# Final Report
# Automated Slugging Detection using Machine Learning

**Deirdree A. Polak**
CID - 00973185
Github - dapolak
Email - deirdree.polak18@imperial.ac.uk
Imperial College London
MSc Applied Computational Science & Engineering
ACSE-9 Independent Research Project

31$^{\text{th}}$ August 2019

*Academic Supervisors:*
Prof Olivier Dubrule & Lukas Mosser,
Imperial College London, Exhibition Road, London SW7 2AZ, United Kingdom
{olivier.dubrule, lukas.mosser15}@imperial.ac.uk

*Company Supervisors:*
Dr Meindert Dillen & Peter Kronberger,
Wintershall Dea GmbH, Friedrich-Ebert-Str. 160, 34119 Kassel, Germany
{meindert.dillen, peter.kronberger}@wintershalldea.com

**Abstract**

A digital transformation is revolutionising the oil and gas industry. With the breadth of data collected through the years, it is now possible for operators to draw on this to lower running costs and improve efficiency.

In this project, a Python package was developed, in partnernship with Wintershall Dea, to predict the occurrence of slug flow, which is a significant cause of downtime in production in oil wells.

The slugdetection package is built around five modules, Data Engineering, Slug Labelling, Flow Recognition, Slug Detection and Slug Forecasting, with each module improving on the last.

The Slug Detection module was determined to be the most successful. It is able to predict slug flow up to an hour in advance; achieving a classification accuracy greater than 95%. This was made possible by using feature engineering techniques and supervised machine learning classifiers. This module also labels slug peaks, and the first slug of a slug flow which was a feature specifically requested by Wintershall Dea engineers.

**Keywords:** Oil Well, Flow Classification, Feature Engineering, Slug Flow, Flow Detection, Flow Forecasting

# Contents

# 1 Motivation and Problem Statement

For years Wintershall Dea has recorded pressure and temperature data from their numerous offshore oil wells. The collected data represents a wealth of information that can be utilised to optimise production output and minimise running costs. Combined with Data Sciences know-how, the data collected can be used to learn how to best approach, predict and mitigate oil production issues. In this particular project, slugging has been identified as a major issue that has lead to one specific well to be non-operational for 12 months, resulting in loss of revenue for Wintershall Dea.

A slug flow in an oil well is characterized by a mass of liquid flowing to the surface, followed by large gas pockets. This type of flow is undesirable, as it has heavy consequences in the downstream production facility. In order to honour that data collected and prevent slugs, a project to develop instrumentation to detect and predict flow regimes was put forward. This instrument is to use machine learning techniques to detect ahead of time the likeliness of a slug happening, and to predict, in the future, the flow patterns.

## 1.1 Aim & Objectives

The aim of this project is to produce a Python package with tools that successfully predict the probability of a slugging event 30 minutes to an hour prior to it occurring, using machine learning. Wintershall Dea's engineers would like to have an indication of the likelihood of slug happening within that time frame.

The objectives of the research project were as follows:

1. Structure the data, and label slug occurrences based on well pressure and temperature. Validate slug labels with Wintershall Dea engineers.
2. Train a model to recognize the different types of flow within the data using feature vectors. Tune hyper parameters to maximise the classification score.
3. Train a model to predict slug occurrence in advance. Tune hyper parameters to maximise classification score.
4. Test models to forecast well pressure into the future, using uni-variate time series analysis models.
5. Transfer software built to Wintershall Dea, and make applications available to engineers.

## 1.2 Data Presentation

The data available for the project is continuous and sorted by time stamps. It contains ten years worth of data, with an entry every one minute. The data is from a single offshore well. The data contains values for:

- Well Head Pressure (WHP) in BarG, from the top of the well located on the sea bed
- Down Hole Pressure (DHP) in BarG, from the bottom of the well located underground
- Well Head Temperature (WHT) in Degree Celsius
- Down Hole Temperature (DHT) in Degree Celsius
- Production Choke Opening (WH choke) in percentage. This value indicates whether the choke at the top of the well is opened (100%) or closed (0%).

Note that BarG is gauge pressure, i.e., pressure in bars above ambient or atmospheric pressure.

The data can be visualized below.

# 2 Background Research

In this section an account of the research performed both before and throughout the project is presented. It also highlights the main issues encountered and the solutions considered to resolve them.

## 2.1 Slugging

A slug flow is a multiphase-fluid flow characterized by large amounts of liquid followed by pockets of gas, as shown in in Figure 2. In the production line, the separators and compressor are not able to handle outbursts of gas, and the system's production capacity is reduced. In the case of this project, slugging caused by well geometry only is considered.

In the data streams provided by Wintershall Dea, slugging flow can most easily be identified through the high variations in WHP. The large amounts of liquid coming up to the surface increase the pressure at the

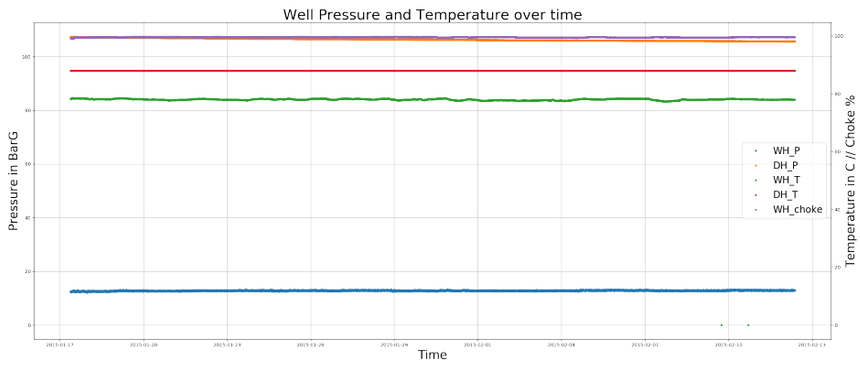



Figure 1: Example data available. The graph represents normal flow, where WHP is blue, DHP is orange, WHT is green, DHT is red and WH choke is purple. This colour scheme will be used throughout this report.

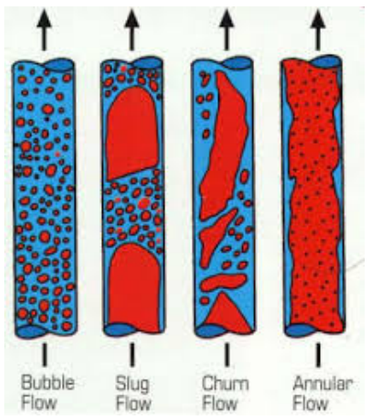



Figure 2: Different Flow Regimes in Vertical Pipes, Red is Gas, Blue is Liquid

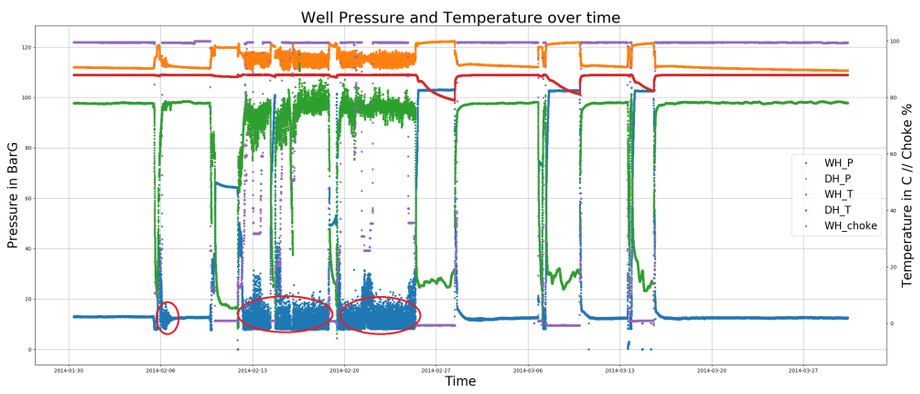



Figure 3: Slug flow as shown in the data, during an interval of 2 months. The slug flows have been circled in red on the WHP data, in blue

well head, which then drops when the liquids have passed through. The loss in pressure leads to more liquids slugging up the well, as the pressure differential becomes sufficient.

Figure 3 demonstrates that the WHP (in blue), DHP (in orange) and WHT (in green) are all indicators of a slug happening. This particular screen shot of data also shows the choke opening (to 100%) and closing (to 0%), and the resulting instability in the different variables. Human interactions, such as these choke adjustments, presented a challenge as they occurred throughout the data set.

Figure 4 shows in detail the trend of each variable during a period of slug flow. The WHP in blue peaks as described, followed by WHT in green and the DHP in orange. The DHT in red has a smaller variance. There are about 1,000 data points for each variable in this specific data screen shot.

Looking at these figures, it shows that smoothing or aggregating the data might obscure trends in the WHP and other variables. Wintershall Dea provided a report that concluded slugs are caused by the terrain and the well geometry, and happen at intervals of 6-15 minutes, thus smoothing of the data could be performed over this time frame.

Through this report slug flow will refer to the continuous slugging of the well over a period of time, whereas

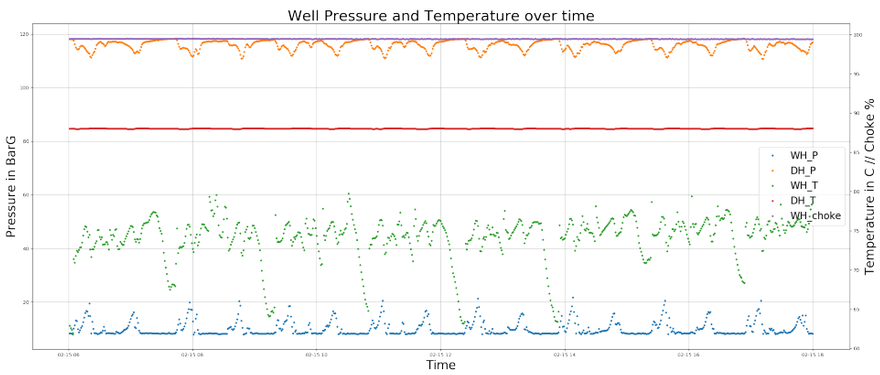



Figure 4: Slug flow as shown in the data, during an interval of 12 hours

slug will refer to individual slugging events shown in the data as a peak in WHP.

## 2.2 Multi-phase Flow Pattern Detection

In literature, research has been performed to categorize flow patterns in horizontal and vertical pipes. In recent research, Guillen-Rondon et al [8] achieved 97% correct classification of three different types of flow using a Support Vector Machine (SVM): Intermittent Flow (like slugs), Dispersed flows (for example with bubbles) and Segregated flows (such as annular flows, where the liquid is flowing solely against the pipe). SVM in this case ignores the time-series aspect of the data and bases the fit on feature vectors. The model does not predict a future slug, but is able to identify a slugging flow from instantanious data. The data used for this research was larger than that available for this project and included flow velocity, viscosity, density, surface tension, inclination angle and pipe diameters.

Further research outlines Artificial Neural Networks (ANN) as tools to determine flow patterns. In particular, E.S. Rosa et al's [14] research recommends using Multi-layer Perceptron networks and Probabilistic Neural networks to categorize the flows. Similarly, M. Al-Naser et al [2] uses a range of Feed Forward neural networks.

Overall, the data available for the research described above is more precise and analytic than what is available on typical offshore wells. The data used is not time dependant but carefully collected variables which give an informative screenshot of the flow. The methods outlined above that use NNs are therefore not directly applicable to this project because of the lack of such qualitative data.

## 2.3 Data Labelling

The first challenge encountered with the data provided is that it lacked labels for any type of flow. This project considers three different methods for labelling the data:

- Internal Labelling, thereby manually labelling the data using a domain expert. The data is labelled with high accuracy, but the process is highly time consuming.
- Unsupervised Learning, allowing machine learning to identify trends and likeliness in the data. The clustered data can then be given labels by the domain expert.
- Data Programming, using a script to label the data under certain conditions (for example, any period with a WHP variance above 4 BarG). This is a quicker method but it can be less accurate.

Other techniques involve using professional data labelling services [1] who specialize in manually labelling thousands of images or data points for data analysis purposes.

## 2.4 Features

### 2.4.1 Feature Vectors

A feature vector is a list of features to be fed into a machine learning model and can be of any size and data type. The feature vector could include all data points for a given period of time, or it can be aggregated over the time period.

Feature vectors were considered in this project as they provide a method of labelling and classifying flows that is independent of time. Using feature vectors allows the use of classification models instead of regression models. Classification is less complex and a more powerful tool, and could yield better results with the corrupted data at hand.

Being time independent allows periods of human intervention in the well operation to be removed from the data set without having a negative impact on the model.

### 2.4.2 Feature Engineering

Feature engineering in machine learning is the process of creating features that work best with machine learning algorithms based on domain knowledge. The data used in this project is unprocessed and continuous, therefore feature engineering is beneficial to condense the data and extract patterns.

Common approaches to feature engineering such as grouping, binning or smoothing of the data were initially considered. However these were found to be ineffective due to the short duration of the slugs, as shown by Figure 4 where the slugs are almost insignificant when viewed at an hourly timescale. A feature that takes into account the slug peaks is the variance. If the variance across short periods of time is binned, it is still be possible to visually identify a slug. Binning is the process of replacing small intervals by a value representative of that interval.

In the context of classification, binary features are beneficial, as they provide a clear divide between data points. One hot encoder could be applicable to the data, and simple binary indicators for an increase or decrease in the gradient of the data streams are also beneficial.

### 2.4.3 Feature Selection

Only a number of well selected features provide useful information for the model. Selecting only the most relevant features reduces over fitting, cuts training time and improves accuracy [10].

Numerous statistical approaches exist for selecting relevant features. A popular approach is the uni-variate selection using a chi-squared ($\chi^2$) statistical test. The $\chi^2$ is calculated between each feature, as seen in equation 1, and the class label, or target. The observed frequency is the occurrence of the feature, and the expected frequency is the occurrence of the class label. The calculated variable will give an indication of the relationship between the feature and the target. The features that yield the highest $\chi^2$ are kept and used for classification. [12] This test is used for categorical labels.This method is also available in the `Scikit Learn` library.

$$\chi^2 = \frac{(Observed\ Frequency - Expected\ Frequency)^2}{Expected\ Frequency} \tag{1}$$

Correlation matrices are another way of selecting the most relevant features. Combined with a heat map, one can clearly see which features are most highly correlated with one another. This matrix is easily computed by using a library such as `Pandas`.

Finally, Tree based classifiers have inbuilt feature importance functions. By running the data through a Decision Tree, the coefficient of utility of each function, based on the GINI Index calculations as explained in section 2.6.3, can be accessed. The coefficients of the full feature sets add up to one. In their research paper, Fulcher and Jones [6] created thousands of features from simple time series data. They selected the most relevant features using greedy forward feature selection and were able to successfully label different wavelets within their data sets, such as "normal", "cyclic", or "increasing".

## 2.5 Unsupervised Learning Classification Models

As mentioned previously, unsupervised classification models are an interesting approach to labelling the data. Similarly to Fulcher and Jones, [6], by using features instead of time series data, the models could cluster the different types of pressure and temperature patterns together, for example as slugs or normal flows.

K-Means is a popular approach to unsupervised learning. The inputs to this algorithm are the number of clusters required and the data set. The algorithm functions by iteratively refining the clusters by creating cluster centroids.

The centroids are defined randomly during the first iteration. At each step, each data point is assigned to its closest centroid, based on the euclidean distance of each of its features. The centroids' locations are then recomputed based on the mean distance to the centroid of all the data points assigned. This is performed until no data point changes cluster anymore and the sum of the distances is minimized[5].

## 2.6 Supervised Learning Classification Models

In order to recognize the different flow patterns using feature vectors, supervised learning classifiers can be considered.

### 2.6.1 Logistic Regression

Logistic Regression is a popular classifier [3], which returns its prediction as a set of probabilities of each input belonging to a certain class. Logistic Regression training works as follows:

1. The estimate of the label is calculated as shown in equation 2. The results are then transformed using a sigmoid function.
$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \tag{2}$$
Where $h(x)$ is the estimate, $x_{1,2,..,n}$ are the features and $\theta_{0,1,2,...,n}$ are the corresponding weights
2. The weights are updated iteratively by calculating the derivative of the cost function and using gradient descent at each step to find the local minima.
3. The training stops once the cost function returns a small enough residual value.
$$Cost(h(x), y) = -ylog(h(x)) - (1 - y)log(1 - h(x)) \tag{3}$$
where $y$ is the true value of the prediction

Logistic Regression is preferred as a classifier as it gives its results as probabilities.

The main parameter that can be used to tune a Logistic Regression model is the regularisation parameter. In `Scikit Learn`, this is the parameter C, which is the inverse of the regularisation strength. Regularisation can be added as bias when the variance of the model is very high. With small values of C, the regularisation strength has been increased which will lead to simple models likely to under fit the data. For large values of C, the regularisation strength is low which implies the the model has a high complexity and is likely to overfit the data.

