. We verified that the normalizer preprocessing without and with shapelets (*Normalizer* and *Shalelets - Normalizer* respectively) obtain the best scores, respectively 100% and 99% of precision, recall, and F1. Even obtaining worse results when using shapelets, for the unprocessed and normalizer preprocessing, we reach more than 90% of precision, recall, and F1 scores. It is worth mentioning that despite having a decrease in results using shapelets, the quantile transformers preprocessing achieved a satisfactory quality. In Figure 4.1 we can see a comparison of F1 scores for each preprocessing by using or not shapelets.

Table 4.1: Performance measures for the Voting classifier predicting FAULT, FRACTURE, HOMOGENEOUS, and LINEAR FLOW scenarios.

Preprocessing	**Precision**	**Recall**	**F1**
Unprocessed	0.96	0.96	0.96
Standard Scaler	0.92	0.95	0.92
Normalizer	1.00	1.00	1.00
Robust Scaler	0.87	0.93	0.87
Power Transformer	0.98	0.99	0.98
Quantile Transformer Normal	0.99	0.99	0.99
Quantile Transformer Uniform	0.98	0.99	0.98
Shapelets - Unprocessed	0.92	0.94	0.93
Shapelets - Standard Scaler	0.67	0.72	0.67
Shapelets - Normalizer	0.99	0.99	0.99
Shapelets - Robust Scaler	0.78	0.85	0.78
Shapelets - Power Transformer	0.80	0.84	0.80
Shapelets - Quantile Transformer Normal	0.89	0.91	0.89
Shapelets - Quantile Transformer Uniform	0.88	0.90	0.88

In Table 4.2, we show the results for the classification of the second group of classes. The normalizer preprocessing (*Normalizer* and *Shalelets - Normalizer* respectively) continue reaching the best scores, 100% when using or not shapelets. The unprocessed and quantile transformers preprocessing remain to obtain solid results, more than 95% of precision, recall, and F1

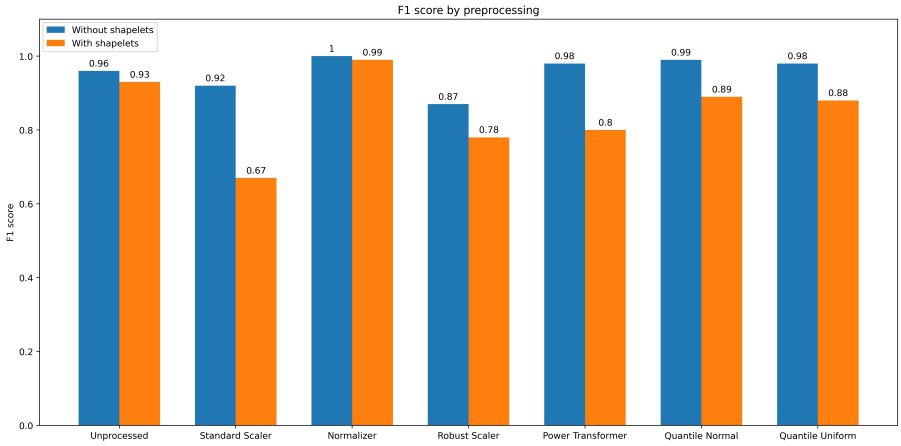


Figure 4.1: Comparison of F1 scores for each preprocessing by using or not shapelets for the Voting classifier predicting FAULT, FRACTURE, HOMOGENEOUS, and LINEAR FLOW scenarios.

scores, with or without shapelets. In Figure 4.2 we can see a comparison of F1 scores for each preprocessing by using or not shapelets.

As noted, the normalizer preprocessing achieved the best quality predicting the two groups of classes, regardless using shapelets or not. Based on that, we show in Table 4.3 the configurations of the best models used in the voting classifier to predict the first group of classes, using normalizer and normalizer with shapelets preprocessing.

Similarly, in Table 4.4, we present the configurations of the best models used in the voting classifier to predict the second group of classes, using the normalizer preprocessing with and without shapelets.

