Politecnico di Milano

SCHOOL OF INDUSTRIAL AND INFORMATION ENGINEERING
Master of Science – Nuclear Engineering

Flow Regime Estimation in Two-phase Flows Employing Machine Learning: Investigation of the Most Promising Dimensionless Feature Set

Supervisor
Dr. Behzad NAJAFI

Co-Supervisor
Prof. Luigi Pietro Maria COLOMBO

Candidate
Alessandro BENETTI – 913114

Academic Year 2020 – 2021

Sommario

Questo lavoro è dedicato alla determinazione dell'algoritmo di machine learning più promettente e del miglior insieme di variabili adimensionali in grado di predirre il tipo di regime bifase all'interno di una generica componente industriale. Pertanto, utilizzando un dataset eterogeneo, una machine learning pipleine che riceve quantità fisiche come inputs e i regimi di flusso come target è sviluppata ed ottimizzata. Pertanto, in un primo step, i dati vengono pre-elaborati, sostituendo valori mancanti o errati, per poi inserire quantità fisiche di interesse. A seguire, vari modelli di machine learning vengono confrontati tra di loro, al fine di individuare quello più promettente per risolvere il problema in questione. Utilizzando il miglior algoritmo identificato nel passaggio precedente, il miglior set di variabili adimensionali è quindi determinato tramite un processo di feature selection iterativo. In seguito, lo step di ottimizzazione dei vari modelli viene ripetuto, questa volta utilizzando solo il sottoinsieme di variabili, ed il miglior algoritmo (in termini di cross-validation accuracy) è individuato. Tale modello utilizza solo cinque variabili adimensionali, ma è in grado di raggiungere una accuratezza del 95,9% ed un F1-score medio del 95,4% sul test set, maggiore di qualsiasi altro modello attualmente disponibile in letteratura. Ai fini di verificare la generalizzabilità del modello, questo è anche utilizzato per riprodurre alcune delle mappe di flusso bifase più note, ed i risultati si dimostrano essere estremamente soddisfacenti. Infine, per garantire una maggiore fruibilità del modello, la pipeline ottimizzata è resa disponibile tramite un software open source.

Parole Chiave: Stima del regime di flusso, Flusso bifase, Machine learning, Selezione dei parametri, Ottimizzazione degli algoritmi.

Abstract

The present work is dedicated to the determination of the most promising machine learning algorithm and the most suitable set of dimensionless features for estimating the the flow pattern in two-phase flows. Accordingly, using a comprehensive and heterogeneous dataset retrieved from the literature, a machine learning based pipeline, which receives dimensionless quantities as inputs and the flow regime as the target is developed and optimized. Employing a benchmark algorithm, a wrapping feature selection algorithm is next applied, and the most promising combination of dimensionless features is determined. Subsequently, the algorithm selection and tuning procedure is performed in order to identity the most suitable machine learning algorithm (and its corresponding hyper-parameters) which leads to the highest achievable cross-validation accuracy. The obtained results demonstrate that the determined optimal pipeline utilizes only five dimensionless features as inputs, which simplified the model's complexity and facilitates the physical interpretation. Furthermore, employing the optimal pipeline a test accuracy of 95.9% and a macro averaged F1-score of 95.4% is achieved, which is greater than the accuracy that can be obtained using any other model that is currently available in literature. The estimations of the proposed pipeline are also compared with those provided by the state-of-the-art multi-phase flow regime maps and the corresponding coherence has been demonstrated. In order to enhance the model's reproducibility and ease-of-use, the optimal pipeline has been made publicly accessible as an open-source software.

Keywords: Flow Regime Estimation, Two-Phase Flow, Machine Learning, Feature Selection, Algorithm Optimization.

Extended Abstract

0.1 Introduction

Several components utilized in the industry such as various types of heat exchangers [1], oil extraction devices [2], and fuel cells [3] include two-phase flows. In these units, gas and liquid flow simultaneously, and the mixture can assume a wide range of different configurations, known as flow regimes or flow patterns. The relative velocity between the phases is heavily influenced by the flow pattern assumed by the mixture, and it is a fundamental quantity to predict during the design phase of these components, as it is related to both frictional and gravitational pressure drop. [4]. However, determining the flow pattern has been proven to be a remarkably challenging task, as it is a function of multiple quantities such as the pipe's internal diameter, the system's tilt, and the superficial velocities of the two phases. Over the years, the scientific community has put extensive efforts to develop a model able to accurately predict the flow regime under a wide range of working conditions [4]. Usually, these models are presented in term of two-dimensional multiphase flow maps, which aim at simplifying the process of determining the flow regime by providing the user with a simple graph where just two quantities (usually related to the phase velocities) are necessary to determine the flow regime. To this day however, no framework is able to work in a generalized environment [5], since the chaotic nature of multiphase flows requires to make strong assumption for physical models, and empirical models are lacking in that they are limited to the experimental evidence they are based on. Moreover, the implementation of these models in a project is often time consuming and lackluster in terms of code portability: the abundance of logical branches and over-reliance on case-specific dimensionless variables hinders the integrability of these codes in an unified environment. However, as experimental studies are periodically performed in this field, the amount of available data continue to increase. This allow us to recast this issue in terms of a classification problem, opening the path for machine learning solution to be employed [5] [6]. When compared to traditional techniques, machine learning has the advantage of being easy to update with new experimental evidence and is able to manage complex interactions between a large number of variables, grasping correlations otherwise not possible to find using traditional statistical techniques. Thus, the present study is focused on optimizing and deploying a machine learning pipeline using data available in open literature. The aim of this pipeline is two fold: first, it generates the best machine learning algorithm for this problem, then, through feature selection, it also identifies what are the best physical quantities able to differentiate the various regimes.

0.2 Flow Regimes

In a two phase system, the geometrical configuration of the interfaces determine the flow regime (flow pattern). While in single phase flows the laminar and turbulent regimes should only need to be distinguished, when gas and liquid flow together inside a pipe, the interaction between the two phases can generate a large number of configurations. As the characterization of multi-phase flow regimes can be at times subjective, there is still some debate towards the actual number of patterns that can exist inside a pipe [7]. In the present work, the initial classification that was adopted in the study conducted by Pereya et al [8] has been followed, in which six different multi-phase flow patterns (provided below) are considered:

- Stratified smooth (SS): the liquid and gas phases are completely separated and the interface between the two phases is flat. Commonly observed in horizontal pipes in which both gas and liquid phases flow at low velocities. This flow regime is completely absent in upward inclined pipes.
- Stratified wavy (SW): similar to the SS, liquid and gas are still completely separated, though, due to the higher liquid or gas velocities, the interface is now wavy. This flow regime is typical of downward tilted pipes and tends to disappear in upward inclinations.
- Dispersed bubble (DB): the gas is moving as dispersed bubbles in the liquid phase. This regime occurs at high liquid velocities, where the turbulence is high enough to dominate over the buoyancy, entrapping gas particles in the liquid flow.
- Bubbly flow (B): the gas is dispersed as discrete bubbles in continuous liquid. This flow regime can only take place in near vertical systems at low liquid velocities, where the turbulence is not enough to break the bubbles and result in transition to DB.
- Intermittent flow (I): gas and liquid alternates one another. This is one of the most common flow patterns and can be present in any type of system configuration, usually for intermediate liquid and gas velocities.
- Annular flow (A): the bulk of the liquid flows on the wall of the pipe as a film, while the gas occupies the center of the duct as a continuous phase.

0.3 Description of the employed dataset

Table 1. **Summary of the studies in the data base.**

Authors	Fluids	ρ_l (Kg/m^3)	ρ_g (Kg/m^3)	μ_l (Pa·s)	σ (N/m)	d(cm)	θ (deg)	Data Points
Shoham [9]	Air-Water	1000	1.18	0.001	0.070	2.54 and 5.1	-90 to +90	5676
Lin [10]	Air-Water	1000	1.12	0.001	0.070	2.54 and 9.54	0	141
Kouba [11]	Air-Kerosene	814	3.00	0.0019	0.029	7.62	0	53
Kokal [12]	Air-Oil	860	4.13	0.007	0.032	2.58 to 7.63	-9 to +9	1668
Van Dresar [13]	Hydrogen-Water	77	0.13	0.00001	0.070	0.874	1.5	83
Wilkens [14]	CO2-Salty Water	1025 to 1059	5.02 to 14.90	0.001	0.070	9.72	-2 to +5	204
Meng [15]	Air-Oil	883 to 889	1.49 to 2.16	0.0047 to 0.0063	0.03	5.01	-2 to +2	153
Manabe [16]	Natural Gas-Oil	789 to 809	8.10 to 26.90	0.0032	0.015	5.49	0 to +90	247
Mata [17]	Air-Oil	879.8	1.3	0.483	0.03	5.08	0	80
Adbuvayt [18] [19]	Nitrogen/Air-Water	1000	5.52 to 23.4	0.001	0.07	5.49 and 10.64	0 to +3	443
Omebere-Iyari [20]	Nitrogen-Naptha	700 and 702	23.4 and 104	0.0003	0.01	18.9	90	98

The experimentally obtained dataset that has been utilized in the present study is the one collected by Pereyra et al. [8], which consists of the data reported in several studies conducted on flow pattern prediction. Each point in the dataset contains information about the system operating conditions (pressure (P), temperature (T), internal diameter (d), inclination angle (θ)), the fluids superficial velocities (U_{sL} and U_{sG}) and the fluid properties (density (ρ), viscosity (μ), superficial tension (σ)). A summary of this database is reported in Table 4.1. Most of the data points are derived from a study carried out by Shoham [9] in 1982, that investigated the behaviour of air-water systems, in 50.8 and 25.4 mm pipes, under all possible inclinations angles. In another study, Lin [10] similarly studied the air-water flow, but utilized 25.4 and 95.4mm pipes while only considering only horizontal configurations. The data from Kouba [11], deals with slug flows in air-kerosene systems. The work of Kokal [12] constitute another major part of the database that is focused on air-oil systems in near horizontal configurations, under different levels of pressure. Next, Van Dresar [13] investigated the behavior of cryogenic fluids under conditions of low mass and heat flux. Wilkens [14], studied the CO2 and salty water flows in a 97.2 mm diameter pipe. Meng [15] investigated low liquid loading in wet gas pipelines. The data from Manabe [16] contains information about the relation between pressure and flow patterns. Mata [17], in 2002, created a flow pattern map for high viscosity oil and water in an horizontal pipe. Abduvyant [18] studied the effects of pressure and pipe diameter on gas-water in near horizontal systems. Lastly, Omebere [20] studied flow patterns in large diameter vertical pipes at high pressures.

In fig 1 we reported the distribution of the most important design conditions according phenomenological models. It is evident that the database provide heterogeneous working configurations, but most of the data points refers to the intermittent (I) flow regime. Since an unbalanced dataset may lead to biased estimators, oversampling techniques will be applied during the training of the models, to give the same importance to all flow regimes. It should be underlined that these oversampling techniques are used exclusively on the training set, and both validation and testing

frames will not have any type of synthetic data in them, avoiding to generate optimistic expectations [21].

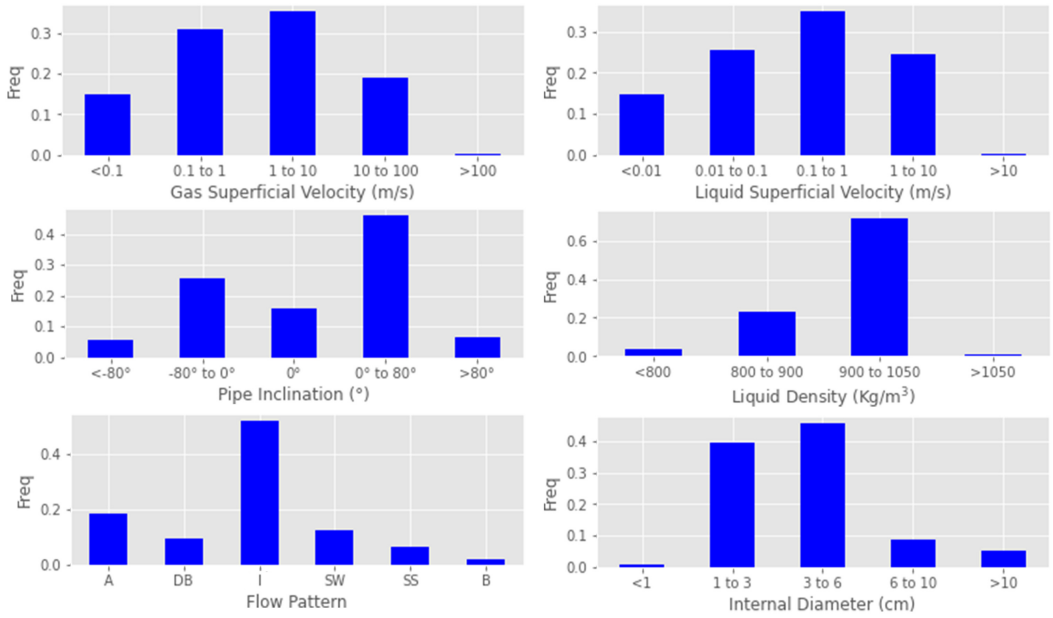

Figure 1. Distribution of the most important variables in the database

0.4 Physical Models

To improve the generalizability of the developed models, the knowledge of currently employed physical models is fundamental. In fact, as demonstrated by Quintino [22] and MustafaAl-Naser [5], utilizing dimensionless quantities as features can help the machine learning algorithms to leverage the known physics and achieve a superior generalization capability. Thus, the basic physical features of the dataset are transformed into dimensionless quantities, that have been widely employed in the physical phenomena-based models such as the ones present in Barnea's unified model [23]. A complete list of the features that are employed in the present work is provided in Table 2.

Dataset's Features	P	T	ρ_L	ρ_G	μ_L	μ_G	σ	d	θ	U_{sL}	U_{sG}
Transformed Features	Re_G	Re_L	Fr_G	Fr_L	X_{LM}	Y	K_G	N_L	We	Eo	T_{TB}
	θ	α_L	X_{LM}^2								

Table 2. Features employed in this work

0.5 Machine Learning Models

In this section we spend a few words on the more technical aspects focusing on the necessary machine learning theory behind our models. In fact, the knowledge of their inner workings can provide useful insights on the validity of the generated outputs. Since the process of optimizing and tuning a model is computationally expensive [24], only those models whose performance can be deemed state-of-the-art were explored in this thesis. First, multiple types of ensemble techniques were considered, such as gradient boosting machines [25] and random forests [26]. In these algorithms, the general aim is to train and aggregate the outputs of a series of models in order to generate a prediction. Then, deep learning solutions were considered, in particular a Multi-Layer-Perceptron [27] and Tab-Net [28]. The first algorithm is the most common type of neural network, made of only perceptrons connected in a feed-forward architecture. Tab-Net [28] on the other hand is an attention based network, and it employs much more complex building blocks such as transformers. In Table 3 all the explored algorithms, as well as their Python packages, are reported.

Explored Algorithms	**Python Packages**
Random Forest [26]	Sklearn [29]
Gradient Boosting [30]	XGBoost [31] LightGBM [32]
Stacking [33]	Custom
MLP [27]	Tensorflow [34]-Keras [35]
Tab-Net [28]	Pytorch [36]

Table 3. Models explored and their relative Python packages

0.6 Machine Learning Based Pipeline

Here we discuss in detail the various methodologies adopted to develop and optimize the machine learning based pipeline. In the first step, considering the governing physical phenomena and the recommendations that have been provided in the previously conducted physical phenomena-based studies, a series of additional features are generated and added to the dataset. Next, the dataset is divided into the training (80% of samples) and testing (20% of samples) sets. The training set, employing a 5 fold cross-validation method, is used to optimize the pipeline. The test set is instead utilized to evaluate the performance of the proposed pipeline in estimating the flow regime on a set of points for which it has not been trained and optimized. In order to preserve the distribution of the classes, the generation of each fold (validation and testing) is performed using a stratified approach. The model selection procedure is then conducted and the most suitable algorithm, while utilizing all of the features, is identified. In order to give the same importance to all the classes, in the cross-validation procedure, SMOTE [37] is applied to each fold of the training data. Next, feature selection is applied to the dataset and employing the Sequential Forward

Floating Selection (SFFS) approach, features are progressively added until a plateau in the cross-validation accuracy is achieved. Finally, the selected subset is utilized to repeat the algorithm optimization step and the most promising algorithm resulting in the highest accuracy is identified.

0.7 Results and discussions

This section is dedicated to discussing the results found in the various steps of the implementation and optimization of the pipeline. In the first section, the performance of the generated models, in terms of accuracy and F1 score, is presented. Next, the results of the feature selection step are provided and the parameters that have been selected in the optimal pipeline are explored employing box-plots. Finally, to further assess the accuracy of the proposed pipeline, the corresponding estimations and the resulting transition boundaries are compared with experimental multi-phase flow maps that are available in literature.

Machine Learning Algorithm	All Features		Feature Subset	
	Accuracy [%]	F1 score [%]	Accuracy [%]	F1 score [%]
LightGBM	95.3	94.8	95.2	94.4
Random Forests	91.5	90.0	90.9	88.8
XGBoost	94.8	93.1	94.9	94.1
Stacking	95.3	94.8	95.2	94.4
MLP	93.4	91.6	89.9	90.6
Tab-Net	93.8	92.8	90.7	88.5

Table 4. Comparison between the various optimized machine learning models

Table 4 reports the achieved performance metrics (determined through cross-validation) metrics associated with all of the explored algorithms are reported. Gradient boosting machines are clearly proven to be the most promising category of models to solve this classification problem, as both XGBoost and LightGBM outperform every other algorithm in any given condition. The performance of random forests and neural networks algorithms is demonstrated to be similar but lower than the other algorithms. Lastly, stacking can achieve the same performance of LightGBM in every scenario, as it uses its outputs to perform predictions. However, given that this is a second level learner which inevitably tends to over-fit the training data, it was still deemed inferior than a simple gradient boosting machine.

In the first model optimization step, the best estimator identified by the pipeline is a gradient boosting machine from the LightGBM library. Said model has a cross validation accuracy of 95.3% and a macro averaged F1 score of 94.8%. The latter value confirms that the bias towards the intermittent flow regime is avoided, due to the oversampling techniques adopted during the training phase. When used to

evaluate the test set, the performance of the model is even slightly better, as the test accuracy is 95.9% and the macro average F1 score is 95.6%. The small increment in the test's metrics is justified by the fact that cross validation tends to provide pessimistic expectations. Additionally, as test and validation scores match, it is evident that no overfitting occurred during the training phase of the model.

Table 5. Cross validation confusion matrix, five features

	A	DB	I	SW	SS	B	Precision [%]
A	1168	0	48	30	5	0	93.4
DB	0	606	20	0	0	0	96.8
I	50	39	3199	20	10	5	96.3
SW	28	2	17	802	11	0	93.3
SS	5	0	10	8	281	0	92.4
B	0	0	4	0	0	118	96.7

Table 6. Test confusion matrix, five features

	A	DB	I	SW	SS	B	Precision [%]
A	291	0	14	6	2	0	93.0
DB	0	151	5	0	0	0	96.8
I	12	9	808	1	1	0	97.2
SW	8	0	3	201	2	0	93.9
SS	0	0	1	0	76	0	98.7
B	0	0	3	0	0	28	90.2

Using the selected feature subset identified through SFFS, the model optimization step is repeated, and the best generated estimator is still a gradient boosting machine from the LightGBM library, but with slightly different hyperparameters. Said model compares favorably with respect to the previous one, as the cross validation accuracy is 95.2%, and the F1 score is 94.4% (Table 5). Little to no information is lost, while the number of employed features is reduced from 13 to 5. Again, the results on the test set are similar, with an accuracy of 95.9% and an F1 score of 95.4% (Table 6).

All Features	θ	Re_G	Re_L	Fr_G	Fr_L	X^2_{LM}	Y	K_G	N_L	We	Eo	T_{TB}	α_L
SFFS Results	θ	Fr_L	Fr_G	Eo	X^2_{LM}								

Table 7. Feature selection results

The results of sequential floating forward selection are reported in figure 2 and Table 7. It is evident that a plateau in the cross validation metrics is achieved as soon as five features are considered to train our estimators. As reported in Table 7, the selected subset contains both liquid and gas Froude velocities, as well as the system tilt. These quantities are the most widely employed by physical models, and wrapping feature selection is able grasp the importance of these features in determining the

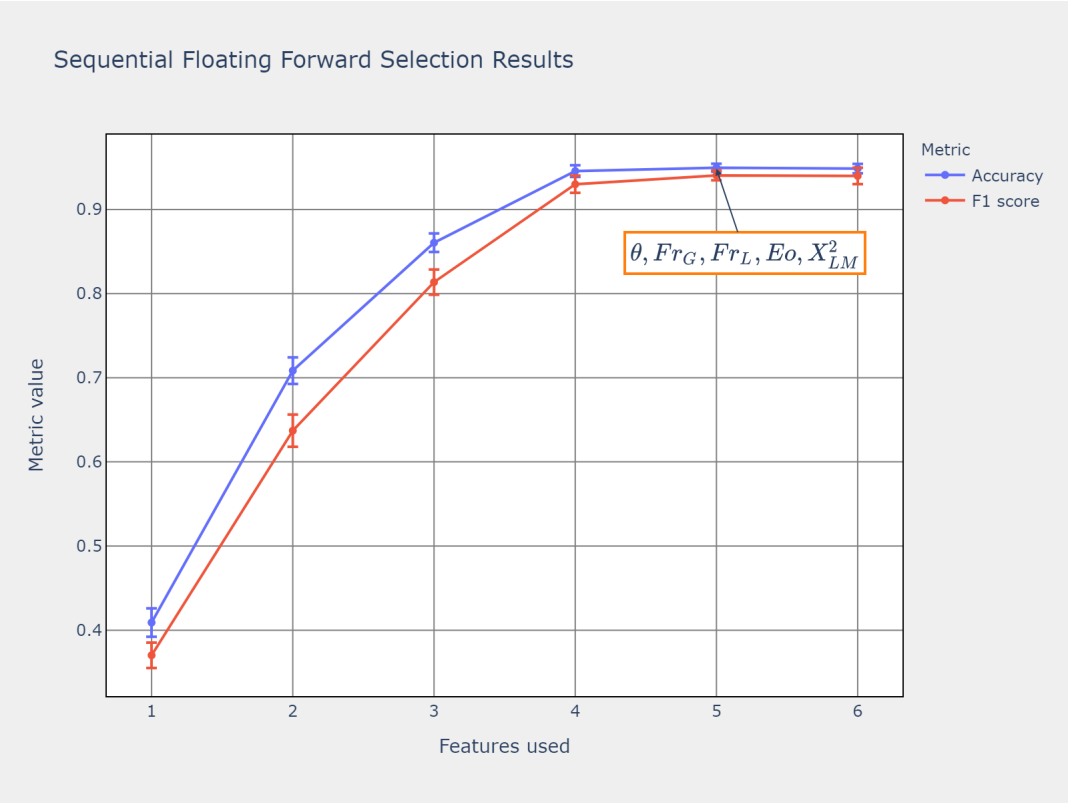

Figure 2. Sequential feature selection results

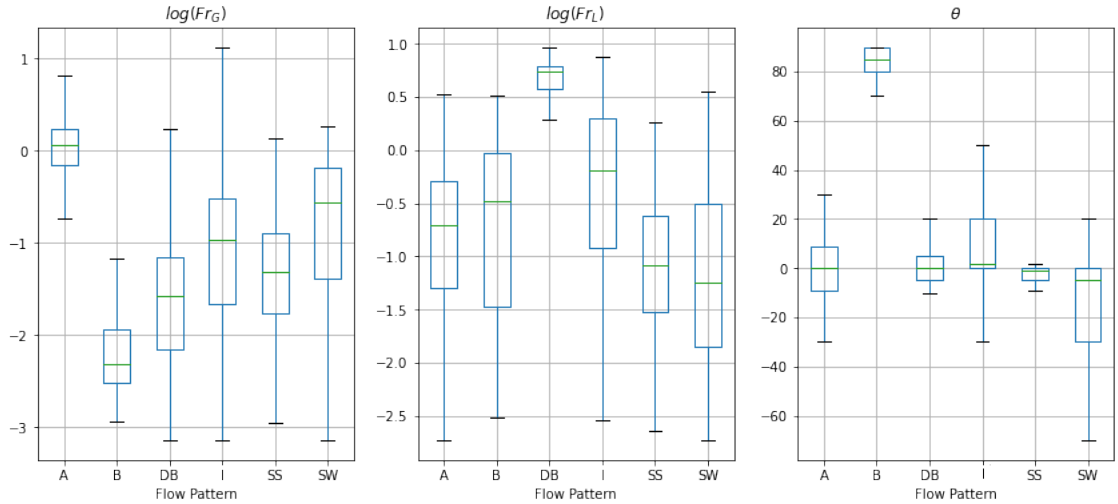

Figure 3. Boxplots of the selected feature subset

correct flow regime. The influence of these features can easily be shown in terms of boxplots (figure 3), where it is evident that they are able to separate the flow regimes by some degree.