### 2.6.2 Support Vector Machines

Another type of Classifier is the Support Vector Machine, or Support Vector Classifier. In short, SVM tries to find the optimised hyper plane that separates two or more classes. The algorithm finds the points within each class that are closest to each other. These points are called support vectors. It then computes a plane that is equi-distant from the two points, so that the margin between the plane and the classes is optimised [4], [11]..

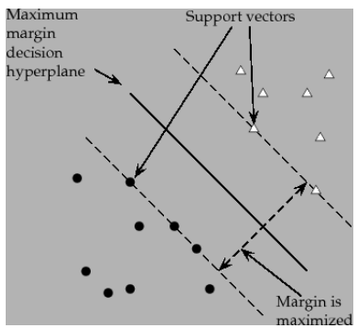



Figure 5: SVM hyper plane [11]. The support vectors are the 5 points right up against the margin of the classifier.

When the data is not linearly separable, SVM has a kernel function hyper parameter that can be updated to transform the data. For example, the kernel function could be linear, polynomial or a radial basis function. The influence factor($\gamma$) which is the coefficient of the kernel function can also be used as a hyper parameter.

### 2.6.3 Decision Trees and Random Forests

A decision tree is a non-parametric supervised learning classification model. During training, a decision tree splits the data based on a condition on one feature, and creates small sub data sets that can further be split based on other conditions. By the end, a decision tree has been designed so that based on the consecutive conditions, it can classify new data points. A sample decision tree is in Figure 6.

The conditional nodes are referred to as decision nodes and the end classifying nodes are called leaf nodes. Each splitting or branch creates a sub tree.

The splitting method considered in this project is based on the GINI Index. This index is a measure of the probability of a feature to be incorrectly identified. When the GINI is high for a node, it means the condition leading to this node was a poor indicator of the two classifications [13].

For example, when creating a Decision Tree for different flow types in the well data, splitting the data based on whether WHP is less than or greater than BarG 0, gives a high GINI. This is because since WHP must physically always be greater than 0, it won't be indicative of any changes. However, splitting the data on

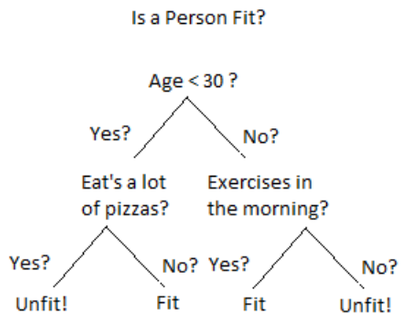



Figure 6: Example decision Tree from medium.com

whether the choke is opened or closed, yields a low GINI index, as it successfully splits the data into choked flow and other flow.

A single tree will create splits on different conditions, and thanks to the GINI index, it will be able to compare the splits and select the one that holds the most relevant results.

A Random Forest is a model which contains multiple Decision Trees. In a random forest, each decision tree is given a different number of features, and a different range of data points. Each tree is then developed and trained. The prediction of the Random Forest comes from the aggregation of all the individual predictions by each Decision Tree. This ensures cross validation within the model, and avoids over fitting.

A Random Forest is also a highly tuneable model, for example by adjusting the number of trees, the depth of the trees, pruning, or by limiting the number of features/data points to be dropped.

## 2.7 Time Series Models

The data provided by Wintershall Dea is continuous over 10 years and therefore a time series was initially considered. However the large number of human interventions through the year made this impractical. Each movement of the WH_choke causes a perturbance in temperature and pressure throughout the well. For this reason, time series modelling was not considered for slug prediction, as the human interactions are not possible to predict.

On the other hand, times series modelling can be used when trained exclusively on known, continuous and independent slug flows by using WHP only. This would be value added for Wintershall Dea, as the forecasting could give them an indication of the amplitude of the slugs. Moreover, once enough data was gathered, the model could also be trained on forecasting WHP when manual slug control is applied, by choking the well or using gas lift.

One model was primarily considered here, due to time constraint : ARIMA

### 2.7.1 ARIMA

ARIMA stands for Auto-Regressive Integrated Moving Average models. It works best for uni-variate forecasting, WHP in the context of this project.

**Parameters**

An ARIMA model requires three parameters [9]:

- `p` - The Auto-Regressive (AR) parameter. It decides on how many past terms, also called lagged observations, are going to be used in the prediction. A weight is applied to each past term.
- `d` - The Integrated (I) parameter, where d represents the degree of differencing. The differencing eliminates the seasonality of the data and ensures that the data is stationary.
- `q` - The Moving Average (MA) parameter which refers to the number of lagged residual errors of a moving average model that should be included. It removes errors and noise in the fit. MA have a fixed window and weights relative to time,

These parameters can be found by doing a Grid search. It is however common to set $p$ to the values out of the confidence range in the Partial Auto Correlation figure and to set $q$ to the values out of the confidence range in the Auto Correlation figure. On the other hand $d$ is usually found by calculating how much differencing of the data is required to make it stationary. This will be discussed in more details in the Software Description Section 4.5.1.

**Stationarity**

The data must be proven stationary in order to start fitting an ARIMA model. A data set is said to be stationary if the mean, variance, auto-correlation are constant over time. An Augmented Dickey Fuller (ADF) test can be used for testing stationarity. The null hypothesis of the ADF test states that the data is non-stationary . The null hypothesis can be rejected if the p-value is lower than the significance level (5%), and it can be inferred that the data is stationary. Furthermore, the more negative the value of the ADF test statistics, the stronger the rejection of the hypothesis [7].

# 3 Software Development Life Cycle

The software developed is contained in the slugdetection package. It is a standalone package to be used for data analysis of offshore oil wells that exhibit slugging flows. It requires Spark, Pandas, Scikit Learn and Stats Model libraries.

## 3.1 Development & Operation Tools

The code for this project was prototyped on DataBricks, a Unified Analytics Platform, optimised to work with Microsoft Azure Cloud and Apache Spark, an open-source distributed computing framework. Since confidential company information was present on the main notebook at use, most of the code was developed within DataBricks. The plateform has its own version control and control management system with a Revision History available.

Unit tests were consistently developed and improved to ensure the classes and their methods were not altered, and performed as expected. The tests were run whenever changes to the code were implemented. The modules once completed were moved to a new notebook and imported to the working notebook.

Once the prototyping was complete and sensitive names and data were removed, Github was used as the source Control Management System for the project. The original software written in the notebooks was converted to scripts within a package and moved to Git. The package was then accessed from the DataBricks notebook.

## 3.2 Development Methodology

The management methodology used for this project is the SCRUM framework. The project was performed with close feedback to the company supervisors, holding two meetings a week when possible. The agenda for each meeting was to give an update on the work done since the last meeting, explain what activities were planned until the next, and discuss any issue currently being faced.

In accordance with the SCRUM framework, each task was aimed to be completed within two week long sprints.

## 3.3 Run Time Considerations

The code was run and developed on DataBricks notebooks, optimised for Spark processes. DataBricks notebook are run on clusters consisting of minimum one driver node and one worker node. The driver node keeps state information of the notebook, and interprets all the commands run from the notebook. The worker node distributes and runs Spark jobs. Adding worker nodes could imporve run time, however the clusters were limited by the company.

For this project, when Spark jobs were started the run time was automatically optimised. For the other tasks performed using other libraries than Spark, the run time was optimised as much as possible for performing short actions. However, the cost of further optimising the code was more than the cost of the company running the nodes, and it was therefore decided to prioritise building the package before optimising the run time. Further optmisation actions can be performed in the future development of the package.

## 3.4 Validation and Verification

Software verification was performed by creating unit tests for each and every method created as part of the package. The code also conforms to PEP8 standards. The name of the variables were kept simple and clear, so that any user within Wintershall Dea with a background in production engineering would be able to understand the steps taken through the analysis.

Results validation was performed by sharing results with Wintershall Dea's engineers and project supervisors, through the development process. As domain experts, they were able to give valuable insight on the accuracy of the data classification. As far as the data labelling is concerned, there are no metrics to calculate their accuracy, so the results depended heavily on the engineers insight and validation. In the context of data classification, confusion matrix were used to quantify and visualise the errors. In the context of data regression, different error metrics such as Mean Absolute Percentage Error were used, along with infographics. This is explained in more depth in Section 4.5.

### 3.5 Package Creation

The package was created on github directly and instantiated in the Databricks notebook. This in itself was a challenge because DataBricks is a self contained platform that does not allow for many external inputs.

From the package setup tools, distribution archives were computed, and the wheel built distribution file can be uploaded directly to DataBricks as a local library. The library can then be linked to a notebook or attached to a cluster.

Due to company ownership restrictions, the package has not been distributed to the Python Package Index (PyPi) as of yet.

## 4 Software Description

A comprehensive data engineering and analysis package named "slugdetection" has been developed during the course of this project to be used by employees of Wintershall Dea.

It is split into five modules in order to make it more accessible for employees with little knowledge of both the well data, and machine learning. Additionally, a Python notebook has been designed, walking the user through the analysis steps, and providing insight into the methods available.

The modules included in the package are:

- **Data Engineering**: Contains basic data engineering and data cleaning tools.
- **Slug Labelling**: Labels slugs using an unsupervised clustering approach,
- **Flow Recognition**: Labels individual data points, groups data into feature vectors and categorises different flow types in the data using supervised classification
- **Slug Detection**, Labels time intervals leading to slug flow and recognises slug flows one hour before occurrence, using feature engineering and supervised classification,
- **Slug Forecasting**, forecasts slug flow one to two hours in the future, using time series modelling.

Whilst Data Engineering uses the data in Spark data frame format, all four classes require the data to have been converted to a Pandas data frame. The Slug Labelling, Flow Recognition and Slug detection classes are all inherited from Data Engineering.

The analysis modules were developed consecutively, and each benefited from the successes and failures of the previous ones. As such, the Slug Detection and Slug Forecasting methods performed best, each attaining over 90% accuracy.

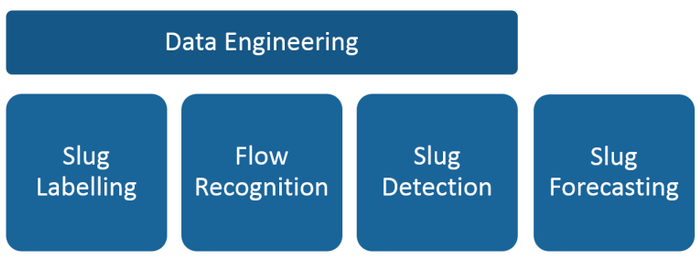



Figure 7: Package structure. Slug labelling, Flow Recognition and Slug Detection are inherited Data Engineering classes (See 4.1 Data Engineering). Slug Forecasting is a standalone class.

### 4.1 Data Engineering

In this section, the Data Engineering module is presented, as well as the independent data cleaning actions required prior to analysis. The pre processing and Data Engineerng methods include joining the .csv files together, assembling the Pyspark data frame, defining data thresholds and cropping the data.

#### 4.1.1 Raw Data Processing

The data available consists of several comma separated value or .csv files. It contains the values of the variables WHP, DHP, WHT, DHT and of the WH choke. For each stream of data, there are between 60 and 64 .csv files provided. This is the format from the company database, where only 80-100 000 lines of data can be provided at a time.

The .csv files were downloaded locally, and joined into one .csv files per stream (ie. WHP, DHP, WHT, DHT, WH choke). The script for this operation is available at `read_files.py`.

#### 4.1.2 Pyspark Loading and Joining

The different data tables are loaded directly into DataBricks' database and converted to a Pyspark table. These tables can be accessed from the notebook directly, and are instantiated in the notebook using Pyspark. The steps to join and load the data are presented in the Python notebook `IRP_slugdetection_2019`.

Each .csv table has two columns: a time stamp column, and a value column which contains the sensor reading. The time stamp column is abridged from `"dd MMM-yy HH:mm:ss.SS"` format that includes the seconds and milliseconds to `"dd MMM-yy HH:mm"` format. This small alteration was performed because the raw data was not downloaded using the same seconds and millisecond stamp. Shortening it to minutes allows to join the different tables together on the minute, using a full join.

The full join ensures that no data is lost even if the time frame of each variable is different. For example WHP data is provided from 04 March 2009, and DHP data is provided from 09 March 2009. Using the full join, all the rows were kept and unknown variable were given a None value.

#### 4.1.3 Data Engineering class

Once the full data spark data frame is created, the Data Engineering class can be instantiated. The class is able to crop the data between dates, threshold the data, clean the data from choked flow, convert the Spark data frame to a Pandas data frame and plot the data between specified dates.

### 4.2 Slug Labelling - Unsupervised Learning

Slug Labelling using unsupervised classification was the first module implemented. The aim was to create feature vectors from windows of time in the continuous data and run a clustering algorithm to group like flows together. Five flows with recognizable patterns were put down as objectives for each cluster:

- Choked Flow (where WH choke is shut)
- Normal Flow (where variations in WHP are small)
- Slug Flow (where variations in WHP are high, and regular)
- Slug after Choke, in transition (where the variations in WHP are high, uneven, and the choke is changing)
- Slug formation, in transition (where the variations in WHP are increasing)

This approach would serve as an indicator of the types of flows, and potentially allow to label the slugs in the data.

#### 4.2.1 Data Pre-Processing

This method required some pre-processing of the data by transforming it into feature vectors to be used for clustering.

A generic function was written to collect the data and build feature vectors, storing them in a copy of the original data frame. New columns are added depending on the size of the time window. A window of time is then swept across the full continuous data frame and the data from each variable within the window of time is added to the feature vector data frame. For a 5 minute time window, the feature vector includes the data for WHP, DHP, WHT, DHT and WH choke at the current time, at -1 minute, -2 minutes, -3 minutes, -4 minutes and -5 minutes. The data can be standardised through normalisation.

Three window of times were tested to label the slugs: 5, 20 and 40 minutes. The windows are overlapping, and the overlap value can be set by the user.

#### 4.2.2 Unsupervised Learning Model

The feature vectors are then used in the KMeans unsupervised clustering model. The number of clusters required can be set the user.

The results of the clustering are presented by plotting the same label windows on the same plot, overlaid. It is then up to the user to give a name to the label of the flow depicted. The labelling requires domain knowledge of slugging and other well flows.

### 4.3 Flow Recognition - Supervised Learning

The second data analysis module implemented was created to recognize the different types of flow in the continuous data. Again, sweeping windows of times was used and the data from each window was labelled using

a sorting code. Supervised learning models were implemented and tested such as Logistic Regression and Support Vector Classification (SVM). The models were compared by performance and prediction accuracy.

The Slug Labelling method labels were not used for this approach. As dicussed in the Results section, the results were inconclusive in labelling slug and normal flows. After attempting unsupervised learning, this approach was deemed more promising as the supervised model could be forced to learn the desired patterns in the data. A risk with this approach is the bias of manually labelling the data. Slugs can visually be identified, however in the data, there are no indicators for a slug flow, or its preceding flow. This entails that the labels will not represent the absolute true flows occurring, rather the interpretation of the engineers.

### 4.3.1 Data Pre-processing

**Data Labelling**

The labelling for the data was straight forward. It was performed point by point, based on WHP, the gradient of WHP and the WH choke. A user defined value of the gradient WHP is defined as the value at which a slug is occurring. The default values were set to 3 BarG, based on discussions with Wintershall Dea's engineers and visual observation.

The individual point labels are:

- "Slug" or 0, where the WHP gradient is larger than 3 BarG or user defined value
- "First Slug" or 1, where a labelled slug is the first slug to occur within a 30 minutes interval, or user defined value
- "Pre-Slug" or 2, the 60 data points prior to the first slug, or user defined value
- "Normal" or 3, points where choke is constant and WHP gradient is less than 3 BarG or user defined value
- "Ignore" or 4, all other points, including choked flow and choke transition

Once a label was computed for each point, labels for the window of times and thus the feature vector was extrapolated based on the number of occurrences of the different labels.

The labelling was not updated as the results yields were low, and instead a new approach to directly predict the slugs was engineered.

**Feature Vectors**

Similarly to the Unsupervised Approach, feature vectors were created using 20 and 40 minutes windows. The data was normalized prior to the creation of the feature vectors.

### 4.3.2 Supervised Learning Model

**Data Splitting**

The feature vectors were split into two groups: a training set and a validation set containing 70% and 30% of the data respectively. The `StratifiedShuffleSplit` method by `Scikit Learn` was used.

**Modelling**

The approach to flow recognition only required a simple classification algorithm. For best practice purposes, two models were run and compared: Logistic Regression and Support Vector Classification.

Each model was implemented in two steps, in two methods in the class, one to train the model using attribute training data, and the other one to predict labels. This predicting function uses the attribute test data by default, but new data can also be used.

The results were visualised by building a confusion matrix of the results and true labels.

The approach taken, recognising different flows using Supervised Learning, was inefficient. This may have been due to five labels being too many, and the model being overwhelmed by the large amount of data.