It is worth mentioning that in both cases the best scores were found using only the three best models in the voting classifier.

We can see that two simple models were used in both voting classifiers, k-NN and decision tree. The other models used were random forest – to compose the voting classifier of the first group of classes – and SVM with linear kernel – to compose the voting classifier for the second group of classes.

For this first experiment, the use of multiple shapelets with normalizer preprocessing proved to be an effective approach. Unprocessed, quantile transformer normal and uniform preprocessing also reached solid results, even having a decrease in the scores when using shapelets.

4.3 Multiclass Classification Using Multiple Shapelets With Noise Data

In this experiment, we added a sine noise $S \in [0.1; 0.5]$ for each time series in the dataset.

Table 4.2: Performance measures for the Voting classifier predicting the INFINITE RADIAL, PSEUDO STEADY STATE, and STEADY STATE scenarios.

Preprocessing	**Precision**	**Recall**	**F1**
Unprocessed	0.99	0.99	0.99
Standard Scaler	0.62	0.63	0.56
Normalizer	1.00	1.00	1.00
Robust Scaler	0.97	0.96	0.96
Power Transformer	0.75	0.87	0.73
Quantile Transformer Normal	0.99	0.99	0.99
Quantile Transformer Uniform	0.99	0.99	0.99
Shapelets - Unprocessed	0.96	0.97	0.96
Shapelets - Standard Scaler	0.62	0.62	0.57
Shapelets - Normalizer	1.00	1.00	1.00
Shapelets - Robust Scaler	0.87	0.86	0.87
Shapelets - Power Transformer	0.74	0.84	0.72
Shapelets - Quantile Transformer Normal	0.97	0.97	0.97
Shapelets - Quantile Transformer Uniform	0.97	0.97	0.97

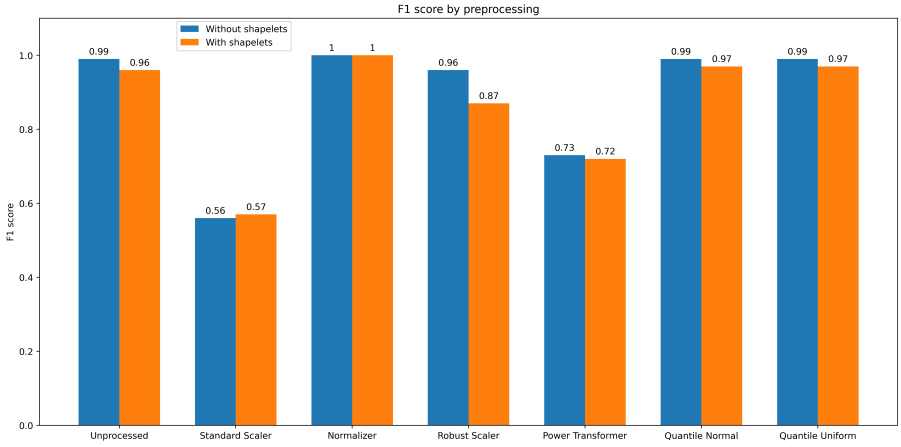


Figure 4.2: Comparison of F1 scores for each preprocessing by using or not shapelets for the Voting classifier predicting INFINITE RADIAL, PSEUDO STEADY STATE, and STEADY STATE scenarios.

Figure 4.3 presents plots of logarithmic pressure derivatives with noise of the scenarios grouped by the classes. Figure 4.3a shows the first group of classes that are composed of scenarios that contain FAULT, FRACTURE, HOMOGENEOUS, and LINEAR FLOW regime. Figure 4.3b presents the second group of classes that consists of scenarios with INFINITE RADIAL, PSEUDO STEADY STATE, and STEADY STATE flow regime.

Table 4.5 presents the results achieved for the classification of the first group of classes. We can verify that for data without shapelets, the quantile transformer normal preprocessing achieves the best result, with precision, recall, and F1 score of 99%. The normalizer preprocessing, as in the previous experiment, obtained a solid result with an F1 score of 98%. We also need to

Table 4.3: Configurations of the best models used to predict the classes FAULT, FRACTURE, HOMOGENEOUS, and LINEAR FLOW, by the use or not of shapelets.

