University of Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialization: MSc. Petroleum Engineering	Spring semester, 2019 Open
Authors: Carlos Andres Izurieta Luis Alejandro Rocha Vargas	[illegible] .... Carlos Andres Izurieta [illegible] .... Luis Alejandro Rocha Vargas

Programme coordinator:
Øystein Arild

Supervisors:
UiS - Prof. Dan Sui
AkerBP ASA - Per Seim

Title of master's thesis:
INTEGRATION OF NEURAL NETWORKS AND WELLBORE STABILITY, A MODERN APPROACH TO RECOGNIZE DRILLING PROBLEMS THROUGH COMPUTER VISION AND MACHINE LEARNING

Credits: 30

Keywords: Machine Learning, Computer Vision, Neural Networks, Cavings, Drilling.	Number of pages: 147 + supplemental material/other: 147 Stavanger, 15th June 2019

Carlos Andres Izurieta

Luis Alejandro Rocha Vargas

Integration of Neural Networks and Wellbore Stability, A Modern Approach to Recognize Drilling Problems Through Computer Vision and Machine Learning

Master Thesis Project for the degree of
MSc in Petroleum Engineering

Stavanger, June 2019

University of Stavanger
Faculty of Science and Technology
Department of Energy and Petroleum Engineering

Universitetet
i Stavanger

Abstract

Cavings are an endless and valuable source of information when drilling operations is being performed. Multiple parameters can contribute to produce cavings which indicate that failure has occurred or is about to occur downhole. This project is an integrated study of Machine Learning, Computer Vision, Geology and Photography so that the recognition of cavings in the shaker is possible and how to link the cavings morphology with causal mechanisms related to wellbore instability problems. The aim of this study is to develop a model which is able to extract caving features such as Shape, Edge Definition, Colour and Size.

The methodology consists in developing a structured image database of cavings from the Norwegian Continental Shelf which it is used to extract features. Different algorithms are used to investigate on the best approach to correctly detect the shape of the caving, from supervised learning, K-Nearest Neighbors proved to be the one with the best results, up to 67% of accuracy but not good enough for an object detection project, therefore unsupervised learning was implemented and different architectures of neural networks used. From the simplest with one hidden layer to state-of-the-art architectures as InceptionV3 and YoloV3 were used. The latest proved to be a robust solution to correctly detect the caving with up to 98% of accuracy.

The edge definition feature involved an analysis using computer vision and a mathematical approach to calculate circularity of objects, and this is performed using top view images of the samples where area and perimeter values are relevant. Also, computer vision and k-means method is used to extract the most dominant colours from a caving treating every single image as MxN pixels which are the data points to be clustered. Furthermore, to compute size feature, computer vision and a color scale as a reference object are used to identify caving centroid and calculate width and height of cavings.

A dataset of 1,064 samples of cavings was built from scratch, it was used to successfully train and validate different algorithms and architectures of Neural Networks. Up to 7 different shapes of caving can be recognized with an accuracy of 98% if the proper conditions of the picture are met. Regarding edge definition, the used approach allows us to have circularity values which can be associated with how smooth the edges of the cavings are. This is suitable for non elongated cavings.This model is able to obtain 3 dominant colours from caving, RGB codes and their percentages. Also, measuring the caving size was accomplished with high level of accuracy.

N.B.: This report includes relevant parts of the code used to pre-process the data, manipulate the data, train and validate the different algorithms and neural networks. The complete code has been delivered to the exam office at the University of Stavanger.

Acknowledgments

I would like to thank the University of Stavanger for the support and sponsorship that made possible this project to be completed. To our supervisor Prof. Dan Sui for her advice and supervision and Øystein Arild for always giving the needed support.

To AkerBP for giving the know-how, caving samples, help and a place to work during the realization of this project, without their knowledge this project would not have been possible. Specially to: David Rodrigues, Per Seim, Graham Eaton, Terje Myklebust and Cecilie Edland.

To my family and friends, for the support, the moments and for being there, a sincere thank you.

Carlos Izurieta

First and foremost, I would like to thank God for giving me the strength, knowledge, patience, and opportunity to face this challenging project and to persevere and complete it satisfactorily.

I would first like to thank my thesis advisor Prof. Dan Sui of the Department of Energy and Petroleum Technology at the University of Stavanger for her support and guidance.Also, I would like to acknowledge Øystein Arild, Head of Department of Energy and Petroleum Technology, at the University of Stavanger who have been supportive of this project.

I would also like to thank Aker BP experts who were involved in the continuous improvement and advice for this research project: David Rodrigues, Graham Eaton, Per Seim, Cecilie Edland, and Terje Myklebust.

Finally, I must express my very profound gratitude to my family and friends for providing me with unfailing support and continuous encouragement throughout my years of study.

Luis Rocha

List of Abbreviations

AI	Artificial Intelligence
BGR	Blue Green and Red
BHA	Bottom Hole Assembly
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CV	Computer Vision
FPS	Frames per second
GPU	Graphics Processing Unit
HSV	Hue Saturation Value
IOU	Intersection Over Union
ISO	International Standards Organization
K-NN	K Nearest Neighbors
LWD	Logging While Drilling
ML	Machine Learning
MNIST	Modified National Institute of Standards and Technology
MWD	Measuring While Drilling
NN	Neural Network
PPI	Pixels per Inch
PX	Pixels
RAM	Random Access Memory
RGB	Red Green and Blue
ROI	Region of Interest
ROP	Rate of Penetration
RPM	Revolutions per minute
SL	Supervised Learning
SVM	Support Vector Machines
UL	Unsupervised Learning
WOB	Weight on Bit

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background, Motivation and Challenge

Drilling operations play a vital role during the execution of an oil and gas project by connecting surface with the reservoir in safe conditions, being cost-effective and always trying to maximize the hydrocarbon production. The study of how to optimize drilling can be approached from different perspectives: optimization of drilling parameters [Wallace et al., 2015], drilling detection events, detection of formations [Khojasteh et al., 2015] and facies classification [Bestagini et al., 2017], prediction of torque and drag [Hegde et al., 2015] or prediction of ROP [Ashrafi et al., 2019] but one key aspect among all is the study of wellbore instability.

Wellbore instability can reach up to 41% of the total non-productive-time [York et al., 2009] in drilling operations, thus making it a very important aspect of study towards drilling optimization. The data gathered by the oil and gas operations can be categorized as *Big Data* [Mishra and Datta-Gupta, 2017] because everyday millions of points of data are gathered by well logs, seismic lines, surface production systems, simulations models, just to mention a few. However, all this data is not being completely used by the front user, only a small, reduced, processed amount is displayed for interpretation in the form of maps, plots, trends and other visual aids.

Caving detection by the use of computer vision to get information from the shape has been addressed before [Galvis et al., 2014] [Skea et al., 2018] [Han et al., 2018] with 3 different shapes and a experimental setup based on laser technology; however, there are other important parameters that need to be taken a second look: roundness, color and size. This experimental setup is constrained to the equipment and technology used to detect the caving. The present project aims to enable portable devices to use its built-in camera to capture pictures and video and analyze it with computer vision.

The challenge is to reduce the time between detecting the caving and taking corrective measurements, judge if the drilling operations are about to have a problem or if they already have a problem. Even though well control events have been studied previously [Unrau et al., 2017] the applicability depends on the recorded database of drill events and the instrumentation used to record, gather and store the data making it difficult to reproduce for other companies. Cavings on the other hand can be present at any well and the samples can be collected and stored by any operator company. The main considerations while analyzing a caving at surface are:

- Should there be cavings?
- How much time has it been since the caving was released into the wellbore? Time between production from the formation and its arrival at surface, also known as travel time.
- Where does the caving come from? Is it from the same formation the bit is in contact at the moment?
- What is the failure mechanism? The physical reason for the caving to exist.
- Why does the caving have the form it has?