## 4.4 Slug Detection - Feature Engineering and Supervised Learning

In the third data analysis module, Slug Detection is performed directly as opposed to flow recognition. The aim of this method is to be able to successfully predict one hour in advance whether a slug flow is about to happen. This is added value to Wintershall Dea as the engineers would now have an indicator showing the probability of a slug occurring in the next hour.

In this approach, only two labels are considered:

- Interval leading to a slug flow (True)
- Interval leading to a normal flow (False)

The length of the interval is decided by the user, and its default is set to 240 data points, equivalent to four hours. The interval directly after the last data point must either be a normal flow or a slug flow. In this approach, slug, choked or choke transition flows are completely ignored.

The time window was increased from 20-40 minutes to four hours, because it was believed from visualizing the data that this time interval was more telling of the flow characteristics. By looking at the variables' behaviour over a four hours window, the aim is to identify the flow type leading to a slug. From visual inspection of the data available, there are two ways slugs seem to occur:

- a slow increase in the frequency and amplitude of the WHP and other variables, from normal flow
- a decrease in the WHP and other variables amplitude, from an unstable post-choke flow.

This can be visualized more clearly in the Results section of this report.

A Random Forest model was then used on intervals of 240 data points, to predict whether they lead to a slug or not. The Random Forest ranks the features, and the 50 most relevant features are then used in a Logistic Regression to classify the short data sets one more time. Logistic Regression is preferred for application purposes since it returns a probability.

### 4.4.1 Data Pre-Processing

The first step in pre-processing the data for this approach was to clean the data from any WH choke value lower than 99%. The remaining data is a mix of continuous slates with one minute intervals, and jumps where the choke was less 99%. To deal with this problem in time and trend continuity, the data frame was split into smaller and continuous data frames. A "jump" column was added where the time difference between two rows is larger than one minute. The chunks of continuous data are given a unique index number. Any chunk of data containing less than 240 data points was discarded.

The continuous slates of data that were longer than 240 were added as individual data frame to a dictionary attribute.

**Data Labelling**

The labelling of the individual data frames is performed in three steps: first, all the slugs in the data are labelled, second, the first slugs are labelled, and third, the data frames are cut into smaller 300 points long data frames, labelled as True or False depending on whether the point immediately following is the first slug of a slug flow.

The labelling of the individual slugs was refined in this approach:

1. The gradient of WHP is calculated at each data point
2. The trend is identified as increasing (True) or decreasing (False)
3. The consecutive increasing values are grouped and given unique indices. The same is performed for the consecutive decreasing values.
4. The total WHP differential is calculated for each trend. This is performed in case the increase in WHP due to slugging occurs over several minutes (data points) as opposed to one minute only, as assumed in the previous supervised approach. For example, if WHP increases over five minutes, the cumulative gradient is calculated for this increase. Similarly if the WHP decreases for three minutes, the cumulative decrease is computed.
5. Finally a point label is given. The data point is labelled as a slug (True) if the cumulative total is larger than 4 BarG and if the next data point has a decreasing trend, otherwise it is labelled as False. The WHP difference threshold can be user defined.

The labelling was then verified with engineers at Wintershall Dea. This is believed to be the best slug indicator developed, as it is robust and logical. The labelling of the first slug, explained below, is also robust and ensures the slugs are regular and not just instabilities.

The list of indexes where the slug flow peaks are occurring are saved and passed through another method to determine the occurrence of first slugs.

The method for first slugs determination checks for four requirements before saving the slug index as a first slug:

- That there's a least 240 points between the current slug index and the previous first slug. This is to ensure that there is enough data to then be used when the data frames are cropped to 240 row long tables. If a first slug occurs within 240 points of the beginning of the data frame, it is still marked, but the data interval will not be used for training the model.
- That this is the first slug to be identified in the past 60 points, or past hour. As exemplified in Figure 8, without this condition a new first slug would be identified every 240 minutes.

- That the current and next two slugs occur within 20 minutes of each other. This value is derived from a report provided by Wintershall Dea, where slugs in a slug flow are normally occurring within 6-15 minutes of each other.
- Finally, that the current and next two slugs have a WHP value within 2 BarG of each other. this insures that the flow is indeed a slug flow, and not only instability in the flow, which could be caused post-choking.

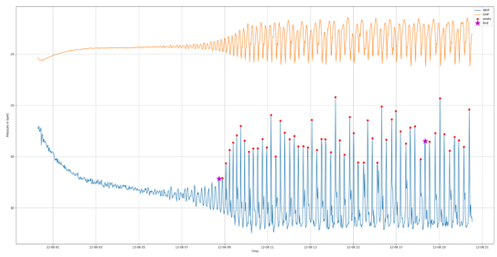


Figure 8: Example of double labelling a first slug in the data. This shows the need for a non-continuous slugging condition. The slug peaks are labelled in red on the WHP, and the first slugs are labelled in magenta.

Once the first slugs indexes are derived, the individual data frames are cropped to size 240, which corresponds to four consecutive hours. For data frames where no first slugs have been identified, the flow throughout is considered as normal, and therefore at any point in that data frame, the previous points are leading to a normal flow. The data frame is split evenly into sub data frames of 240 points. The frames are allowed to overlap.

For data frames containing first slugs, the 240 points before the first slugs are selected and labeled as True: leading to a slug flow. If there aren't enough points before the first slug, it is ignored. The logic of sorting each clean data frame is shown in Figure 9.

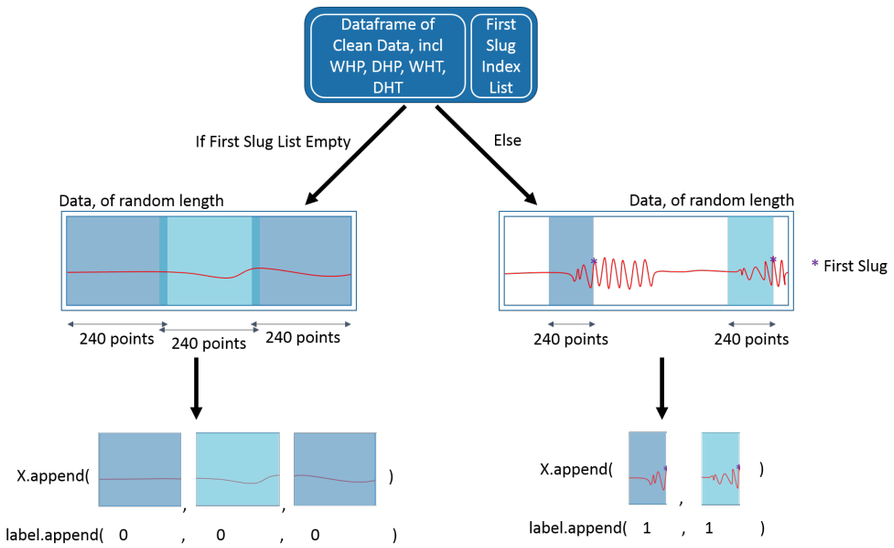



Figure 9: Splitting of the data in same sized data frame, and labelling

It is clear from the design that there will be more normal flows than flows leading to a slug in the final list of data windows. An argument was set in place to give the user the choice of restricting the number of normal flows in the final data.

**Feature Engineering**

This supervised learning uses windows in time of four hours, or 240 data points. Using the same feature vector as in the previous section would lead to an enormous vector. As the quality of the results and analysis have been low, a new feature vector was designed instead.

Visually, from observing five hours worth of data at a time, there are two trends leading to a slug. WHP instability caused by reopening a choke that settles into a slug, or normal flow with constant WHP that gradually becomes more and more unstable. Looking at both the rolling average of the standardized mean and at the standard deviation has the potential to show this increase over time. More precisely looking at the mean and standard deviation differential over time should clearly show the trend in a feature vector.

From the four hours of data in each data frame, the last hour was ignored. This is to train the model on predicting slugs one hour before they occur. The remaining three hours were arbitrarily split into five sections, from which the mean and standard deviation were calculated. Figure 10 is a diagram explaining how the mean and standard deviation were procured.

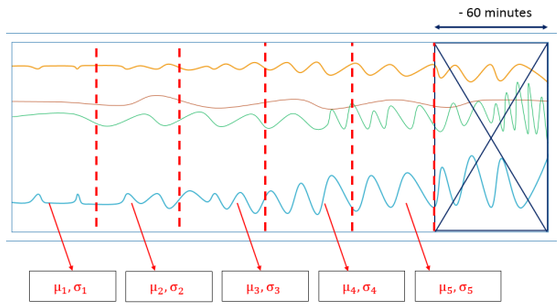



Figure 10: Aggregating the data variables into five groups

Furthermore, binary features were also considered because of how well they work with binary classifiers such as Random Forest and Logistic Regression. Features for the trend (increasing/decreasing) and for the difference significance (large difference/low difference) for both the mean and variance were created, as well as the count of significant increases, and the count of significant decreases.

The full list of features within the feature vector is shown below. Note that WH choke data is no longer used since all values below 99% were cropped out. In total, there are 152 features.

Table 1: Features developed for each variable. Here only WHP is shown, but the features are repeated for DHP, WHT and DHT.

| | | | |
|---|---|---|---|
| WHP | Section 1 | Mean | Float |
| | | Standard Deviation | Float |
| | Section 2 | Mean | Float |
| | | Standard Deviation | Float |
| | Section 3 | Mean | Float |
| | | Standard Deviation | Float |
| | Section 4 | Mean | Float |
| | | Standard Deviation | Float |
| | Section 4 | Mean | Float |
| | | Standard Deviation | Float |
| | Section 1 - 2 | Mean Difference | Float |
| | | STD Difference | Float |
| | | Binary Trend Indicator | 0 - Increase<br>1 - Decrease |
| | | Binary Significance Indicator | 0 - Increase<br>1 - Decrease |
| | Section 2- 3 | ... | ... |
| | Section 3 -4 | ... | ... |
| | Section 4 - 5 | ... | ... |
| | | Significant Increases Count | Int |
| | | Significant Decreases Count | Int |

### 4.4.2 Supervised Learning Model

**Data Splitting**

A Stratified Shuffle Split was used again for data splitting, with a default 70%-30% training to validation split. These values can be adjusted by the user.

**Modelling**

The model required for this analysis is a binary classifier. As mentioned in the objectives, it is an added benefit for the company to get a probability of a first slug occuring in the next hour, and therefore Logistic Regression is preferred. However, there are 152 features for each data point and feature selection would be beneficial to maximise the performance of Logistic Regressions.

As mentioned in the Background Research, there are multiple readily available tools for feature selection. In the interest of time however, it was decided to train a Random Forest model on the data, and extract from it the Feature Importance matrix. The most important features could then be used to train the Logistic Regression model.

The top 50 features were selected from the Random Forest model, and used to train a Logistic Regression model. This switch in model was decided both because the Logistic Regression returns probabilities which are of more value to the production engineer i.e. how likely is it that a slug is happening in one hour, and because the model is more transparent and communicable to the future users.

## 4.5 Slug Forecasting - Time Series Modelling

The fourth data analysis module implemented aims to forecast slugs into the future. It is the only analysis class that is not inherited from the data Engineering class. The Slug Forecasting class requires a Pandas DataFrame with at least WHP and time stamp columns to instantiate. The data should only contain a slugging interval, free of human interactions.

In order to forecast slugs, an ARIMA model was implemented to be trained on slug flows lasting at least four hours. This time series model was considered because it requires univariate data, and works best with a repetitive, sinusoidal like trend. It is a good fit for forecasting WHP trend during slug flow.

### 4.5.1 Parameters Selection

As discussed in the Background Resaerch section, ARIMA requires three parameters to fit the data: $p$, $d$ and $q$. Whilst there are some packages such as `pmdarima` that can gird search for the best parameters, it was thought preferable for the end-user comprehension to implement the model step by step.

The first step to ensure that the ARIMA model is correctly implemented with the data available is to ensure the data is stationary. For this, an Augmented Dickey-Fuller test was implemented, returning the test and the p-value. If the p-value is larger than 0.05 and the null hypothesis is accepted, the data is not stationary and differencing should be applied. The user can then repeat the test with different values for $d$, until the data is stationary.

Note that from domain knowledge, the assumption is made that the WHP does not have a trend. It is therefore also assumed that differencing is enough to make the data stationary. For other more complex data sets, additional transformations will be required to make the data stationary.

Moving on to deciding on the $p$ and $q$ parameters, the user can display the Auto Correlation and Partial Auto Correlation plots.

From the partial auto correlation figure plot (PACF), the AR or $p$ parameter can be approximated. 'Auto Regressive' means the regression model uses its own lags as predictor, and therefore the partial auto correlation gives a good indication. If there is a significant correlation between the series and its lag, the value of the lag should be used as the $p$ parameter. The partial correlation will be looked at for the first 25 lags because that range allows for the correlation to another slug to be shown.

From the auto correlation figure plot (ACF), the MA or $q$ component can be approximated. The "Moving Average" term tells the number of lagged forecasts error that should go into the model. The ACF plot tells how many MA terms are required to remove any autocorrelation. Again 25 lags will be considered as it should show the auto correlation between at least two slugs. If there is a significant correlation between the series, the value of the lag should be used as the $q$ parameter.

Examples of ACF and PACF plots are shown in Figure 19. The process of choosing the parameters is explained and stepped through in the Python notebook `IRP_slugdetection_2019`

### 4.5.2 Model Fitting and Error Metrics

Once the parameters are approximated, the model can be fitted. The user only has to input the three parameters, and the real data will be shown along with the fitted curve.

Error metrics can be printed and viewed. They are:

- Mean Absolute Percentage Error: MAPE gives a statistical measure of how accurate a regression is. In general, a small MAPE value means an accurate forecast.

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

- Mean Squared Error: Assesses the quality of a forecast. A low value indicates a good prediction.

$$MSE = \frac{1}{n}\sum_{t=1}^{n}\left(A_t - F_t\right)^2$$

- Root Mean Squared Error: RMSE is the standard deviation of the residuals, and thus gives an indication of how spread out the errors are. A small RMSE is preferable.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}\left(A_t - F_t\right)^2}{n}}$$

- R2 Coefficient: illustrates the strength of the relationship between the true and the forecast values. The closer it is to 1, the stronger the relationship, and presumably the better the fit. It is calculated by computing the total sum of squares in the data and the sum of squares of the residuals:

$$R^2 = 1 - \frac{SS_{residual}}{SS_{total}} = 1 - \frac{\sum_{t=1}^{n}\left(A_t - F_t\right)^2}{\sum_{t=1}^{n}\left(A_t - \mu\right)^2}$$

Where $n$ is the number of samples, $A_t$ is the actual value at sample $t$, $F_t$ is the forecast value at sample $t$ and $\mu$ is the mean of the data.

Moreover, the residuals and errors can be visualised, as seen in Figure 11. The error plot method was implemented to give users with only a basic knowledge of statistics an easy tool to refine the model. Each graphic has its significance:

- Residuals plot: Shows the range of the the residuals. In this example it can be seen that the residuals have a low amplitude, however, they appear to have a repetitive pattern.
- R2 plot: shows the forecasted values against the real values. The values should lie closely to the red line, which indicates a perfect correlation. Here it can be seen that the forecasted values seem to be systemically lower than the true values, as shown by the higher concentration of points above the red line.
- ACF plot of the residuals: Ideally, residuals should not be correlated and the lagged correlation should remain within the confidence interval. In the example, the residuals are correlated at lag 12.
- Density plot: shows the density distribution of the residuals as compared to the normal distribution. The histogram is the actual distribution, whereas the orange line is the kernel estimated density function(kse). If the fit is good, the kse should closely resemble the normal distribution. In this case, it can be seen that the mean of the residuals is offset, which evidences a bad fit. Differencing could be a solution to reduce the error margin of this set.