The influence on the flow pattern for the remaining features is harder to explain, as they cannot be used alone to separate the flow regimes. From a physical point of view, the Eo number represent the ratio between gravitational and capillary forces, and could have been used by our model to introduce information about the system surface tension. The Lockhart-Martinelli parameter X^2_{LM} on the other hand could have been used to further assess stratified to non stratified transitions based on the ratio of the pressure drop between the two phases.

While the proposed pipeline shows remarkable accuracy when used to on experimental conditions similar to the ones of training, real world applications require a more robust model evaluation. Thus we assessed the generality capabilities of the proposed pipeline by comparing its outputs with some multiphase flow maps retrieved from literature, with design conditions different from the ones of training. As the name suggest, these maps are two-dimensional representation of the problem at hand, where one can use a pair of physical quantities (such as the velocities of the two phases) to calculate the flow regime. Of course, these maps are only valid under very strict design conditions, and tends to fail when generalized. To this end, the best machine learning model was used to reproduce the map of Madhane [38], and the results can be seen in figure 4. The transition boundaries closely match the ones of the author, and the model only shows some discrepancies, where the classification of the regime is debatable.

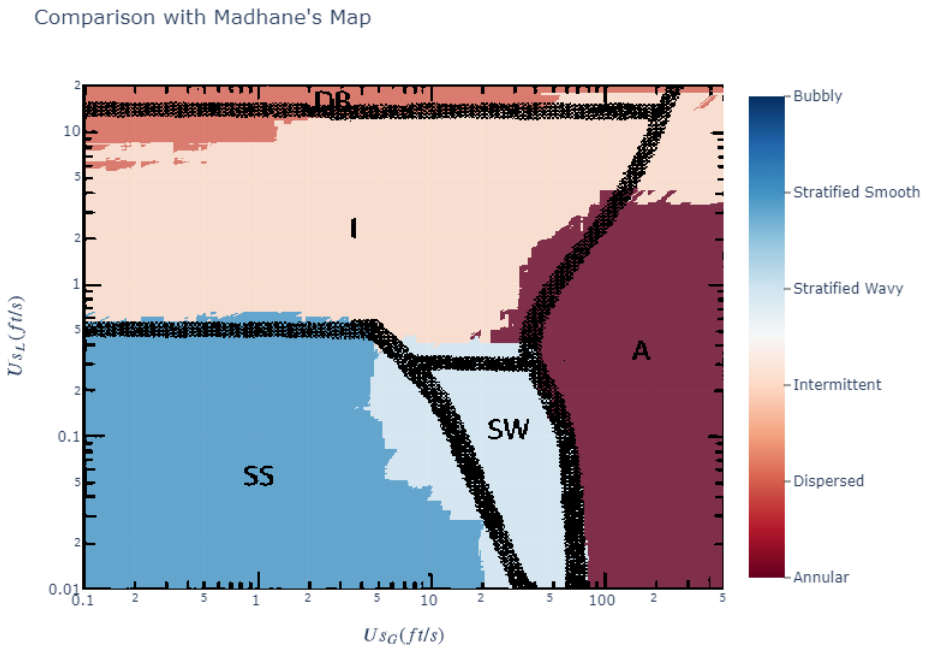

Figure 4. Comparison with the map of Madhande (in black) and the one generated with the proposed pipeline (colored regions)

0.8 Conclusion

The main objective of this work, was to develop and optimize a machine learning based pipeline in order to predict the multiphase flow regime under a wide range of working conditions. Modeling the flow regimes through the physical models was fundamental in order to increase the explainability of the generated machine learning model. While using all the features, the best generated algorithm obtained through the implemented procedure is a gradient boosting machine from the LightGBM library, with a cross validation accuracy of 95.3% and a macro averaged F1 score of 94.8%. The latter metric denotes that the model has no bias towards any of the regimes. When used to evaluate the test set, the metrics are similar to the cross validation ones, with a test accuracy of 95.9% and a macro averaged F1 score of 95.5%. Given these values, it is evident that overfitting was avoided. As the second core objective of the pipeline was to determine the most important physical quantities, sequential floating forward selection was applied to the dataset. This iterative feature selection technique showed that, by greedily adding variables to an empty set, the near-optimal feature combination is: system tilt, liquid densimetric Froude number, gas densimetric Froude number, Eötvös number and the Lockhart-Martinelli parameter. With this subset, the new optimized estimator is again a gradient boosting machine from LightGBM library, but with slightly different hyperparameters. This model has a cross validation accuracy of 95.1%, and a macro averaged F1 score of 94.2%. Given the similarity on the metrics between the first and the second model, we can state that little to no information was lost using the optimized feature subset. When used to evaluate the test set, it showed no signs of overfitting, as the accuracy was 95.9% and the F1 score 95.4%. To further assess the quality of the generated outputs, this model was also used to simulate and compare different multiphase flow maps found in literature. The transition region identified by the model closely matched the one of theory. Given the overall performance of the pipeline on both feature selection and model optimization, this work can be considered a success. It should be underlined however that while the dataset employed for this thesis offers a wide range of working conditions, all the data points refer to smooth macro tubes. As a machine learning model is only as good as the data it has been trained with, it won't be able to consider complex physical phenomena related to surface roughness or micro channels. However, this deficiency easily be solved with the introduction of more data related to those phenomena.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Several components utilized in the industry such as various types of heat exchangers [1], oil extraction devices [2], and fuel cells [3] include two-phase flows. In these units, gas and liquid flow simultaneously, and the mixture can assume a wide range of different configurations, known as flow regimes or flow patterns. The relative velocity between the phases is heavily influenced by the flow pattern of the mixture, being highest when the gas phase is separated from the liquid one and lowest when the gas is dispersed inside the turbulent liquid phase. As this quantity is related to the liquid holdup, the accurate simulation and estimation of this phenomenon has thus been of notable importance. In fact, the liquid holdup is used for the calculation of both frictional and gravitational pressure drop [4], which are essential parameters in the design procedure of industrial units. However, determining the flow pattern has been proven to be a remarkably challenging task, as it is a function of multiple quantities such as the pipe's internal diameter, the system's tilt, and the superficial velocities of the two phases. Over the years, the scientific community has put extensive efforts to develop physical-phenomena based models that are able to accurately predict the flow regime under a wide range of working conditions [4]. Taitel and Duckler [39] in 1976 provided a methodology to to determine the flow pattern in horizontal systems. Their work showed great accordance with the experimental evidence from Madhane [38]. Expanding the latter model, Barnea [23] proposed an unified model, able to work in a more generalized framework, for the whole range of pipe inclinations. These models are often served through multiphase flow maps, two-dimensional representation of the phenomena at hand, where one can use a pair of suitable physical quantities to determine the flow regime. While these models provide useful insights for the problem at hand, they tends to fail in a generalized environment. [5]. In fact, the chaotic nature of multi-phase flows requires to make strong assumptions for physical models and the obtained empirical models are limited to the experimental evidence that they are based on. Moreover, the implementation of these models in a project is often time consuming and lackluster in terms of code portability: the abundance of logical branches and over-reliance on case-specific dimensionless variables hinders the integrability of these codes in an unified environment. However, as experimental studies are periodically performed in this field, the amount of available data continue to increase. This allow us to recast this issue in terms of a classification problem, opening the path for machine learning solution to be employed [5] [6]. When compared to traditional techniques, machine learning has the advantage of being easy to update

with new experimental evidence and is able to manage complex interactions between a large number of variables, thus grasping correlations otherwise not possible to find with traditional statistical techniques. While machine learning has been heavily employed in other areas of thermo-hydraulics such as pressure drop [40] and heat transfer estimation, its use in this specific domain remain relatively unexplored. One of the first studies is from Xie et al [41], which employed an artificial neural network (ANN) using pressure signals as inputs to predict the flow regime in a three phase system. Then, MustafaAl-Naser et al [5] also used an ANN trained from synthetic data from the unified model, but focused on horizontal two-phase systems. Recently, Mask et al [6] tried different machine learning models to solve the task at hand, but mentioned the necessity of proper model validation. Lastly, Quintino et al [22] proposed an hybrid-physics-data machine learning approach, using a set of manually selected dimensionless features to enhance the generality capabilities of the trained models. The aforementioned works however either focus on a specific type of machine learning (ML) model or handpick the features used to train the models. Additionally, none of them provide an open-source framework that can be updated as new data is published. Motivated by the mentioned research gap, the present study is focused on optimizing and deploying a machine learning pipeline using data available in open literature. The aim of this pipeline is two fold: first, it generates the best machine learning algorithm for this problem, then, through feature selection, it also identifies what are the best physical quantities able to differentiate the various regimes.

1.1 Objective and Outline of the Thesis

As stated before, machine learning is becoming more and more prominent in the field of thermo-hydraulics, thanks to its advanced capabilities when compared to traditional techniques. For the problem at hand however, currently employed machine learning solutions fail to provide an open framework that can be updated as new data becomes available, and they perform little to no validation for the machine learning model produced. While performing exploratory data analysis, the latter shortcoming was found to be particularly serious for studies using the same dataset as the one employed for this thesis. In fact, the data contains some transcription errors that would hinder the quality of the outputs in real world applications if not addressed correctly. Thus, the scope of this thesis is to develop and optimize a machine learning pipeline using data available in open literature in order to predict the multiphase flow regime under a wide range of working conditions. To this end, the first objective of the pipeline is to determine the most promising machine learning model to solve the problem at hand. Then, in order to enhance the model's interpretability, characteristic dimensionless physical quantities are introduced as variables, and through an iterative feature selection algorithm, the best ones to differentiate the flow regimes are identified. Following, the pipeline is evaluated in terms of adimensional metrics to summarize the overall performance, and proper model validation is also carried out. Thus, using famous multiphase flow maps from literature, and an external dataset, the generality of the model is assessed. Once the results of the generated estimators are deemed acceptable, the best machine learning model is also containerized as a web application, providing an open-source tool to the scientific community. Lastly, the whole code

is also made available in an online repository, to allow the reproducibility of the results.

Thus, Chapter 2 is first dedicated to discuss some fundamental concepts of multiphase-flow theory. In particular, the first section is used for the introduction of the most used quantities in this field and the description of the pressure drop models used for multiphase flow, focusing on how the most accurate one is related to the flow regime of the system. Then, the physical reasoning behind the various regime transitions is examined, and the dimensionless quantities used to model such changes are introduced.

Chapter 3 instead pertains the description of the various machine learning models analyzed for this thesis. First the reasoning behind ensemble methods is discussed, and the most used algorithms from this machine learning (ML) branch are introduced (random forests (RF), gradient boosting (GB), and stacking). Then, deep learning solutions are analyzed. The first model discussed is a multi-layer-perceptron (MLP) along with the fundamental elements present in a neural network (NN). Lasly, an attention based network is introduced and confronted with the previous architecture.

Chapter 4 is dedicated to the description of the implemented procedure to train and optimize a machine learning based pipeline. First, the methodology for the whole optimization process is discussed. This part encapsulates the whole reasoning behind each action taken in this project. Then each single step of the implemented procedure is discussed in detail, from data analysis to the deployment of the best model.

Finally, Chapter 5 analyzes the best performing models obtained during the optimization of the pipeline. They are first benchmarked in terms of accuracy and F1 score on the validation set. In order to prove that overfitting was avoided, the most promising estimator is also confronted with a test set, taken from the same dataset used for training. The generality capabilities of the model are then discussed by comparing its output with Madhane's multiphase flow map [38]. Then, using a different external database taken from literature, the best performing model is benchmarked again in terms of F1 score and Accuracy against data with previously unseen system's design configurations.

Chapter 2

Multiphase Flows Theory

In order to understand the importance of determining the flow regime in a pipe duct, one has first to introduce some fundamental aspects of multiphase flows. The first part of this chapter is then dedicated to discuss some physical quantities of interest for multiphase system, as well as to briefly describe the most used models to calculate the pressure drop. The second part of this chapter is instead used to introduce the physical recommendation currently employed to distinguish the various flow regimes. While the aim of this thesis is to approach this classification problem using machine learning, introducing the known physics in terms of dimensionless quantities through feature engineering can greatly enhance the capabilities of our machine learning models to find the correct criteria to separate the flow patterns.

Basic quantity	Symbol	Description	Unit
Pressure	P	Pressure inside the pipe	[MPa]
Temperature	T	Temperature of the system	[°C]
Diameter	D	Pipe's internal diameter	[cm]
Viscosity	μ	Viscosity of a given phase	[Pa · s]
Density	ρ	Density of a given phase	[Kg/m^3]
Surface tension	σ	Surface tension between the phases	[N/m]
System's tilt	θ	System's tilt (zero for horizontal configurations)	[deg]
Superficial Velocity	U_s	Superficial velocity of a given phase	[m/s]

Table 2.1. Basic physical quantities describing the system

2.1 Fundamental Relations in Multiphase Flows

The term "multiphase systems" describe a situation where two or more phases (liquid, gas, solid) flow together, typically inside a pipe duct. In order to describe these systems, the knowledge of thermodynamics and hydraulics is fundamental, and the

physical quantities usually employed in these fields find their way in the description of multiphase systems. The basic quantities usually characterizing the systems are summarized in Table 2.1. Only the quantities exclusive to multiphase systems will be discussed in this section. Additionally, since this thesis only deals with two-phase systems, the definitions will refer to these specific type of systems.

The first and most important quantity to discuss is the average void fraction, defined as the ratio between the volume of the diapered (gas) phase V_G within the volume V:

$$\alpha = \frac{V_G}{V} \qquad \text{with V} > V_0 \tag{2.1}$$

The condition $V > V_0$ ensures that stationarity is achieved despite of the motion of elements of the dispersed phase such as particles, drops and bubbles.

Similarly to the void fraction, the liquid holdup is defined as the ratio between the continuous (liquid) phase V_L and the total volume V:

$$\alpha_c = \frac{V_L}{V} \qquad \text{with V} > V_0 \tag{2.2}$$

It is clear that due to the volume conservation principle, the sum of the liquid hold and the void fraction has to be equal to 1:

$$\alpha_c + \alpha = 1 \tag{2.3}$$

The void fraction (or liquid holdup) can be used to calculate important quantities used in multiphase, such as the actual velocities of the phases flowing in the pipe.

$$u_G = \frac{\Gamma_G}{\alpha A \rho_G} \qquad \text{and} \qquad u_L = \frac{\Gamma_L}{(1-\alpha) A \rho_L} \tag{2.4}$$

Where Γ_G and Γ_L are the two phases mass flows, A is the cross section, ρ_G and ρ_L the two phases densities.

The correct evaluation of the void fraction is quite challenging, and it is often preferred to work with apparent quantities, which are defined as the physical properties of the single phase as if it was flowing alone inside the pipe. When dealing with the phase velocities, they are defined as the superficial velocities:

$$U_{sG} = \frac{\Gamma_G}{A \rho_G} \qquad \text{and} \qquad U_{sL} = \frac{\Gamma_L}{A \rho_L} \tag{2.5}$$

Notice that when compared to the actual velocity, the superficial is actually lower, as it is divided by the whole area A. The great advantage of using this quantities is that they are easy to determine, as they are function of known properties of the systems.

Another important quantity to introduce is the vapor or mass quality x, defined as the ratio between the mass flux of the gas and the total mass flux:

$$x = \frac{\Gamma_g}{\Gamma_g + \Gamma_l} \tag{2.6}$$

This is not the only "quality" used in multiphase, and during the development of a project, the thermodynamic quality (2.7) and the volume quality (2.8) are also usually calculated:

$$x_{th} = \frac{h_b - h_l}{h_g - h_l} \qquad (2.7) \qquad\qquad x_v = \frac{Q_g}{Q_g + Q_l} \qquad (2.8)$$

In eq 2.7, h_b is the bulk enthalpy, while h_g and h_l are gas and liquid entalphies respectively. The thermodynamic quality is the only quality that can assume values that are lower than zero (sub-cooled liquid) and greater than one (over-heated gas). While in thermodynamic equilibrium, it is equal to the vapor quality. In 2.8 instead, Q_g and Q_l are the gas and liquid volumetric flow respectively. When comparing the definition of mass (2.6) and volume (2.8) qualities, it is clear that they are bounded by the following relation:

$$\frac{1 - x_v}{x_v} = \frac{\rho_g}{\rho_l}\frac{1 - x}{x} \qquad (2.9)$$

Where ρ_g and ρ_l are the densities of the two phases.

Except special cases discussed below, the gas and liquid velocities differ from one another. It is then convenient introducing the slip ratio, defined as the ratio between the actual velocities:

$$S = \frac{u_g}{u_l} \qquad (2.10)$$

This quantity is depended on the configuration assumed by the mixture and can be used to relate the void fraction to the vapor quality:

$$\frac{1 - \alpha}{\alpha} = S\frac{1 - x_v}{x_v} \qquad (2.11)$$

2.2 Two-Phase Pressure Drop Models

A fundamental aspect of multiphase flows is how to calculate the pressure gradient. As the slip ratio between the two phases changes, so does the liquid holdup and the liquid volume fraction (2.11). This quantity is strictly related to the flow regimes, being highest when in stratified flow conditions and lowest in bubbly condition. Since the determination of the flow pattern is challenging, different models have been developed to relive the engineer from the burden of determining it. Needless to say, these models are a simplification of a real case, and their predictions can even differ in one order of magnitude from the actual pressure drop of the system.

2.2.1 Homogeneous Flow Model

The concept of homogeneous flow was developed by Soviet scientist in the 1960s. In the homogeneous flow model, the multiphase fluid is assumed to be an uniform mixture

($S = 1$) with averaged properties of the two fluids. This means that traditional single phase theory can be applied in order to calculate the frictional pressure drop:

$$\frac{-dp_f}{dz} = 2\frac{f_{tp}G^2 v_b}{D} \tag{2.12}$$

The fanning friction factor of the mixture (f_{tp}) can be derived from the Reynolds number, once a suitable correlation for the mixture viscosity is chosen:

$$\text{Owens}: \quad \mu_{tp} = \mu_l \tag{2.13}$$

$$\text{Cicchitti}: \quad \mu_{tp} = x\mu_g + (1-x)\mu_g \tag{2.14}$$

$$\text{MC Adams}: \quad \frac{1}{\mu_{tp}} = \frac{x}{\mu_g} + \frac{1-x}{\mu_l} \tag{2.15}$$

Since the slip ratio is equal to 1, the void fraction is equal to the volume quality ($x_v = \alpha$, see eq 2.11). Both the accelerative term and the gravitational term can be calculated from known design quantities:

$$\frac{-dp_g}{dz} = \rho_{tp} g \sin\theta = [\alpha\rho_g + (1-\alpha)\rho_l] g \sin\theta \tag{2.16}$$

$$\frac{-dp_a}{dz} = \frac{d}{dx}\left[\rho_g \frac{U_g^2}{\alpha} + \rho_l \frac{U_l^2}{1-\alpha}\right] \tag{2.17}$$

While the homogeneous model simplifies the problem by trivializing the determination of the void fraction, it can only be applied when the properties of the two phases are similar to one another, giving satisfying results when $\rho_l/\rho_g < 10$ or G_g+G_l>2000 kg/(m^{2}s).

2.2.2 Separated Flow Model

Lockhart and Martinelli proposed a model where the two phases display different properties and flow at different velocities, without interacting with one another. Here the frictional pressure drop is determined by terms of the two phase multiplier Φ_L^2 or Φ_G^2:

$$-\left(\frac{\mathrm{d}p_\mathrm{F}}{\mathrm{d}x}\right) = \Phi_L^2 \left(-\frac{\mathrm{d}p_\mathrm{F}}{\mathrm{d}x}\right)_{SL} = \Phi_G^2 \left(-\frac{\mathrm{d}p_\mathrm{F}}{\mathrm{d}x}\right)_{SG} \tag{2.18}$$

Where $(\mathrm{d}p_\mathrm{F}/\mathrm{d}x)_{SL}$ and $(\mathrm{d}p_\mathrm{F}/\mathrm{d}x)_{SG}$ are the frictional pressure gradients of the single phases, as if they were flowing alone inside the pipe. The estimation of the two phase multipliers is done through the Lockhart-Martinelli parameter X_{LM}^2 defined as the ratio between the pressure gradients of the single phases:

$$X_{LM}^2 = \frac{(\mathrm{d}p_\mathrm{F}/\mathrm{d}x)_{SL}}{(\mathrm{d}p_\mathrm{F}/\mathrm{d}x)_{SG}} \tag{2.19}$$

$$\Phi_L^2 = 1 + \frac{C}{X_{LM}} + \frac{1}{X_{LM}^2} \tag{2.20}$$

$$\Phi_G^2 = 1 + CX_{LM} + X_{LM}^2 \tag{2.21}$$

C is a tabulated quantity, and depends on the single phase flow regime (laminar or turbulent). In order to calculate the accelerative and gravitational pressure gradients, the slip ratio is also estimated by terms of X_{LM}^2:

$$S = k(X_{LM}^2)^d \tag{2.22}$$

Where k and d are empirical coefficients that vary based on the correlation considered.