Part of the motivation is to optimize drilling by taking the right steps into a safer drilling conditions, this means to reduce the production of cavings. As a material that is "caving into" the wellbore, its presence is not planned and it is unwanted, the result of drilling should only be the expected cuttings for analysis.

1.2 Objectives and Scope

The present study focuses on detecting different features of rock samples called *"Cavings"*, these will be considered as pieces of rocks significantly bigger than the regular cuttings and often related with wellbore deterioration while drilling a well. Even though the project can be very complex and long in terms of what can be deployed as a solution for AkerBP, the present project needs to be defined within a time-frame of 5 months in order to get the best results and prove that the first steps towards extracting information from cavings and giving possible solutions to the drilling problems can be done with machine learning.

To reflect these thoughts, the following objectives are proposed:

- To make use of the available samples of cavings to build a structured database of pictures.
- To extract information from the pictures, this will be called features for the project and the most relevant are: shape, roundness, size and color.
- To use supervised and unsupervised learning to train different machine learning models with the information extracted from the features to identify the type of caving.
- To identify the root cause and the failure mechanism by using pictures, video or live stream video of cavings.

The selected features: shape, roundness, size and colour are chosen because it contains the most basic information related with the interaction of the wellbore with the rock. Size would give an idea of the volume of rock detached from the walls, roundness can tell if the transport of particles (hole cleaning) is the correct for a given size of caving and the mud rate (washout and packing problems), color can indicate the formation the caving belongs to and if the caving matches the formation that is being currently drilled and last, shape is related with the interaction of differential pressure of the wellbore and the rock (wellbore instability).

The project will use 2D pictures of cavings and the coding will be done in Python; the code can be run in windows or linux as the different architectures of neural networks (NN) might require a more natural environment for compilation. The final step is to implement caving detection using a mobile device as the input parameter for video and live-stream video.

1.3 Methodology

Coding is the backbone of the project, all the writing will be done in Jupyter [Kluyver et al., 2016] as it can handle several programming languages and allows the use of environments for the used version packages. The preferred programming language is Python [Van Rossum and Drake, 2011] due to its development, resources and widely support from developers and enthusiasts. A complete guide on how to start using Python and Jupyter can be found in [Galea and Capelo, 2018]. The most important packages that will be used and its versions are summarized in Appendix B.1, this is shown for reproducibility.

Since there is no database for cavings, this will be build using samples provided by AkerBP; all of the samples will be washed and clean, this decision is taken because one of the features to be extracted will be colour. 2D photographs of single samples will be stored in JPEG with the highest possible quality of picture, this format is used due to limitations with the camera and no RAW pictures will be stored.

The manipulation of the images, extraction of information, standardization and storage of data will be done using python and the mentioned packages. With the information from the images, the project aims to extract four different features: shape, edge definition (roundness), size and colour.

The machine learning algorithms, classifiers, will be fed with the new database, training and evaluation will be performed and depending of the performance of the classifiers, the project will focus on developing the most promising to improve accuracy. The best method to compare performance is the use of a confusion matrix among the classes to see how the algorithm learned to recognize them.

Computer vision will be used to judge the size of the caving, this means extract height and width of every caving. A way to do it is to extract the borders of the caving and calculate the centroid, and from it calculate the distance to the borders to establish the half-leght of width and height.

The method of K-mean clustering is useful to associate the different pixels that composes the caving and establish a pre-determined number of clusters, number of dominant colours, to calculate the closeness of all the pixels to the centroid of the cluster and thus determine the percentage of the dominant colors.

Edge definition will be calculated by calculating the roundness of the sample or how close to a perfect circle the sample is. In other terms, the use of the perimeter in relation with the internal area. This calculation needs to address the elongated cavings and the tabular cavings as the first tend to depart from sphericity and the latter can be very angular but close to a sphere.

1.4 Limitations and Blockers

One of the main limitation with this project is the data, it is difficult to obtain caving samples without having the support from an operator company. AkerBP has agreed to provide as much samples as they can, however the dataset needs to be balanced and selecting the needed type of caving becomes a great task as these are not to be picked, the well with a problem will produce a certain type of caving but certain shapes are more common than others.

Another limitation is the computer power, using regular laptops will perform well on basic tasks like plotting a function or loading a picture but when the calculation involves thousands of pictures and thousands of data points per picture the limitation might be on how much RAM memory the machine can allocate to the task and if the calculation surpasses the available memory it can cause the computer to crash.

The use of data mining for extracting features and later use in NN can become problem if the performance of the computer is slow, and it can give the results in a matter of days or even a week instead of a few hours; as the accuracy of the model is revised after the NN is trained and validated, the changes to get the expected results might involve another iteration in training and testing, thus making it a long process that depends on the specifications of the machine.

The final limitation of the project is the database itself, while the applicability of the the studies relies on pictures of cavings that will be taking on-site, the final database will contain only one sample per image, the sample will be cleaned and the background will always be white. This makes it difficult to translate its full potential to application where the pictures does not meet the specifications. There are two possible solutions to this inconvenient:

- Change the photography conditions.
- Use image/data augmentation.

The first option is more suited for complex architectures of NN and this project will try to concentrate the efforts on keeping the problem and the solutions as simple as possible. In the event that the simplest approach is not enough to extract the features from the cavings, the dataset will not be a problem for the NN as state-of-the-art architectures are designed to handle complex photographs and it will have no problem by using the simplest dataset.

The second option is more suitable when the dataset is not big enough to define the different classes the project is trying to classify or detect. By using this option, one single picture can become several by using different filters and transformation of the original under different conditions that will depart from the ideal picture. Some of the transformations are: different light conditions, blur, size, orientation, mask color, saturation and others.

Chapter 2

State of the Art

2.1 The Role of Machine Learning in the Oil & Gas Industry

During the last two decades, there has been two major oil crisis: in 2008 and in 2014. Both of them had significant impact on the operations and the best indication is the worldwide rig count [Hughes, 2019]. As seen in Figure 2.1 whenever there is a crisis the number of rigs drops dramatically and the recovery can be fast as in 2010 or slow as in 2017 and onward. However, if the number of papers related to "Machine Learning" [SPE, 2019] in the oil and gas industry is plotted, there is a sharp increase in the last 5 years. Thus, the research towards digitalization and the potential of data science has no relation with the price per barrel of oil, it only means that the effort on combining the two disciplines is increasing.

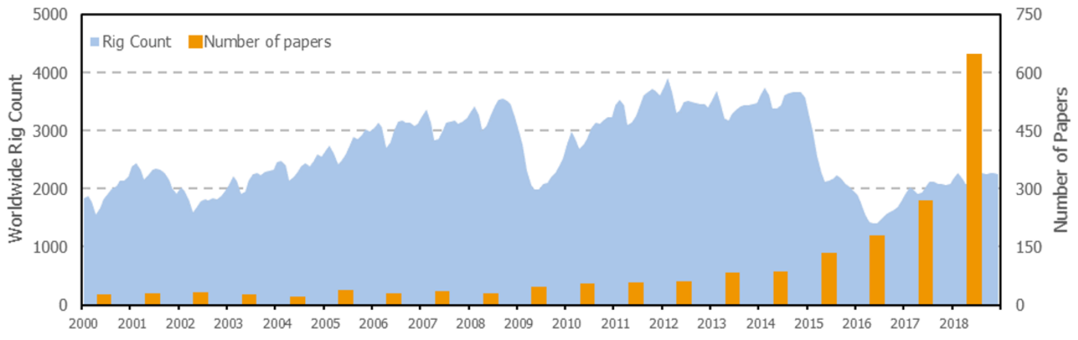


Figure 2.1: Worldwide Rig count and number of papers related to "Machine Learning" published in www.onepetro.org

The oil and gas is moving towards a safer and optimized environment where the uncertainty of the operations and analysis needs to be reduced. Now that the computing power is able to analyze all the data gathered through the decades of operation, the E&P companies are looking to extract the most out of their information. The new roles in the industry are the Data Scientists and Data Engineers, both with different roles and yet similar capabilities, the main differences is that the Data Engineer needs to be the one with experience in the sector, the one who can tell that the ROP cannot have negative values.