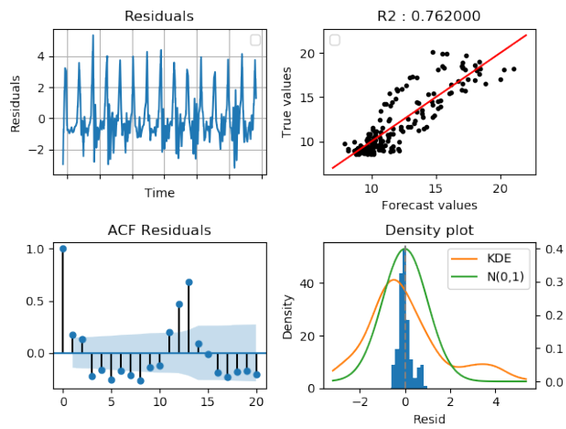



Figure 11: Example Error Plot

### 4.5.3 Slug Forecasting

Similarly, a function is in place to forecast slugs in the future. There is no limit to the time prediction, and the accuracy should technically stay the same if the slug flow pattern remains unchanged. The prediction time is at the discretion of the user. The same error metrics are available to calculate the accuracy of the model. If the accuracy is low, the model should be re-fitted with new parameters

# 5 Software Meta Data

| | **Name** | **Version** |
|---|---|---|
| **Programming Language** | Python | 3.7 |
| **Required Libraries** | Numpy | 1.17 |
| | Pandas | 0.25 |
| | Matplotlib | 2.2.0 |
| | Pyspark | 2.4.3 |
| | Scikit Learn | 0.21.3 |
| | StatsModel | 0.10.1 |
| **Proprietary Software Requirements** | None | |
| **License** | MIT | |
| **Operating System** | Any | |
| **Databricks Requirements** | Cluster : 5.3 ML (includes Apache Spark 2.4.0, Scala 2.11)<br>Driver Node: Standard_DS4_v2, 28.0 GB Memory, 8 Cores<br>Worker Node: Standard_F4s, 8.0 GB Memory, 4 Cores | |
| **if run locally Hardware Requirements** | RAM: 3.7 GB recommended | |

The code (version 1.0) is available at :

`https://github.com/msc-acse/acse-9-independent-research-project-dapolak`

Documentation can be found at:

`https://dapolak.github.io/acse-9-independent-research-project-dapolak/`

The package was created to be used from a DataBricks platform, Python notebook or Python script

# 6 Software Implementation and Results

In this section the code implementation, tuning and results for each of the modules are discussed.

## 6.1 Data Engineering Implementation

The Data Engineering class was implemented as an effort to clean the data. The data collection depends entirely on real sensors, positioned at the head and bottom hole of the well. Erroneous values have occasionally been recorded where these sensors have failed. For example, nonphysical values such as -814 BarG for the DHP have been recorded.

Two strategies addressing this issue were considered: Cropping the data at a certain value, or replacing the corrupted values outside of threshold values by the previous value. Both methods require thresholds for each of the variables outside of which, the data needs to be replaced.

The relevance of these two techniques was dependent on the analysis taking place. As explained in Sections 4.2 and 4.3 dealing with slug labelling and flow recognition, out of range values will heavily corrupt the analysis, as the actual sensor values are directly used. It is therefore required that the values be within range and do not suddenly jump. Hence, replacing the corrupted value by the last in range value is the preferred solution. As far as Section 4.4, which focuses on slug predicting is concerned, the values are not as important, as the mean and standard deviation are used for training the model and predicting, instead of raw values. As for the slug forecasting model described in Section 4.5, out of range values are not a relevant issue, as only a so called clean run of selected WHP are used.

The threshold values for both techniques were developed along with the company engineer's knowledge. Other techniques to interpolate the thresholds were experimented but were not reliable, such as detecting outlier data points with the standard deviation, or using a percentile from the data mean value. As shown in the data, WHT jumps from a range of 10 to 20 C when the choke is opened, and to 70-80 C when it is closed. Defining outliers as 50% above and below the mean is inefficient, as it will crop the data to approximately 10 and 40 C, thus missing the WHT information when the choke is closed.

The thresholds were saved as class attributes in a dictionary. In order to apply the thresholds using Pyspark, new "lag" columns were created, and when the data didn't fall within the threshold values, it was replaced by the value in the "lag" column. This is briefly illustrated below:

```
well_df = well_df.withColumn(WHP,
F.when( F.col(WHP) < tresh[0],
F.col(WHP_lag_0)).otherwise(F.col(WHP)))
```

A problem arose where some sensor defects occurred over multiple time steps. A normal approach would be to loop through the data frame at hand, and replace every out of range value with the previous one. However Pyspark does not offer a straight forward solution to accommodate this. The approach taken to replace all the out of range values, was to add a large number of lag columns (e.g. 30), each with increasing lag (1, 2, 3, ..), and to perform the above operation from one lag column to the previous one. An alternative was implemented where in a while loop, Pyspark was made to count the number of values within the columns that were outside of threshold range. However Pyspark is not optimized for such operations, and each iteration of the loop would take over five minutes per variable to count the values. Similarly, a recursive approach was tested to limit the number of lag columns, with counting or masking the out of range values. Pyspark still performed better and faster creating 30+ columns and changing values for each of them.

The reason for the long computation times for simple operations such as counting rows is that they are the few operations where computation is actually forced. When a new colum is added or a filter is applied, in short when a transformation takes place, Pyspark registers it as a "lazy" operations. the transformation is registered but the computation is not forced. Using Pyspark `.count()` forces computation and triggers all previous transformation, such as the creation of a new lag column, resulting in a long of computational time.

It was therefore elected to limit the Pyspark operations to lazy transformations only, and to add the large number of lag columns. The only forced computation is the transformation to Pandas. This library switch was also largely decided upon because of the limitations of Pyspark to perform simple operations in a timely manner.

## 6.2 Slug Labelling Implementation & Results

### 6.2.1 Results

Slug labelling was applied on windows of 5, 20 and 40 minutes respectively. The data from 2014 to 2019 was used, with all windows overlapping by more than half their length.

The results were inconclusive on the 5 minutes window data. The K-mean clustering was performed using 5 clusters, however only two flow clusters were physically recognizable: closed choke flow and normal flow. The 20 minutes window clustering was performed and similarly, only these two flows could be observed, using from 3 to 6 clusters. The main trend in the other clusters was transition in the WH choke and the other variables. Those trends can be viewed in Appendix A.

In order to counter the interference of the noisy and transitioning flows, two data pre-processing approaches were considered:

- Only consider data where the WH choke is above 99%.
- Crop the data where the choke is closed (0%) and where the choke gradient is more than 3%, which is indicative of human interaction.

The choke cleaning was performed as part of the Data Engineering module, using Pyspark transformations. One resulting problem however, was the creation of discontinuous data. As a feature vector creation takes 20 or more consecutive points in the data frame, the discontinuity in time results in a jiggered looking flow where each variable jumps from its pre-choke value to its post-choke one.

After choke cleaning was performed, the analysis was run again, this time using only 20 and 40 minute windows. Between 3 and 5 clusters were selected each time. Kmeans classification for 40 minutes and 20 minutes window sizes, with five clusters, can be seen in Figure 12. The results are also available in the Appendix A and are only presented here for reference.

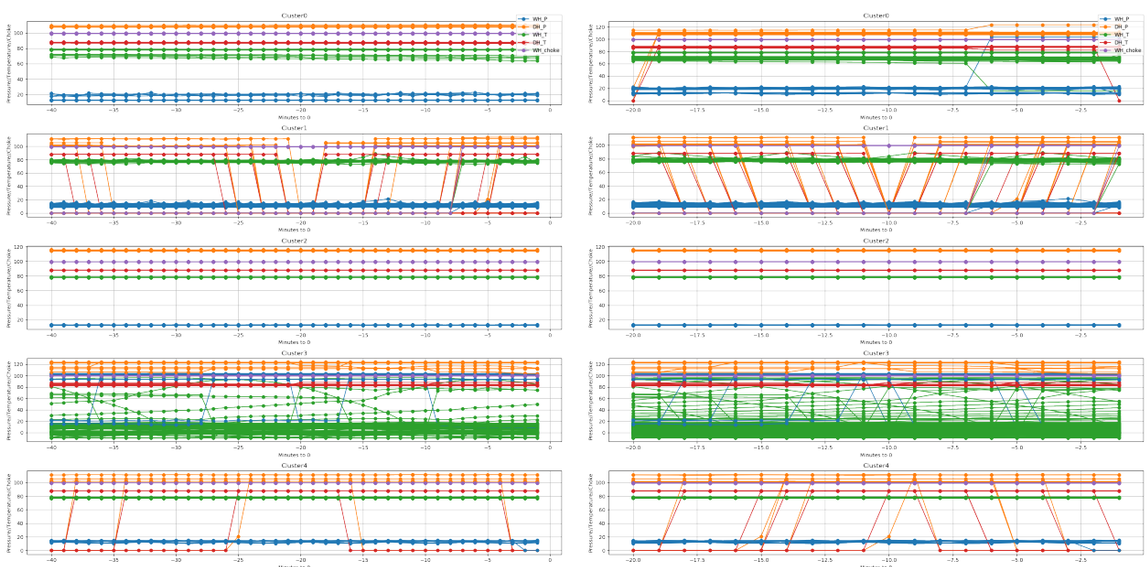
In Figure 12, the results for 5 clusters on the 20 and 40 minutes window size are presented, showing some similarity in the grouping.

For example, in both results, cluster four seems to regroup the high variation in WHT data (in green).

Another example is cluster three for both 12a and 12b where the windows seem to exhibit normal flow characteristics, with no variation in any of the variables.

Cluster one in 12a and cluster two in 12b are also similar, as they have a slightly larger variance in WHP, and could therefore be considered as slug behaviours clustering together. However it can also be noticed that the DHP (in orange) is evolving in a totally different range, from cluster two to cluster five in 12a. In cluster 2, DHP is approximately 120 BarG, whilst for cluster five it is closer to 100 BarG. That change in DHP is due to the pressure in the reservoir decreasing over the years, as oil is being pumped out. It is therefore unsure whether the feature vectors were clustered together because of normal and slug flow or because of the different range of DHP.

Moreover, there is also a lot of noise associated with the data. Each cluster presents large changes, for example from one DHP point to another, which cannot be explained as being independent behaviour.




(a) 40 minutes window

(b) 20 minute window

Figure 12: Example of Kmeans

### 6.2.2 Discussion

Overall, whilst some trends and characteristic flows can be observed in the different clusters, this labelling method is not accurate enough to label the data. Too much noise is associated with each window of time, that cannot be identified.

The lessons learnt to move the data labelling forward and create more data analysis models were the following:

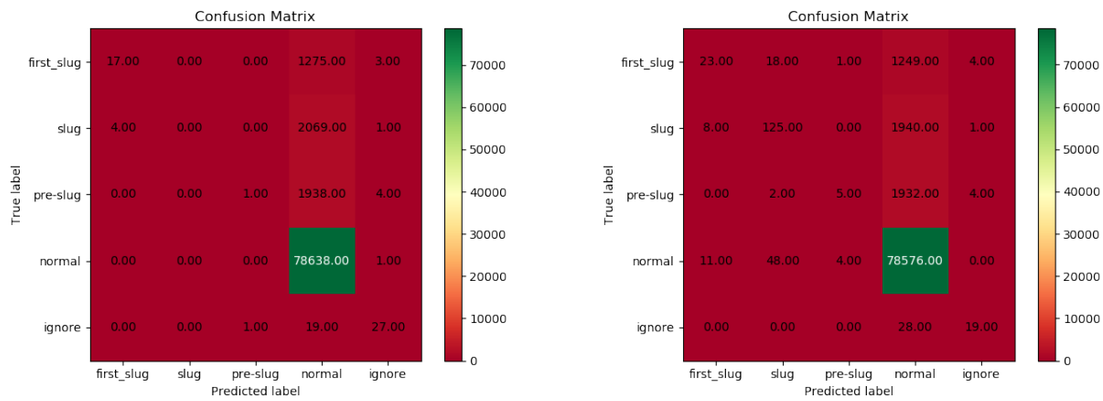
- Unsupervised learning cannot separate flows into sought after patterns. Flow labelling needs to be performed manually.
- No indicator of pre-slug has been identified.
- Data where the choke is closed or changing is not necessary and can be removed from the datasets. However, the resulting data discontinuity generates misleading values that should be dealt with.

## 6.3 Flow Recognition Implementation & Results

### 6.3.1 Results

Flow Recognition method results were poor. Whilst the overall performance of the Logistic Regression classifier on the first run were high, above 90% when using 20 minutes windows of time, looking at the results more closely revealed that there was a large bias towards classifying the window as normal flow.

The confusion matrices from Logistic Prediction and SVM, shown below in Figure 13, clearly indicate that the normal flow is over predicted for all the classes, and gets a 99.7% accuracy classifying itself. Since there are over 170 thousand normal flow feature vectors, it heavily influences the last accuracy score. In comparison, pre-slug and first slug flow feature vectors are barely correctly classified. For SVM, 0 slug or pre-slug flows are correctly classified. Even for the ignore label which takes into account choked flows, the classification accuracy is less than 50 %. Note that the pre-slug period was set to the 40 minutes flow before a first slug happening.




(a) SVM

(b) Logistic Regression

Figure 13: Confusion Matrix of results for the Flow Recognition Classification

Whilst the models could probably use some hyper parameter tuning, it is unlikely that the model would be able to significantly improve regarding the smaller classes such as first slug, pre-slug and slug. The problem certainly comes from the method, as opposed to the model tuning. There's a high chance that the windows of times, which in this case are 20 minutes long, have been mis-labelled. Knowing that a slug on average occurs every 6 to 15 minutes, it could be likely that a window of time labelled as slug would mostly show normal properties. Similarly a pre-slug label could show slug ish behaviour in the case of a choke re-opening and the WHP settling down into a slug. Large spikes would counter the classification.

Moreover, because of the large amount of data and classes, the training of the model took a considerable amount of time. Using the full data set, it took over seven minutes to complete on DataBricks.

### 6.3.2 Discussion

The points to take forward to create more robust labels and get better accuracy results from the models were:

- Labelling the data point by point is not very effective. Moreover, labelling individual peaks still leads to all the points in the immediate time frame, before and after, to be labelled as normal. Because of this the window labels are incorrect.

- Instead of labelling slugs individually, labelling the whole slug flow might be more interesting. The most important slug is the first slug, from which the pre-slug flow can be identified.
- Labelling choked flow is inefficient, and it is preferable to completely eliminate the choked flow data. However, the discontinuity needs to be dealt with.
- Classifier was overwhelmed by the number of labels. Binary classification would be more efficient.

Overall it was decided to move on to a new method to label the data differently and start moving away from the fixed windows of time with only standardised features.

## 6.4 Slug Detection Results

The Slug Detection was the most successful approach of all three methods implemented. For the tuning and final results of the data analysis, the data from January 2012 to September 2018 were considered. The training and testing set ratio were 70% and 30% of the data.

There were a total of 695 feature vectors used to train and validate the Logistic Regression model, of which

- 139 intervals to a slug flow (True)
- 556 intervals leading to a normal flow (False)

### 6.4.1 Hyper Parameters Tuning

There are numerous hyper parameters that can be considered for the slug prediction, such as model specific ones or data engineering ones.

The hyper parameters that will be considered for tuning are:

- Random Forest Hyper Parameters: Number of Trees (`n_estimators`) , Depth of Trees(`max_depth`) and Bootstrap (whether columns/rows are dropped per Decision Tree)(`bootstrap`)
- Logistic Regression: Regularization value (`C`) and Maximum iteration number (`max_iter`)
- Data Engineering Method: Window size (default 4 hours) and number of splits (default 5) in feature vector

The hyper parameters tuning code and result can be viewed in the `IRP_HyperParameterTuning.ipynb` in the code github repository.

**Method Tuning**

The method was tuned on two parameters: the length of the interval to consider before a first slug, and the number of splits. The second parameter decides of the number of features in the feature vector. The chosen range of values is :

| | |
|---|---|
| Window Size | 240, 300, 3600 |
| Number of splits | 4, 5, 6 |

A manual grid search was performed over the two parameters to find the best combination that yields the highest results. Only the Random Forest classifier was used here, for a faster check.

One issue with tuning these parameters is that the training data set is different for each iteration, and so is the split of the data. This means there will be some bias in the score that depends on the data being sorted at each iteration. For this reason, the grid search was performed three times to see which combinations would consistently perform better.

Overall for window size, length of 240 and 300 minutes (4 hours and 5 hours) performed best. The size of 240 minutes was chosen. This is for practicality reason, as there may be more intervals of data leading to slug when the minimum size required is 240 as oppose to 300.

In terms of the number of splits, and therefore number of features in the feature vector, there was no number that performed exceptionally better than the other. It was decided to keep values of 5 splits.

**Models Tuning**

The hyper parameter tuning was performed using `ScikitLearn.GridSearchCV`, the grid search cross validation for the Random Forest and the Logistic Regression models. The results of the Grid Search as cross validated using a three-fold stratified K-Fold.

The following values were tested :

The results were as follow for the Random Forest Classifier:

```
Best: 0.993534 using 'bootstrap': True, 'max_depth': None, 'n_estimators': 15
```

| | | | |
|---|---|---|---|
| Random Forest | Number of trees | 5, 10, 15, 20 | (default is 10) |
| | Maximum depth of the trees | None, 50, 10, 5, 2 | (default is None) |
| | Bootstraps or feature drops | True, False | (default is True) |

| | | | |
|---|---|---|---|
| Logistic Regression | Regularisation parameter (C) | 0.01, 0.1, 0.5, 1.0, 2.0 | (default is 1.0) |
| | Maximum number of iterations | 50, 75, 90, 100, 110, 120, 150 | (default is 100) |

For the Logistic Regression model with the full set of features:

```
Best:  0.954741 using {'C': 1.0, 'max_iter':  50}
```

For the Logistic Regression model with only the top 50 features as set by Random Forest:

```
Best:  0.956897 using {'C': 2.0, 'max_iter':  50}
```

The two Logistic Regression models fit the data with a high C regularization value. This means the model have low bias, and could be prone to overfitting. The best scores are very high at 95%. This indicates that the data is sufficiently different, and that it offers a clear divide between the windows of time leading to a slug and those leading to normal flow.