Preprocessing	*Model*	*Hyperparameters*	*Value*
Normalizer	k-NN	Number of neighbors	16
	Decision Tree	Measure quality of a split	*gini*
		Split criterion	*best*
		Minimum number of samples required at a leaf node	2
	Random Forest	Measure quality of a split	*gini*
		Number of tress	10
		Minimum number of samples required at a leaf node	2
Normalizer - Shapelets	k-NN	Number of neighbors	4
	Decision Tree	Measure quality of a split	*entropy*
		Split criterion	*random*
		Minimum number of samples required at a leaf node	8
	Random Forest	Measure quality of a split	*gini*
		Number of tress	10
		Minimum number of samples required at a leaf node	2

mention that the unprocessed and quantile transformer uniform preprocessing reached satisfactory results, with a respective F1 score of 97% and 98%. For data with shapelets, we could see a decrease in quality. The best scores were obtained with the unprocessed and normalizer preprocessing with an F1 score of 91% and 89% respectively. For the normal and uniform quantile transformations preprocessing that obtained solid results without shapelets, we reached an F1 score of 83% and 82%, respectively. In Figure 4.4 we can see a comparison of F1 scores for each preprocessing by using or not shapelets.

In Table 4.6, we show the results for the classification of the second group of classes. The normalizer preprocessing achieves the best scores regardless the use of shapelets, with precision, recall, and F1 scores of 99%. The unprocessed and quantile transformers preprocessing also achieved strong results, with an F1 score equals to or higher than 95%, using shapelets or not. In Figure 4.5 we can see a comparison of F1 scores for each preprocessing by using or not shapelets.

In Table 4.7, we present the configurations of the best models used in the voting classifier to predict the first group of classes, using quantile transformer normal preprocessing without shapelets and unprocessed data with shapelets.

Likewise, in Table 4.8, we show the configurations of the best models

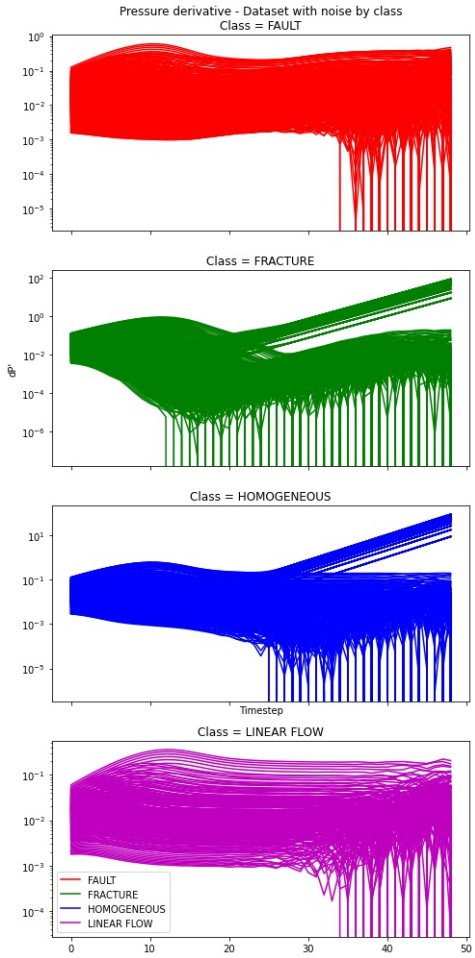
Table 4.4: Configurations of the best models used to predict the classes INFINITE RADIAL, PSEUDO STEADY STATE, and STEADY STATE, by the use or not of shapelets.