The Oil & Gas business is definitely an example of Big Data due to its velocity, volume and variety [Mishra and Datta-Gupta, 2017], and all this data is gathered through high speed sensor, translating pressure, fluid rate, temperature, vibrations into points of data that are stored. The principal function of these data is to predict different events such as: maintenance of equipment, well rate decline, problems or unwanted events and the way of doing it is by creating models.

A model can be made from using the laws of physics but in machine learning, the user replaces the law of physics by using the data as input material for computer learning algorithm as if it were *experience* that the machine needs to understand to recognize different patterns, behaviors and the meaning of concepts. Normally all this *experience* would be written as a matrix denoted X with x_i elements or *features*, the complex relationship of the features is what creates the response Y or *class* output.

If the model can learn to reproduce the outcome, then the user can choose to optimize the operations by finding the combination of features that created the label or event: *problem*. There is a great interest in the digital twin, a model that runs in parallel with the normal operations but tries to be ahead of the real time. This look at the *future* prevents the real operation of behaving towards a problem event, this is another example of optimization.

There is plenty of applications of machine learning in the oil and gas industry, just to mention a few: decline curve analysis [Li et al., 2017] [Cao et al., 2016], field development [Tandon et al., 2019] , prediction of life-span on artificial lift equipment [Sneed, 2017], etc. This means that any operation that can gather data can be modelled and optimized through ML. Among the most important libraries in Python for machine learning and deep learning projects are:

- Scikit-Learn
- Numpy
- Keras
- Imutils
- Matplotlib
- OpenCV
- Tensorflow
- Pandas

2.2 The Need to Study Cavings, Wellbore Stability

Wellbore stability can cost up to 1000 million dollars per year to the Oil & Gas Industry [Chen et al., 2003] and in a frame of cost optimization and safer drilling conditions, it makes it a challenging field of study. There are two aspects in wellbore stability: the factors that can be controlled and the factors that cannot. The first are the fluid pressure while drilling and the chemical composition of the mud; the factors that cannot be controlled are the intrinsic parameters from the rock, formation and earth stresses [Cheatham Jr et al., 1984] [Aadnoy et al., 1987]. Figure 2.2 summarizes some of the problems a well can expect in different formations due to the interaction of the fluid pressure and earth stresses. Some of the problems are:

Cause	**Effect**
Hydration of swelling shales	Hole Enlargement
Excessive wellbore pressure	Fracture - Lost circulation
Low wellbore pressure	Hole collapse or Blowout
Excessive production rates	Solid particle influx & Hole enlargement

Table 2.1: Wellbore instabilities causes, modified from [Cheatham Jr et al., 1984]

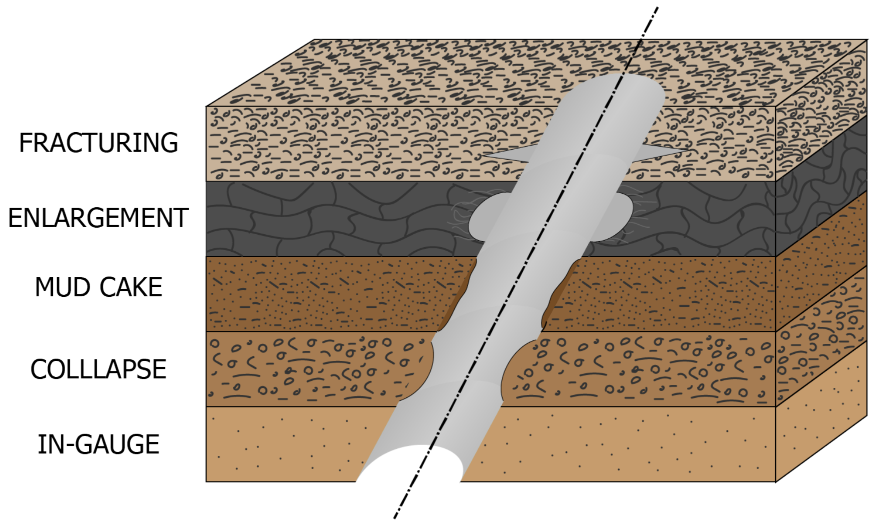


Figure 2.2: Wellbore instability, modified from [Cheatham Jr et al., 1984] after [Bradley, 1979]

The ideal condition while drilling a well is to drill under normal conditions, which can be difficult to define but to the extent of this project it will be understood as drilling inside the mud window, *"stable borehole"*. This means that the pressure in the wellbore will never exceed the fracture pressure nor the pore pressure.

The differential pressure between the hydrostatic mud weight and the rock stress can be seen in Figure 2.3. Shear failure is the responsible of making the elongated or concave-convex cavings, the excess of pressure in the rock makes it to fail and most of the times, the face of the wellbore can be identified in a closer inspection of the caving. If the difference in pressure is low enough, the cavings will tend to have a pyramidal or triangular shape with angular edges.

The previously defined *"stable borehole"* can be seen in Figure 2.4 as the green region defined within the pore pressure and the fracture pressure. This region is used to select the casing setting depth which represent the length of casing to be used in each section and the integrity of the wellbore. Part of the optimization is to reduce the number of sections while reassuring wellbore integrity according to company and government standards.

The probability of having a problem can be calculated from the risk analysis perception [Moos et al., 2003], where a simplified version of the problems described in Table 2.1 can be

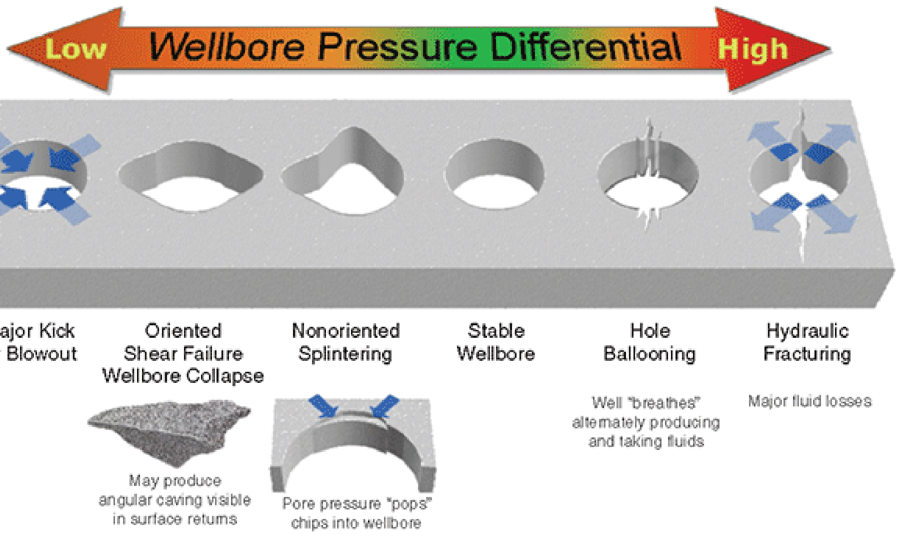
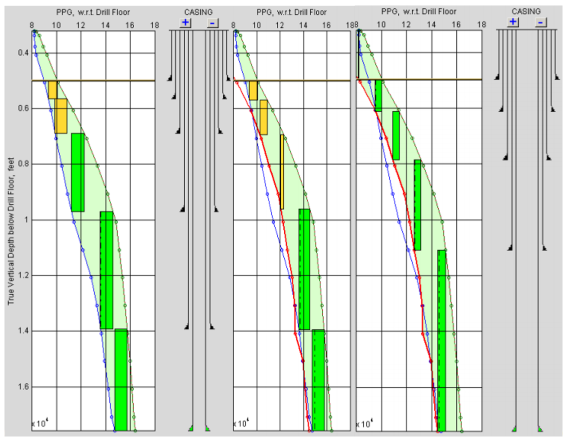
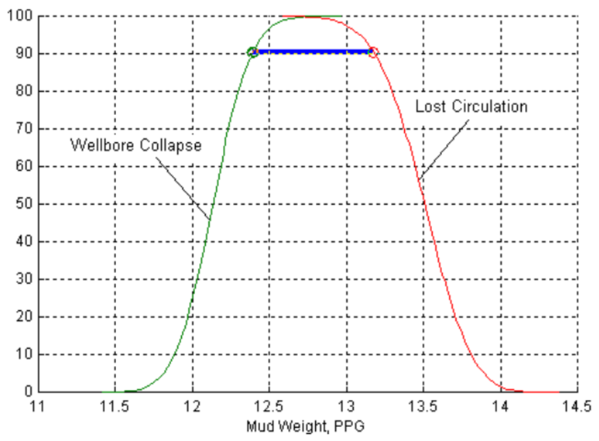