2.2.3 Drift Flux Model

The models discussed above are a great baseline to estimate the range of the pressure drop inside a system. In order to further increase the accuracy of our predictions however, flow pattern dependent models are required. The drift flux model was introduced by Zuber and Findlay in 1965, and consider the relative motion between the phases by terms of a kinematic constitutive relation. Using the drift flux model, the actual gas velocity u_g can be expressed as the contribution of a drift term u_{gj} and the total superficial velocity U:

$$u_G = U_{Gj} + C_0 U \tag{2.23}$$

C_0 is called the distribution parameter, and takes into account for the non-uniformity of the volumetric flux and the void fraction profiles across the duct. U_{Gj} instead is the drift velocity of the gas and represent the average effect of the local relative velocity between the phases. Of the model discussed until now, the drift flux model is the most physically sound, as it takes in consideration the interaction between the phases. Given C_0 and u_{Gj}, which are closely related to the flow pattern, the void fraction can be obtained by combining eq 2.23, 2.5 and 2.4:

$$\alpha = \frac{U_G}{u_G} = \frac{U_g}{U_{Gj} + C_0 U} \tag{2.24}$$

Once α is known, one can proceed to calculate the pressure drop of the system.

2.3 Multiphase Flow Regimes

The following section will be dedicated to discuss the physical models developed over the years to determine the transition boundaries between the various flow regimes. Aside from being extremely important to understand the problem, introducing physical quantities of interest in our machine learning models through feature engineering could help for the generalization of the model. For reference, a summary of the features employed by the model is reported in Table 2.2

Characteristic quantity	Symbol and equation	Description	Unit	Eq. No.
System's tilt	θ	System's tilt (zero in horizontal configurations)	[deg]	(-)
Reynolds number	$Re = \frac{\rho D U_S}{\mu}$	Ratio of inertial forces to viscous forces	[-]	(2.30)
Densimetric Froude number	$Fr = U_s\left(\frac{\rho}{gD\Delta\rho}\right)^{0.5}$	Ratio between inertia forces and gravity	[-]	(2.27)
Weber number	$We = \frac{U_{SL}^2 \rho_L D}{\sigma}$	Ratio between drag and cohesion forces	[-]	(2.33)
Eotvos number	$Eo = \frac{\Delta\rho g D^2}{\sigma}$	Ratio between gravitation and capillary forces	[-]	(2.34)
Lockhart–Martinelli parameter	$X_{LM}^2 = \frac{(\mathrm{d}p_\mathrm{F}/\mathrm{d}x)_{SL}}{(\mathrm{d}p_\mathrm{F}/\mathrm{d}x)_{SG}}$	Ratio between the pressure drop of the two phases as if they were flowing alone	[-]	(2.25)
Chisholm parameter	$Y = \frac{\Delta\rho g sin(\theta)}{(dp_F/dx)_{SG}}$	Ratio between gravity forces and pressure drop of the gas	[-]	(2.26)
Dimensionless gas velocity	$K_G = \frac{Fr_G Re_L^{1/2}}{cos(\theta)}$	Dimensionless gas velocity	[-]	(2.31)
Dimensionless liquid velocity	$N_L = \frac{f_{SL} Fr_L^2}{cos(\theta)}$	Dimensionless gas velocity	[-]	(2.32)

Table 2.2. Utilized characteristic quantities and the corresponding descriptions

2.3.1 Type of regimes

In a two phase system, the geometrical configuration of the interfaces determine the flow regime (flow pattern). While in single phase flows the laminar and turbulent regimes should only need to be distinguished, when gas and liquid flow together inside a pipe, the interaction between the two phases can generate a large number

of configurations. As the characterization of multi-phase flow regimes can be at times subjective, there is still some debate towards the actual number of patterns that can exist inside a pipe [7]. In the present work, the initial classification that was adopted in the study conducted by Pereya et al [8] has been followed, in which six different multi-phase flow patterns (provided below and in figure 2.1) are considered:

- Stratified smooth (SS): the liquid and gas phases are completely separated and the interface between the two phases is flat. Commonly observed in horizontal pipes in which both gas and liquid phases flow at low velocities. This flow regime is completely absent in upward inclined pipes.
- Stratified wavy (SW): similar to the SS, liquid and gas are still completely separated, though, due to the higher liquid or gas velocities, the interface is now wavy. This flow regime is typical of downward tilted pipes and tends to disappear in upward inclinations.
- Dispersed bubble (DB): the gas is moving as dispersed bubbles in the liquid phase. This regime occurs at high liquid velocities, where the turbulence is high enough to dominate over the buoyancy, entrapping gas particles in the liquid flow.
- Bubbly flow (B): the gas is dispersed as discrete bubbles in continuous liquid. This flow regime can only take place in near vertical systems at low liquid velocities, where the turbulence is not enough to break the bubbles and result in transition to DB.
- Intermittent flow (I): gas and liquid alternates one another. This is one of the most common flow patterns and can be present in any type of system configuration, usually for intermediate liquid and gas velocities.
- Annular flow (A): the bulk of the liquid flows on the wall of the pipe as a film, while the gas occupies the center of the duct as a continuous phase.

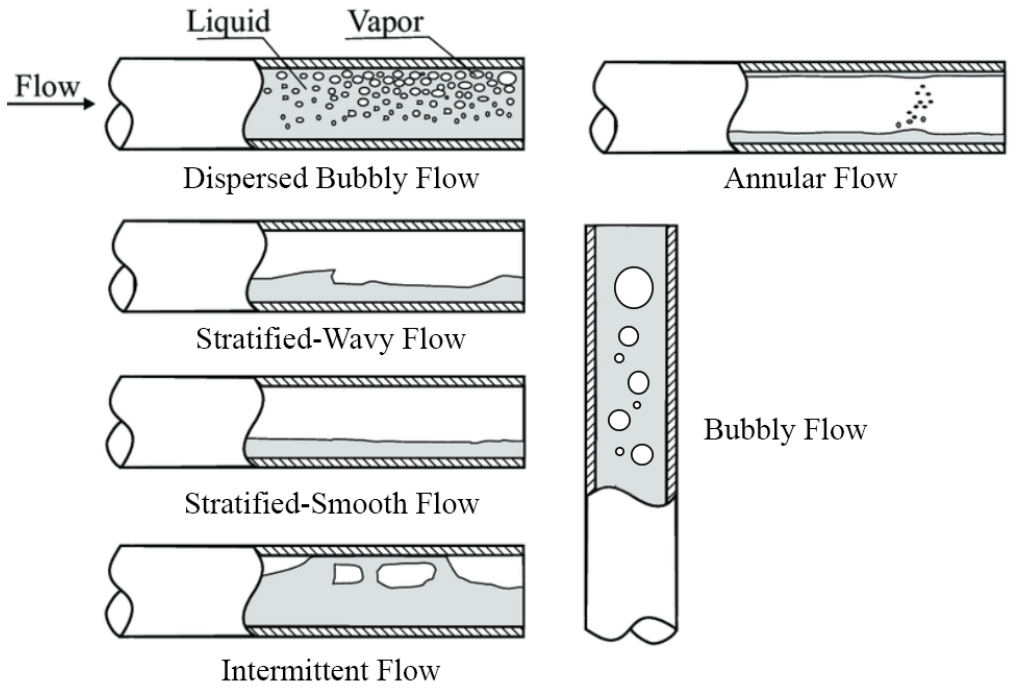

Figure 2.1. Types of flow regimes

2.3.2 Stratified to non-stratified transition

The transition between stratified to non stratified regime can be described by the Kelvin-Helmholtz instability. Considering a pipe in stratified flow conditions, in which a wave of finite length is present, in order to maintain the same flow rate, the gas will accelerate over the wave's crest causing its expansion due to a drop in pressure. On the other hand, the gravity acting on the wave will promote its suppression. The transition between the regime thus depends on whether the acceleration forces can become dominant over the gravity ones. Taitel [39] and Barnea [23] proposed the following dimensionless quantities to determine the transition criterion:

$$X_{LM}^2 = \frac{(\mathrm{d}p_\mathrm{F}/\mathrm{d}x)_{SL}}{(\mathrm{d}p_\mathrm{F}/\mathrm{d}x)_{SG}} \tag{2.25}$$

$$Y = \frac{\Delta\rho g sin(\theta)}{(dp_F/dx)_{SG}} \tag{2.26}$$

$$Fr_G = U_{sG}\left(\frac{\rho}{gD\Delta\rho}\right)^{0.5} \tag{2.27}$$

The first quantity X_{LM}^2 is the Lockhart–Martinelli parameter and is defined as the ratio between the pressure drop of the two phases as if they were flowing alone. This parameter, which is widely used to provide an estimation of the two-phase frictional pressure drop multiplier, is utilized in the present study along with the Chisholm parameter Y (the ratio between the gravity forces and the pressure drop of the gas phase), to solve the momentum equations for each phase in stratified flow. In the

present work, aiming at determining the single-phase pressure drop that is required for the calculation of both X_{LM}^2 and Y, the following correlations have been utilized in order to determine the fanning friction factor:

$$\text{Laminar:} \qquad f = \frac{16}{Re} \qquad \text{for Re} < 2300 \tag{2.28}$$

$$\text{Turbulent [42]:} \quad f = 0.0625 \left[\log\left(\frac{150.39}{Re^{0.98865}} - \frac{152.66}{Re}\right) \right]^{-2} \qquad \text{for Re} > 2300 \tag{2.29}$$

Where Re is the Reynolds number, which is the ratio between inertial and viscous forces (Eq. 2.30)

$$Re = \frac{\rho D U_S}{\mu} \tag{2.30}$$

The third employed quantity, Fr_G, is the gas densimetric Froude number that defined as the ratio between the inertia forces and the gravity. In the study conducted by Taitel [39], this quantity was divided by a cosine. To avoid numerical instability, quantities divided by the system tilt have been modified. While this approach might negatively impact the performance of each individual feature, machine learning algorithms are be able to combine the mentioned information with ease.

2.3.3 Stratified Smooth to Stratified Wavy

In the conditions in which the gas velocity is high, but not elevated enough to satisfy the Kelvin-Helmholtz instability, waves will form on the surface of the liquid. Tatiel-Duckler [39] were able to identify a transition criteria based on the dimensionless gas velocity K_G (Eq. 2.31).

$$K_G = \frac{Fr_G Re_L^{1/2}}{cos(\theta)} \tag{2.31}$$

2.3.4 Stratified wavy to annular

In the particular case of downward-inclined pipes, in which the stratified liquid level is low and the liquid velocity is elevated, Barnea [23] and Tatiel [39] proposed that droplets may be torn from the liquid surface and be deposited on the upper wall initiating the annular regime. This transition has been modeled employing the dimensionless liquid velocity N_L (Eq. 2.32).

$$N_L = \frac{f_{SL} Fr_L^2}{cos(\theta)} \tag{2.32}$$

2.3.5 Dispersed to non-dispersed bubble transition

The transition from dispersed to non-dispersed bubble has been attributed to two phenomena: bubble agglomeration and bubble creaming. Since the two mechanisms are related the surface tension of the bubbles, Pereyra [8] proposed the use of the following dimensionless groups to determine this transition:

$$We = \frac{U_{SL}^2 \rho_L D}{\sigma} \qquad (2.33) \qquad\qquad Eo = \frac{\Delta \rho g D^2}{\sigma} \qquad (2.34)$$

Eo is the Eotvos number, which is defined as the ratio between the gravitational and the capillary forces, while We is the Weber number that is defined as the ratio between the drag and cohesion forces.

2.3.6 Transition to dispersed bubble

In near-horizontal systems, if the turbulence is high enough to dominate over the buoyancy forces keeping the gas at the top of the pipe, transition to the dispersed bubble regime can happen. This transition has been modeled by Taitel [39] using the dimensionless quantity T_{TB}, which is the ratio between the turbulent and buoyancy forces:

$$T_{TB} = \frac{(\mathrm{d}p_F/\mathrm{d}x)_{SL}}{\Delta \rho g cos(\theta)} \qquad (2.35)$$

2.3.7 Annular to non-annular transition

This flow regime transition has been attributed to the liquid blocking the gas core, which consequently promotes intermittent flow. According to Barnea [23], this phenomena can be due to the instability of annular flow or spontaneous blockage due to the wave growth in the liquid film. These two phenomena can be modeled employing the liquid holdup α_L parameter in annular flow, which is obtained by solving the following non-algebraic equation [23] (Eq. 2.36):

$$Y = \frac{1 + 75\alpha_L}{(1-\alpha_L)^{2.5}\alpha_L} - \frac{1}{\alpha_L^3} X_{LM}^2 \qquad (2.36)$$

Chapter 3

Machine Learning Theory

The most important and computationally time consuming part of the pipeline is without doubt the model optimization step. When trying to solve a classification problem through machine learning, we are actually dealing with a function estimation problem. Suppose of having a systems consisting of "outputs" y and a set of "input" features $\mathbf{x} = \{x_1, \ldots, x_n\}$. Given a training sample $\{y_i, x_i\}_i^N$ of known (y,$\mathbf{x}$) values, the aim of a machine learning algorithm is to find a function $F^*(\mathbf{x})$ that can estimate y using $\mathbf{x}$ as an input. As the various strategies to obtain $F^*(\mathbf{x})$ differ based on the machine learning model adopted, multiple solutions are offered for the same estimation problem. Unfortunately, the best machine learning algorithm and its hyperparameters cannot be known a priori. The only reasonable solution for this optimization step is then to make educated guesses on the types of machine learning algorithms that can best solve our classification problem, and tune their hyperparameters using suitable heuristic techniques. This chapter is then dedicated by to describe the theory behind the selected algorithms explored during this thesis.

3.1 Ensemble Methods

Ensemble methods are a family of powerful machine learning techniques widely used to solve classification problems when the data is in tabular form [43]. They are some of the most performing algorithms on Kaggle [44], a site that host competitions between data scientists, that gives an unbiased estimate on the performance of the various machine learning algorithms. While a wide variety of ensemble methods are present in literature, the idea behind them is the same: as one often consults multiple experts in order to make a difficult decisions, ensemble methods aggregate the outputs of multiple models to generate a prediction [43]. How the "base" models are generated, and the method used to aggregate the outputs will differentiate the various ensemble methods discussed below.

3.1.1 Boosting

Boosting is an ensemble technique based on converting a series of weak learners in a strong learner [43]. A weak learner is defined as a model that only slightly perform better than a random guess, and is typically a simple function, like a decision tree

with limited depth. Boosting differentiate itself from other ensemble technique in that these weak learners will be trained in a sequential fashion, and each new model will aim to correct the errors of the previous ones. Given a dataset S composed of features $\mathbf{x} = \{x_1, \ldots, x_n\}$ and a target y, boosting tries to determine a function $F^*(\mathbf{x})$ that can estimate y using $\mathbf{x}$ as an input, such that over the joint distribution of all (y,$\mathbf{x}$) values, the expected value $\mathbf{E}$ of some specified loss function Ψ(y, $F(\mathbf{x})$) is minimized:

$$F^*(\mathbf{x}) = \arg\min_{F(\mathbf{x})} E_{y,\mathbf{x}} \Psi(y, F(\mathbf{x})) \tag{3.1}$$

The loss function Ψ is chosen to be differentiable, for regression problems is usually set to the least squares, while for classification problem with K classes it is usually a log-loss function:

$$\Psi(y, F(\mathbf{x})) = -\sum_{k=1}^{K} y_{i,k} \log(p_{i,k}) \tag{3.2}$$

Where y is a binary indicator, set to one when k is the correct class, multiplied by the predicted probability p of k given given observation i.

Boosting will try to approximate the function $F(\mathbf{x})$ using multiple weak learner h($\mathbf{x}$;$\mathbf{a}$):

$$F(\mathbf{x}) = \sum_{m=0}^{M} \beta_m h(\mathbf{x}; \mathbf{a}_m) \tag{3.3}$$

Again, "base learners" are usually simple functions of $\mathbf{x}$, with parameters $\mathbf{a} = \{a_1, \ldots, a_n\}$, that will perform poorly if used alone. The expansion coefficient $\{\beta\}_0^M$ and the parameters $\{\mathbf{a}_m\}_0^M$ are used in order to fit the training data in a forward "stage wise" manner. In the first step, an initial random guess for $F_0(\mathbf{x})$ is generated, and then updated using the following schema for m = 1,2,...,M:

$$(\beta_m, \mathbf{a}_m) = \arg\min_{\beta,\mathbf{a}} \sum_{i-1}^{N} \Psi(y_i, F_{m-1}(\mathbf{x}_i) + \beta h(\mathbf{x}_i; \mathbf{a})) \tag{3.4}$$

and

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \beta_m h(\mathbf{x}_i; \mathbf{a}) \tag{3.5}$$

Solving equation 3.4 can be challenging, and while other techniques are present in literature (eg. AdaBoost [45]), here the methodology adopted by gradient boosting is described, as this technique will generate the best performing algorithm of the whole pipeline. In order to determine the solution to said equation, gradient boosting [30] [46] uses a two step procedure. First, the function h($\mathbf{x}$; $\mathbf{a}$) is fit using least squares:

$$\mathbf{a}_m = \arg\min_{\mathbf{a},\rho} \sum_{i=1}^{N} [\widetilde{y}_{im} - \rho h(\mathbf{x}_i; \mathbf{a})]^2 \tag{3.6}$$

where $\widetilde{y}_{im}$ are the "pseudo"-residuals:

$$\widetilde{y}_{im} = -\left[\frac{\partial \Psi(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})} \tag{3.7}$$

Once h($\mathbf{x}$; $\mathbf{a}_m$) is determined, the optimal value for β_m is calculated by minimizing the loss function:

$$\beta_m = \arg\min_{\beta} \sum_{i=1}^{N} \Psi(y_i, F_{m-1}(\mathbf{x}_i) + \beta_m h(\mathbf{x}_i; \mathbf{a}_m)) \tag{3.8}$$

Using this strategy, solving eq 3.4 becomes trivial, by using first a least squared method 3.6 and then a single parameter optimization 3.8. As the only constraint for this strategy is that the loss function Ψ has to be differentiable (eq 3.7), this strategy is suitable for both classification and regression problems.

In gradient tree boosting, the base learners h($\mathbf{x}$; $\mathbf{a}$) are regression trees of limited depth L. Since the depth of these tree is limited, they will be characterized by high bias, but will be extremely fast to train, and won't make any strong assumption on the size of the dataset. At each m-th iteration, a regression tree will divide the $\mathbf{x}$ space in L-disjoint regions $\{\mathrm{R}_l m\}_{l=1}^{L}$ and predict a separate constant value (the flow regime in our problem) in each one:

$$h(\mathbf{x}; \{R_{lm}\}_l^L) = \sum_{l-1}^{L} \bar{y}_{lm} \mathbf{l}(\mathbf{x} \in \mathbf{R}_{lm}) \tag{3.9}$$

Where $\bar{y}_{lm}$ is the mean of (3.7) in each region of R_{lm}. The parameters $\{\mathbf{a}_m\}_0^M$ are the splitting variables and split points defined by the tree. Since eq 3.9 predicts a constant value for $\bar{y}_{lm}$, the outputs of the function $F_{m-1}(\mathbf{x})$ can at best modified by a quantity γ_{lm}:

$$\gamma_{lm} = \arg\min_{\gamma} \sum_{\mathbf{x}_i \in R_{lm}} \Psi(y_i, F_{m-1}(\mathbf{x}_i) + \gamma) \tag{3.10}$$

In order to avoid overfitting, a constant learning parameter ν is used to update formula of $F_{m-1}(\mathbf{x})$ (eq 3.5), that then becomes:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \gamma_{lm} \mathbf{l}(\mathbf{x} \in \mathbf{R}_{lm}) \tag{3.11}$$

This "shrinkage" parameter, can assume values between 0 and 1. Determining the optimal value for ν can be done through grid search or other hyper-parameters optimization techniques, but usually small values will lead to a better generalization error, as overfitting is avoided.

As stated before, gradient tree boosting is not the only type of boosting that can be found in literature. Another popular option is adaptive boosting, that will modify the weights β_m in eq 3.3 in order to give greater emphasis to the base learners that are able to correctly classify the errors of the previous models.

XGBoost

Extreme gradient boosting (XGBoost [47] [31]) is a scalable, open source, end-to-end tree boosting system. This package has been extensively used by the scientific community in order to win data mining and machine learning competitions. XGBoost

follow the theory discussed above, and builds the candidates decision trees of each step in a parallel fashion. Additionally, XGBoost offers the possibility of using a regularized loss function, built in order penalize model complexity and improve the generality. The hyperparameters for XGBoost are then reported in 3.2

XGBoost hyperparameters	Description	Type	Range
num-of-trees	Number of boosting rounds	Int	50 - 300
max-depth	Maximum depth of each tree	Int	1 - 9
eta	Step size shrinkage	Float	1e-8 - 10
gamma	Minimum loss reduction required to make a further partition on a leaf node	Float	1e-8 - 10
lambda	L2 regularization term	Float	1e-8 - 10
alpha	L1 regularization term	Float	1e-8 - 10

Table 3.1. XGBoost hyperparameters and their explored ranges in this work

LightGBM

LightGBM [25] [32] is another library for boosted trees published by Microsoft. While XGBoost provided remarkable improvements when compared to other gradient boosting machines in terms of scalability, it still required to analyze all the samples and the features in order to generate a split of a decision tree. LightGBM reduces the time required to build said split by using Gradient-based One Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). GOSS is based on the idea that under-trained instances, characterized by a large gradient, will be able to better minimize the loss function. When building a split, GOSS will then only select those instances, as well as a percentage of the other ones (that will be scaled to preserve the original data distribution). EFB on the other hand is used to reduce the number of features to analyze. When the data is high-dimensional, the probability of having mutually exclusive features is large. EFB will then bundle those feature together, effectively reducing the time needed to build the tree. Given these two implemented techniques, the trees built using LightGBM will be different from the ones of XGBoost, and a difference in the performance of the two algorithms has to be expected. Nonetheless, this difference will be minimal since they still belong to the same class of ML models. The hyperparameters for LightGBM are then reported in 3.2

LightGBM hyperparameters	Description	Type	Range
num-of-leaves	Maximum number of leaves in each tree	Int	2 - 300
learning-rate	Step size shrinkage	Float	1e-8 - 10
num-of-trees	Number of boosting rounds	Int	50 - 300
min-samples-leaf	Minimal number of data in one leaf	Int	1 - 300
lambda	L2 regularization term	Float	1e-8 - 10
alpha	L1 regularization term	Float	1e-8 - 10

Table 3.2. LightGBM hyperparameters and their explored ranges in this work

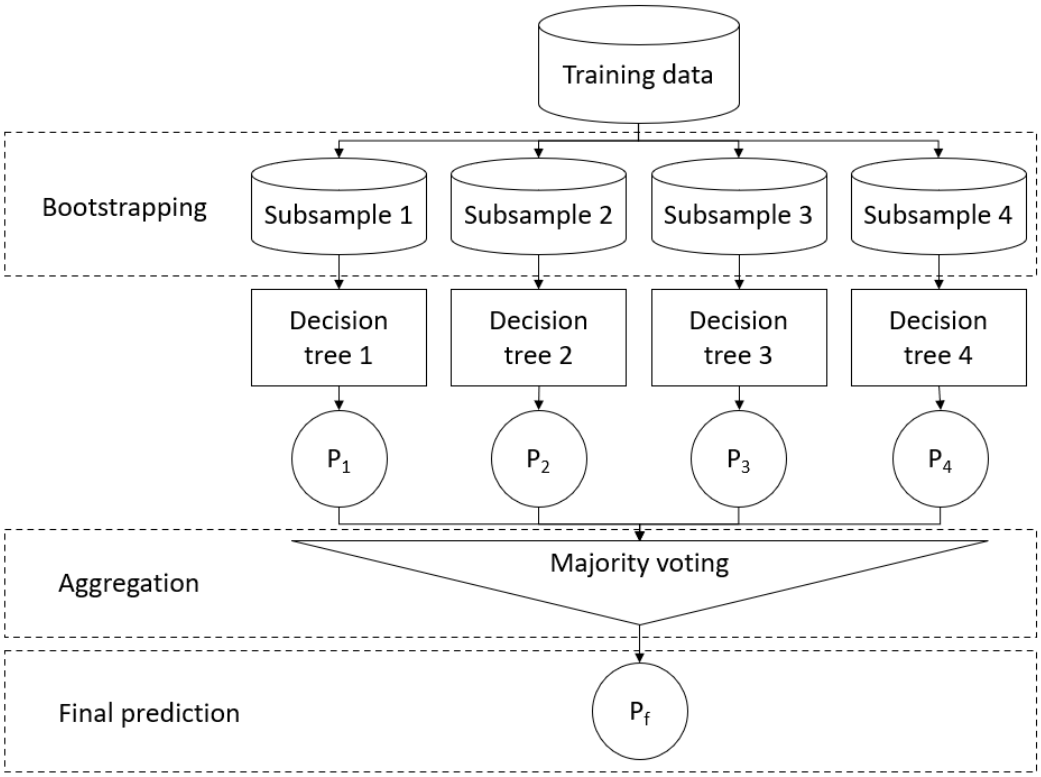

Figure 3.1. Bagging process overview. In the first step the training data is divided in multiple sub-samples, and each one of them is used to individually grow a decision tree. The ensemble then perform prediction by majority voting.