Preprocessing	*Model*	*Hyperparameters*	*Value*
Normalizer	k-NN	Number of neighbors	4
	Decision Tree	Measure quality of a split	*gini*
		Split criterion	*best*
		Minimum number of samples required at a leaf node	2
	SVM - Linear	Penalty	l_1
		Solve dual optimization problem	*False*
		Regularization parameter	6
		Maximum number of iterations	10000
		Multi-class strategy	*one-versus-rest*
Normalizer - Shapelets	k-NN	Number of neighbors	4
	Decision Tree	Measure quality of a split	*gini*
		Split criterion	*best*
		Minimum number of samples required at a leaf node	2
	SVM - Linear	Penalty	l_2
		Solve dual optimization problem	*True*
		Regularization parameter	1
		Maximum number of iterations	10000
		Multi-class strategy	*one-versus-rest*

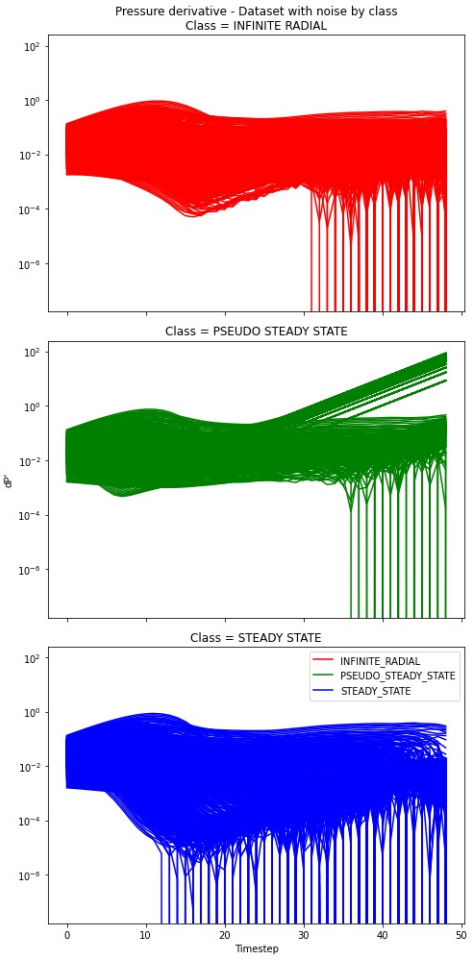
Table 4.5: Performance measures for the Voting classifiers predicting the FAULT, FRACTURE, HOMOGENEOUS, and LINEAR FLOW scenarios using data with noise.

Preprocessing	**Precision**	**Recall**	**F1**
Unprocessed	0.97	0.98	0.97
Standard Scaler	0.91	0.94	0.91
Normalizer	0.98	0.97	0.98
Robust Scaler	0.82	0.88	0.83
Power Transformer	0.97	0.98	0.97
Quantile Transformer Normal	0.99	0.99	0.99
Quantile Transformer Uniform	0.98	0.99	0.98
Shapelets - Unprocessed	0.91	0.92	0.91
Shapelets - Standard Scaler	0.68	0.72	0.68
Shapelets - Normalizer	0.89	0.92	0.89
Shapelets - Robust Scaler	0.79	0.84	0.79
Shapelets - Power Transformer	0.78	0.85	0.78
Shapelets - Quantile Transformer Normal	0.83	0.85	0.83
Shapelets - Quantile Transformer Uniform	0.82	0.84	0.82

employed to predict the second group of classes, using the normalizer preprocessing with and without shapelets.



(a) Logarithmic pressure derivative with noise of the scenarios divided by the classes FAULT, FRACTURE, HOMOGENEOUS, and LINEAR FLOW.



(b) Logarithmic pressure derivative with noise of the scenarios divided by the classes INFINITE RADIAL, PSEUDO STEADY STATE, and STEADY STATE.

Figure 4.3: Logarithmic pressure derivatives with noise of the scenarios grouped by the classes that were predicted in each step by the machine learning models.

Once again, in both cases, the best scores were found using only the three best models for the voting classifier.

We observed that the most recurrent models used in the voting classifiers were random forest and multi-layer perceptron. SVM also was considerably used but with different kernels.