Low
Wellbore Pressure Differential
High
Major Kick or Blowout
Oriented Shear Failure Wellbore Collapse
Nonoriented Splintering
Stable Wellbore
Hole Ballooning
Hydraulic Fracturing
May produce angular caving visible in surface returns
Pore pressure "pops" chips into wellbore
Well "breathes" alternately producing and taking fluids
Major fluid losses

Figure 2.3: Wellbore pressure differential, taken from [Caenn, 2015] adapted from [Zhang et al., 2008]



(a) Mud window



(b) Probability of avoiding a problem

Figure 2.4: Stable wellbore, taken from [Moos et al., 2003]

seen as a function of the used mud weight and the probability of avoiding the problem. The stable borehole can be seen as a blue line with a 90% chance of avoiding wellbore collapse and lost circulation.

The shape of the cavings are linked to the interaction of the forces in the wellbore and the formations; by taking into account the different events, wellbore characteristics and environment conditions, the shape of the cavings can be linked with these conditions. The chord diagram presented in Figure 2.5 is an example of how the drilling parameters and conditions are related with the wellbore stability problems. A close inspection to the roundness will tell that any can be present in any wellbore stability problem, however a caving related with Mud Chem-

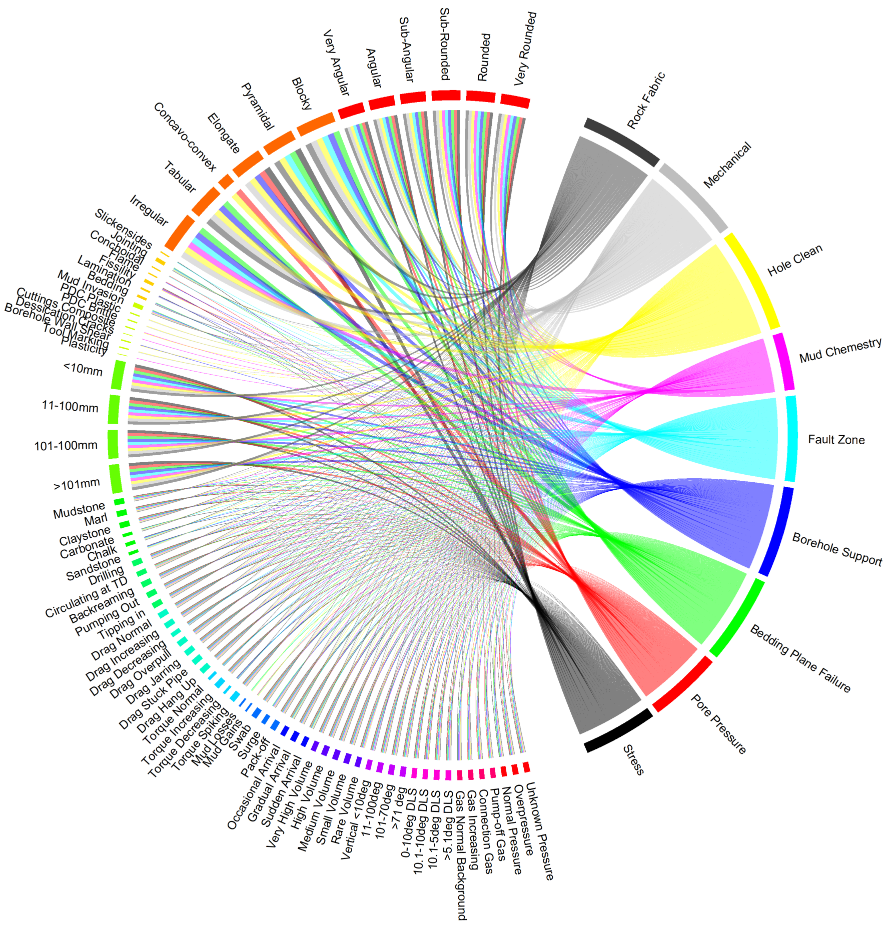


Figure 2.5: Drilling parameters linked to wellbore stability problems, modified from AkerBP data

istry cannot be Very Angular, only rounded. When it comes to shape, certain type of shapes are more related with some specific problems: Blocky, Pyramidal and Irregular shapes are common for a Fault Zone related problem while the Elongated and Concave/Convex is related with Pore Pressure. Figure 2.5 can be seen in detail in Appendix E.1.

This interpretation is based on observation and needs to be validated with statistics and a structured database, thus it does not represent a ground truth, the input of the drilling crew, the geologist and the analysis of all the parameters at the same time are needed.

2.3 Computer Vision and Photography

Computer Vision and Photography play an important role in this project in addition to Geology, Drilling, Neural Networks and Machine Learning. Essentially, knowing the basic concepts in relation to these fields helps to reach the objectives for this work. CV is an extensive field which is about how computers can be able to understand and extract information from videos and images, and this involves a lot of processes; also, it is used to automate different operations that a human can do [Solem, 2012]. This technology is not new since there are many applications in fields like health, manufacturing, etc.. However, there has been a preference to use machine learning and deep learning for O&G applications such as facies prediction [Bestagini et al., 2017] and process for rock classification [Sidahmed et al., 2017], but computer vision is an emerging technology in O&G digitalization more often in the recent years. Besides, it has been used for recognizing cuttings in 3D environments [Han et al., 2018] and caving depth prediction [Galvis et al., 2014]. In addition, understanding photography is essential because this has a big impact on the results; as a result, basic concepts related to this Caving project will be explained in this section.

2.3.1 OpenCV functions and Applications

The project is developed using Python as programming language, and one of the big advantages is that it has many open source libraries. One of this library is OpenCV (Open Source Computer Vision Library) which has application for computer vision and machine learning providing tools for recognizing objects, persons, animals and so on from images and videos [Bradski, 2000]. As a result, an explanation will be provided in this section about the tools from this library used in this project.

Image Processing

Image processing is fundamental to perform different calculations and extract information from images and videos. There are multiple tools to do this, and this depends on the desired outcome. A brief explanation of the tools used in this project based on the OpenCV Manual[Itseez, 2014].

- **Contour Features:** This tool is powerful because it allows us to extract information like perimeter, area, bounding box and centroid. First, moment parameters are very helpful to calculate the center of an object, and they are also used to find object area. Besides, contour perimeter option is essential for roundness feature, but it is highly important to explain the difference about perimeter and convex perimeter. Convex hull is the smallest convex shape which consists in curves around the object (Figure 2.6). The convex perimeter is the perimeter of the convex hull which contains the object (Figure 2.7). The normal perimeter is calculated around the whole object edge.