3.1.2 Bagging

Bagging [43], or bootstrap aggregation (figure 3.1), is an ensemble technique that is based on the parallel training of multiple models using different parts of the same dataset S. In the first part of the algorithm, sampling with replacement (bootstrapping) is applied to the dataset, generating a subset S_i that will be used to train a single model h_i. Typically, this model is decision tree, that will be fully grown, as the data used for its construction is limited. Once the whole ensemble has been generated, the output of the various model is aggregated, and classification is performed through majority voting.

Random Forest

Random forest [26] is a machine learning algorithm based on bagging, and represent one of the state-of-the-art ensemble method. In addition to bootstrapping aggregation, random forests introduce a randomized feature selection during the training process of each tree. The hyperparameters for Random Forest are then reported in 3.3. The pseudo code for random forests is instead reported in Algorithm 1. The algorithm starts with a an empty ensemble H, a specified number of trees B, and a dataset S with F features. For each individual tree, a subset S_i is extracted from the whole dataset and used for the training process. At each node, a small subset of features f is chosen (the original author suggested to be around the logarithm of F) and used to generate the split. Once the tree h_i is built, it is added to the ensemble H.

Algorithm 1: Random Forest

```
S ← {(x_1, y_1), ..., (x_n, y_n)}
F ← number of features
B ← number of trees
H ← ∅
for i = 1, ..., B do
    S_i ← bootstrap sample from S
    h_i ← ∅
    for Each node do
        f ← subset of F
        m ← split with the best feature in f
        h_i ← h_i ∪ {m}
    end
    H ← ∪ {h_i}
end
```

Random Forests hyper-parameters	Description	Type	Range
num-of-trees	Number of trees in the ensemble	Int	10 - 1000
max-depth	Maximum depth of each tree	Int	10 - 150
criterion	How to measure the quality of a split	Categorical	Gini or Entropy
min-samples-split	The minimum number of samples required to split an internal node	Int	2 - 50

Table 3.3. Random Forest hyperparameters and their explored ranges in this work

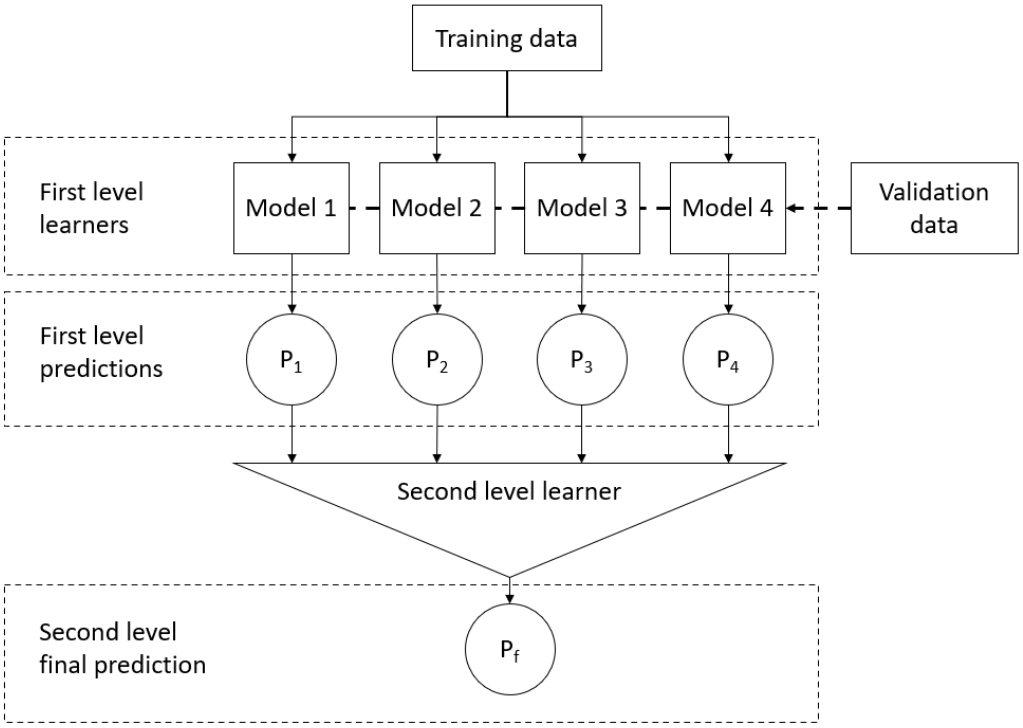

Figure 3.2. General procedure for blending/stacking. Models are trained with an initial dataset, and their output are then combined using a second level learner. This model combine the output of the base model using an holdout validation dataset

3.1.3 Blending and Stacking

Blending and stacking [33] are meta learning machine learning techniques. The difference between these two methods only relies on weather cross validation data (stacking) or a new set (blending (3.2)) is used to train the meta classifier. While previously discussed techniques aimed at training a large number of relatively simple models like decision trees, the scope of blending and stacking is to aggregate a relatively small number of strong estimators to further increase the overall performance. There is no constrain in the type of models that can constitute the ensemble: it can contain neural networks, support-vector-machines, gradient boosting machines, random forests etc. The overall process of blending is reported in figure 3.2. In the first step, a series of already trained models $\delta = \{\delta_0, ..., \delta_J\}$ is used on a new dataset $S = \{\mathbf{x}, y\}_0^m$ and a series of output vectors $\hat{y} = \{P_0(\mathbf{x}), ..., P_J(\mathbf{x})\}$ is generated. Then, using a second level learner, the output vectors are combined such that a chosen objective function Φ is maximized (or minimized):

$$F^*(\mathbf{x}) = \arg\max_{F(\mathbf{x})} \Phi(y, \hat{y}) \tag{3.12}$$

When compared to traditional ensemble techniques, the aim of blending is just to train a second level learner using the predictions of the already trained models. To this end, stacking and boosting can either employ complex second level learners (such as Random-Forests, Gradient-Boosting-Machines etc) or relatively simple ones, such as logistic regression. The benefit of the latter is that it can determine the optimal weights $w = \{w_0, ..., w_j\}$ for each single first level model, making this technique more explainable. Additionally, it should be underlined that both learning and stacking will not act on the already trained models, but will only combine them in the most performing manner.

3.2 Neural Networks

Neural networks [27] are a type of machine learning technique heavily employed for classification tasks. They are often used for speech recognition, time series analysis or image classification, where their performance remain to this day mostly unmatched [28]. The concept of neural networks stems from the idea of mimicking the human brain, by creating a system composed of signal processing elements (the neurons) interconnected with each-other using synaptic weights [27]. Given their ability of solving complex problems, over the years extensive effort has been put to generate new architectures and building blocks that can be used inside a network, allowing them to be far more flexible than other machine learning technique. The design of a neural network then require a strong domain knowledge, in order to consider only those architectures that can properly solve the problem considered. When the data is in tabular form, and there is no temporal relation between the inputs, the most used architecture is without doubt the Multi-Layer-Perceptron (MLP), which employ multiple perceptrons in a sequential fashion in order to preform classification. Very recently however, the field of deep learning has been changed by attention based networks, that supposedly exceed the performance of traditional networks in many applications [48]. Thus, the performance of an attention based network: Tab-Net [28] [36], is also explored.

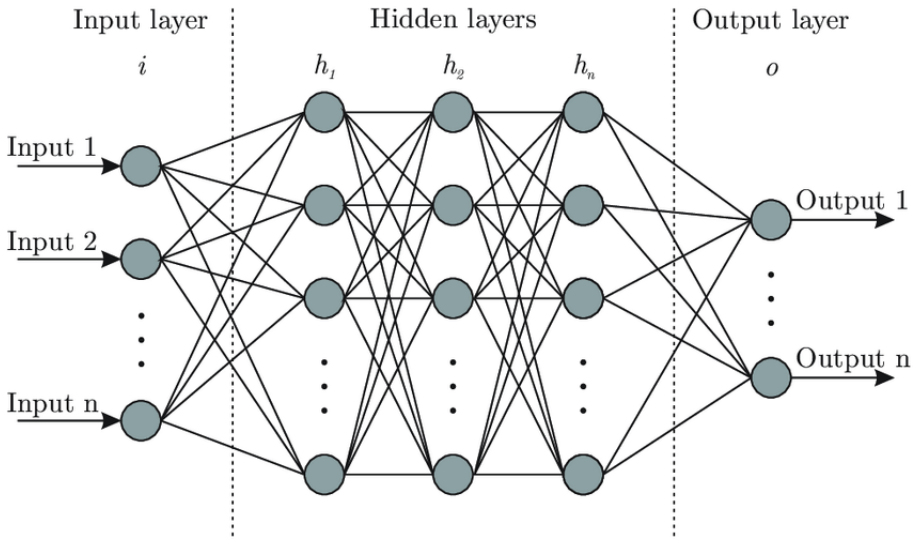

Figure 3.3. A fully connected feed-forward neural network

3.2.1 Multi-Layer-Perceptron

Multi-layer perceptrons [27] are the most common type of neural networks. Their architecture is relatively simple, as the name suggests, they are composed of multiple layers of perceptrons (fig 3.3). Their feed-forward structure allow the to only be passed from one layer to the next one, there is no feedback like in recurrent neural networks. When a vector $\{\mathbf{x}\}$ is passed through the network, a series of different activation functions transform the information, obtaining the function $F(\mathbf{x})$:

$$F(\mathbf{x}) = f^{(n)}(f^{(n-1)}(...f^{(1)}(x))) \tag{3.13}$$

Where n is the number of layers in the network. Typically, these activation functions are different from each other, and can be sigmoids, rectified linear units or hyperbolic tangents. The activation function of the output layer on the other hand is chosen in order to match the form of the target y. For classification purposes, it is usually a soft-max function:

$$\mathrm{P(y{=}j \mid x)} = \frac{e^{x^T w_j}}{\sum_{k=0}^{K} e^{x^T w_k}} \tag{3.14}$$

During the training of the network, the function $F(\mathbf{x})$ is driven iteratively to match the real unknown function $F^*(\mathbf{x})$ by minimizing a suitable loss function, like the one defined in eq 3.2. The aim of the training procedure is the same described for ensemble methods (eq 3.14), and is usually done through stochastic gradient descent. As the convergence of the network is not guaranteed and the training of these models is expensive (both in terms of time and resources), choosing the correct hyperparameters is fundamental, as well as understanding what are the unpromising trials that needs to be pruned. Notably, there is no constrain on the output of middle layers constituting the network, effectively making this systems a "black-box". In this thesis, the multi-layer-perceptrons were generated using Tensorflow 2.4.1 [34]. The only modification to the architecture discussed above was the addition of a batch normalization layer

at the beginning of the network, to avoid scaling the input features beforehand and the possibility of adding dropout layers. The hyperparameters for the MLP are then reported in Table 3.4

MLP hyperparameters	Description	Type	Range
num-of-layers	Number of layers in the neural network	Int	1 - 7
num-of-units	Number of neurons in each layer	int	64 - 256
dropout	Fraction of the input units to drop in each layer	Float	0 - 0.4
learning-rate	Learning rate for the network	float	1e-3 - 1e-1
epochs	Number of training epochs	Int	100 - 1500
optimizer	Type of optimizer used during the training	Categorical	Adam
loss	Type of loss used for training	Categorical	Cross-entropy
batch-size	Number of samples each iteration	Int	8192

Table 3.4. MLP hyperparameters and their explored ranges in this work. The number of units and the dropout rate are different for each layer.

Tab-Net hyperparameters	Description	Type	Range
steps	Number of decision blocks in the architecture	Int	2 - 5
gamma	Coefficient for feature reusage in the masks	Float	1 - 2
n-independent	Number of independent Gated Linear Units layers at each step	Int	1 - 3
n-shared	Number of shared Gated Linear Units layers at each step	Int	1 - 3
n-d	Width of the decision prediction layer	Int	8 - 64
n-a	Width of the attention embedding for each mask	Int	8 - 64
learning-rate	Learning rate for the network	float	1e-3 - 1e-1
epochs	Number of training epochs	Int	100 - 1500
optimizer	Type of optimizer used during the training	Categorical	Adam
loss	Type of loss used for training	Categorical	Crossentropy
batch-size	Number of samples each iteration	Int	8192

Table 3.5. Tab-Net hyperparameters and their explored ranges in this work.

3.2.2 Tab-Net

The type of neural network discussed above, is a great introduction to discuss attention based networks like Tab-Net [28]. While the logic behind the training is still the same (back-propagation with a suitable optimizer), architecture like tab-net are no longer made of only perceptrons, but additional elements, like feature transformers and masks are added between each layer. In this architecture, the aim is no longer to emulate the human brain, but to obtain the benefits of decision trees, such as the ability to use only the most performing features, in a deep learning setup. This is achieved through feature selection blocks, that will learn how to restrict the numbers of available features in

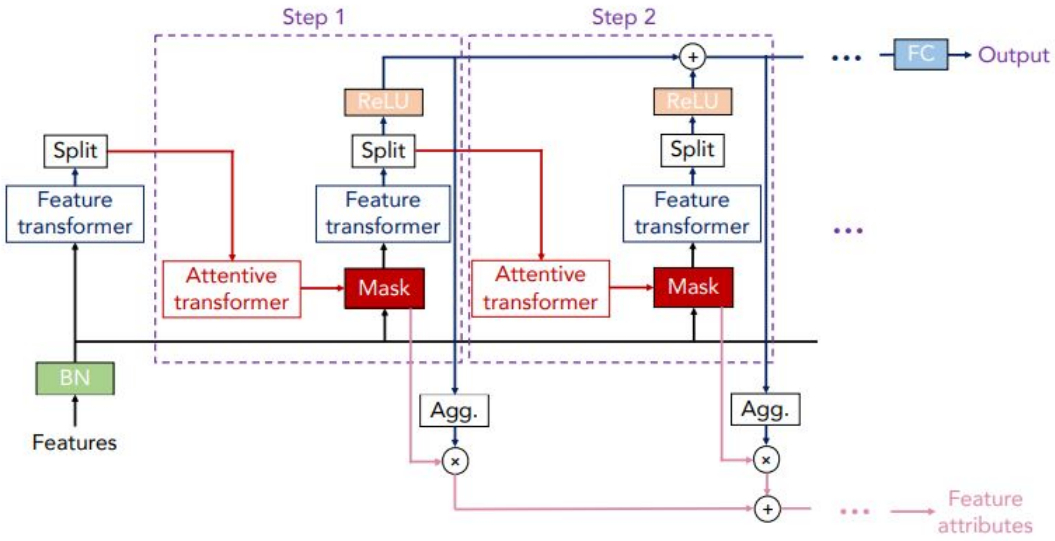

Figure 3.4. Tab-Net architecture

each step, much like a decision tree uses the most performing feature in each node split.

The schematic of Tab-Net is reported in figure 3.4. The network takes as input raw data, and uses batch normalization and categorical feature embedding as a preprocessing step. The pre-processed data is then passed through all the N decision blocks of the network. Each of these blocks also take as an input the processed information of the previous step, in order to train a learn-able mask along with an attentive transformer to restrict the features used in each step. This attentive transformer is composed of a fully connected layer, a batch normalization and a sparse-matrix normalization block (figure 3.5), which control how many times a feature can be used throughout the net via a relaxation parameter γ. When $\gamma = 1$ a feature can be used only in a single step, while as the value of γ increases, more flexibility is provided to the data. Once the pool of restricted feature has been selected, they are passed through a feature transformer block. This unit is composed of a part shared across all the steps of the network in order to enhance robustness during training, and a part that is specific to each step. The length of said block can be tuned through suitable hyperparameters, and is always composed by a series of fully connected layers, a batch normalization and gated linear units. Once the information has been transformed, it is passed through a Rectified Linear unit and added to the output of the previous steps, following the logic of decision trees ensembles. The hyperparameters for Tab-Net are then reported in 3.5

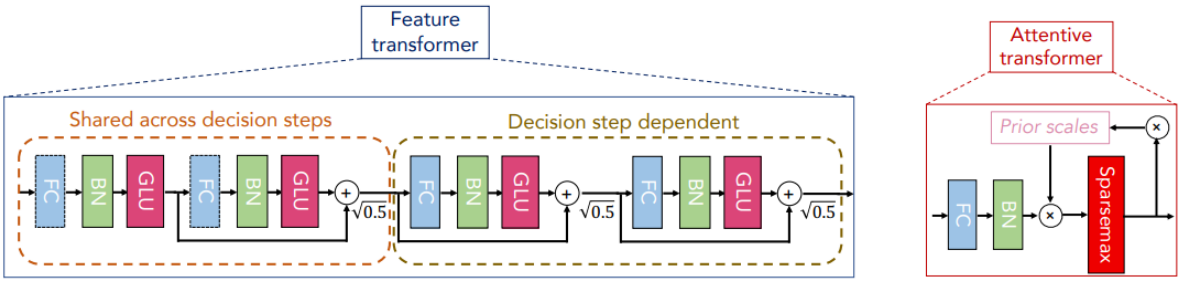

Figure 3.5. Feature transformer and attention transformer architecture

Chapter 4

Machine Learning Based Pipeline

When dealing with a machine learning problem, one has to take in consideration that most of the time the dataset used to train the algorithms cannot cover all the possible configurations that a model could encounter over time. Moreover, while one estimator might perform extremely well on the training and test data, making mathematical assumptions on its generality capabilities is often impractical. As new experimental evidence becomes available, it is then essential to guarantee a simple methodology to update the optimal machine learning pipeline. In the present work, in order to obtain the optimal pipeline, a multi-staged iterative procedure has been implemented and conducted which includes the following steps: data cleaning, feature engineering, model selection, feature selection, model evaluation and model deployment (figure 4.1). Accordingly, the general methodology that has been adopted to train and optimize the pipeline is described and the theoretical and practical aspects behind some of the most important step of the pipeline are illustrated.

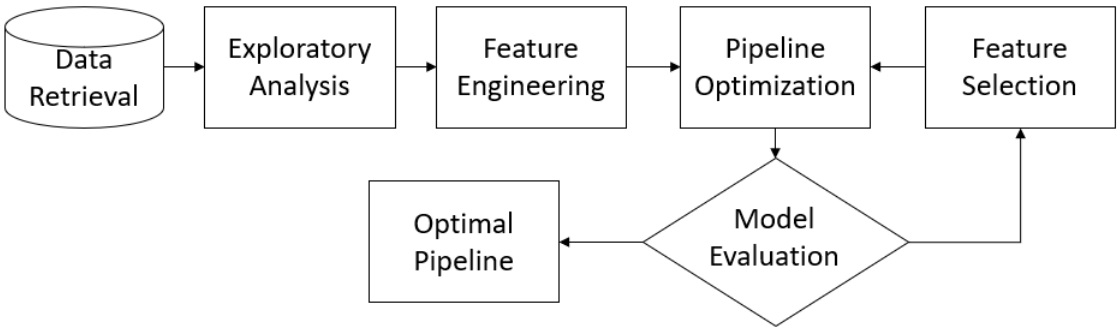

Figure 4.1. A schematic of the machine learning pipeline developed for this thesis

4.1 Methodology

In the first step, considering the governing physical phenomena and the recommendations that have been provided in the previously conducted physical phenomena-based studies (discussed in section 2), a series of additional features are generated and added to the dataset. Next, the dataset is divided into the training (80% of samples) and

testing (20% of samples) sets. The training set, employing a 5 fold cross-validation method, is used to optimize the pipeline. The test set is instead utilized to evaluate the performance of the proposed pipeline in estimating the flow regime on a set of points for which it has not been trained and optimized. In order to preserve the distribution of the classes, the generation of each fold (validation and testing) is performed using a stratified approach. The model selection procedure is then conducted and the most suitable algorithm, while utilizing all of the features, is identified. In order to give the same importance to all the classes, in the cross-validation procedure, SMOTE [37] is applied to each fold of the training data. Next, feature selection is applied to the dataset and employing the Sequential Forward Floating Selection (SFFS) approach, features are progressively added until a plateau in the cross-validation accuracy is achieved. Finally, the selected subset is utilized to repeat the algorithm optimization step and the most promising algorithm resulting in the highest accuracy is identified.