For this second experiment, if we compare the results between data with and without shapelets, we observe a considerable decrease in the scores for the first group of classes compared to the second one. We also noticed that the unprocessed, normalizer, quantile transformer normal and uniform preprocessings achieved solid results. However, the normalizer preprocessing was again the more consistent data preprocessing with and without shapelets.

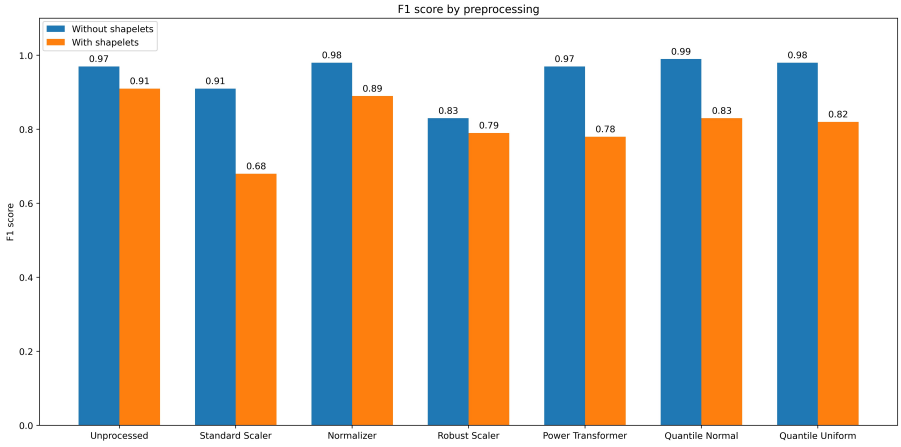


Figure 4.4: Comparison of F1 scores for each preprocessing by using or not shapelets for the Voting classifiers predicting FAULT, FRACTURE, HOMOGENEOUS, and LINEAR FLOW scenarios using data with noise.

Table 4.6: Performance measures for the Voting classifier predicting the INFINITE RADIAL, PSEUDO STEADY STATE, and STEADY STATE scenarios using data with noise.

Preprocessing	**Precision**	**Recall**	**F1**
Unprocessed	0.99	0.99	0.99
Standard Scaler	0.72	0.74	0.70
Normalizer	0.99	0.99	0.99
Robust Scaler	0.92	0.91	0.91
Power Transformer	0.90	0.93	0.90
Quantile Transformer Normal	0.99	0.99	0.99
Quantile Transformer Uniform	0.98	0.99	0.98
Shapelets - Unprocessed	0.96	0.96	0.96
Shapelets - Standard Scaler	0.67	0.72	0.64
Shapelets - Normalizer	0.99	0.99	0.99
Shapelets - Robust Scaler	0.89	0.88	0.89
Shapelets - Power Transformer	0.85	0.90	0.85
Shapelets - Quantile Transformer Normal	0.95	0.95	0.95
Shapelets - Quantile Transformer Uniform	0.97	0.97	0.97

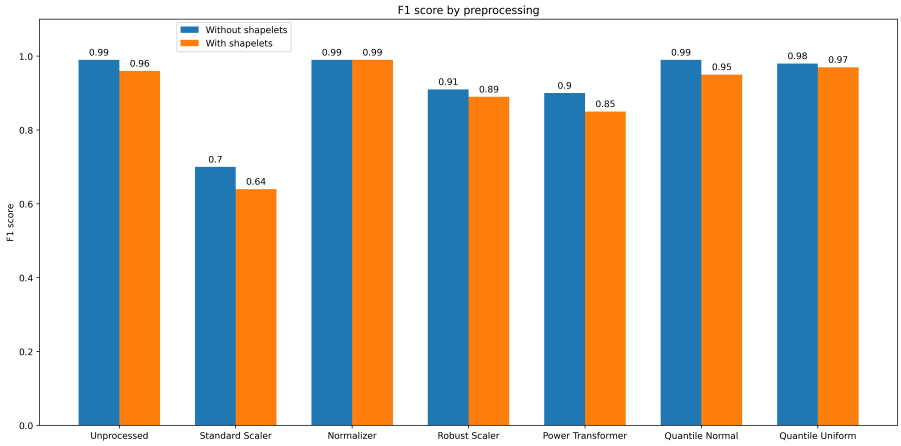


Figure 4.5: Comparison between the F1 scores for each preprocessing by using or not shapelets for the Voting classifier predicting the INFINITE RADIAL, PSEUDO STEADY STATE, and STEADY STATE scenarios using data with noise.