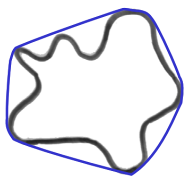


Figure 2.6: Convex Hull taken from [Wirth, 2001]

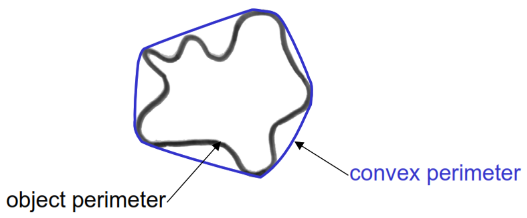


Figure 2.7: Difference between convex perimeter and object perimeter taken from [Wirth, 2001]

- **Image Thresholding:** A fundamental step to work with images is establishing thresholds. The concept is very simple since a value is established as a limit; as a result, if the pixel value is higher than this limit, an outcome is produced which could change pixel to black or white color. The tools used for this project are *BINARY* and *BINARY_INV*.

- **Smoothing Images:** All images have some noise, and filtering these is important to process the pictures. *Blurring* option is used for color and roundness feature, and this uses low-pass filter kernel; as a result, remove noise with high frequency.

- **Morphological Transformations:** Using this option, multiple operations are performed employing binary images. *EROSION* is used, as its name says, to erode pixels based on kernel principle. Also, *DILATION* does the opposite of the previous option, which is increasing the white pixel values.

- **Histograms:** It is a type of plot or graph which is used to extract information from images about the RGB or Binary pixel values.

Imutils Package

This package is being used widely to develop algorithms in this project in order to extract features such as size, colour and roundness. It is a powerful package which is open source, and it has multiple functions regarding image processing such as re-sizing, translation, etc. [Rosebrock, 2015]. Besides, it is applicable to OpenCV to work together.

Camera Calibration

The concept of calibrating camera for using Computer Vision needs to be clarified, and there are some terms which needs to be taken into account. All cameras generate distortion, and their properties influence on the final result of images. There are two important distortions. The first one is *Radial Distortion* is related to the perception that straight lines in a picture looks like curves. The second distortion is *Tangential Distortion* refers to the misalignment between the camera lens and the image plane. This can generate that the real distance is not the right one. There are some algorithms to avoid or correct these type of distortions. In this project, using the right white-box setup and lighting conditions help to minimize these distortions.

2.3.2 Photography Concepts

It is well-known that photography is an extensive field, and it provides many options to obtain good quality images which are the input information for CV and ML. There are intrinsic and extrinsic parameters of the camera [Itseez, 2014]. Intrinsic parameters refers specifically to a camera which are focal length and optical centers, and extrinsic parameters are related to the translation of coordinates between 3D points and coordinate system. In this section, some important parameters, which have been used and modified in order to look for optimal images results based on [Peterson, 2016], will be discussed.

- **ISO:** This term refers to the camera sensor sensitivity to light. A higher ISO increases the sensitivity, and less light is required to produced the right exposure; on the contrary, a lower ISO produces the opposite situation. Also, it is important to know that a high ISO generate more light and more noise in pictures. As a result, low ISO is desired, but this should be balanced in relation to other parameters like aperture and shutter speed.

- **Shutter Speed:** This parameter refers to how long the camera shutter remains open so that it determines the amount of light which enters to the camera. If the camera shutter is left open more time, more light enters to the camera. In other words, this can be explained as if faster shutter speed is set, the result is less light and freeze movement. On the other hand, a slow shutter speed is set, the result is more light and blur movement.

- **Aperture:** This is related to the lens aperture which has a diaphragm in the lens. This parameter regulates the amount of light that passes through the lens. In conclusion, a large aperture means that more light and shallow depth of field are obtained. Conversely, a small aperture means that less light and large depth of field are obtained.

- **Chromatic Aberration:** This is a type of optical distortion caused by the impossibility of a lens to focus all colors at a single point of convergence.There are methods to reduce this situation, and there many cameras that already have an included system to compensate this distortion. For this project, the camera which was used has an automatic system to reduce this distortion.

2.3.3 Image/Data Augmentation

For the purposes of this project, image augmentation is a crucial part in developing a good dataset that can be used with the ML algorithms, this is achieved by generating minor changes to the images such as: rotation, re-size, change of brightness, cropping, mirroring and others; all these differences are not part of the dataset that has a limited set of conditions. This will result in a substantial increment of the dataset; most of the standard datasets have tens of thousands samples that can feed with enough variety of conditions.

As discussed by [Hernández-García and König, 2018], image augmentation does have a positive impact in the performance of the classifiers and it allows it to increase the accuracy between 5% and 12% depending on the dataset and how heavy the image augmentation is. The creation of synthetic data can go beyond the flipping or re-sizing images, as shown by [Zhu et al., 2017]. Images in Figure are an example of image translation where two inputs are used to create a new image that does not exist in the real world.

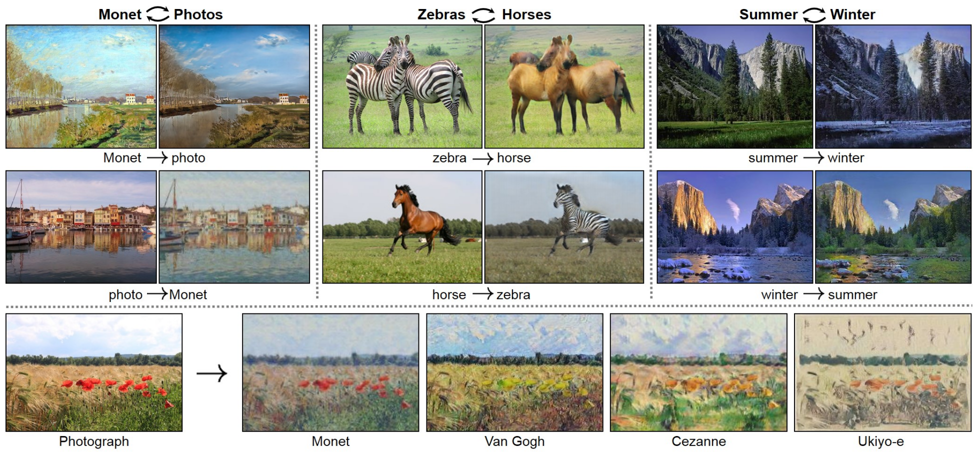


Figure 2.8: Example of image augmentation

2.3.4 Object Detection

To define an object in a picture it is necessary to recognize the area of the object in it. This can be done with a bounding box that delimits the top, bottom, right and left margin of the object. It is within the interest of this project to train a neural network with the information of bounding boxes encapsulating the caving samples.

When an image with a specific object in it is used to train the algorithm, there will always be two boxes: the ground truth and the prediction box (see Figure 2.8). In the ideal case that the algorithm is well trained, both boxes will superpose. The parameter to measure how well the algorithm is performing is to use the *Intersection Over Union*, or simply IoU. This metric is obtained by dividing the area of the overlap by the area of the union.

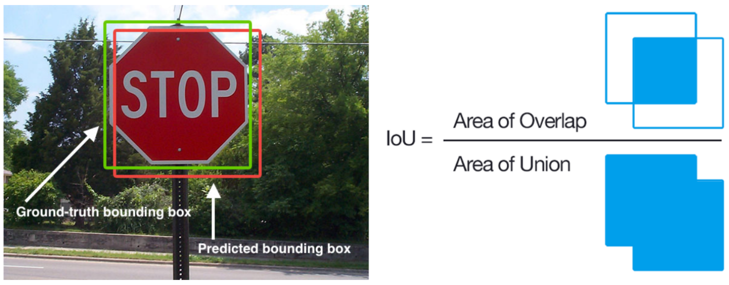


Figure 2.9: Intersection Over Union, taken from [Rosebrock, 2017b]

Depending on the value of IoU, the performance of the algorithm can be quantified. Values close to 1 will be the ones useful for the project and less than 0.5 will have a poor performance as seen in Figure 2.10, this would mean that the algorithm needs more training.