4.2 Data Pre-processing

Table 4.1. **Summary of the studies in the data base.**

Authors	Fluids	ρ_l (Kg/m^3)	ρ_g (Kg/m^3)	μ_l (Pa·s)	σ (N/m)	d(cm)	θ (deg)	Data Points
Shoham [9]	Air-Water	1000	1.18	0.001	0.070	2.54 and 5.1	-90 to +90	5676
Lin [10]	Air-Water	1000	1.12	0.001	0.070	2.54 and 9.54	0	141
Kouba [11]	Air-Kerosene	814	3.00	0.0019	0.029	7.62	0	53
Kokal [12]	Air-Oil	860	4.13	0.007	0.032	2.58 to 7.63	-9 to +9	1668
Van Dresar [13]	Hydrogen-Water	77	0.13	0.00001	0.070	0.874	1.5	83
Wilkens [14]	CO2-Salty Water	1025 to 1059	5.02 to 14.90	0.001	0.070	9.72	-2 to +5	204
Meng [15]	Air-Oil	883 to 889	1.49 to 2.16	0.0047 to 0.0063	0.03	5.01	-2 to +2	153
Manabe [16]	Natural Gas-Oil	789 to 809	8.10 to 26.90	0.0032	0.015	5.49	0 to +90	247
Mata [17]	Air-Oil	879.8	1.3	0.483	0.03	5.08	0	80
Adbuvayt [18] [19]	Nitrogen/Air-Water	1000	5.52 to 23.4	0.001	0.07	5.49 and 10.64	0 to +3	443
Omebere-Iyari [20]	Nitrogen-Naptha	700 and 702	23.4 and 104	0.0003	0.01	18.9	90	98

The experimental database considered for this study is the one collected by Pereyra et al. [8], which consists of the most important studies conducted on flow pattern prediction. Each point in the dataset contains information about the system operating conditions (pressure (P), temperature (T), internal diameter (d), inclination angle (θ)), the fluids superficial velocities (U_{sL} and U_{sG}) and the fluid properties (density (ρ), viscosity (μ), superficial tension (σ)). A summary of this database is reported in Table 4.1. Most of the data points derive from a study conducted by Shoham [9] in 1982, that investigated the behaviour of air-water systems, in 50.8 and 25.4mm pipes, under all possible system tilts. Another author, Lin [10], also studied air-water systems, but used 25.4 and 95.4mm pipes, considering only horizontal configurations. The data from Kouba [11], deals with slug flows in air-kerosene systems. The work of Kokal [12] constitute another major part of the database, and it focuses on air-oil systems in near horizontal configurations, under different values of pressure. Next, Van Dresar [13] investigated the behavior of cryogenic fluids under conditions of low mass and heat flux. Wilkens [14], in 1997, studied the effects refers to CO2 and salty

water systems in a 97.2 mm diameter pipe. Meng [15] investigated low liquid loading in wet gas pipelines. The data from Manabe [16] contains information about the relation between pressure and flow patterns. Mata [17], in 2002, created a flow pattern map for high viscosity oil and water in an horizontal pipe. Abduvyant [18] studied the effects of pressure and pipe diameter on gas-water in near horizontal systems. Lastly, Omebere [20] studied flow patterns in large diameter vertical pipes at high pressures.

4.2.1 Exploratory Data Analysis

Using the collected dataset, the first step of the implemented procedure is to perform data analysis. The aim of this step is to identify and correct erroneous values, and to provide useful insights on the data. To this end, in figure 4.2 the histograms of some of the most important variables according to phenomenological models are reported. It is evident that the dataset provides a wide range of working configurations, but most of the data points refer to the intermittent flow regime. As an unbalanced dataset may result in biased models, proper data augmentation techniques will be necessary in future steps. In particular, for this thesis, Synthetic Minority Oversampling TEchnique (SMOTE [37]) is used to perform oversampling of the minority classes. It should be underlined that this method is used exclusively on the training data, and both validation and testing frames will not have any type of synthetic points in them, avoiding to generate optimistic expectations [21].

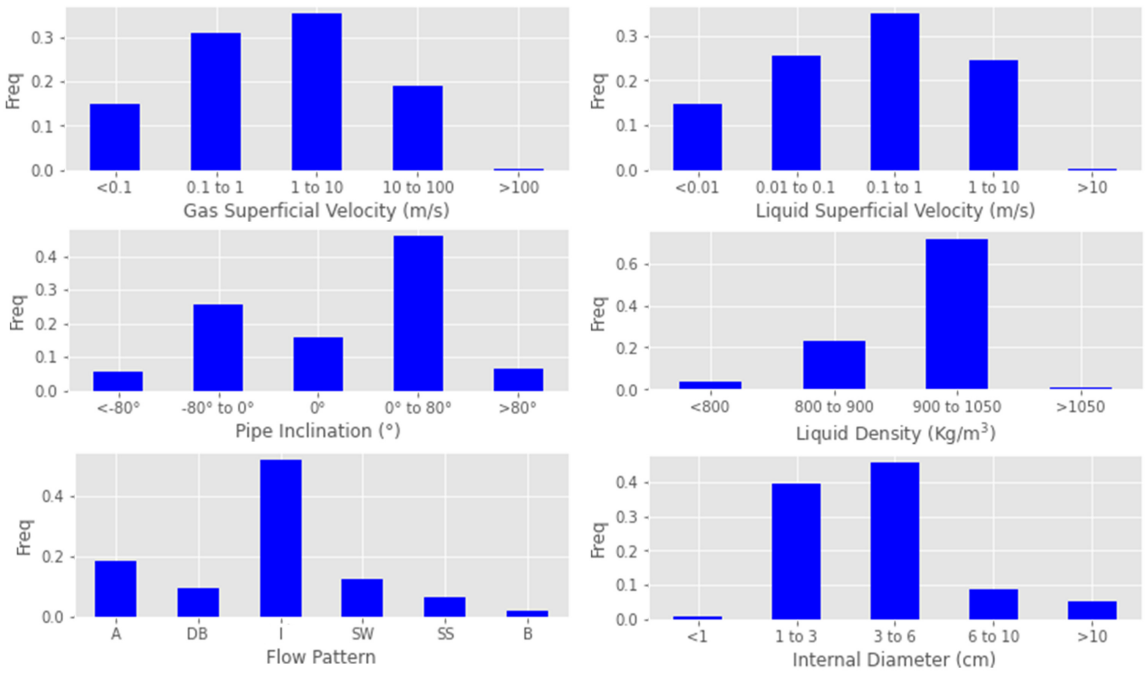

Figure 4.2. Distribution of some of the most important variables in the database

Aside from constructing useful graph to depict the state of the data, exploratory data analysis is also essential to identify what are possibles outliers of our dataset. In fact, even if the data used for this thesis is taken from a renowned journal, and hence

passed peer review, some values were found to be incorrect. For this specific problem, it should be underlined that the dataset is clearly compiled by an human using a spreadsheet, and thus common transcription errors are present. Some of these errors can be easy to spot (for example, the surface tension from the study of Abduvayt [18] is actually 0.070 N/m, it does not reach the absurd value of 230.070 N/m), while other require more advanced approaches to be identified. In particular, we first used Principal Component Analysis (PCA) to collect possible outliers. This techniques can be used to reduce the number of dimension while keeping as much variance as possible. Then, by hashing the system operating conditions we recreated the various multiphase flow maps in order to spot all the remaining incorrect points not identified using previous technique. It should be underlined that the identified outliers were modified (or removed) exclusively when the information of the original studies did not match the database's one. All other outliers, that were due to exotic experimental configurations, but still constituted valid evidence, were kept.

Principal Component Analysis

Principal Component Analysis (PCA) is an unsupervised technique that aims at reducing the dimensionality of a problem while preserving the majority of the information. The idea behind PCA is to project the data in a subspace which preserves as much variance as possible. In order to achieve this goal, first the mean of the data and the variance-covariance matrix are calculated:

$$\overline{X} = \frac{1}{N}\sum_{n=1}^{N} X_n \tag{4.1}$$

And:

$$\mathbf{S} = \frac{1}{N-1}\sum_{n=1}^{N}(X_n - \overline{X})(X_n - \overline{X})^T \tag{4.2}$$

As this matrix is symmetric by definition, it is always possible to determine a series of real positive eigenvalues λ_k and eigenvectors e_k. These eigenvalues are calculated and sorted in descending order, such that the largest λ_1 is the first principal component. Each principal component λ_k will then capture a portion of the total variance equal to:

$$\text{Explained Variance} = \frac{\lambda_k}{\sum_i \lambda_i} \tag{4.3}$$

The feature space can then be projected using the first k principal components $E_k = (e_1, ..., e_k)$:

$$\tilde{X} = XE_k \tag{4.4}$$

Notably, as the eigenvectors are calculated using the co-variance matrix $\mathbf{S}$, it is clear that before applying the PCA the data needs to be normalized if the scales of the variables are different, like the ones employed in this work.

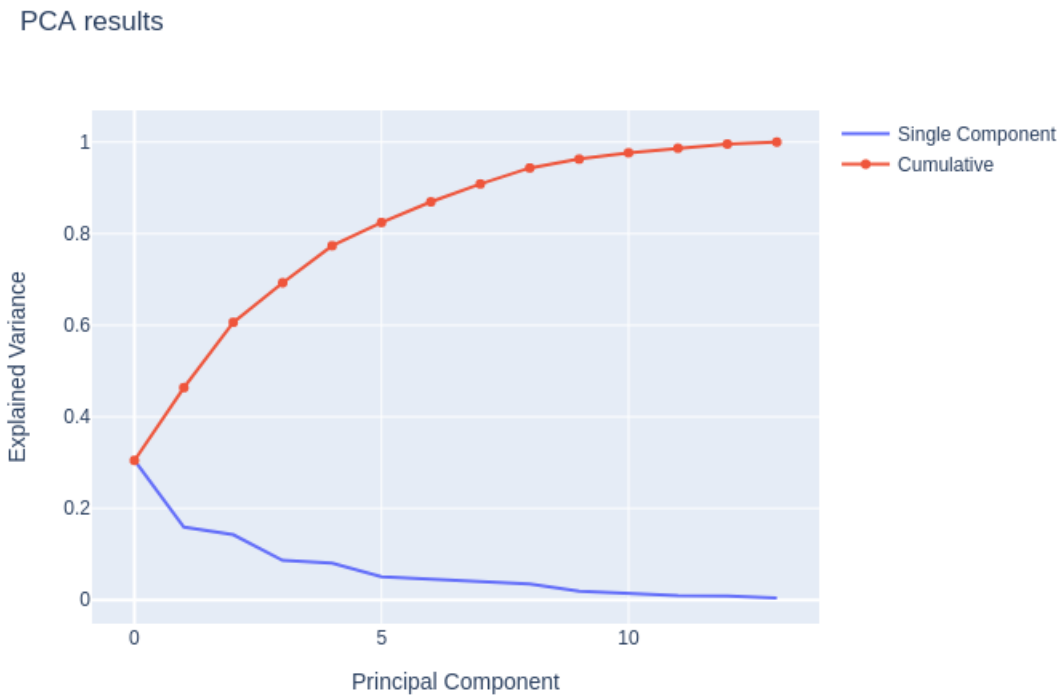

Figure 4.3. Variance explained by the principal components

In figure 4.3, the results of PCA are reported, and it is evident that most of the variance is explained by the first few principal components. The graph of the first two principal components is reported in figure 4.4. It is evident that some outliers are present, that needs to be investigated. In particular, cluster of points that are far from the majority of the data will most likely represent exotic experimental conditions, while lone data points are most likely associated with transcription errors. In figure 4.4 the same image is reported, by isolating the *stratified wavy* regime. It is evident that three data points are widely distanced from the rest of the cluster. Unsurprisingly, they were found to be incorrect, and have hence been removed. However, the same logic cannot be applied for all the data points of interest identified through PCA. In fact, some of the outliers for the *intermittent* flow regime were found to be perfectly valid data points, so they have been preserved. From PCA, it is also evident that by projecting the features in the principal component space, the flow patterns can be somewhat separated. While this could be an argument towards including principal components as features to enhance the performance of the various machine learning model, they are void of any physical significance, and have hence been employed for outliers detection/data-visualization purposes only.

Figure 4.4. Feature space visualized using the first and the second principal components

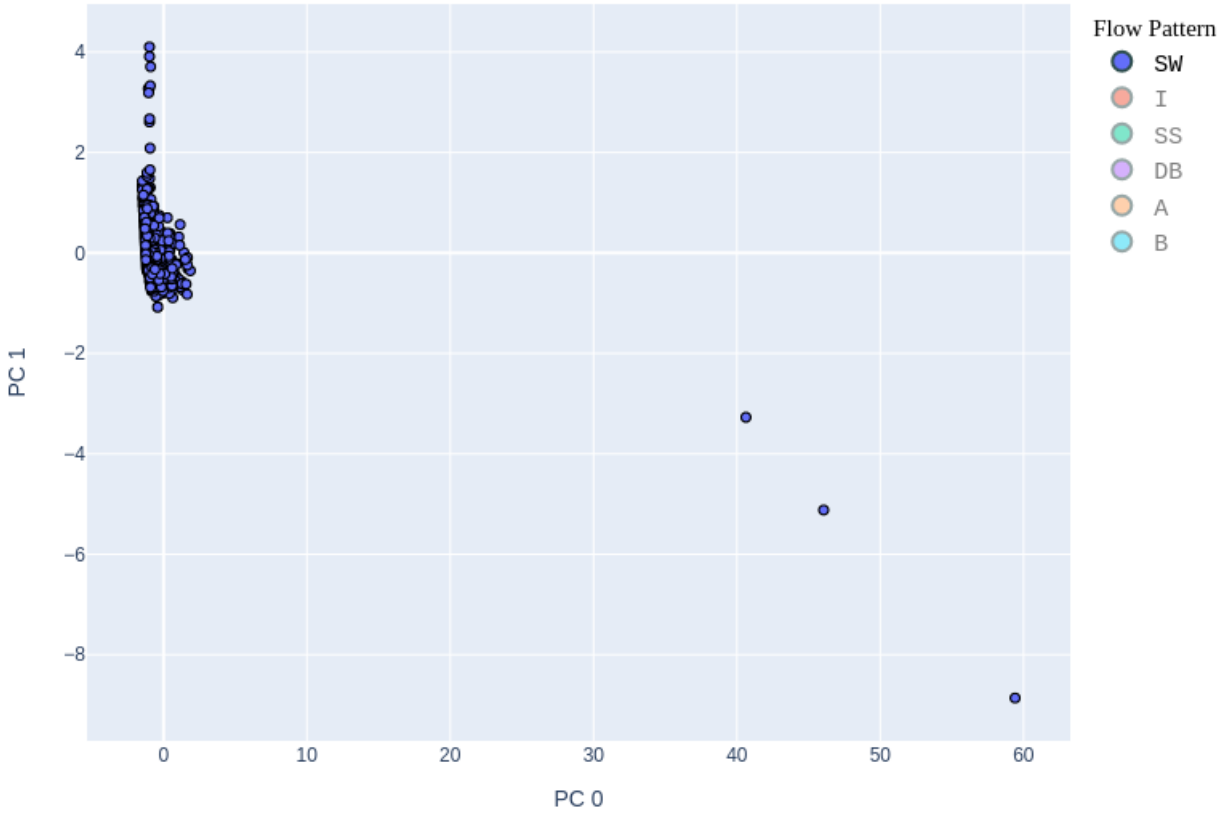

Figure 4.5. Details on the stratified wavy regime. Three points are clearly far from the cluster

Map Analysis

After the removal of incorrect data points identified through the employed outlier detection techniques, it was clear from the analysis of the results that some errors in the dataset were still present. These errors are not outliers, but are related to clear transcription errors, where the columns of similar features are inverted, such as the gas and liquid superficial velocities. To this end, the only method able to identify these errors is through visual inspection. This was achieved first by grouping the dataset by experimental setup. In particular, this step was performed by hashing together pressure, temperature, surface tension, internal diameter, system tilt. Then, through visual inspection, every map was analyzed to check weather or not it is consistent within itself. It should be underlined that this technique is far from being perfect, as it cannot be automatized. Nonetheless, the dataset employed for this study is relatively small, so visual inspection can be performed without being extremely burdensome. An example of incorrect data points uncovered with this method is reported in figure 4.6 and 4.7. It is evident that the liquid and gas superficial velocities columns were inverted for the dispersed bubbly regime, as the patterns on the corrected map actually match reasonable experimental evidence [23].

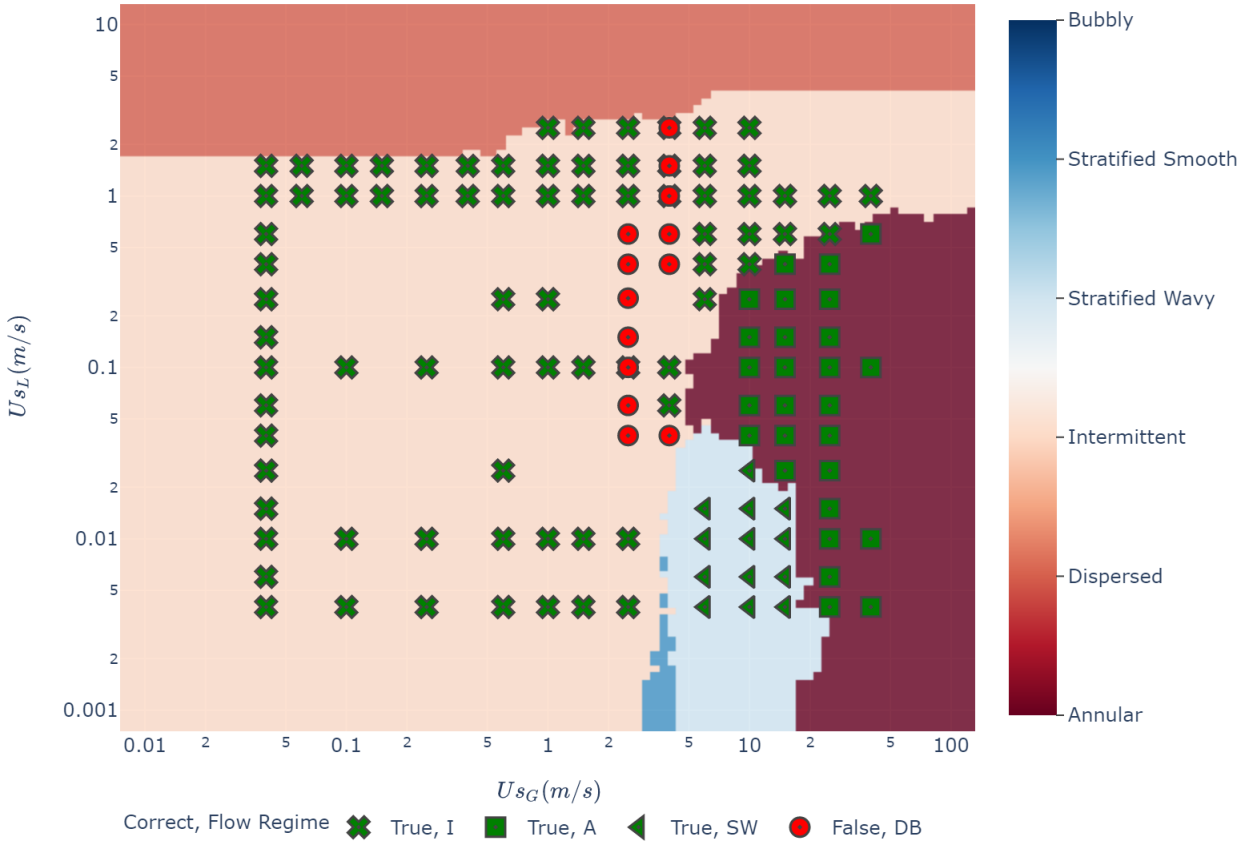

Figure 4.6. Data in Shoham study (2.54cm ID, 0.5° upward tilt): the superficial velocities for the dispersed bubbly regime are clearly swapped

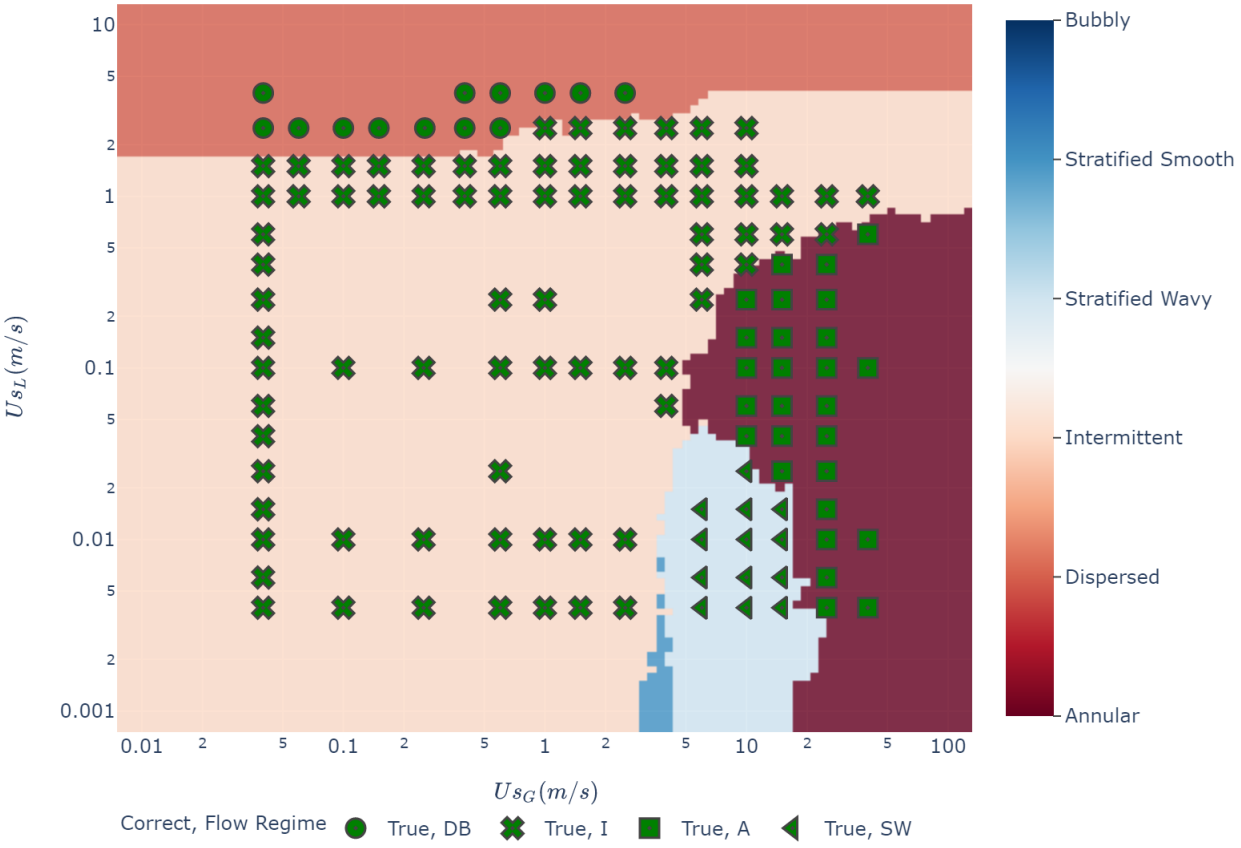

Figure 4.7. Corrected multiphase flow map

4.2.2 Feature Engineering

Following the data cleaning procedure, the next step in the implemented procedure is to introduce the physical quantities of interest discussed in chapter 2, summarized in Table 4.2. It should be underlined again that some of the dimensionless features were slightly modified: any cosine/sine at the denominator was removed for numerical stability as the database contains vertical and horizontal systems. By performing non linear transformation of the "basic" features, the aim is to allow the various machine learning models to leverage the known physics in order to generalize better.