Table 4.7: Configurations of the best models used to predict the classes FAULT, FRACTURE, HOMOGENEOUS, and LINEAR FLOW with noise data, by the use or not of shapelets.

Preprocessing	*Model*	*Hyperparameters*	*Value*
Quantile Transformer Normal	Random Forest	Measure quality of a split	*gini*
		Number of tress	110
		Minimum number of samples required at a leaf node	2
	Multi-layer Perceptron	Activation function for the hidden layer	*tanh*
		Solver for weight optimization	*lbfgs*
		Early stoping	*True*
		Proportion of training data to set aside as validation set for early stopping	0.2
	SVM - RBF	Regularization parameter	6
		Maximum number of iterations	10000
Unprocessed - Shapelets	Decision Tree	Measure quality of a split	*entropy*
		Split criterion	*best*
		Minimum number of samples required at a leaf node	2
	Random Forest	Measure quality of a split	*entropy*
		Number of tress	180
		Minimum number of samples required at a leaf node	2
	Multi-layer Perceptron	Activation function for the hidden layer	*tanh*
		Solver for weight optimization	*lbfgs*
		Early stoping	*True*
		Proportion of training data to set aside as validation set for early stopping	0.2

Table 4.8: Configurations of the best models used to predict the classes INFINITE RADIAL, PSEUDO STEADY STATE, and STEADY STATE with noise data, by the use or not of shapelets.

Preprocessing	*Model*	*Hyperparameters*	*Value*
Normalizer	Multi-layer Perceptron	Activation function for the hidden layer	*tanh*
		Solver for weight optimization	*lbfgs*
		Early stoping	*True*
		Proportion of training data to set aside as validation set for early stopping	0.2
	SVM - RBF	Regularization parameter	4
		Maximum number of iterations	10000
	SVM - Polynomial	Polynomial degree	2
		Regularization parameter	1
		Maximum number of iterations	10000
Normalizer - Shapelet	k-NN	Number of neighbors	4
	SVM - Linear	Penalty	l_1
		Solve dual optimization problem	*True*
		Regularization parameter	6
		Maximum number of iterations	10000
		Multi-class strategy	*crammer singer*
	Random Forest	Measure quality of a split	*entropy*
		Number of tress	80
		Minimum number of samples required. at a leaf node	2

5
Conclusion

The fundamental tool used in the interpretation of a well-test are pressure derivative curves. Although these curves have a signature of the reservoir, it is a challenging task to correctly classify them, since different reservoir models can have similar pressure derivative signatures.

In this work, we proposed a system – using several data preprocessing in conjunction with a time series primitive named shapelets – to extract signatures for each reservoir model and use them to create a new feature space based on the distance between a pressure derivative curve and a shapelet. Then we use this new feature space to train classical machine learning models and use them in a voting classification manner to predict the reservoir model class.

For this purpose, we built an analytical multiclass dataset and divided the classes into two groups to perform the classification. The first group indicates whether the reservoir is homogeneous, fracture, fault, or linear flow. The second one shows whether the reservoir is infinite radial, pseudo-steady-state, or steady-state.

Finally, we compared the classification quality using data with shapelets to data without shapelets.

In the literature of well-test model classification, neural networks are the most used technique to deal with this problem. However, it demands plenty of annotated data to train those neural networks. One of the differences of our work to the literature is the use of only classical machine learning models to perform the classification of the pressure derivative curves. The main reason for our choice is the insufficiency of annotated data when dealing with real-life problems. Based on the results of our experiments, the use of classical machine learning in a voting manner, with or without shapelets, proved to be an effective approach for reservoir classification using pressure derivative plots. Among the models used, we can highlight k-NN, decision tree, random forest, multi-layer perceptron, and SVM, which were the most used models in our voting algorithms, achieving the best scores for each experiment.