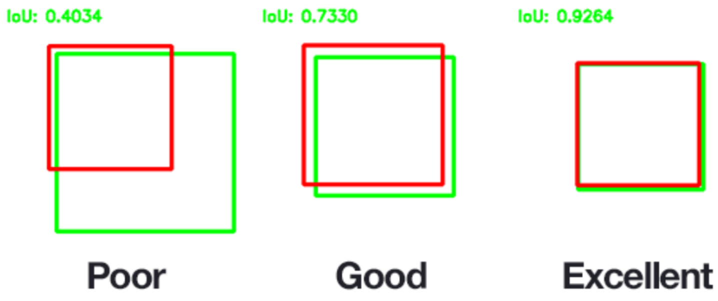


Figure 2.10: Examples of IoU, taken from [Rosebrock, 2017b]

Part of the metrics in object detection are:

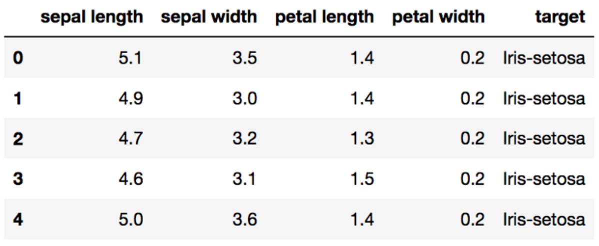
- $Object$**:** It represents the probability that the bounding box contains the object of interest.
- $Class$**:** It represents the probability that the detected object corresponds to the right class.
- $Class|Object$**:** It represents the probability that the detected object corresponds to the right class given that the object is present.

2.4 Supervised Learning

As defined by [Galea and Capelo, 2018]: "*Supervised Learning Techniques center on mapping some set of information to another by providing the training process with the input information and the desired outputs, then checking its ability to provide the correct result*". The basic applications of supervised learning is classification or sorting of data, teaching an algorithm how to differentiate between apples and oranges, a digit from another, dogs and cats or the face of a famous actor from others and then return the the output that the user stated is the correct classification or sorting. To summarize in supervised learning the input is provided and the algorithm is training this information to corresponding known answers.

There are some concepts and methods that will be recurrently used and are worth to mention:

- **Features:** Set of characteristics from the input data, this can be any distinctive characteristic that makes the class unique, by its value, and can help distinguish it from the rest of the dataset. The ML algorithms will look for the combination of these characteristics and their values for each class.

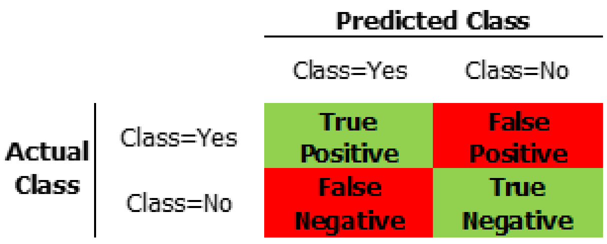


	sepal length	sepal width	petal length	petal width	target
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Figure 2.11: Example of features, IRIS dataset

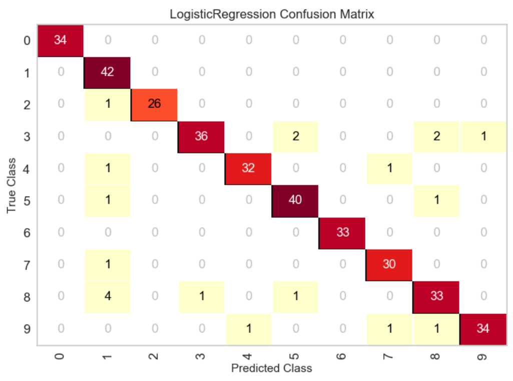
- **Confusion Matrix:** Design to count false positives and false negatives, it will summarize the number of correct outcomes from the algorithms, it is also know as a score visualizer. Ideally a confusion matrix of a well trained model will tend to show a diagonal with the test data correctly distributed among the classes.

  The simplest form of a confusion matrix can be seen in Figure 2.12, the real or actual classes are listed on the left as rows and the predicted classes are columns. Figure 2.13 shows a confusion matrix with 10 classes with good accuracy on prediction and some false positives and false negatives. Correctly predicted positive classes are known as *True Positives* and the correctly predicted negative classes as *True Negatives*, both values will try to be as large as possible in a defined algorithm. *False Positives* and *False Negatives* are classes where the predicted class is different from the actual class.



		Predicted Class	
		Class=Yes	Class=No
Actual Class	Class=Yes	True Positive	False Positive
	Class=No	False Negative	True Negative

Figure 2.12: Confusion Matrix - Simplest form



LogisticRegression Confusion Matrix

True Class \ Predicted Class	0	1	2	3	4	5	6	7	8	9
0	34	0	0	0	0	0	0	0	0	0
1	0	42	0	0	0	0	0	0	0	0
2	0	1	26	0	0	0	0	0	0	0
3	0	0	0	36	0	2	0	0	2	1
4	0	1	0	0	32	0	0	1	0	0
5	0	1	0	0	0	40	0	0	1	0
6	0	0	0	0	0	0	33	0	0	0
7	0	1	0	0	0	0	0	30	0	0
8	0	4	0	1	0	1	0	0	33	0
9	0	0	0	0	1	0	0	1	1	34

Figure 2.13: Confusion Matrix, taken from [Yellowbrick, 2018]

- **Overfitting:** As defined by [Perner, 2007], *"A machine learning model is said overfit the training data relative to a simplest model if the first model is more accurate on the training data but less accurate on the test data."* This means that even though the model has a perfect or almost perfect accuracy with the training samples, it fails when a new samples is brought for testing. This problem is often related with how the dataset is structured, if the dataset is balanced or not or if the conditions of the dataset does not meet the reality of the application.

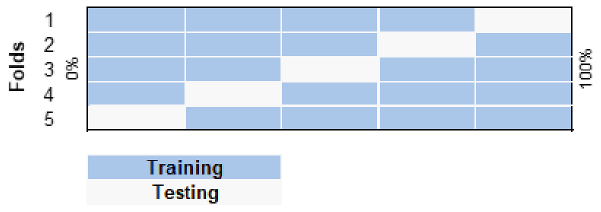


Figure 2.14: K-fold cross validation

- **Cross Validation:** Is a common technique for getting the most out of the dataset for training. Every time a model is trained, a part is removed to be used as testing data; and this is a portion of the dataset that can hold important information for the model to learn, a consequence of this loss is an under-fitting model. In the present study, this problem is faced by using K-fold cross validation. As seen in Figure 2.14 the dataset is separated in different folds and each will be tested using a portion of the dataset while the rest is being used for training. This will not only increase the accuracy but it will reduce the variance.

- **Evaluation Metrics:**

  To evaluate a model, different parameters need to be revised. A model performs well not only because it presents a good accuracy but because it can be applicable. The accuracy is defined as Equation 2.1 where the interest is to know from all the samples in the dataset, how many were correctly predicted.

$$Accuracy = \frac{true\_positives + true\_negatives}{total\_samples} \tag{2.1}$$

  However, it is good to know how many of the positive values are correct (see Equation 2.2), the Precision will calculate the percentage over all the positive classes the model predicted (true positives and false positives). A good model will try to have as less false positives as possible in order to be reliable.

$$Precision = \frac{true\_positives}{true\_positives + false\_positives} \tag{2.2}$$

  Recall is also useful when the model predicts false negatives (see Equation 2.3), this means that in reality the class is NO but the model predicts it as YES, in an alarming system this can be concerning as the sensitivity of the model needs to be attached to real events that truly triggers the alarm. Recall needs to be as high as possible to reduce the false alarm incidents.

$$Recall = \frac{true\_positives}{true\_positives + false\_negatives} \tag{2.3}$$

  The way to evaluate all the previous parameters at the same time is by using the F1 Score (see Equation 2.4). This metric will evaluate how many false positives and false negatives the model has, a good model will try to have a F1 score close to 1 meaning that it can correctly identify the classes when detected and that it can avoid false alarms events.

$$f1\_score = 2 * \frac{precision * recall}{precision + recall} \tag{2.4}$$

The most known supervised algorithms are:

- Support Vector Machines
- Linear Regression
- K-Nearest Neighbors
- Random Forest
- Decision Tree

This project will use all of the previously mentioned and more but only KNN will be explained as during the investigation it was the one that gave the best results. Among the different algorithms, there are *"Classification Algorithms"* and *"Regression Algorithms"*. The different learning methods have their similarities and their differences, the loss function, parameter estimation algorithm, the model complexity reduction, regression function shape, predicion algorithms and other details can be seen in Tables 2.2 and 2.3.