Furthermore, the max iteration values for tuning the weights values in the regression is relatively low, both for the full features model and the reduced features number model. This is indicative again that the classification occurs smoothly with the data, and that the parameters are relatively easily calculated.

The `Slug_Detection` class was updated for these parameters, which can also be accessed and changed as key words arguments. The full list of ranking for the parameters is available in Appendix B.

### 6.4.2 Results

**Data Labelling**

The labels developed were the most representative as compared to the other methods. The slug peak was consistently labelled, and thanks to the first slug condition, irregular flows such as post-choke were ignored.

The results for the slug prediction were highly conclusive, as illustrated in Figure 14.

As mentioned before, the labelling cannot be verified objectively. Instead, figures showing the pressure and temperature with labelling indicators were shared and discussed with the company's engineers. Some slugging definitions like the value of the WHP gradient that can be interpreted as slug and the time interval between regular slugs were also discussed.

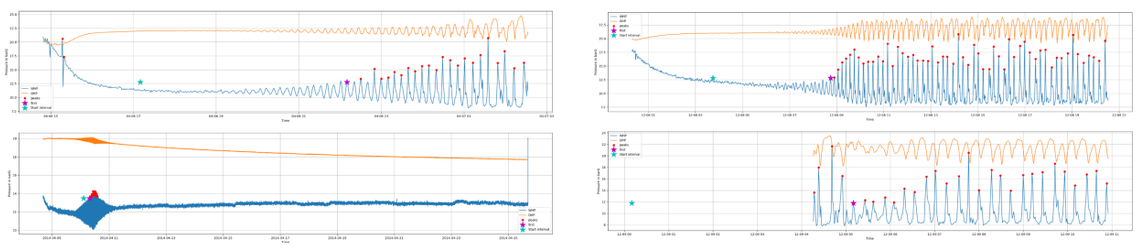


Figure 14: Example of slug (red dots) and first slug (magenta star) labelling. The interval that will be considered for classification starts at the Start cyan star, and ends at the magenta first slug star. The DHP in orange has been scaled.

The second part of the data labelling was to label four hours long, or in other words 240 points long, data frames. The labeling was set to 0 or 1, depending on whether the data was leading to normal flow, or was leading to a slug flow pattern. The first labelling showed clear differences between the two types of flow, as shown in Figure 15. note that during model training, the last 60 points are deleted, as the model is trained to recognize the flow that leads to a slug one hour before it happening.

**Classification**

As can be seen in Figure 16, the performance of both classifiers is outstanding. The Random Forest had an accuracy classification of **100.0 %**, whilst the Logistic Regression using the top features had an accuracy of **95.694 %**. These result were obtained using the best hyper parameters for each model.

These results are satisfactory, as the accuracy is high.

Looking into more details at the flows that were mis-classified by the Logisitics Regression Classifier (included in Appendix C)

Three of the nine mis-classified intervals were pre-slug intervals, with seemingly constant but higher than usual variance as seen in Figure 17. Since constant variance and mean is characteristic of normal flow, it actually makes sense that the classifier would label it as such.

Another four mis-classified flows showed random and unstable characteristics. The last two flows shows what would be thought typical pre-slug characteristics, with an increasing mean and variance over the period of time. More data could be needed to train the model.

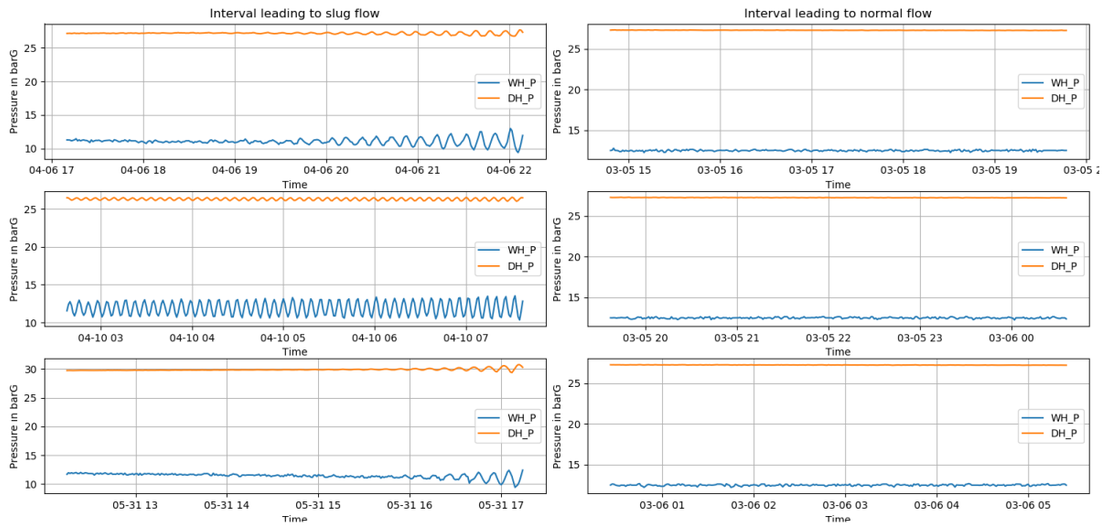
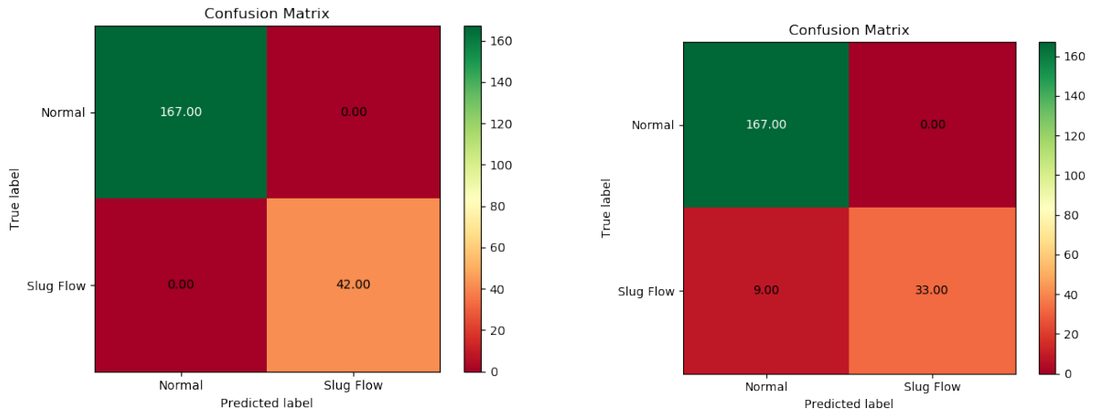



Figure 15: Labels for classification. WHP is in blue, and DHP (scaled) is in orange. Data shown is four hours.




(a) Random Forest: 100.0 %

(b) Logistic Regression: 95.694 %

Figure 16: Classification Results

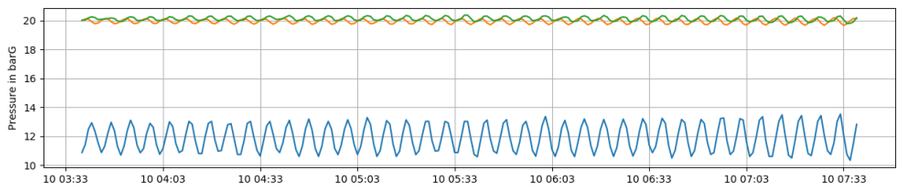



Figure 17: Misclassified flow

### 6.4.3 Discussion

The results for this method were highly accurate. This method successfully labelled slugs and first slug, and allowed for the identification of pre-slug flow. This is a successful approach that hopefully will add value to Wintershall Dea. It could in the future be implemented for real time data, and thus create an indicator that slugs are happening in the next hour. This paves the way to being able to control slugs through choking the well and gas lifting.

Logistic regression was implemented as a preferred classifier over Random Forest. However it yields slightly lower results, and requires the additional steps of fitting the random Forest model to get the top features. Upon inspection, the top features are different from one fit to another. This questions whether using the top features only for the Logistic Regression is beneficial.

## 6.5 Slug Forecasting Results

The Slug Forecasting results were highly satisfactory. As explained in the Code Description Section, the module takes in a data frame containing WHP during a slug flow. An ARIMA model is fitted to the data, and the flow can then be forecasted.

### 6.5.1 Parameters Selection

For the chosen WHP slug flow interval of size 240, the Augmented Dickey-Fuller test for stationarity was applied.

```
ADF Statistic: -4.192408
p-value: 0.000679
Critical Values:
        1%: -3.459
        5%: -2.874
        10%: -2.573
```

(a) d = 0

```
ADF Statistic: -11.127478
p-value: 0.000000
Critical Values:
        1%: -3.458
        5%: -2.874
        10%: -2.573
```

(b) d = 1

```
ADF Statistic: -10.603076
p-value: 0.000000
Critical Values:
        1%: -3.459
        5%: -2.874
        10%: -2.573
```

(c) d = 2

Figure 18: d parameters tuning

As can be seen, when the differencing is 0, the data is stationary since the p-value is less than 5% and the ADF statistic is negative. For $d = 1$ or 2, the p-value decreases further and the same happens for the ADF statistics. The chosen parameter for this interval will therefore be 0 since it is enough, but $d = 1$ could also be experimented with.


Figure 19: Auto Correlation and Partial Auto Correlation Plots

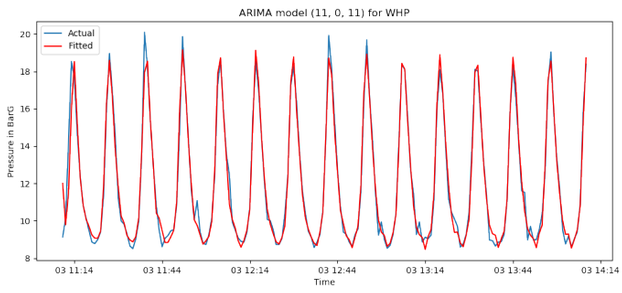
From the ACF plot, it can clearly be seen that the slug is a regular stationary pattern, since the correlation pattern repeats itself. The $q$ parameter can be set to 1, 2, 4, 5, 6, 11 or 12 looking at the out of range values. From the PACF plot, it is clear that the $p$ parameter should be set to 11 since it's the most striking out of range value. However, a value of 1 could also be tested.

Overall there is a range of parameters that can be used to fit the ARIMA model.

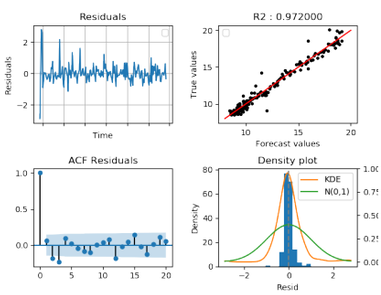
### 6.5.2 Results

The results were highly satisfactory. The full range of $p$, $d$, $q$ values that were tested for the fit can be viewed in Appendix D.

For the selected data range of WHP, the ARIMA model was trained on a 180 points, to predict 60 points. In other words, the model was traine on 3 hours of data to predict slugs in the next hour. The best parameters were p=11:




(a) Fit




(b) Error information

Figure 20: ARIMA model p=11, d=0, q=11

As can be seen in Figure 20, the residuals are relatively low, the actual and fitted values are highly correlated with each other, the auto correlation of the residuals stay within the confidence interval and the residuals are normally distributed. The ARIMA model is a good fit for the data.

The forecast value were very close to the actual values. The slug timing was predicted highly accurately. As can be seen in Figure 21, the actual and forecast line merge together as the pressure increases and decreases.

Similarly the confidence interval at that stage is very slim. The model however gives less accurate predictions at the peaks and troughs of the slug, with the confidence interval also getting larger.

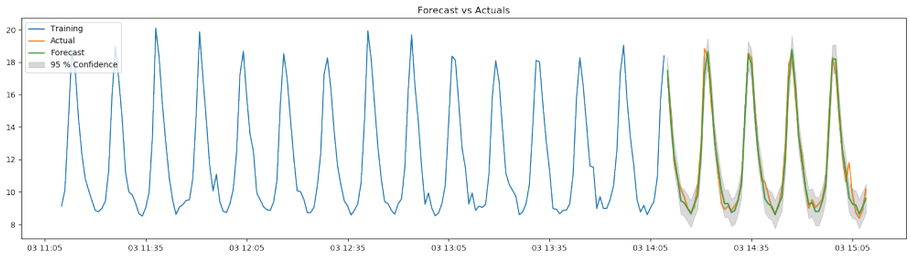



Figure 21: Forecast ARIMA model

The error metrics are still however showing good results. The residuals are small, and not auto correlated. The data is normally distributed, although there is a bias towards negative residuals, meaning the forecast is usually lower than the true values

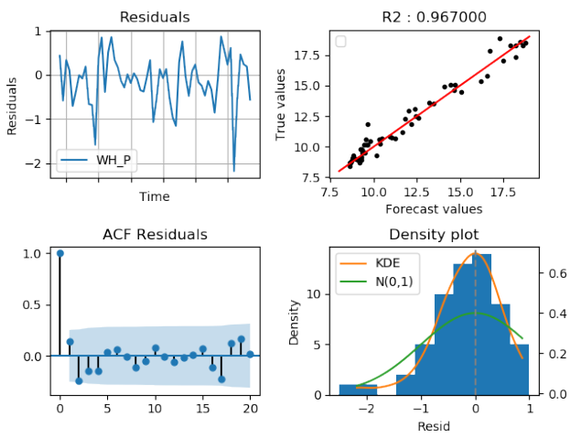



Figure 22: Forecast ARIMA model

### 6.5.3 Discussion

The ARIMA model performed very well. The error metrics were satisfactory:

- Mean Absolute Percentage Error, the data fitting scored -0.13, and the prediction scored -0.131. Both values are close to zero
- Mean Squared Error, data fit scored 0.321 and prediction scored 0.344, two relatively low values.
- Root Mean Squared Error, data fit scored 0.566 and prediction scored 0.587, again two very good statistics
- R2 Determination, data fit scored 0.972 and prediction scored 0.967 showing that that the regression is a very good fit.

However, the model is not transferable and can only predict an independent WHP. This means that if this were to be applied to incoming data, the ARIMA parameters would constantly have to be recalculated and refitted. The same parameters were used to predict the same slug flow, however 24 hours later, the model would not fit. The ACF and PACF figures showed widely different correlation lag values. This is all accessible in the Appendix.

The ARIMA was also trained on pre slug data, as identified in the Slug Detection, to see if it could predict the slug frequency and amplitude. The results accuracy was low, as the ARIMA was unable to reflect the increase in amplitude, had a very large confidence interval and did not match the true slug peaks. This can be viewed in Figure 23.

Moreover, ARIMA works best with independent variables. If slug control were to be applied, it is unlikely that the model could forecast flows under different circumstances. The model therefore does not represent added value to Wintershall Dea.

The model performs very well for slug flow, but probably another modelling approach would yield better results for the required applications.

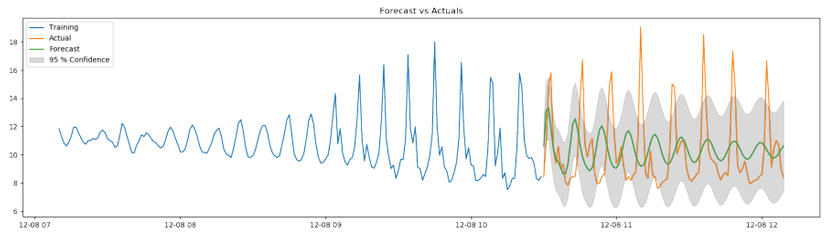



Figure 23: Forecast ARIMA model for Pre Slug flow

# 7 Discussion and Conclusions

## 7.1 Project Challenges

The main challenge identified during the project was handling the data. In total there were approximately 5 million data points available for each variable. The data was raw and required a data engineering to be performed It wasn't clear which behaviours were independent and which were not, and the slug definition given was not conclusive. Over the 10 years of data available, the natural flow patterns were on numerous occasions disrupted by human interventions like choking. The problem was initially posed as a time series problem but the uneven data made this unrealistic.

Moreover, a second challenge with the data was the very discrete slugging patterns. Whilst on a monthly outlook slugs are easily identified visually on a plot, on an hourly basis there are only 3 to 4 slugs occurring. That is 60 individual points, with the majority of them looking like a normal flow and less than 10 pointing to a slug. This leads to any smoothing of the data to be impossible as it would loose the patterns.