Another difference between our work to previous works in literature was that we extensively tested different data preprocessings. Since real-life data is noisier and can be more dispersed, our motivation was finding those prepro-

cessings that provide the best fit in the context of reservoir classification, using pressure derivative plots. We found that normalizer and quartile transformers preprocessings achieved positive results using shapelets or not. Although in one case the unprocessed data obtained better results, it is expected that, when working with real data, those preprocessings would be useful.

Although the use of shapelets provides a gain of quality of machine learning models in many cases in literature, we also have many examples where that does not happen. In cases where there is a decrease in quality, however, it is not a restriction not to use shapelets, since when we use them, we have benefits, such as gain of interpretability and faster training of machine learning models. In our case, we believe that the gain of interpretability – where the specialist can have a visual representation of the shapelets used in the training of the machine learning model and thus can verify if they are correctly capturing the signals of the pressure derivative – brings more benefits than the small decrease in scores obtained when using shapelets in our experiments.

As future work, we would suggest the automatic estimation of reservoir model-related parameters, such as permeability, skin factor, wellbore storage, and flow rate, based only on the pressure derivative plots obtained in a reservoir test.

Bibliography

[Aggarwal, 2015] AGGARWAL, C. C.. **Data Mining**. Springer International Publishing, Cham, 2015.

[Ahmadi et al., 2017a] AHMADI, R.; SHAHRABI, J. ; AMINSHAHIDY, B.. **Automatic well-testing model diagnosis and parameter estimation using artificial neural networks and design of experiments**. Journal of Petroleum Exploration and Production Technology, 7(3):759–783, Sept. 2017.

[Ahmadi et al., 2017b] AHMADI, R.; AMINSHAHIDY, B. ; SHAHRABI, J.. **Well-testing model identification using time-series shapelets**. Journal of Petroleum Science and Engineering, 149:292–305, Jan. 2017.

[Al-Kaabi and Lee, 1993] AL-KAABI, A.; LEE, W. J. ; OTHERS. **Using artificial neural networks to identify the well test interpretation model (includes associated papers 28151 and 28165)**. SPE formation evaluation, 8(03):233–240, 1993.

[Allain and Horne, 1990] ALLAIN, O. F.; HORNE, R. N.. **Use of Artificial Intelligence in Well-Test Interpretation**. Journal of Petroleum Technology, 42(03):342–349, Mar. 1990.

[Athichanagorn and Horne, 1995] ATHICHANAGORN, S.; HORNE, R. N.. **Automatic parameter estimation from well test data using artificial neural network**. In: PROCEEDINGS - SPE ANNUAL TECHNICAL CONFERENCE AND EXHIBITION, volumen Omega, p. 249–262, 1995. Cited By :22.

[Bourdet, 2002] BOURDET, D.. **Well test analysis: the use of advanced interpretation models**. Elsevier, 2002.

[Bourdet et al., 1989] BOURDET, D.; AYOUB, J. ; PIRARD, Y.. **Use of Pressure Derivative in Well Test Interpretation**. SPE Formation Evaluation, 4(02):293–302, June 1989.

[Chu et al., 2019] CHU, H.; LIAO, X.; DONG, P.; CHEN, Z.; ZHAO, X. ; ZOU, J.. **An Automatic Classification Method of Well Testing Plot Based**

on Convolutional Neural Network (CNN). Energies, 12(15):2846, July 2019.

[Deng et al., 2000] ZH, D. Y.. **The Artificial Neural Network Method of Well-Test Interpretation Model Identification and Parameter Estimation**. p. 7, Oct. 2000.

[Hameurlain et al., 2017] Hameurlain, A.; Küng, J.; Wagner, R.; Madria, S. ; Hara, T., editors. **Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXII**, volumen 10420 de **Lecture Notes in Computer Science**. Springer Berlin Heidelberg, Berlin, Heidelberg, 2017.