2.4.1 K Nearest Neighbors (K-NN)

K-NN is a method or algorithm based on similarity of the data, by creating a defined number clusters, it will classify the data using the provided features and the closeness of one sample to the others in the same category. According to [Parsian, 2015] K-NN is: "*...an algorithm for classifying n-dimensional objects based on their similarity to the other n-dimensional objects. In machine learning, Nearest Neighbors analysis has been developed as a way to recognize patterns of data without requiring an exact match to any stored patterns or objects. Similar n-dimensional objects are near each other and dissimilar n-dimensional objects are distant from each other. Thus the distance is between two cases is a measure of their dissimilarity.*"

Learning Method	**Generative or Discriminative**	**Loss Function**	**Decision Boundary**	**Parameter Estimation Algorithm**	**Model Complexity Reduction**
Gaussian Naive Bayes	Generative	-log*(X,Y)*	Equal variance: linear boundary. Unequal variance: quadratic boundary	Estimate $\hat{\mu}$, σ^2, and *P(Y)* using maximum likelihood	Place prior on parameters and use MAP estimator
Logistic Regression	Discriminative	-log*P(Y|X)*	Linear	No closed form estimate. Optimize objective function using gradient descent	L_2 regularization
Decision Trees	Discriminative	Either -log*P(Y|X)* or zero-one loss	Axis-aligned partition of feature space	Many algorithms: ID2, CART, C4.5	Prune tree or limit tree depth
K-Nearest Neighbors	Discriminative	zero-one loss	Arbitrarily complicates	Must store all training data to classify new points. Choose *K* using cross validation	Increase *K*
Support Vector Machines (with slack variables, no kernel)	Discriminative	hing loss: $\|1-y(w^Tx)\|_+$	linear (depends on kernel)	Solve quadratic program to find boundary that maximizes margin	Reduce *C*
Boosting (with decision stumps)	Discriminative	exponential loss: $exp\{-yf(x)\}$	Axis-aligned partition of feature space	Adaboost	Reduce the number of iterations

Table 2.2: Comparison of Classification Algorithms of SL, adapted from [Singh, 2010]

Learning Method	**Loss Function**	**Regression Function Shape**	**Parameter Estimation Algorithm**	**Prediction Algorithm**
Linear Regression (assuming Gaussian noise model)	square loss: $(\hat{Y}-Y)^2$	Linear	Solve $\beta=(X^TX)^{-1}X^TY$	$\hat{Y}=X\beta$
Nadaraya-Watson Kernel Regression	square loss: $(\hat{Y}-Y)^2$	Arbitrary	Sotre all training data. Choose kernel bandwith *h* using cross validation	$f(x)=\frac{\sum_i y_i K(x_i,x)}{\sum_j K(x_j,x)}$
Regression Trees	square loss: $(\hat{Y}-Y)^2$	Axis-aligned partition of feature space	Many: ID3, CART, C4.5	Move down tree based on x, predict value at the leaf

Table 2.3: Comparison of Regression Algorithms of SL, adapted from [Singh, 2010]

The best example is a database of movies with the genre and the key aspects of the plot, after watching a few, the user would develop a certain taste for some of them. The suggestion that search engines use in the catalogue is based on these features and it will then present similar titles based on the watch history. Figure 2.15 shows how important is to select the correct value of k and the impact of it, by using a value of k=3 the new sample (star symbol) would be classified as a red circle but with a value of k=7 it is classified as a blue triangle as it is more close to them.

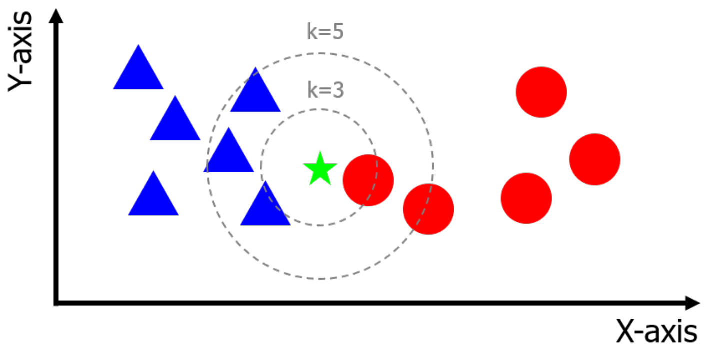


Figure 2.15: K-NN, the classification depends on the chosen k value

2.5 Unsupervised Learning

Unsupervised learning methods use the input information to train the algorithms but the difference with supervised methods is that it does not utilize known answers or outputs. Training is conducted by rules and constrains within the algorithm during the training process. The main algorithms for unsupervised learning are: Clustering and Neural Networks (NN).

As defined by [Gurney, 2014]: *"A neural network is an interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the interunit connection strengths, or weights, obtained by a process of adaptation to, or learning from, a set of training patterns"*. The neural network *"learns"* by experience, from the data provided and this *"experience"* is stored in the weights (W) of the NN (see Figure 2.16).

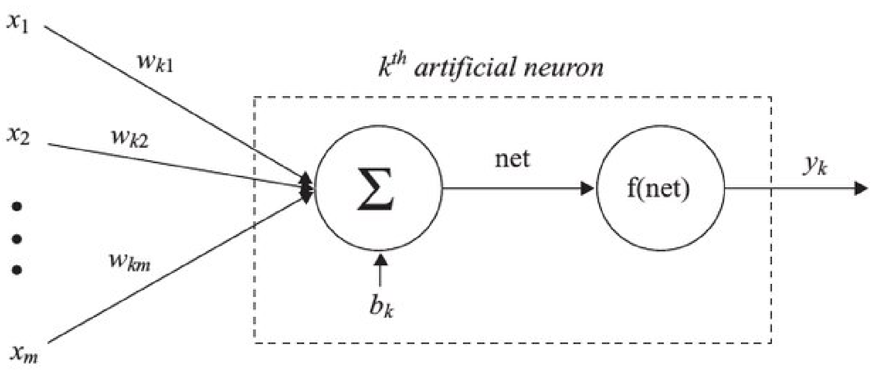


Figure 2.16: Artificial neuron sketch, taken from [Kantardzic, 2011]

Figure 2.17 shows an example of how the neural network finds the features from each category and how it links them with the higher level layers to understand the object in the picture and judge how different from the rest of the categories is. The output of the architecture of the NN is a probability of object being one of the trained classes.