A large part of the project was spent on computing slug indicators. In the Flow Recognition module each point was given a label, and pre-slug patterns were roughly indicated by overriding 40 points before a poorly defined first slug. Identifying the first slug was a break through in itself, as it allowed to label the full slug flow as one instead of labelling the individual points. With the robust slugging peaks and first slug labels in the Slug Detection module, it was finally possible to break free from the constraint of the individual point and binning and other feature engineering was possible.

Another challenge still related to slug flow and pre-slug flow, was the low amount of information available from the Wintershall Dea engineers, as to how to visually detect slug flow happening. A slug flow is caused by the geometry of the well, however in the data there is no clear indicator, except maybe couple minutes or less prior to individual slugs. Predicting a slug one minute before it happens is however no value added for the company. The slugging detection problem was also never posed as detecting a slug flow, but rather detecting and identifying individual slugs.

Finally, from a Software Development point of view, Data Bricks was not a good tool to develop the software and perform the analysis. It was used from the beginning of the project mainly to get access to the company Data Lake database, however were the project to be done again, Azure DevOps or any other platform made available by Winterhsall Dea would largely be preferred and recommended.

## 7.2 Project Strengths

The strength of the project is that each method was created consecutively and allowed a learning opportunity for the following approach. Slug detection was the most successful module both because of the high accuracy results and because it brings value to the company. It can easily be applied to real time data and can be implemented on a dashboard as a warning sign, or as an always updating percentage on whether a slug is occurring.

With Slug Detection module working with a high accuracy, the Slug Labelling and Flow Recognition could now be improved from the success. The data processing and feature engineering could be applied to the full data set, and by updating and changing the Flow Recognition module to classify the new feature vectors, the results could be more conclusive. The intervals considered could be made shorter, such as one hour, and the labels updated to slug flow, pre-slug flow and normal flow. Similarly, the feature engineering could also be used for the KMeans in the Slug Labelling module. Whilst this would be less value added to the company, since the algorithm designed to label the slugs is very robust, it could yield better results and allow to cluster and label the data through machine learning.

Another strength of the project is that the module was designed to be used by the production engineers at Wintershall Dea. The package uses production related words for names of variables and little understanding of machine learning or computer sciences is required to use the package. A Python notebook was also created to walk the users through the different resources of the package. The package answers the brief of the company and can be built on to accommodate for further well monitoring utilitisation?

## 7.3 Project Limitations

The package was very bespoke for production engineers which is good for its applications but also means it is not suitable for other usages. It is very specific to slugs in the well considered and has not been tested on other wells where slugging characteristics might be different. Whilst it is possible to add more variables to the Data Engineering modules and the inherited modules, and other flexible utilities such as adding thresholds or defining the slug characteristics for the Slug Detection, the package remains very rigid. It is highly bespoke and can hardly be applied to other situations.

The Slug Detection module which is the package biggest success also has some limitations. There are numerous steps required to be able to start classifying the feature vectors. The feature vectors themselves are clearly different from one class to another as seen in Figure 15, pondering over the usefulness of having such a lengthy data processing and whether the method is over engineered. If the intervals are visually different, a simpler approach could also have been taken. The goal to use Logistic Regression to classify is also a limitation. The model was chosen over Random Forest as it is less of a black box and it is more familiar to users. However, it yields lower results than the Random Forest, and requires the additional step of fitting the Random Forest to get the best features.

The main limitation of this project and of the package elaborated is the limitation in time available to test the package and check results. Hyper parameter was performed for the Slug Detection and Slug Forecasting modules, however more time could still have been invested on refining the results and the data pre processing. The development of each module was stopped shortly after the results were found. Since the project was run using SCRUM methodology, each development was quick pace and more importance was given to accomplishing the task than tuning methods to the best possible.

## 7.4 Project Continuation

The next steps for this project would to apply models to incoming real time data and to forecast WHP behaviour when the slug flow is controlled through choking the well and gas lift. These two steps are the methods that would bring the most added value to Wintershall Dea.

**Online Learning for Slug Detection**

As far as online learning and applying the Slug Detection module are concerned, the real time data stream will be made available once a pipeline has been set up by the company. Implementing online learning would therefore be about tuning the fitted model to work with incoming data, and making the data engineering and pre processing into feature vectors as smooth as possible. The feature vectors would have to be updated frequently to take in the new data. It probably would be beneficial to perform a better feature selection to avoid having to create 150 features at every round.

**Slug Control Forecasting**

The ARIMA model was trained to forecast slug flow, as slug flow is already occurring. It is very reliable to predict the future slug peaks happening. However, it cannot predict slugs accurately from training the model on pre slugging data. A more robust time series model is therefore required.

An option that was investigated is the use of Recursive Neural Networks (RNNs), and more particularly of Long Short Term Memory Neural Network (LSTM).

## 7.5 Conclusion

The project reached all five objectives set at the beginning:

- A slug flow indicator was developed. Slug peaks were identified and the first slug of a slug flow was correctly labelled for the entire data set. This can be found in the Slug Detection module.
- Normal and pre-slug flow were recognized and labelled in the Slug Detection module. The Flow Recognition module was unsuccessful but could be developed to recognize slugs, pre-slug, normal and even choked flows if the feature vectors and pre processing of Slug Detection module are applied.
- Slug occurrence can be detected in over 96% of tests one hour in advance.
- Time series modelling was trialed, and accurately predicted slug flow behaviour by training the model on slug flows. The module Slug Forecasting was developed contained through error metrics and plots, to assist users in judging the accuracy of the model.
- Effort was put in the building of the package to use human understandable variable names and to keep the data analysis flow clear.

Overall the project was successful and has potential for future developments, both in detecting slugs in real time data and forecasting WHP behaviour for slug control when training data becomes available. The slugdetection package is readily available for Wintershall Dea engineers to use, and has the potential to be applied to other wells. This package could furthermore have applications beyond the Oil&Gas industry and be applied to any high variance fluctuations data sets.

# References


[1] How to organize data labeling for machine learning: Approaches and tools, www.altexsoft.com, Mar 2018.

[2] Mustafa Al-Naser, Moustafa Elshafei, and Abdelsalam Al-Sarkhi. Artificial neural network application for multiphase flow patterns detection: A new approach. *Journal of Petroleum Science and Engineering*, 145:548–564, 2016.

[3] Christopher M. Bishop. *4. Linear Models for Classification*, page 205–206. Springer-Verlag, New York, 2016.

[4] Christopher M. Bishop. *7. Sparse Kernel Machines*, page 325–358. Springer-Verlag, New York, 2016.

[5] Christopher M. Bishop. *9. Mixture Models and EM*, page 423–430. Springer-Verlag, New York, 2016.

[6] Ben D. Fulcher and Nick S. Jones. Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):3026–3037, Jan 2014.

[7] Wayne A. Fuller. Introduction to statistical time series. *Wiley Series in Probability and Statistics*, 1995.

[8] Pablo Guillen-Rondon, Melvin D Robinson, Carlos Torres, and Eduardo Pereya. Support vector machine application for multiphase flow pattern prediction. Jun 2012.

[9] R J Hyndman and G Athanasopoulos. Arima models. *Forecasting: principles and practice*, page 221–274, 2018.

[10] Huan Liu and Hiroshi Motoda. *Computational methods of feature selection*. Chapman Hall/CRC, 2008.

[11] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. Support vector machines and machine learning on documents. *Introduction to Information Retrieval*, page 293–320, 2008.

[12] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. Text classification and naive bayes. *Introduction to Information Retrieval*, page 234–265, 2008.

[13] Lior Rokach and Oded Maimon. *Data mining with decision trees: theory and applications*. World Scientific, 2015.

[14] E.s. Rosa, R.m. Salgado, T. Ohishi, and N. Mastelari. Performance comparison of artificial neural networks and expert systems applied to flow pattern identification in vertical ascendant gas–liquid flows. *International Journal of Multiphase Flow*, 36(9):738–754, 2010.

# Appendices

## A KMeans Cluster Results

### A.1 Kmeans Window 5 Minutes

```
Cluster 0 contains 67.35 % of the data
Total of  404055  points
Cluster 1 contains 32.488 % of the data
Total of  194907  points
Cluster 2 contains 0.162 % of the data
Total of  973  points
```

(a) 3 Clusters

```
Cluster 0 contains 32.293 % of the data
Total of  193739  points
Cluster 1 contains 0.162 % of the data
Total of  973  points
Cluster 2 contains 67.42 % of the data
Total of  404478  points
Cluster 3 contains 0.124 % of the data
Total of  745  points
```

(b) 4 Clusters

```
Cluster 0 contains 47.699 % of the data
Total of  286161  points
Cluster 1 contains 0.162 % of the data
Total of  973  points
Cluster 2 contains 20.622 % of the data
Total of  123721  points
Cluster 3 contains 0.124 % of the data
Total of  745  points
Cluster 4 contains 31.393 % of the data
Total of  188335  points
```

(c) 5 Clusters

Figure 24: Clusters

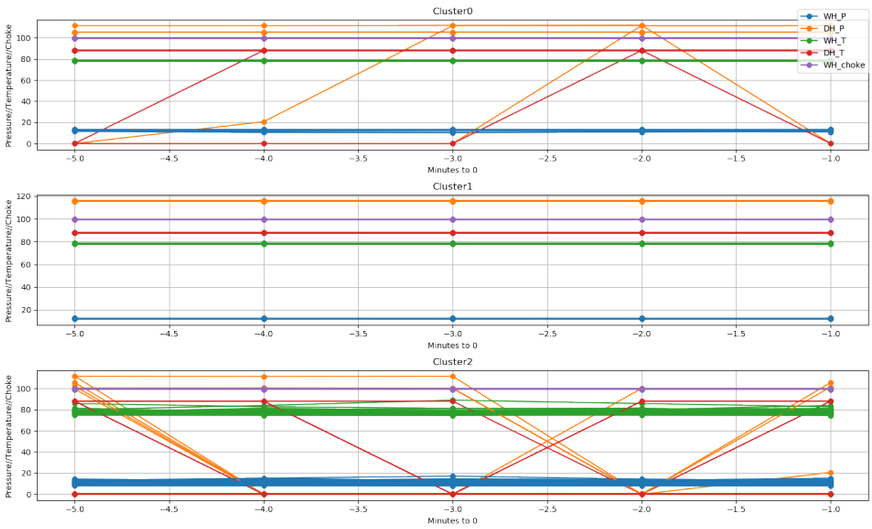



Figure 25: Kmean with 3 Clusters

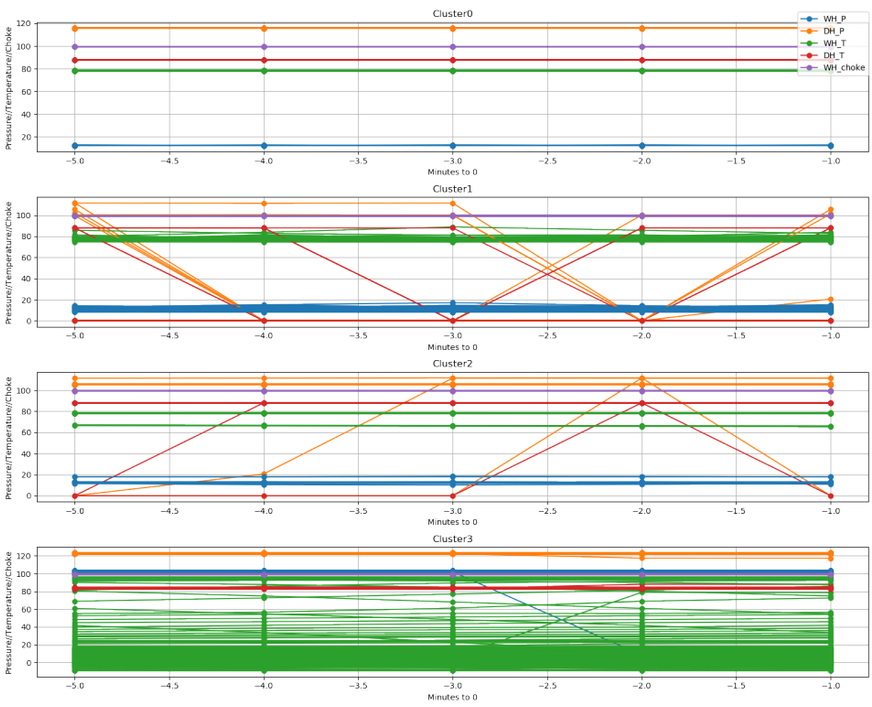



Figure 26: Kmean with 4 Clusters

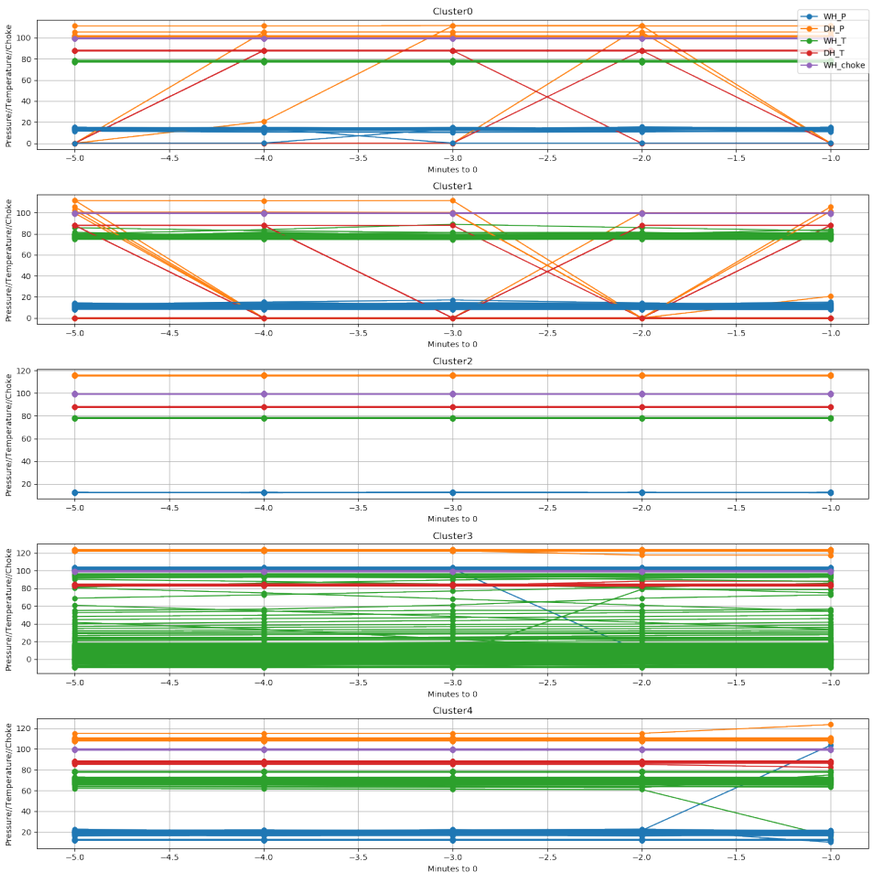



Figure 27: Kmean with 5 Clusters

## A.2 Kmeans Window 20 Minutes

```
Cluster 0 contains 33.189 % of the data
Total of  59733  points
Cluster 1 contains 66.648 % of the data
Total of  119952  points
Cluster 2 contains 0.163 % of the data
Total of  294  points
```

(a) 3 Clusters

```
Cluster 0 contains 32.938 % of the data
Total of  59281  points
Cluster 1 contains 66.774 % of the data
Total of  120180  points
Cluster 2 contains 0.163 % of the data
Total of  294  points
Cluster 3 contains 0.124 % of the data
Total of  224  points
```

(b) 4 Clusters

```
Cluster 0 contains 47.736 % of the data
Total of  85914  points
Cluster 1 contains 20.735 % of the data
Total of  37318  points
Cluster 2 contains 0.163 % of the data
Total of  294  points
Cluster 3 contains 0.124 % of the data
Total of  224  points
Cluster 4 contains 31.242 % of the data
Total of  56229  points
```