[Han et al., 2011] HAN, J.; KAMBER, M. ; PEI, J.. [**9780123814807**] **Data Mining: Concepts and Techniques**. 2011.

[Horne, 1995] HORNE, R. N.. **Modern well test analysis**. Petroway Inc, 1995.

[Kotsiantis et al., 2006] KOTSIANTIS, S. B.; KANELLOPOULOS, D. ; PINTELAS, P. E.. **Data preprocessing for supervised leaning**. International Journal of Computer Science, 1(2):111–117, 2006.

[Lee, 1982] LEE, J.. **Well testing: New york: Society of petroleum engineers**. Original edition, 1982.

[Lines et al., 2012] LINES, J.; DAVIS, L. M.; HILLS, J. ; BAGNALL, A.. **A shapelet transform for time series classification**. In: PROCEEDINGS OF THE 18TH ACM SIGKDD INTERNATIONAL CONFERENCE ON Knowledge DISCOVERY AND DATA MINING - KDD '12, p. 289, Beijing, China, 2012. ACM Press.

[Mansbridge et al., 2018] MANSBRIDGE, N.; MITSCH, J.; BOLLARD, N.; ELLIS, K.; MIGUEL-PACHECO, G.; DOTTORINI, T. ; KALER, J.. **Feature Selection and Comparison of Machine Learning Algorithms in Classification of Grazing and Rumination Behaviour in Sheep**. Sensors, 18(10):3532, Oct. 2018.

[Patri et al., 2014] PATRI, O. P.; PANANGADAN, A. V.; CHELMIS, C.; MCKEE, R. G. ; PRASANNA, V.. **Predicting Failures from Oilfield Sensor Data using Time Series Shapelets**. In: SPE Annual Technical Conference AND Exhibition, Amsterdam, The Netherlands, 2014. Society of Petroleum Engineers.

[Patri et al., 2016] PATRI, O. P.; TEHRANI, A. S.; PRASANNA, V. K.; KANNAN, R.; PANANGADAN, A.; REYNA, N. ; OTHERS. **Data mining with shapelets for predicting valve failures in gas compressors**. In: SPE WESTERN REGIONAL MEETING. Society of Petroleum Engineers, 2016.

[Polikar, 2006] POLIKAR, R.. **Ensemble based systems in decision making**. IEEE Circuits and Systems Magazine, 6(3):21–45, 2006.

[Sung et al., 1996] SUNG, W.; YOO, I.; RA, S. ; PARK, H.. **Development of the HT-BP Neural Network System for the Identification of a Well-Test Interpretation Model**. SPE Computer Applications, 8(04):102–105, July 1996.

[Vaferi et al., 2015] VAFERI, B.; ESLAMLOUEYAN, R. ; AYATOLLAHI, S.. **Application of Recurrent Networks to Classification of Oil Reservoir Models in Well-testing Analysis**. Energy Sources, Part A: Recovery, Utilization, and Environmental Effects, 37(2):174–180, Jan. 2015.

[Vaferi et al., 2016] VAFERI, B.; ESLAMLOUEYAN, R. ; GHAFFARIAN, N.. **Hydrocarbon reservoir model detection from pressure transient data using coupled artificial neural network—Wavelet transform approach**. Applied Soft Computing, 47:63–75, Oct. 2016.

[Xing et al., 2011] XING, Z.; PEI, J.; YU, P. S. ; WANG, K.. **Extracting Interpretable Features for Early Classification on Time Series**. In: PROCEEDINGS OF THE 2011 SIAM International Conference ON Data Mining, p. 247–258. Society for Industrial and Applied Mathematics, Apr. 2011.

[Ye and Keogh, 2009] YE, L.; KEOGH, E.. **Time series shapelets: a new primitive for data mining**. In: PROCEEDINGS OF THE 15TH ACM SIGKDD INTERNATIONAL CONFERENCE ON Knowledge DISCOVERY AND DATA MINING - KDD '09, p. 947, Paris, France, 2009. ACM Press.