Basic Features	P	T	ρ_l	ρ_g	μ_l	μ_g	σ	d	θ	U_{sl}	U_{sg}	
From Physical Models	Re_G	Re_L	Fr_G	Fr_L	X^2_{LM}	Y	K_G	N_L	We	Eo	T_{TB}	α_L

Table 4.2. Physical features employed

Once feature engineering is applied, the dataset contains a total of 13 features. Needless to say, most them are redundant, and by no means are all required to generate a strong model. However, as one of the aims of the pipeline is to determine what are the most important physical quantities, those features are only be used in the first

optimization step. Later on in the implemented procedure, feature selection methods are applied, and the final model only uses but a fraction of them.

4.2.3 Data Visualization with t-SNE

While principal component analysis underlined the presence of strong outliers in the dataset, its ability of differentiating the various classes in a two or three dimensional space is limited, as it is not actually its main purpose. Once the data has been cleaned from incorrect values, it is also important to grasp the relative position between the classes. This step is interesting in that it can be used to validate the assumptions derived from theory. To this end, t-SNE [49] was used for this work. The aim is similar to PCA in that we are also trying to reduce a highly dimensional data set $X = \{x_1, x_2, ..., x_n\}$ in a two or three dimensional space $y = \{y_1, y_2, ..., y_3\}$ that can easily be displayed in a scatter plot. This technique is based on Stocastic Neighboor Embedding [50], that converts the high-dimensional Euclidean distances between two points x_j and x_i into a conditional probability $p_{i|j}$. This quantity represents the probability for x_i to pick x_j as its neighbour if neighbours were selected using a Gaussian probability density function centered in x_i:

$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq i} \exp(-||x_k - x_j||^2/2\sigma^2)} \tag{4.5}$$

It is clear that for near data points, this probability is relatively high, while it tends to disappear as the euclidean distance starts to increase. Notably, this approach can also be applied in order to calculate the similarity between two points in the reduced feature space. In t-SNE this similarity is calculated using a Student t-distribution with one degree of freedom:

$$q_{j|i} = \frac{(1 - ||y_i - y_j||^2)^{-1}}{\sum_{k \neq i}(1 + ||y_k - y_j||^2)^{-1}} \tag{4.6}$$

The aim of t-SNE is then to map X to y such that the conditional probability between all the points is preserved as much as possible. The goodness of the projection can then be estimated through the Kullback-Lieber divergence over all the data points:

$$C = \sum_i KL(P_i|Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \tag{4.7}$$

The minimization of equation 4.7 is performed by means of gradient descent, and the mapping is obtained.

The t-SNE technique was used to map the feature space in a two dimensional space (figure 4.8) and a three dimensional space (figure 4.9). As expected, the relative position captured by this technique reflect the one from traditional multiphase flow maps:

- Intermittent (I) regime neighbors all other classes
- Stratified regimes (SS and SW) are close to each other

- Annular (A) flow confines with (SW) and (I)
- Dispersed bubbly (DB) and Bubbly (B) are mostly isolated from other classes, with the exception of (I)

Even while being an unsupervised method, the t-SNE is able to cluster the various classes effectively, and the majority of the data points reflect the behaviour expected from theory.

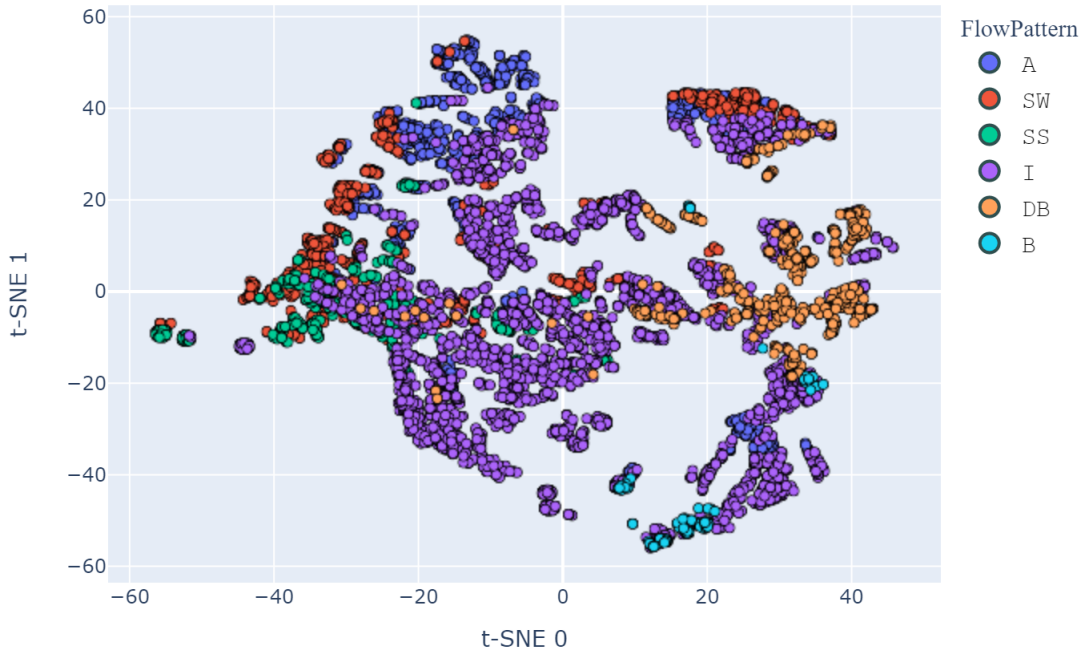

Figure 4.8. t-SNE projection: two components

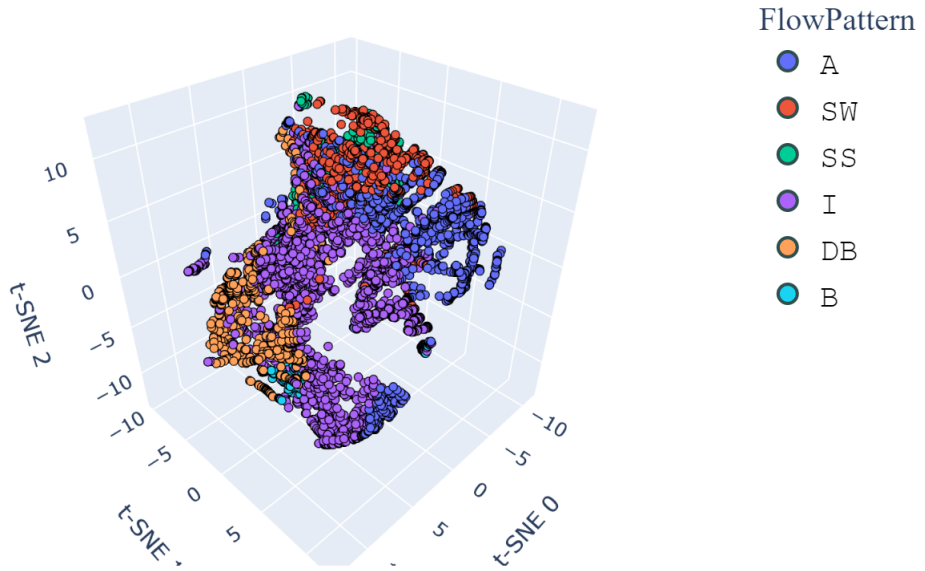

Figure 4.9. t-SNE projection: three components

4.2.4 Feature Scaling

Another important step to mention when performing data preprocessing, is feature scaling. While most of the time this step is applied to the dataset before the training of the estimators, in this specific problem, it is omitted. This choice is due to the nature of the machine learning algorithms employed for this study: ensemble methods are based on decision trees, that are not influenced by the scale of a feature, and the neural networks employed actually have a batch normalization layer at the beginning. By removing the feature scaling step, we can avoid making assumption on the shape of the data, as well as to avoid potential pitfalls caused by information leakage across the dataset.

4.3 Model Optimization

Once the data is pre-processed, the next step is to generate an optimized machine learning model. As discussed in chapter 3, for a general problem, the best performing model and its corresponding hyperparameters cannot be known a priori: the only reasonable solution is to train multiple classifiers and compare them using a suitable metric. In the interest of time, we limited our search to the algorithms described in chapter 3, and tuned their hyperparameters using a Bayesian approach. For the latter goal, Optuna [24] was employed. The choice of using this platform is related to its flexibility, ease of use and most importantly the possibility of performing Bayesian inference in the optimization process. When compared to traditional heuristic

techniques such as grid search, Bayesian approaches have the advantage of being able to predict the most promising set of hyperparameters using information from previous iterations (trials), obtaining the optimal solution with fewer iterations. To optimize the hyperparameters of an estimator, Optuna uses a custom defined objective function, to score the performance of the various models, and a sampler, to decide the next set of hyperparameters. The chosen objective function for the pipeline is the cross validation accuracy of the generated estimator, to be maximized. The sampler instead is a tree pazer estimator [51] [52], that fits two different Gaussian mixture models (GMM) to the previous hyperparameters of the study. The first GMM ($l(x)$) is fitted to those parameters associated with the best objective function values, while the second ($g(x)$) is associated to the remaining ones. At each iteration, the hyperparameters are chosen such that the ratio $l(x)/g(x)$ is minimized. Using this approach, the algorithm will spend more time to search the hyperparameter space near optimal regions, neglecting zones where it tends to perform poorly.

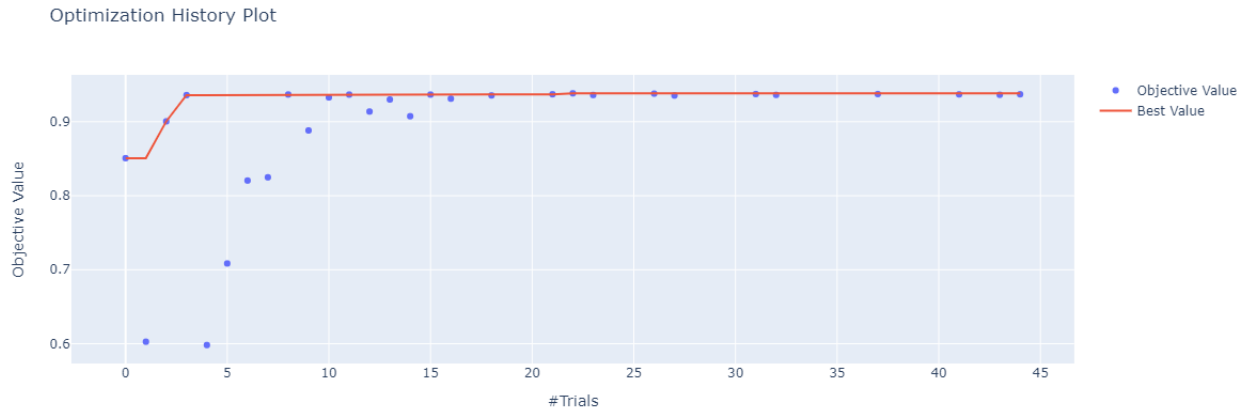

Figure 4.10. Optuna process overview

While the Bayesian sampler alone already increase the hyperparameters search performance, it is often useful to prune unpromising trials. To this end, a median pruner was chosen: if the validation accuracy of a single fold is lower than median of the previous trials, the estimator is discarded. A sample output of Optuna is reported above (figure 4.10). It is evident that at the beginning of the optimization, the hyperparameters are chosen randomly as reflected by the value of the objective function. As the study progresses however, unpromising trials are pruned, and the objective function tends to assume values near its optimum.

4.3.1 Utilized Metrics

In order to evaluate the performance of the generated machine learning models, accuracy score was used. In a classification context, this metric is defined as the ratio between the samples classified correctly and the total number of samples:

$$Accuracy = \frac{\text{Correct Classifications}}{\text{Total Number of samples}} \tag{4.8}$$

While this metric is indeed suited to describe the overall performance of the model, in unbalanced problem like this one, it can be misleading. For this reason, when discussing the performance of the proposed pipeline, the macro averaged F1 score is also reported. Defined as the harmonic mean between precision and recall, this quantity is able to can clearly show if the model has some bias towards any type of regime. When considering a single class 'j', precision is defined as the ratio between the correctly labeled element (TP) among the relevant ones (TP and FN):

$$Precision_j = \frac{TP_j}{TP_j + FP_j} \tag{4.9}$$

Recall is instead the ratio between the TP and the total number of elements classified as positives (TP and FP):

$$Recall_j = \frac{TP_j}{TP_j + FN_j} \tag{4.10}$$

The F1 score of the class j is then:

$$F1_j = 2 \cdot \frac{\text{Precision}_j \cdot \text{Recall}_j}{\text{Precision}_j + \text{Recall}_j} \tag{4.11}$$

The macro averaged F1-score is the unbiased mean of the single class F1-score values:

$$F1 = \frac{\sum_j^J F1_j}{J} \tag{4.12}$$

By using the macro averaged metric, we give the same importance to all the regime types, by assuming that we have no preference in being able to detect one more than the others.

4.4 Feature Selection

Once the first model optimization step is completed, it is now time to focus on removing redundant or useless features. The aim of reducing the dimensionality of the problem is two-fold: first the model will generalize better once the irrelevant features and the noise are removed, then the model will also be easier to explain and will be less computationally demanding [53]. Aside from embedding methods (that depends on the type of model used), feature elimination methods can be broadly classified into filter and wrapper methods [53]. The first category relies on a suitable ranking criterion to perform feature selection by ordering, removing all the features that do not pass a certain threshold. While these methods are fast, the selected subset might not be optimal in that redundant features might be chosen. Wrapper methods on the other hand will use the predictor as a black box, and the predictor performance as the metric to evaluate the goodness of the variable subset. These methods are able to take into consideration the interaction between the variables, and the subset will not include redundant features. The main drawback of these type of algorithms is that they are computationally expensive, so they may be applied only if the dataset contains only a small number of features. A good compromise can then be applying

first a filtering method, that will eliminate irrelevant features, and then a wrapping method that will discard the redundant ones.

4.4.1 Analysis of Variance

Analysis of variance (ANOVA) is a statistical method used to compare the difference in means between classes and reject the null hypothesis. It is widely used for filtering feature selection [54] [55], as it provide a robust and fast method to remove irrelevant features from the dataset. According to the principle of ANOVA, the statistical relevance of a feature λ is given by F(λ), that compares the average variability between the groups to the average variability within the groups:

$$F(\lambda) = \frac{\text{MSB}}{\text{MSW}} \tag{4.13}$$

MSW is the mean square within groups and is calculated by dividing the the Sum of Squares within groups with the degrees of freedom (n is the number of samples and k is the number of classes):

$$\text{MSW} = \frac{\sum_{i=1}^{k}\sum_{j=1}^{n_i}(x_{ij} - \overline{x})^2}{n-k} \tag{4.14}$$

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{k}\sum_{j=1}^{n_i} x_{ij} \tag{4.15}$$

MSB is the mean square between groups, and is calculated by dividing the sum of squares between the groups and the groups degree of freedom:

$$\text{MSB} = \frac{\sum_{i=1}^{k} n_i(\overline{x_i} - \overline{x})^2}{k-1} \tag{4.16}$$

$$\overline{x_i} = \frac{1}{n_i}\sum_{j=1}^{n_i} x_{ij} \tag{4.17}$$

A greater value of F(λ) denotes a larger statistical significance, hence, features that scores below a certain threshold, or the lowest performing ones, can be deemed irrelevant and removed from the dataset.

4.4.2 Sequential Floating Forward Selection

Sequential floating forward selection (SFFS) is wrapper method that uses a greedy algorithm [56] to build the a near optimal model by iteratively adding features to an initial empty subset [57]. The aim of this algorithm is to find near optimum solutions following a series of local optimal choices, reducing d-dimensional problem to a k-dimensional one. At any point SFFS perform the most promising choice among a set of all the possible decisions, and then tries to reduce the complexity of the solution by removing redundant information. Let S being the final subset of k feature considered, such that is possible to write $S = \{s^1, ..., s^k\}$. Let P be defined as an optimization problem, where a cross-validation score of an estimator, for example the accuracy f(S), needs to be maximized, so that is possible to write:

$$S^* = \arg\max_S f(S) \tag{4.18}$$

At each step i the algorithm considers a set $\mathbf{S}^i$ of the available features. The selected element s* to be added to the current solution $\widetilde{S}$={s^1, ..., s^{i-1}} correspond to the local optimum in the current iteration, satisfying the condition:

$$s^* = \arg\max_{s^i \in S^i} f\left(\widetilde{S} \cup \{s_j^i\}\right) \tag{4.19}$$

Then, SFFS, perform conditional removal if the following condition is met for an element $\{s^*\}$ of $\widetilde{S}$={s^1, ..., s^{i-1}}:

$$f(\widetilde{S} \backslash \{s^*\}) > f(\widetilde{S}) \tag{4.20}$$

SFFS will continue to insert features in the subset $\widetilde{S}$ until k features are added. The pseudo-code for this wrapper method is reported in A.2.

Algorithm 2: Sequential Forward Floating Selection for a maximization problem

```
S̃ ← ∅
m ← f(S̃)
for i = 1, ..., n do
    S^i = set of available features
    s* = best feature;
    for all s^i ∈ S^i do
        if f(S̃ ∪ {s^i}) > m then
            s* ← s^i
            m ← f(S̃ ∪ {s^i})
        end
    end
    S̃ ← S̃ ∪ s*
    for all s* ∈ S̃ do
        if f(S̃ \ {s*}) > m then
            S̃ ← S̃ \ s*
            m ← f(S̃)
        end
    end
end
```

4.5 Model Deployment

Once the non-relevant features are removed from the dataset, the model optimization step is repeated again, and the best machine learning model resulting in the highest cross-validation accuracy is identified. The final step of the implemented methodology

is then its deployment on an online environment. This step is extremely important, as it ensure that the results from the model are reproducible by anyone, making its output available to the public. In fact, while the whole pipeline can be found on an online repository, setting up a specific programming environment can be burdensome, and constitutes a considerable barrier for those who don't know machine learning (or Python). To this end, a app was created and deployed using Python Flask and JavaScript. Said app is then containerized using Docker and hosted using Google Cloud Run. Getting a prediction from the optimized pipeline is as simple as filling some blank values with the system's design condition. Another major advantage of this step, is that it allows for Continuous Integration and Continuous Delivery paradigms to be applied. The whole pipeline can now be updated as soon as new data becomes available with relatively low efforts.

Chapter 5

Model Evaluation

This chapter is dedicated to the discussion of the results obtained during the various optimization step of the implemented procedure. In the first section, the performance of the generated models is discussed, in terms of accuracy and F1 score. Then, the results of the feature selection are presented, and the quantities used by the best model are analyzed in terms of boxplots where possible. To further assess the quality of the pipeline, the best identified model is also used to generate some of the most important multiphase flow maps present in literature, as well as to predict a new set of data points whose experimental conditions differ from the one used for training. Finally, the possibility of tuning the probability threshold to increase the generality of the model is discussed.

5.1 Best Generated Models

Machine Learning Algorithm	All Features		Feature Subset	
	Accuracy [%]	F1 score [%]	Accuracy [%]	F1 score [%]
LightGBM	95.3	94.8	95.2	94.4
Random Forests	91.5	90.0	90.9	88.8
XGBoost	94.8	93.1	94.9	94.1
Stacking	95.3	94.8	95.2	94.4
MLP	93.4	91.6	89.9	90.6
TabNet	93.8	92.8	90.7	88.5

Table 5.1. Model comparison: best of family

The general results of the model optimization step are reported in table 5.1, where the cross validation metrics associated to all the explored algorithms are reported. As expected, the performance of XGBoost and LightGBM is similar, as they are still based boosted decision trees. When comparing the two deep learning architectures, it is evident that Tab-Net outperforms the MLP when the problem employs a large number of features. This can be due to the fact that Tab-Net encourage sparseness,

and it is able to automatically exclude irrelevant features before the feature selection step. On the other hand, when the number of features is reduced, and embedded feature selection is no longer necessary, Tab-Net has the same performance as the MLP. Additionally, the performance of these two models is the most heavily impacted when the number of features is reduced. This can be due to the fact that as a gradient boosting machine was used for the feature selection step, the remaining features are sub-optimal to perform deep learning. Another possibility is that by reducing the number of features, deep learning models have to use more complex architectures to extract the same level of information as before. Nonetheless, the hyperparameter tuning of deep learning models is extremely expensive (both in time and costs, even when using Optuna), and given the exceptionally good performance of ensemble techniques in every scenario, no further optimization has been carried out for these types of algorithms. Random Forests are able to achieve a reasonable performance, but they are still inferior to gradient boosting machines. Lastly, stacking can achieve the same performance of LightGBM in every scenario, as it uses its outputs to perform predictions. However, given that this is a second level learner which inevitably tends to over-fit the training data, it was still deemed inferior than a simple gradient boosting machine.

Hyper-parameters	Description	Type	Range	All Features	Feature Subset
num-of-leaves	Maximum number of leaves in each tree	Int	2 - 300	29	87
learning-rate	Step size shrinkage	Float	1e-8 - 10	0.1	0.1
num-of-trees	Number of decision trees in the ensemble	Int	50 - 300	250	223
min-samples-leaf	Minimal number of data points in one leaf node	Int	1 - 300	58	208
lambda	L2 regularization term	Float	1e-8 - 10	2e-8	9e-8
alpha	L1 regularization term	Float	1e-8 - 10	1e-3	3e-3

Table 5.2. LightGBM hyperparameters

In the first model optimization step, the best estimator identified by the procedure is a gradient boosting machine from the LightGBM library, whose hyperparameters

are reported in Table 5.2. Said model has a cross validation accuracy of 95.3% and a macro averaged F1 score of 94.8% (Table 5.3). The latter value confirms that the bias towards the intermittent flow regime is avoided, due to the oversampling techniques adopted during the training phase. Furthermore, the confusion matrix reported in Table 5.3, shows the advantage of using SMOTE, as the bubbly regime (B) has a single class precision of 95,9%, greater then the average even while being a minority class. When used to evaluate the test set (Table 5.4), the performance of the model is even slightly better, as the test accuracy is 95.9% and the macro average F1 score is 95.6%. The small increment in the test's metrics is justified by the fact that cross validation tends to provide pessimistic expectations. In fact, after the hyperparameter tuning step is complete, the whole data can be used to train the model, allowing for greater learning opportunities. Additionally, as test and validation scores match, it is evident that no overfitting occurred during the training phase of the model.