(c) 5 Clusters

Figure 28: Clusters

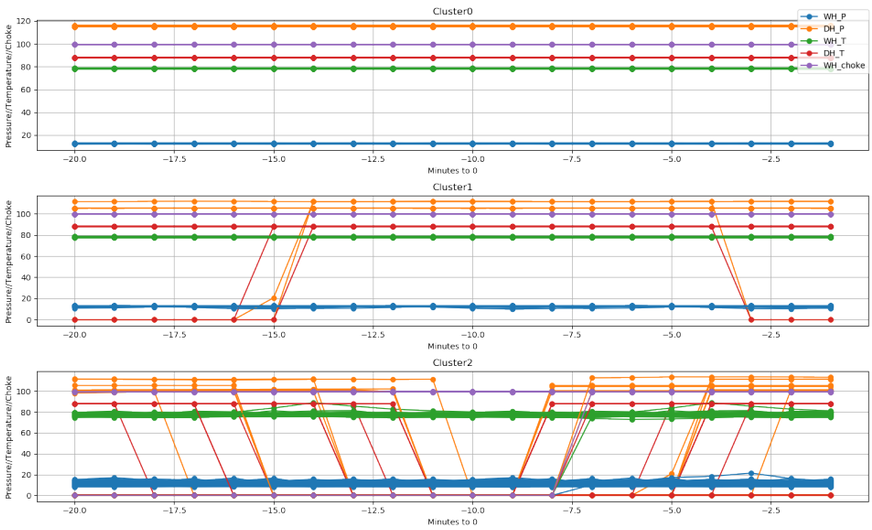



Figure 29: Kmean with 3 Clusters

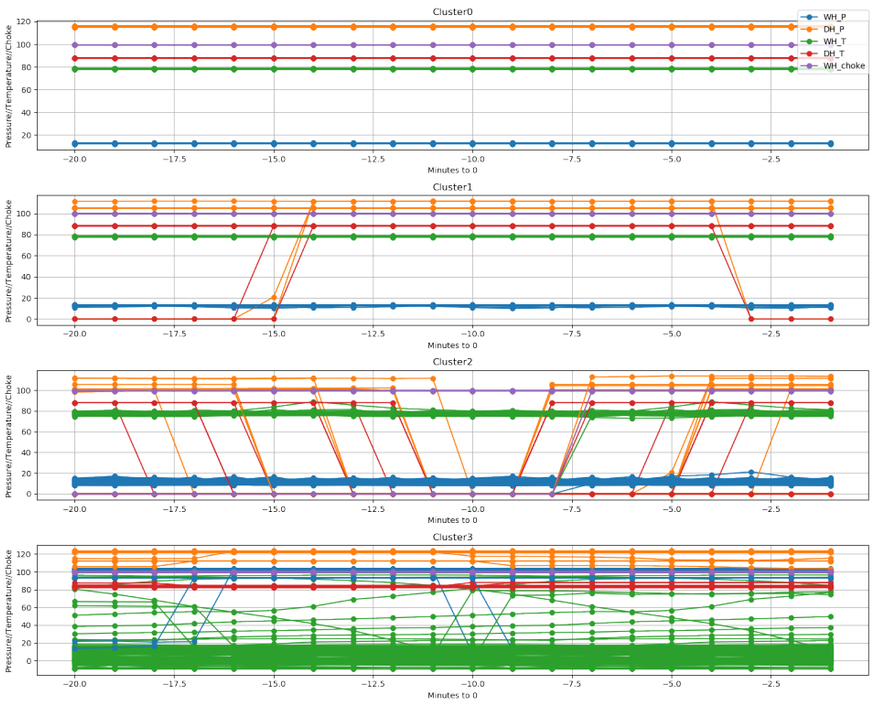



Figure 30: Kmean with 4 Clusters

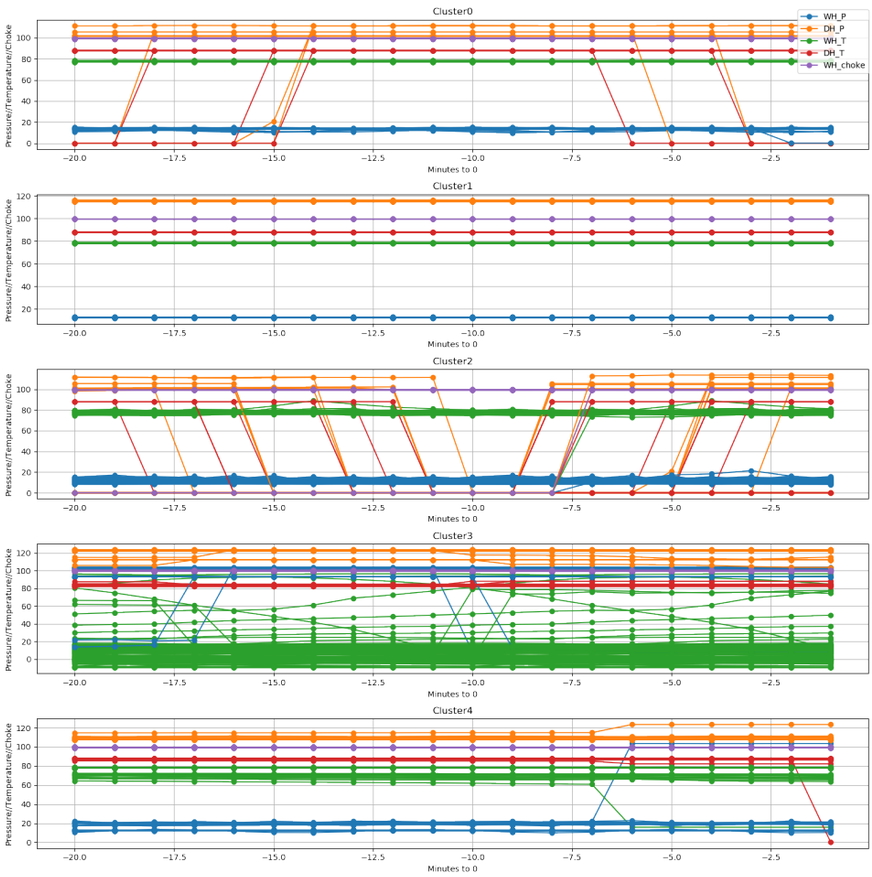



Figure 31: Kmean with 5 Clusters

## A.3 Kmeans Window 40 Minutes

```
Cluster 0 contains 66.259 % of the data
Total of  59626  points
Cluster 1 contains 0.164 % of the data
Total of  148  points
Cluster 2 contains 33.576 % of the data
Total of  30215  points
```

(a) 3 Clusters

```
Cluster 0 contains 66.394 % of the data
Total of  59747  points
Cluster 1 contains 33.316 % of the data
Total of  29981  points
Cluster 2 contains 0.126 % of the data
Total of  113  points
Cluster 3 contains 0.164 % of the data
Total of  148  points
```

(b) 4 Clusters

```
Cluster 0 contains 30.007 % of the data
Total of  27003  points
Cluster 1 contains 0.164 % of the data
Total of  148  points
Cluster 2 contains 19.775 % of the data
Total of  17795  points
Cluster 3 contains 0.126 % of the data
Total of  113  points
Cluster 4 contains 49.928 % of the data
Total of  44930  points
```

(c) 5 Clusters

Figure 32: Clusters

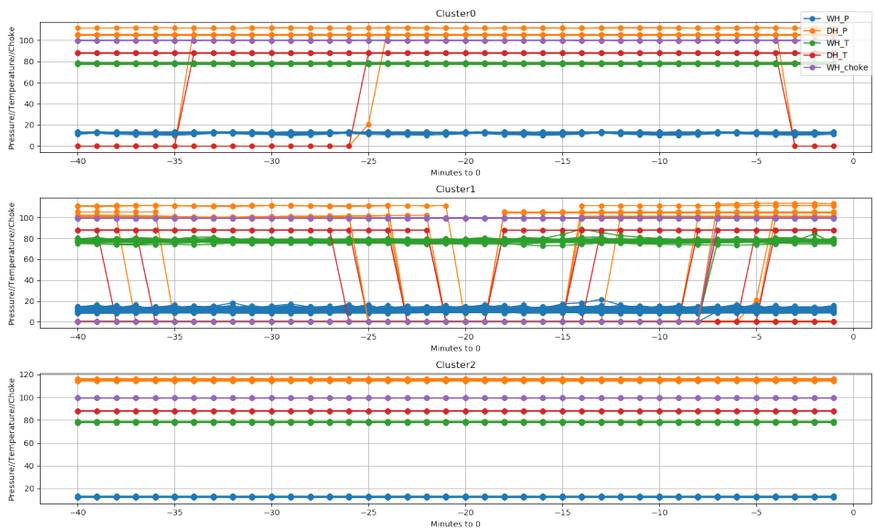



Figure 33: Kmean with 3 Clusters

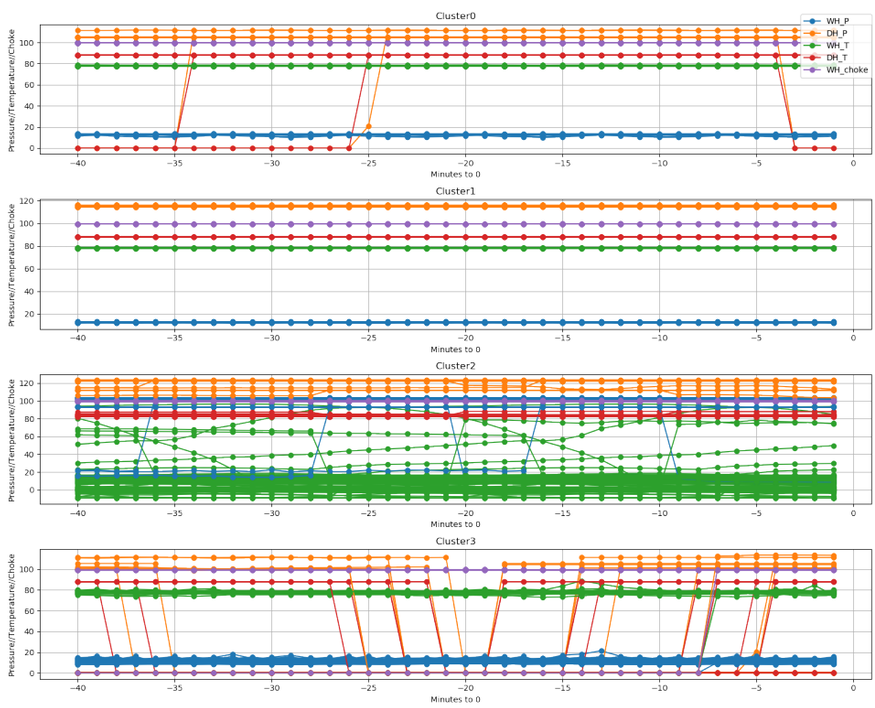



Figure 34: Kmean with 4 Clusters

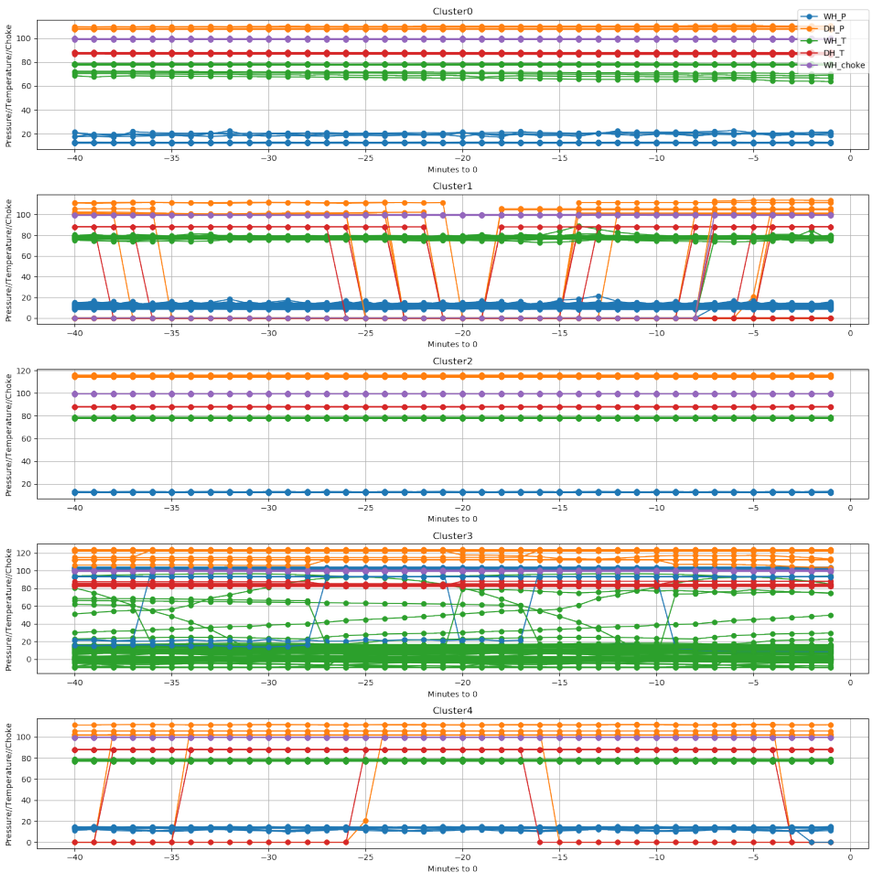



Figure 35: Kmean with 5 Clusters

# B Grid Search results

| rank_test_score | param_C | param_max_iter | mean_test_score | mean_train_score |
|---|---|---|---|---|
| 1 | 0.01 | 90 | 0.9675675675675676 | 0.9594807238394965 |
| 1 | 0.01 | 75 | 0.9675675675675676 | 0.9594807238394965 |
| 1 | 0.01 | 110 | 0.9675675675675676 | 0.9594807238394965 |
| 1 | 0.01 | 120 | 0.9675675675675676 | 0.9594807238394965 |
| 1 | 0.01 | 150 | 0.9675675675675676 | 0.9594807238394965 |
| 1 | 0.01 | 100 | 0.9675675675675676 | 0.9594807238394965 |
| 1 | 0.01 | 50 | 0.9675675675675676 | 0.9594807238394965 |
| 8 | 0.1 | 75 | 0.9621621621621622 | 0.9675889500830492 |
| 8 | 0.1 | 90 | 0.9621621621621622 | 0.9675889500830492 |
| 8 | 0.1 | 100 | 0.9621621621621622 | 0.9675889500830492 |
| 8 | 0.1 | 110 | 0.9621621621621622 | 0.9675889500830492 |
| 8 | 0.1 | 120 | 0.9621621621621622 | 0.9675889500830492 |
| 8 | 0.1 | 150 | 0.9621621621621622 | 0.9675889500830492 |
| 8 | 0.5 | 50 | 0.9621621621621622 | 0.9864935746131654 |
| 8 | 0.5 | 75 | 0.9621621621621622 | 0.9864935746131654 |
| 8 | 0.1 | 50 | 0.9621621621621622 | 0.9675889500830492 |
| 8 | 0.5 | 120 | 0.9621621621621622 | 0.9864935746131654 |
| 8 | 0.5 | 90 | 0.9621621621621622 | 0.9864935746131654 |
| 8 | 0.5 | 100 | 0.9621621621621622 | 0.9864935746131654 |
| 8 | 0.5 | 110 | 0.9621621621621622 | 0.9864935746131654 |

Figure 36: Grid Search results for Logistic Regression will full features set

| rank_test_score | param_C | param_max_iter | mean_test_score | mean_train_score |
|---|---|---|---|---|
| 1 | 1.0 | 120 | 0.9675675675675676 | 0.9675889500830492 |
| 1 | 0.1 | 150 | 0.9675675675675676 | 0.9675889500830492 |
| 1 | 1.0 | 110 | 0.9675675675675676 | 0.9675889500830492 |
| 1 | 0.5 | 100 | 0.9675675675675676 | 0.9675889500830492 |
| 1 | 0.1 | 120 | 0.9675675675675676 | 0.9675889500830492 |
| 1 | 0.5 | 50 | 0.9675675675675676 | 0.9675889500830492 |
| 1 | 1.0 | 100 | 0.9675675675675676 | 0.9675889500830492 |
| 1 | 1.0 | 90 | 0.9675675675675676 | 0.9675889500830492 |
| 1 | 0.5 | 120 | 0.9675675675675676 | 0.9675889500830492 |
| 1 | 0.5 | 150 | 0.9675675675675676 | 0.9675889500830492 |
| 1 | 0.5 | 90 | 0.9675675675675676 | 0.9675889500830492 |
| 1 | 1.0 | 50 | 0.9675675675675676 | 0.9675889500830492 |
| 1 | 0.5 | 110 | 0.9675675675675676 | 0.9675889500830492 |
| 1 | 1.0 | 75 | 0.9675675675675676 | 0.9675889500830492 |
| 1 | 0.5 | 75 | 0.9675675675675676 | 0.9675889500830492 |
| 1 | 0.1 | 50 | 0.9675675675675676 | 0.9675889500830492 |
| 1 | 0.1 | 75 | 0.9675675675675676 | 0.9675889500830492 |
| 1 | 0.1 | 90 | 0.9675675675675676 | 0.9675889500830492 |
| 1 | 0.1 | 100 | 0.9675675675675676 | 0.9675889500830492 |
| 1 | 0.1 | 110 | 0.9675675675675676 | 0.9675889500830492 |

Figure 37: Grid Search results for Logistic Regression will limited features set (top features only

| rank_test_score | param_max_depth | param_n_estimators | param_bootstrap | mean_test_score | mean_train_score |
|---|---|---|---|---|---|
| 1 | 5 | 10 | false | 0.9945945945945946 | 1.0 |
| 1 | 10 | 5 | true | 0.9945945945945946 | 0.9946018008567181 |
| 1 | 2 | 15 | false | 0.9945945945945946 | 1.0 |
| 1 | null | 15 | false | 0.9945945945945946 | 1.0 |
| 1 | 50 | 20 | true | 0.9945945945945946 | 1.0 |
| 1 | 50 | 20 | false | 0.9945945945945946 | 1.0 |
| 1 | null | 15 | true | 0.9945945945945946 | 1.0 |
| 8 | 10 | 20 | false | 0.9891891891891892 | 1.0 |
| 8 | 50 | 15 | false | 0.9891891891891892 | 1.0 |
| 8 | null | 10 | true | 0.9891891891891892 | 0.997289972899729 |
| 8 | 10 | 10 | false | 0.9891891891891892 | 1.0 |
| 8 | 10 | 15 | false | 0.9891891891891892 | 1.0 |
| 8 | 10 | 5 | false | 0.9891891891891892 | 1.0 |
| 8 | 2 | 20 | false | 0.9891891891891892 | 1.0 |
| 8 | null | 20 | true | 0.9891891891891892 | 1.0 |
| 8 | 10 | 20 | true | 0.9891891891891892 | 1.0 |
| 8 | 5 | 5 | true | 0.9891891891891892 | 0.997289972899729 |
| 8 | null | 5 | false | 0.9891891891891892 | 1.0 |
| 8 | 50 | 15 | true | 0.9891891891891892 | 1.0 |
| 8 | 50 | 5 | false | 0.9891891891891892 | 1.0 |

Figure 38: Grid Search results for Random Forest

```
Window size: 240 Number of Splits: 4 % Score: 98.494
Window size: 240 Number of Splits: 5 % Score: 98.343
Window size: 240 Number of Splits: 6 % Score: 98.645
Window size: 300 Number of Splits: 4 % Score: 98.524
Window size: 300 Number of Splits: 5 % Score: 98.616
Window size: 300 Number of Splits: 6 % Score: 98.616
Window size: 360 Number of Splits: 4 % Score: 97.72
Window size: 360 Number of Splits: 5 % Score: 98.914
Window size: 360 Number of Splits: 6 % Score: 97.828
```

```
Window size: 240 Number of Splits: 4 % Score: 98.72
Window size: 240 Number of Splits: 5 % Score: 98.795
Window size: 240 Number of Splits: 6 % Score: 98.72
Window size: 300 Number of Splits: 4 % Score: 97.97
Window size: 300 Number of Splits: 5 % Score: 98.524
Window size: 300 Number of Splits: 6 % Score: 98.708
Window size: 360 Number of Splits: 4 % Score: 98.48
Window size: 360 Number of Splits: 5 % Score: 98.697
Window size: 360 Number of Splits: 6 % Score: 97.937
```

```
Window size: 240 Number of Splits: 4 % Score: 98.795
Window size: 240 Number of Splits: 5 % Score: 98.193
Window size: 240 Number of Splits: 6 % Score: 98.494
Window size: 300 Number of Splits: 4 % Score: 98.063
Window size: 300 Number of Splits: 5 % Score: 98.708
Window size: 300 Number of Splits: 6 % Score: 98.801
Window size: 360 Number of Splits: 4 % Score: 98.371
Window size: 360 Number of Splits: 5 % Score: 98.48
Window size: 360 Number of Splits: 6 % Score: 97.72
```

Figure 39: Grid search for Method Hyper Parameters

# C Slug Detection Mis-classified samples

The mis-classified flows were as follow: They are briefly discussed in the Results Section

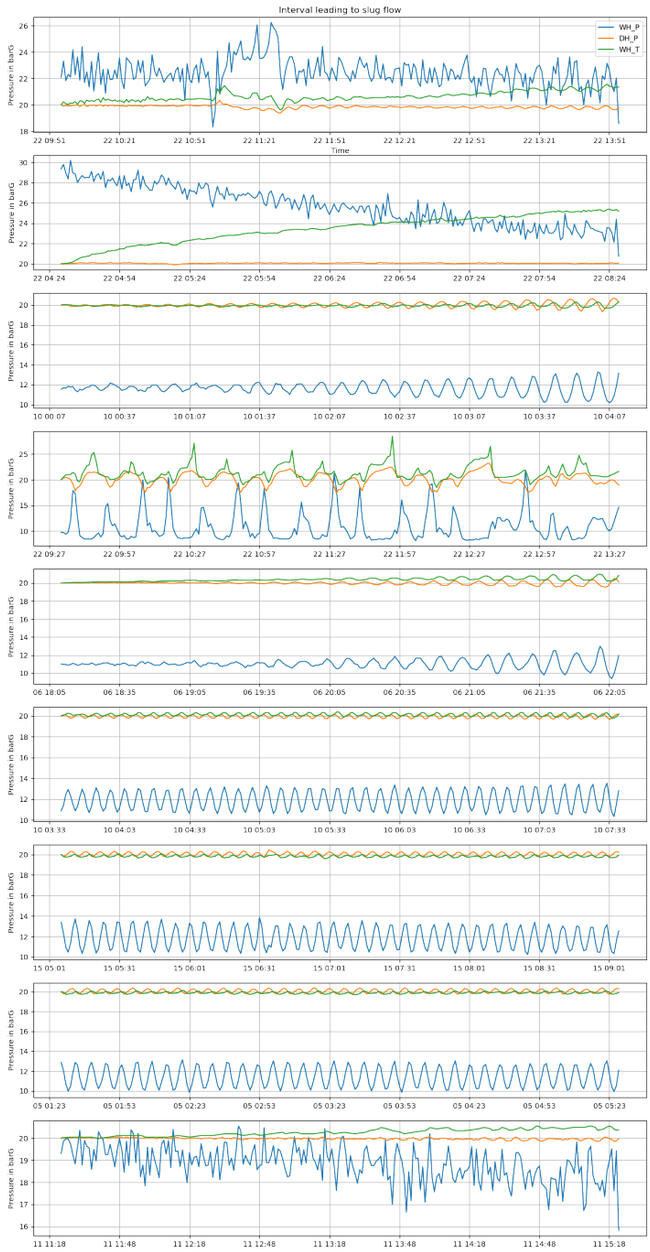
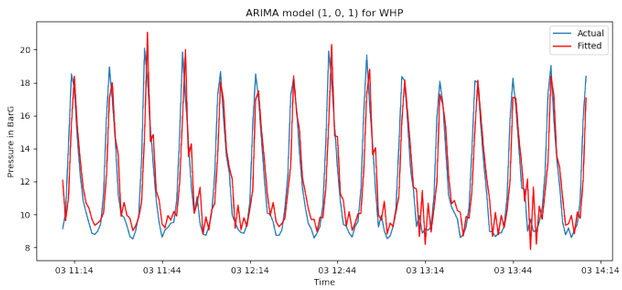


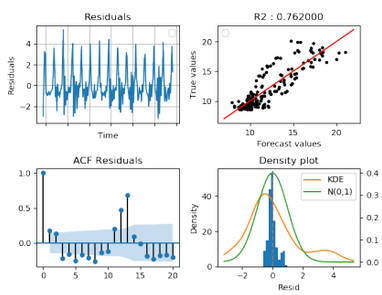

Figure 40: Misclassfied Results for Slug Detection

# D ARIMA Results

## D.1 Parameters: p =1, d = 0, q =1




(a) Fit

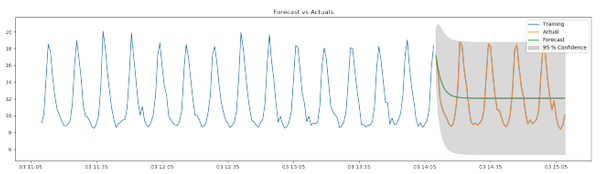



(b) Error information

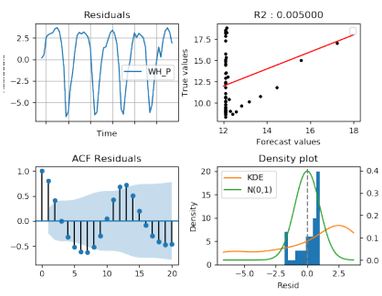
Figure 41: ARIMA model Fit

```
Mean Absolute Percentage Error:  -1.655
Mean Squared Error: 2.741000
Root Mean Squared Error: 1.656000
R2 Determination: 0.762000
```

Figure 42: Error Metrics



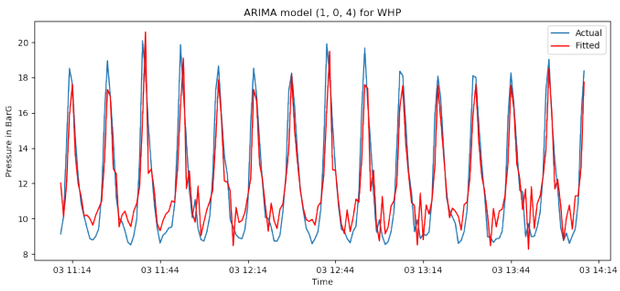

(a) Fit



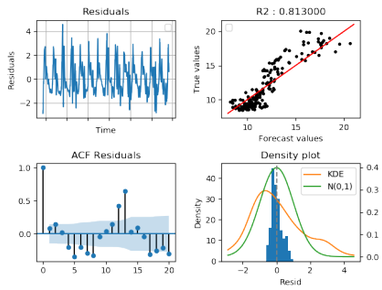

(b) Error information

Figure 43: ARIMA model Forecast

## D.2 Parameters: p =1, d = 0, q =4




(a) Fit

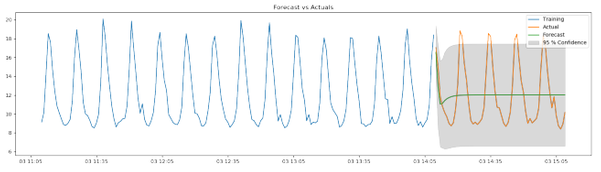



(b) Error information

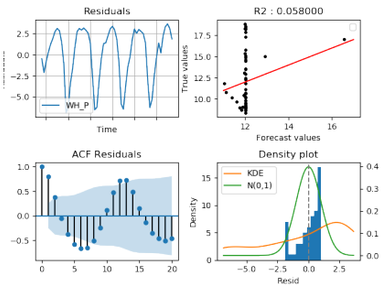
Figure 44: ARIMA model Fit

```
Mean Absolute Percentage Error:  -2.017
Mean Squared Error: 2.153000
Root Mean Squared Error: 1.467000
R2 Determination: 0.813000
```

Figure 45: Error Metrics



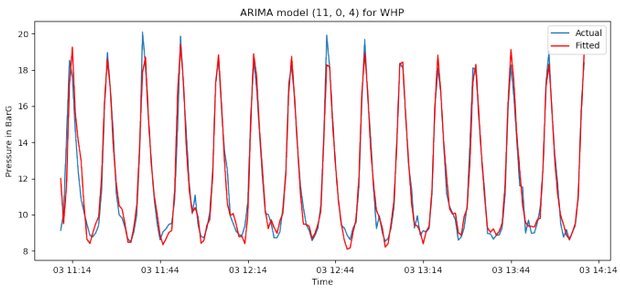

(a) Fit



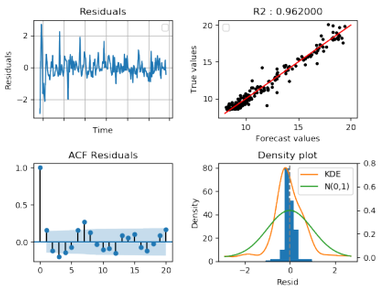

(b) Error information

Figure 46: ARIMA model Forecast

## D.3 Parameters: p =11, d = 0, q =4




(a) Fit

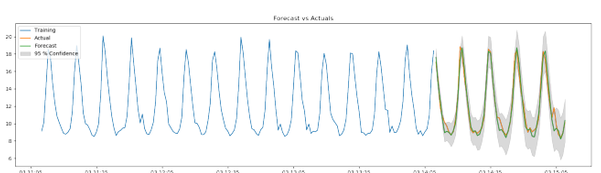



(b) Error information

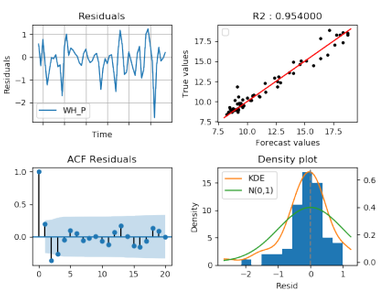
Figure 47: ARIMA model Fit

```
Mean Absolute Percentage Error:  -0.365
Mean Squared Error: 0.437000
Root Mean Squared Error: 0.661000
R2 Determination: 0.962000
```

Figure 48: Error Metrics



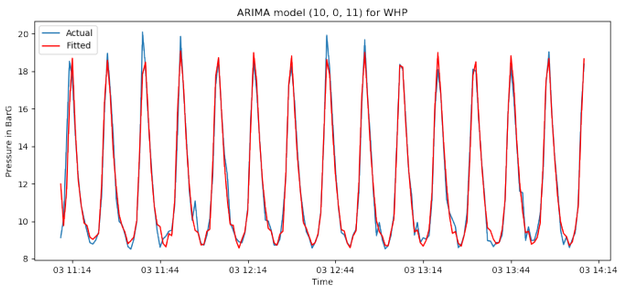

(a) Fit



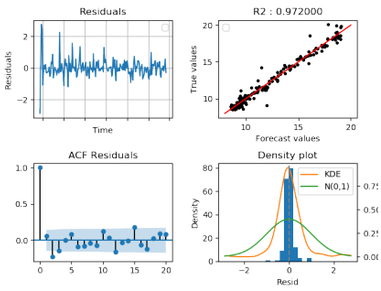

(b) Error information

Figure 49: ARIMA model Forecast

## D.4 Parameters: p =10, d = 0, q =11




(a) Fit

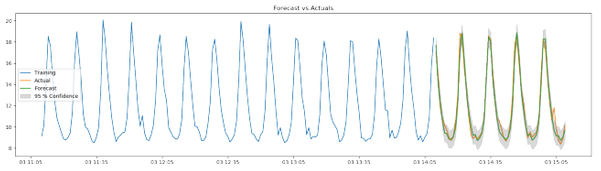



(b) Error information

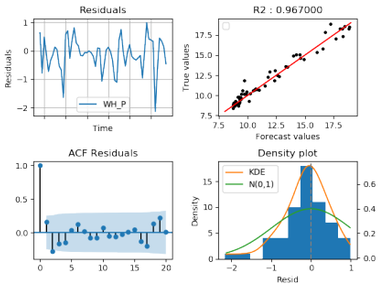
Figure 50: ARIMA model Fit

```
Mean Absolute Percentage Error:  -0.084
Mean Squared Error: 0.323000
Root Mean Squared Error: 0.568000
R2 Determination: 0.972000
```

Figure 51: Error Metrics



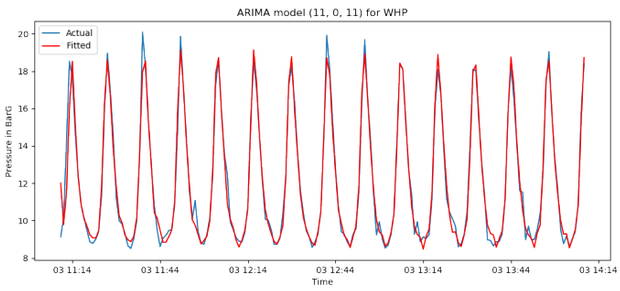

(a) Fit



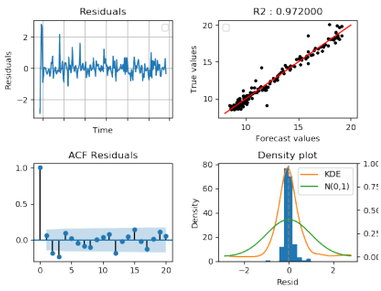

(b) Error information

Figure 52: ARIMA model Forecast

## D.5 Parameters: p =11, d = 0, q =11




(a) Fit

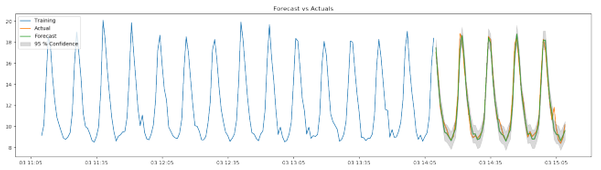



(b) Error information

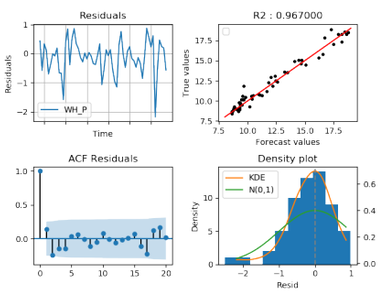
Figure 53: ARIMA model Fit

```
Mean Absolute Percentage Error:  -0.131
Mean Squared Error: 0.321000
Root Mean Squared Error: 0.566000
R2 Determination: 0.972000
```

Figure 54: Error Metrics




(a) Fit




(b) Error information

Figure 55: ARIMA model Forecast