Table 5.3. Cross validation confusion matrix, all features

	A	DB	I	SW	SS	B	Precision [%]
A	1176	0	34	36	5	0	94.0
DB	0	598	28	0	0	0	95.5
I	57	42	3180	21	13	10	95.7
SW	24	2	17	811	6	0	94.3
SS	3	0	6	5	290	0	95.3
B	0	0	5	0	0	117	95.9

Table 5.4. Test confusion matrix, all features

	A	DB	I	SW	SS	B	Precision [%]
A	296	0	11	6	0	0	94.5
DB	0	151	5	0	0	0	96.8
I	14	10	802	3	2	0	96.5
SW	9	0	1	203	1	0	94.9
SS	0	0	1	0	76	0	98.7
B	0	0	3	0	0	28	90.3

Using the selected feature subset identified through SFFS, the model optimization step is repeated, and the best generated estimator is still a gradient boosting machine from the LightGBM library, but with slightly different hyperparameters (Table 5.2). Said model compares favorably with respect to the previous one, as the cross validation accuracy is 95.2%, and the F1 score is 94.4% (Table 5.5). Little to no information is lost, while the number of employed features is reduced from 13 to 5. Again, the results on the test set are similar, with an accuracy of 95.9% and an F1 score of 95.4% (Table 5.6).

Table 5.5. Cross validation confusion matrix, five features

	A	DB	I	SW	SS	B	Precision [%]
A	1168	0	48	30	5	0	93.4
DB	0	606	20	0	0	0	96.8
I	50	39	3199	20	10	5	96.3
SW	28	2	17	802	11	0	93.3
SS	5	0	10	8	281	0	92.4
B	0	0	4	0	0	118	96.7

Table 5.6. Test confusion matrix, five features

	A	DB	I	SW	SS	B	Precision [%]
A	291	0	14	6	2	0	93.0
DB	0	151	5	0	0	0	96.8
I	12	9	808	1	1	0	97.2
SW	8	0	3	201	2	0	93.9
SS	0	0	1	0	76	0	98.7
B	0	0	3	0	0	28	90.3

It is evident that the pipeline is extremely capable of predicting the flow regimes in experimental conditions similar to the ones of training. While these metrics provide useful insights about the performance of the model, it is also important to underline that most of the data point are usually collected near the transition region, given that it is the most interesting zone to conduct experiments. It is then important to asses how the model behaves in a more general environment, with the same experimental conditions of some of the studies in the training set. Thus, some multiphase flow maps are reproduced, and the regions are confronted with the experimental evidence from the original authors. In figure 5.3 and 5.2 some maps from Shoham [9] are displayed, while figure 5.2 used experimental condition from Kokal [12]. The generated boundaries closely match the experimental evidence from the authors, and all the errors (if any) occurs near the transition boundaries, where the flow regime can be at times debatable.

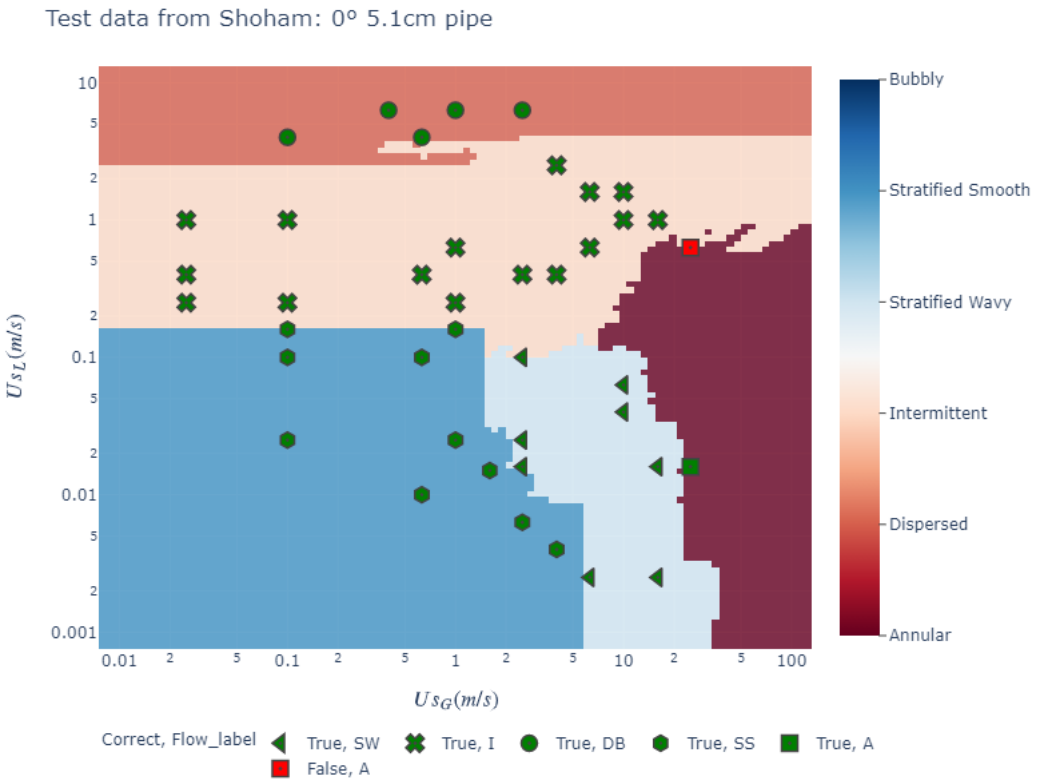

Figure 5.1. Test data from the study of Shoham, 5.1cm and 0 degrees of tilt

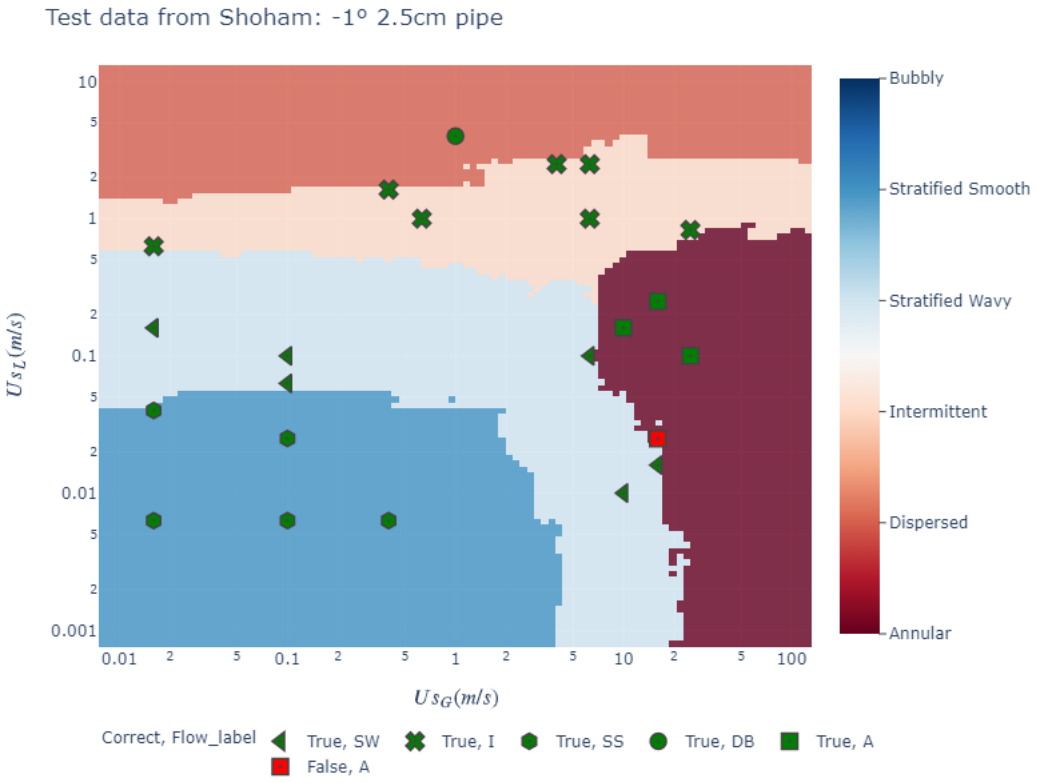

Figure 5.2. Test data from the study of Shoham, 2.5cm and -1 degrees of tilt

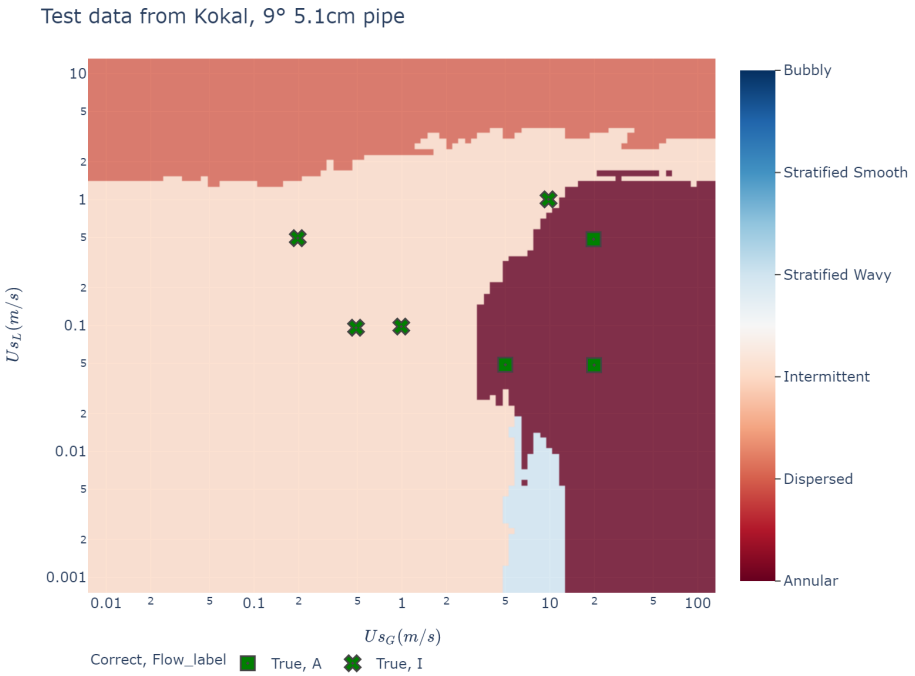

Figure 5.3. Test data from the study of Kokal, 5.1cm and 9 degrees of tilt

Finally, considering the built-in-capability of ensemble methods to predict a probability and not the class directly, the possibility of excluding some points from the classification was assessed. The results of this analysis are reported in figure 5.4: it is evident that the performance of the pipeline increases as the probability threshold to cast a prediction becomes higher. Nonetheless, as this approach allows the model to reject the classification of some data points (where the model is "uncertain"), its application should be considered only when accuracy is fundamental.

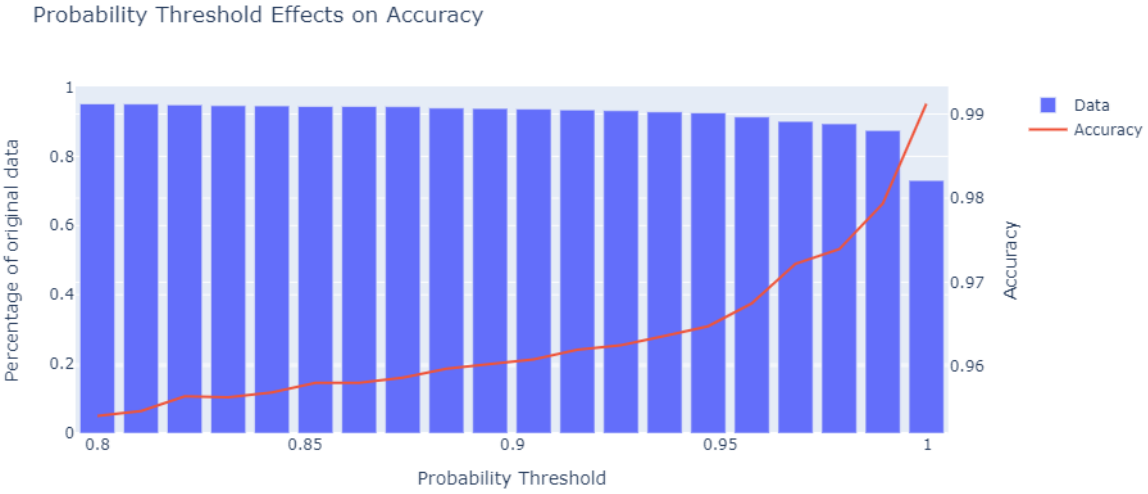

Figure 5.4. Effects of increasing the probability threshold on the test accuracy

5.2 Feature Selection

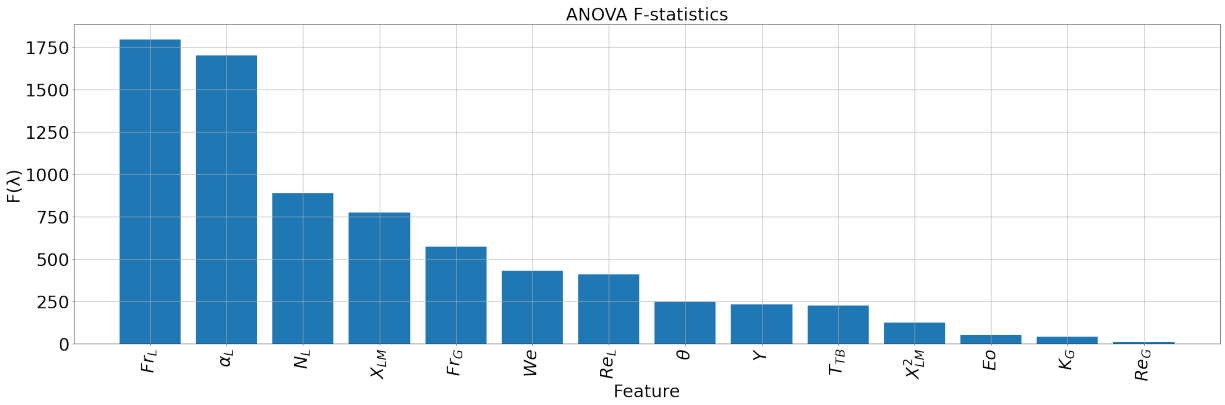

Figure 5.5. ANOVA results

The results from ANOVA feature selection are reported in figure 5.5. As expected, many of the features employed by the first optimized model are actually unable to differentiate the flow regimes if used alone. However, all of them pass the required threshold in order to be preserved. In fact, while some quantities might contain only very marginal information to separate the flow regimes, they might be combined with the most promising features in order to further enhance the quality of our machine learning algorithms. Given its $F(\lambda)$, the most performing feature is clearly the liquid Froude number Fr_L, and will likely produce the best results if used alone. Following this variable, the other most performing features are all related to the phase velocities, as it should be expected, according to theory. Of course, most of these quantities are redundant, but by using ANOVA, there is no mean of determining what is their best combination, hence the need of sequential floating feature selection.

All Features	θ	Re_G	Re_L	Fr_G	Fr_L	X^2_{LM}	Y	K_G	N_L	We	Eo	T_{TB}	α_L
SFFS Results	θ	Fr_L	Fr_G	Eo	X^2_{LM}								

Table 5.7. Feature selection results

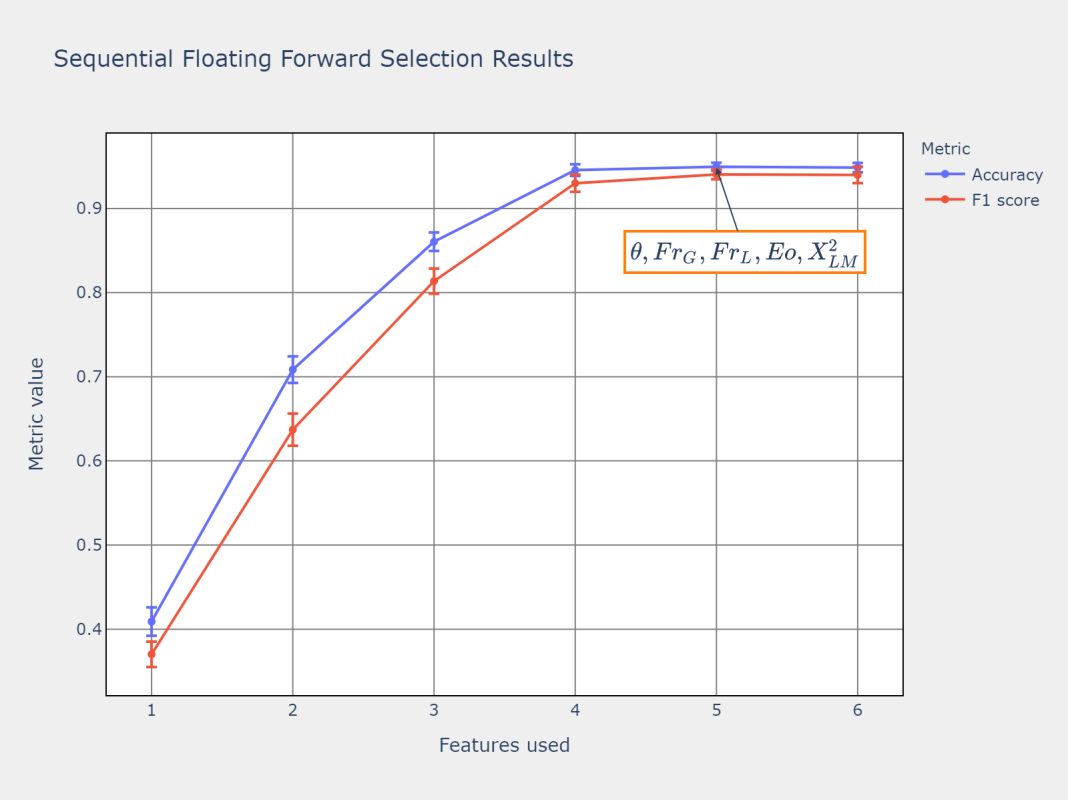

Figure 5.6. Sequential floating feature selection results

The results of sequential floating forward selection are reported in figure 5.6 and Table 5.7. It is evident that a plateau in the cross validation metrics is achieved as soon as five features are considered to train our estimators. As reported in Table 5.7, the selected subset contains both liquid and gas Froude velocities, as well as the system tilt. These quantities are widely employed by physical models, and wrapping feature selection is able grasp the importance of these features in determining the correct flow regime.

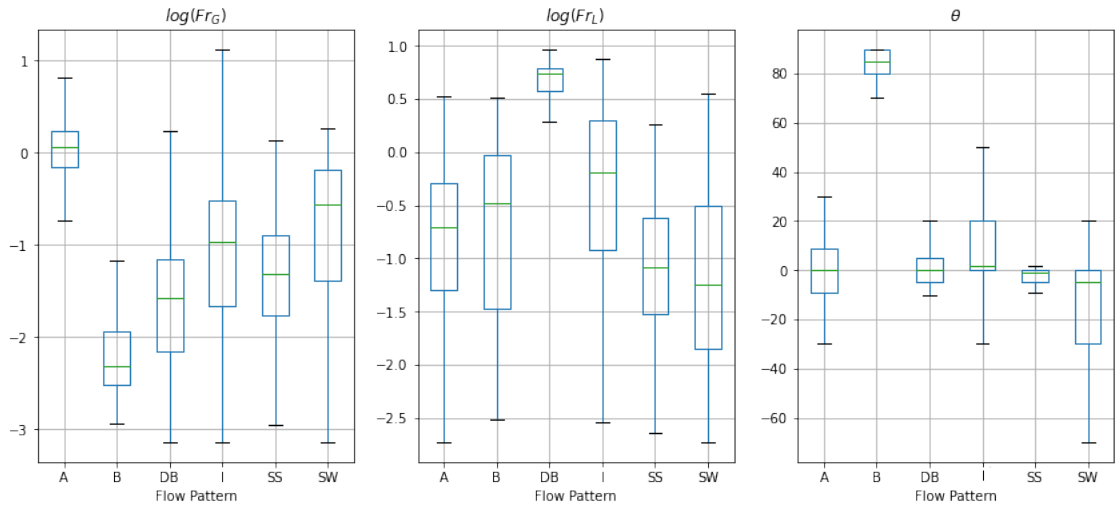

Figure 5.7. Boxplots of the selected feature subset

The influence of these features can easily be shown in terms of boxplots (figure 5.7), where it is evident that they are able to separate the flow regimes by some degree. The influence on the flow pattern for the remaining features is harder to explain, as they cannot be used alone to separate the flow regimes, as highlighted by the ANOVA test. From a physical point of view, the Eo number represent the ratio between gravitational and capillary forces, and could have been used by our model to introduce information about the system surface tension. The Lockhart-Martinelli parameter X^2_{LM} on the other hand could have been used to further assess stratified to non stratified transitions based on the ratio of the pressure drop between the two phases.

5.3 Generated Maps

While the proposed pipeline shows remarkable accuracy when used on experimental conditions similar to the ones of training, real world applications require a more robust model evaluation. In fact, while the training data is heterogeneous, it is evident that it is not near large enough to cover all the possibles working configurations. Thus, we assessed the generality capabilities of the proposed pipeline by comparing its outputs with some multiphase flow maps retrieved from literature, with design conditions somewhat different from the ones of training. First, we used the optimal pipeline to reproduce the map of Madhane [38], and the results can be seen in figure 5.8.

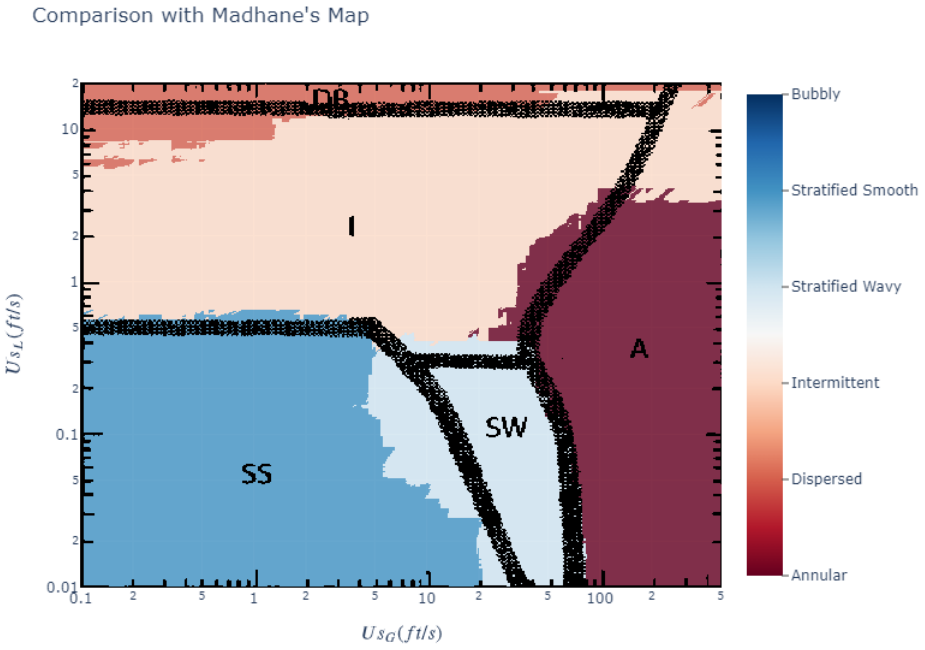

Figure 5.8. Comparison with the map of Madhande (in black) and the proposed pipeline (colored regions)

The transition boundaries closely match the ones of the authors, and the model only shows some discrepancies, where the classification of the regime is debatable.

5.4 Generalization

Table 5.8. Summary of the studies in the external data base.

Authors	Fluids	ρ_l (Kg/m^3)	ρ_g (Kg/m^3)	μ_l (Pa·s)	σ (N/m)	d (cm)	θ (deg)	Data Points
Yamaguci	Air-Water	998.2	1.2	0.001	0.07	8	-90	57
Shumeli	Water/Oil-SF6	998-848	46.2	0.001-0.1	0.02-0.06	6.9	0	33
Li	Air-Water	999	1.2	0.001	0.07	20.3	-90 or 0	172
Hanafizadeh	Air-Water	998	1.2	0.001	0.07	0.2	90	258
Saljoshi	Air-Water	998	3.2	0.001	0.07	0.1 to 0.3	0	318
Khaledi	SF6-Oil	854	25.54	0.032	0.06	6.9	0	287
Crowley	Air-Water/Others	995 to 710	1.2 to 32.1	0.00035 to 0.002	0.02 to 0.07	8.9 to 30	-2 to 4	757
Almabrok	Air-Water	998	13.3	0.001	0.07	10	-80	265
Brito	Air/Kerosene-Oil	825 to 920	11.34	0.0013 to 0.996	0.028 to 0.034	5.1	0	237
Kristiansen	Air/Kerosene-Water/Oil	812 or 998	1.2 to 52	0.001 or 0.002	0.0072 or 0.002	6.9	-0.1 to 0.1	113
Usui	Air-Water	998	1.2	0.001	0.07	2.4	-90	110
Ohnuki	Air-Water	995	1.2	0.0008	0.07	20	90	58
Ansari	Air-Water	999	0.96	0.001	0.07	4 or 7	90	337
Al-Ruhaimani	Air-Oil	884	8.1	0.5	0.03	5.1	90	83

It is now time to compare the output of the model to a generalized environment, with experimental conditions far from the ones of the training set. In order to achieve this comparison, another dataset was retrieved from literature, composed of different studies, never seen before by the model. It should be noted that the mentioned dataset only uses four different flow regimes (SS+SW=Stratified and DB+B=Dispersed) so the output of the model have been modified accordingly. This dataset is taken from the work of Quintino [22], is composed of 3259 data points, and more importantly, of 14 different studies (summarized in Table 5.8). The accuracy of the best generated model is reported in the bar plot of figure 5.9.

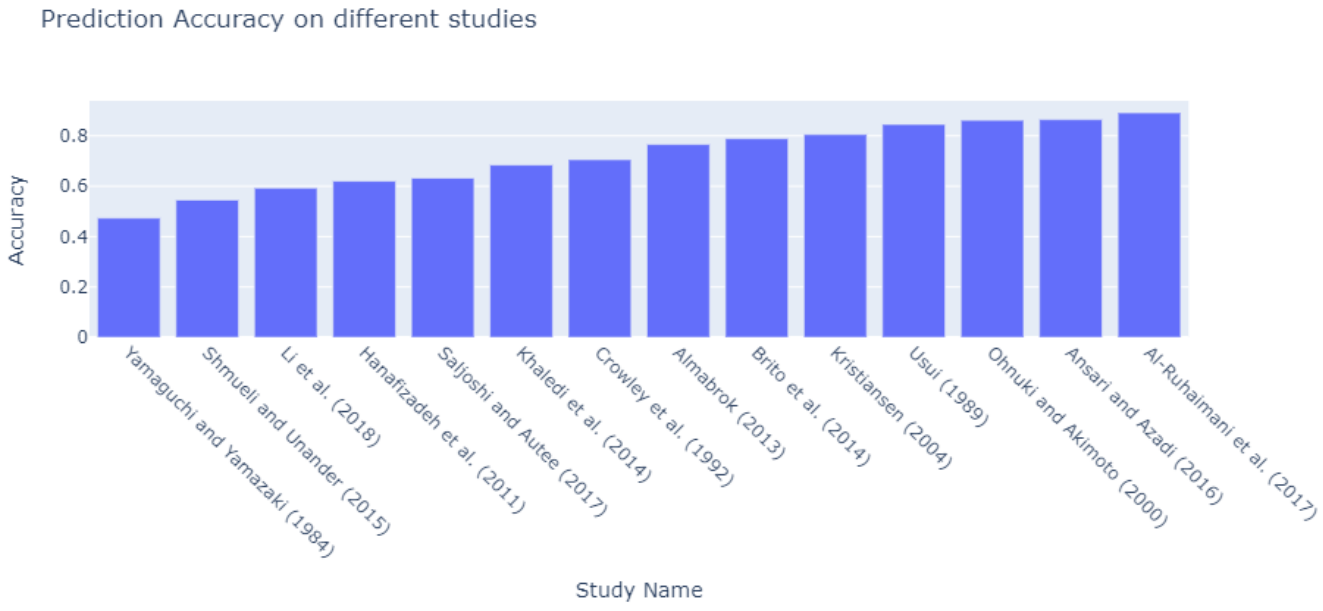

Figure 5.9. Accuracies using a new set of different studies

It is evident that the performance of the estimator is much lower than before. Nonetheless, it is still better than a random guess, as it achieves a general accuracy of 72.9% as well as a macro averaged F1 score of 70.0%. From this figure, it also evident that the difference in performance across the various studies is not negligible, and the model tends to perform well on studies whose experimental conditions resemble those of the training set. Ansari for example used air-water in vertical macro pipes, much similar to Shoham. On the other hand, studies that deal with micro tubes tend to be miss-classified by the model. This can be justified by the fact that, as shown in the exploratory data analysis of chapter 4, these types of systems are absent in our training data. To visualize the results on this generalized test set, two different maps are reproduced by the model, first one from Ansari (figure 5.10) and another one from Li (figure 5.11), that instead investigated micro tubes.

Figure 5.10. Comparison with data from the work of Ansari

From these images, it is evident that the model struggles to find the correct transition boundaries from unseen working configuration, but it is interesting to note that most of the miss classifications are still near the proper flow regime. This is an important property, as it could be exploited to assure that a certain flow regime is avoided during the design phase of a component. In fact, the top 2 accuracy of the model on this new dataset is 92.1%. Looking at this metric when the number of classes is 4 is not something to be excited about, but it shows the ability of the model to exclude some regimes. Lastly, we can again explore the effects on how implementing a probability threshold would impact the overall accuracy on this new external test set (fig 5.12). Unsurprisingly the overall accuracy increases, and it reaches a maximum of 86%, if we avoid classifying as much as half of the data.

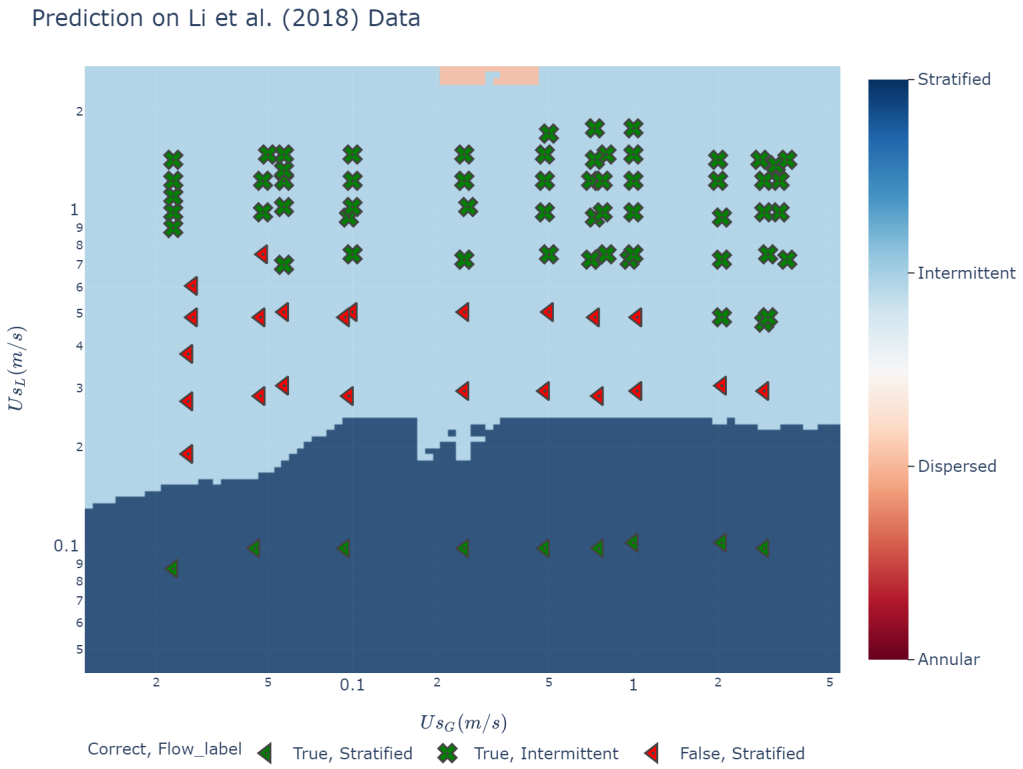

Figure 5.11. Comparison with data from the work of Li

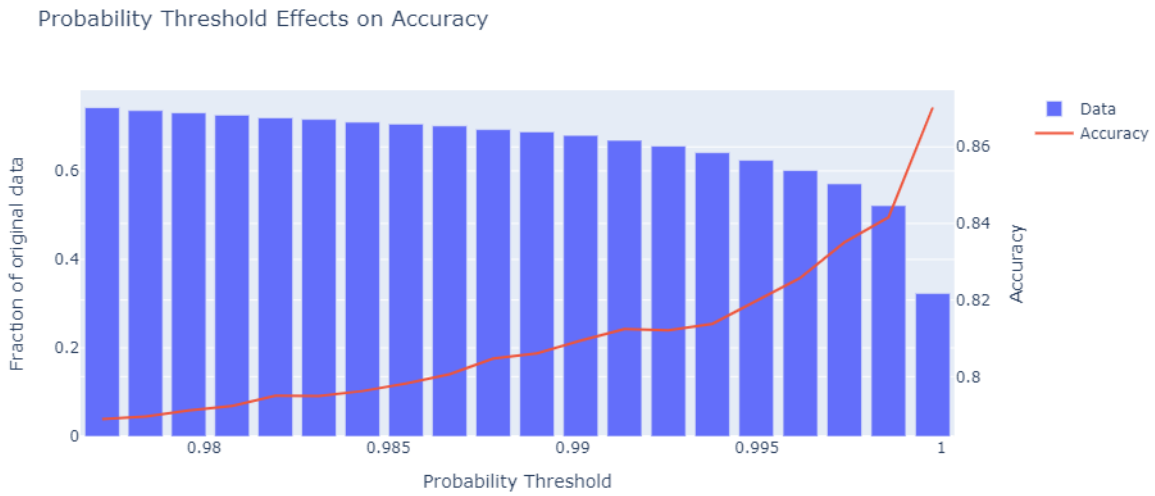

Figure 5.12. Effects of increasing the probability threshold on the test accuracy for the new dataset

Conclusions

The main objective of this thesis, was to develop and optimize a machine learning based pipeline in order to predict the multiphase flow regime under a wide range of working conditions. Modeling the flow regimes through the physical models was fundamental in order to increase the explainability of the machine learning estimators and to provide a rigorous analysis of the generated outputs. In the first step of the implemented procedure, the dataset is corrected and considerations are made about the shape of the data. Then, through feature engineering, the available physical knowledge was introduced in the dataset, in terms of dimensionless variables. The best machine learning model obtained through the first optimization step was a gradient boosting machine from the LightGBM library, with a cross validation accuracy of 95.3% and a macro averaged F1 score of 94.8%. The latter metric denotes that the model had no detectable bias towards any of the regimes. When used to evaluate the test set, the metrics are similar to the cross validation ones, with a test accuracy of 95.9% and a macro averaged F1 score of 95.6%. Given these values, it evident that overfitting was avoided. As the second core objective of this study was to determine the most important physical quantities, feature selection methods were applied to the dataset. Analysis of variance showed that, as expected from theory, quantities related to the phase velocities are of fundamental importance to differentiate the regimes. While ANOVA had proven to be an useful tool, it could not consider interaction between the variables. In order to determine the best features subset to solve this classification problem, sequential forward floating selection was used. This wrapping technique showed that, by greedily adding variables to an empty set, the near-optimal feature combination is: system tilt, liquid densimetric Froude number, gas densimetric Froude number, Eötvös number and the Lockhart-Martinelli parameter. With this subset, the model optimization step was repeated, and the new optimized estimator was found to be again a gradient boosting machine from LightGBM library, but with slightly different hyperparameters. This model has a cross validation accuracy of 95.2%, and a macro averaged F1 score of 94.4%. Given the similarity on the metrics between the first and the second model, we can state that little to no information was lost using the optimized feature subset. When used to evaluate the test set, it showed no signs of overfitting, as the accuracy was 95.9% and the F1 score 95.4%. To further assess the quality of the generated outputs, this model was also used to simulate and compare famous multiphase flow maps found in literature. The transition region identified by the model closely matched the one of theory. Given the overall performance of the pipeline on both feature selection and model optimization, this work can be considered a success. It should be noted however, that while the dataset employed for this thesis offers a wide range of working conditions, all the data points refer to smooth macro

tubes. Since a machine learning model is only as good as the data it has been trained with, it is not capable to consider complex physical phenomena related to surface roughness or micro channels. However, this deficiency easily be corrected with the introduction of more data related to those phenomena. Lastly, the whole implemented methodology is available on GitHub, while the optimized pipeline is deployed as an open source software using Google Cloud Run. Here, the outputs of the model are freely available to everyone, without the requirement of setting up a specific Python environment.

Appendix A

Appendix 1

A.1 Online Repository of the Thesis

The online repository for the code used in this thesis can be found on GitHub using the following link: https://github.com/Benetti-Hub/Multiphase-Flow-Regimes. The repository also contains the link to the related application hosted on Google Cloud Run.

Acronyms

ML	Machine learning
RF	Random Forests
GBM	Gradient Boosting Machine
ANN	Artificial neural network
MLP	Multi-layer-perceptron
AutoML	Automated Machine Learning
PCA	Principal Component Analysis
t-SNE	t-distributed Stochastic Neighbour Embedding
CV	Cross-Validation
SMOTE	Syntetic minority oversampling technique
SFFS	Sequential Floating Forward Selection
SS	Stratified Smooth Flow
SW	Stratified Wavy Flow
I	Intermittent Flow
DB	Dispersed Bubbly Flow
A	Annular Flow
B	Bubbly Flow

Bibliography

[1] R. L. Webb and K. Chung, "Two-phase flow distribution to tubes of parallel flow air-cooled heat exchangers," *Heat Transfer Engineering*, vol. 26, no. 4, pp. 003–018, 2005.

[2] A. R. Hasan, C. S. Kabir *et al.*, "A study of multiphase flow behavior in vertical wells," *SPE Production Engineering*, vol. 3, no. 02, pp. 263–272, 1988.

[3] L. You and H. Liu, "A two-phase flow and transport model for the cathode of pem fuel cells," *International journal of heat and mass transfer*, vol. 45, no. 11, pp. 2277–2287, 2002.

[4] C. T. C. (editor), *Multiphase Flow Handbook*, 1st ed., ser. Mechanical Engineering Series. CRC Press, 2005.

[5] M. Al-Naser, M. Elshafei, and A. Al-Sarkhi, "Artificial neural network application for multiphase flow patterns detection: A new approach," *Journal of Petroleum Science and Engineering*, vol. 145, pp. 548–564, 2016.

[6] G. Mask, X. Wu, and K. Ling, "An improved model for gas-liquid flow pattern prediction based on machine learning," *Journal of Petroleum Science and Engineering*, vol. 183, p. 106370, 2019.

[7] J. S. Hernandez, C. Valencia, N. Ratkovich, C. F. Torres, and F. Muñoz, "Data driven methodology for model selection in flow pattern prediction," *Heliyon*, vol. 5, no. 11, p. e02718, 2019.

[8] E. Pereyra, C. Torres, R. Mohan, L. Gomez, G. Kouba, and O. Shoham, "A methodology and database to quantify the confidence level of methods for gas–liquid two-phase flow pattern prediction," *Chemical Engineering Research and Design*, vol. 90, no. 4, pp. 507–513, 2012.

[9] O. Shoham, "Flow pattern transition and characterization in gas-liquid two phase flow in inclined pipes," PhD dissertation, Tel Aviv, 1982.

[10] P.-Y. Lin, *Flow regime transitions in horizontal gas-liquid flow.* University of Illinois at Urbana-Champaign, 1985.

[11] G. E. Kouba, "Horizontal slug flow modeling and metering," Tulsa Univ., OK (USA), Tech. Rep., 1986.

[12] S. L. Kokal, *An experimental study of two phase flow in inclined pipes*. University of Calgary, 1987.

[13] N. T. Van Dresar and J. D. Siegwarth, "Near-horizontal two-phase flow patterns of nitrogen and hydrogen at low mass and heat flux," *NASA technical memorandum*, vol. 2001, no. NASA technical memorandum, 2001.

[14] R. J. Wilkens, "Prediction of the flow regime transitions in high pressure, large diameter, inclined multiphase pipelines," Ph.D. dissertation, Ohio University, 1997.

[15] W. Meng, X. T. Chen, G. E. Kouba, S. Cem, J. P. Brill *et al.*, "Experimental study of low-liquid-loading gas-liquid flow in near-horizontal pipes," *SPE Production & Facilities*, vol. 16, no. 04, pp. 240–249, 2001.

[16] R. Manabe, "A comprehensive mechanistic heat transfer model for two-phase flow with high-pressure flow pattern validation." 2002.

[17] C. Mata, E. Pereyra, J. Trallero, and D. Joseph, "Stability of stratified gas–liquid flows," *International journal of multiphase flow*, vol. 28, no. 8, pp. 1249–1268, 2002.

[18] P. Abduvayt, R. Manabe, N. Arihara *et al.*, "Effects of pressure and pipe diameter on gas-liquid two-phase flow behavior in pipelines," in *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, 2003.

[19] P. Abduvayt, N. Arihara, R. Manabe, and K. Ikeda, "Experimental and modeling studies for gas-liquid two-phase flow at high pressure conditions," *Journal of the Japan Petroleum Institute*, vol. 46, no. 2, pp. 111–125, 2003.

[20] N. Omebere-Iyari, B. Azzopardi, and Y. Ladam, "Two-phase flow patterns in large diameter vertical pipes at high pressures," *AIChE journal*, vol. 53, no. 10, pp. 2493–2504, 2007.

[21] M. S. Santos, J. P. Soares, P. H. Abreu, H. Araujo, and J. Santos, "Cross-validation for imbalanced datasets: Avoiding overoptimistic and overfitting approaches [research frontier]," *ieee ComputatioNal iNtelligeNCe magaziNe*, vol. 13, no. 4, pp. 59–76, 2018.

[22] A. M. Quintino, D. L. L. N. da Rocha, R. Fonseca Júnior, and O. M. H. Rodriguez, "Flow pattern transition in pipes using data-driven and physics-informed machine learning," *Journal of Fluids Engineering*, vol. 143, no. 3, p. 031401, 2021.

[23] D. Barnea, "A unified model for predicting flow-pattern transitions for the whole range of pipe inclinations," *International Journal of Multiphase Flow*, vol. 13, no. 1, pp. 1 – 12, 1987.

[24] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," *CoRR*, vol. abs/1907.10902, 2019. [Online]. Available: http://arxiv.org/abs/1907.10902

[25] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, pp. 3146–3154, 2017.

[26] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[27] A. C. Ian Goodfellow, Yoshua Bengio, *Deep Learning*. The MIT Press, 2016.

[28] S. O. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," 2020.

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

[30] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001. [Online]. Available: http://www.jstor.org/stable/2699986

[31] "Xgboost library," https://xgboost.readthedocs.io/en/latest/.

[32] "Lightgbm library," https://lightgbm.readthedocs.io/en/latest/.

[33] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[34] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[35] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[36] "Tab-net pytorch libray." [Online]. Available: https://github.com/dreamquark-ai/tabnet

[37] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[38] J. Mandhane, G. Gregory, and K. Aziz, "A flow pattern map for gas—liquid flow in horizontal pipes," *International journal of multiphase flow*, vol. 1, no. 4, pp. 537–553, 1974.

[39] Y. Taitel and A. E. Dukler, "A model for predicting flow regime transitions in horizontal and near horizontal gas-liquid flow," *AIChE journal*, vol. 22, no. 1, pp. 47–55, 1976.

[40] B. Najafi, K. Ardam, A. Hanušovskỳ, F. Rinaldi, and L. P. M. Colombo, "Machine learning based models for pressure drop estimation of two-phase adiabatic air-water flow in micro-finned tubes: Determination of the most promising dimensionless feature set," *Chemical Engineering Research and Design*, vol. 167, pp. 252–267, 2021.

[41] T. Xie, S. Ghiaasiaan, and S. Karrila, "Artificial neural network approach for flow regime classification in gas–liquid–fiber flows based on frequency domain analysis of pressure signals," *Chemical Engineering Science*, vol. 59, no. 11, pp. 2241–2251, 2004.

[42] X. Fang, Y. Xu, and Z. Zhou, "New correlations of single-phase friction factor for turbulent pipe flow and evaluation of existing single-phase friction factor correlations," *Nuclear Engineering and Design*, vol. 241, no. 3, pp. 897–902, 2011.

[43] Z.-H. Zhou., *Ensemble methods : foundations and algorithms*, ser. Chapman Hall/CRC Machine learning pattern recognition series. Chapman Hall / CRC Press, 2012.

[44] "Historical data science trends on kaggle," https://www.kaggle.com/shivamb/data-science-trends-on-kaggle, last visited on 2021-04-10.

[45] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[46] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in Neurorobotics*, vol. 7, 2013. [Online]. Available: https://doi.org/10.3389/fnbot.2013.00021

[47] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[49] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[50] G. Hinton and S. T. Roweis, "Stochastic neighbor embedding," in *NIPS*, vol. 15. Citeseer, 2002, pp. 833–840.

[51] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *25th annual conference on neural information processing systems (NIPS 2011)*, vol. 24. Neural Information Processing Systems Foundation, 2011.

[52] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *International conference on machine learning*. PMLR, 2013, pp. 115–123.

[53] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers Electrical Engineering*, vol. 40, no. 1, pp. 16 – 28, 2014, 40th-year commemorative issue. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045790613003066

[54] H. Ding, P.-M. Feng, W. Chen, and H. Lin, "Identification of bacteriophage virion proteins by the anova feature selection and analysis," *Molecular BioSystems*, vol. 10, no. 8, pp. 2229–2235, 2014.

[55] N. O. F. Elssied, O. Ibrahim, and A. H. Osman, "A novel feature selection based on one-way anova f-test for e-mail spam classification," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 7, no. 3, pp. 625–638, 2014.

[56] R. Marti, *Handbook of heuristics*. Cham, Switzerland: Springer, 2018.

[57] S. Raschka, "Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack," *The Journal of Open Source Software*, vol. 3, no. 24, Apr. 2018. [Online]. Available: http://joss.theoj.org/papers/10.21105/joss.00638