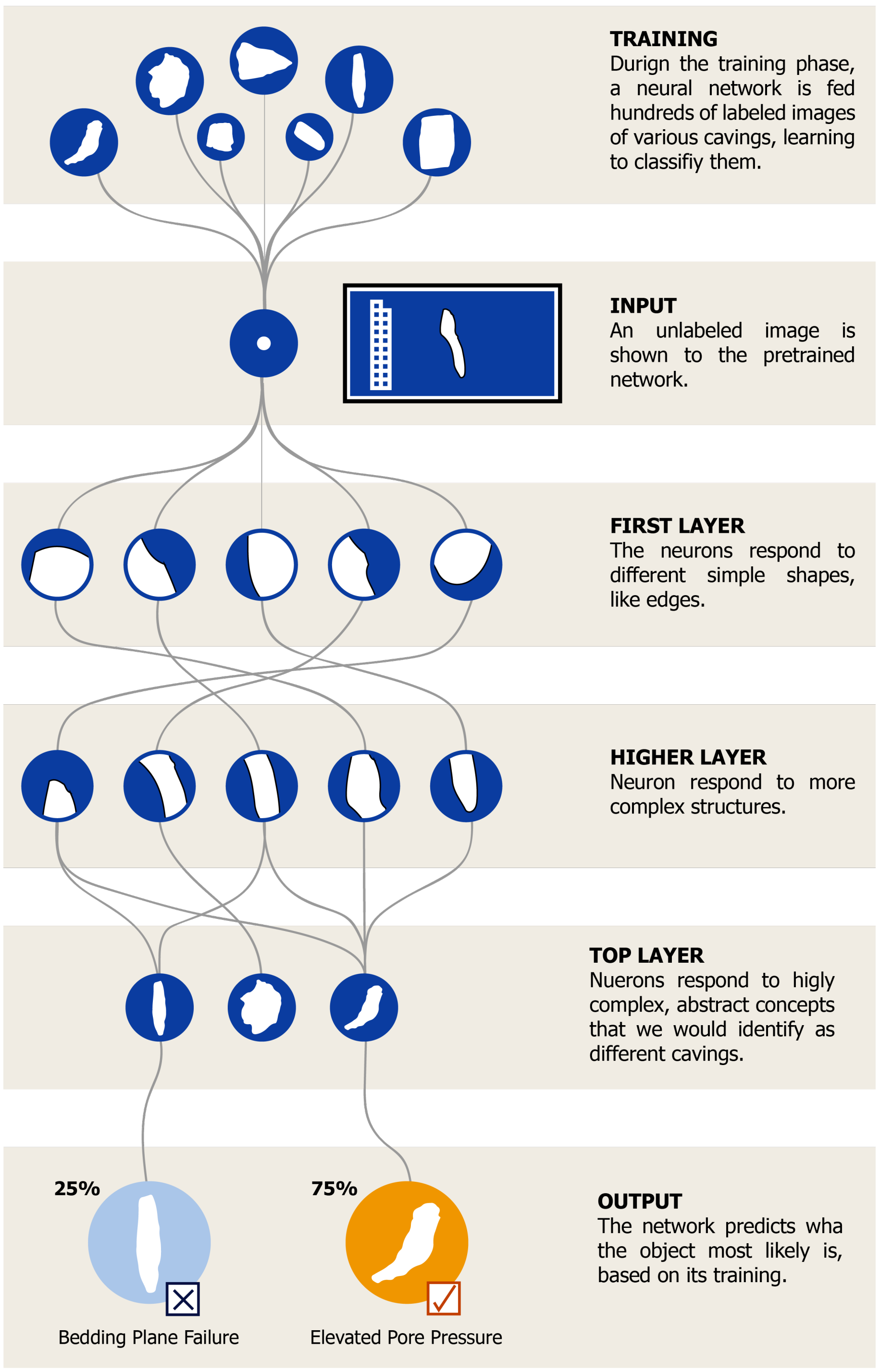

TRAINING
Durign the training phase, a neural network is fed hundreds of labeled images of various cavings, learning to classifiy them.
INPUT
An unlabeled image is shown to the pretrained network.
FIRST LAYER
The neurons respond to different simple shapes, like edges.
HIGHER LAYER
Neuron respond to more complex structures.
TOP LAYER
Nuerons respond to higly complex, abstract concepts that we would identify as different cavings.
25%
75%
OUTPUT
The network predicts wha the object most likely is, based on its training.
Bedding Plane Failure
Elevated Pore Pressure

Figure 2.17: Neural Network workflow sketch, adapted from [Parloff, 2016]

There two kinds of algorithms in unsupervised learning: *Classification Algorithms* and *Clustering Algorithms*. Tables 2.4 and 2.5 summarizes the most important parameter for both algorithms, among them: loss function, prediction algorithm, parameter estimation algorithm, among others.

Learning Method	**Generative or Discriminative**	**Loss Function**	**Parameter Estimation Algorithm**	**Prediction Algorithm**	**Model Complexity Reduction**
Bayes Nets	Generative	-log$P(X,Y)$	MLE	Variable Elimination	MAP
Hidden Markov Models	Generative	-log$P(X,Y)$	MLE	Viterbi or Forward Backward, depending on prediction task	MAP
Neural Networks	Discriminative	Sum-squared error	Back propagation	Forward propagation	Reduce number of hidden layers, regularization, early stopping

Table 2.4: Comparison of Classification Algorithms of UL, adapted from [Singh, 2010]

Learning Method	**Loss Function**	**Number of clusters: Predetermined or Data-dependent**	**Cluster shape: isotropic or anisotropic**	**Parameter Estimation Algorithm**
K-means	Within-class squared distance from mean	Predetermined	Isotropic	K-means
Gaussian Mixture Models (identity covariance)	-log$P(X)$, (equivalent to withing-class squared distance from mean)	Predetermined	Isotropic	Expectation Maximization (EM)
Single-Link Hierarchical Clustering	Maximum distance between a point and its nearest neighbor within a cluster	Data-dependent	Anisotropic	Greedy agglomerative clustering
Spectral Clustering	Balanced cut	Predetermined	Anisotropic	Run Laplacian eigenmaps followed by K-means or thresholding eigenvector signs

Table 2.5: Comparison of Clustering Algorithms of UL, adapted from [Singh, 2010]

2.5.1 K-means Clustering

K-means clustering [MacQueen et al., 1967] is an unsupervised learning method which is simple and one of the most used to solve problems related to clustering. This method automatically partitions a database into k groups, which are non-overlapping clusters. The main idea of this method is using a first group which are randomly selected centroids. These centroids are the beginning points of every cluster, and perform iterative calculations to optimize the positions of the centroids. The process continues adding each instance to its closest cluster, and each cluster center is constantly updating to be the center of its instances. Furthermore, it is fundamental to specify the number of k-cluster for this method.

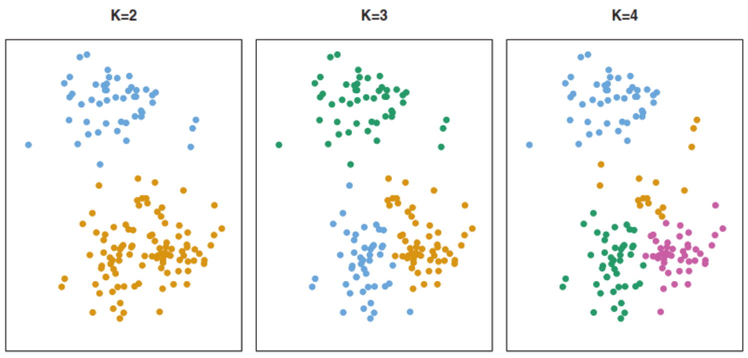


Figure 2.18: 150 observations from a simulated data with different k-clusters taken from [Hastie et al., 2013]

In order to apply this method in this project, an open source library is used which is *Scikit-Learn* [Pedregosa et al., 2011b]. This library is applicable in python, and it performs multiple tasks and operations such as classification, regression, clustering, dimensionality reduction, etc.. It is a powerful tool for data analysis and data mining.

Chapter 3

Database Elaboration

3.1 Caving Samples

For any machine learning project, the vital part is the data; depending on what information the project is based on, its manipulation, standardization and organization may be different. In the present project, the available data is rock samples that will be photographed to get 2D images.

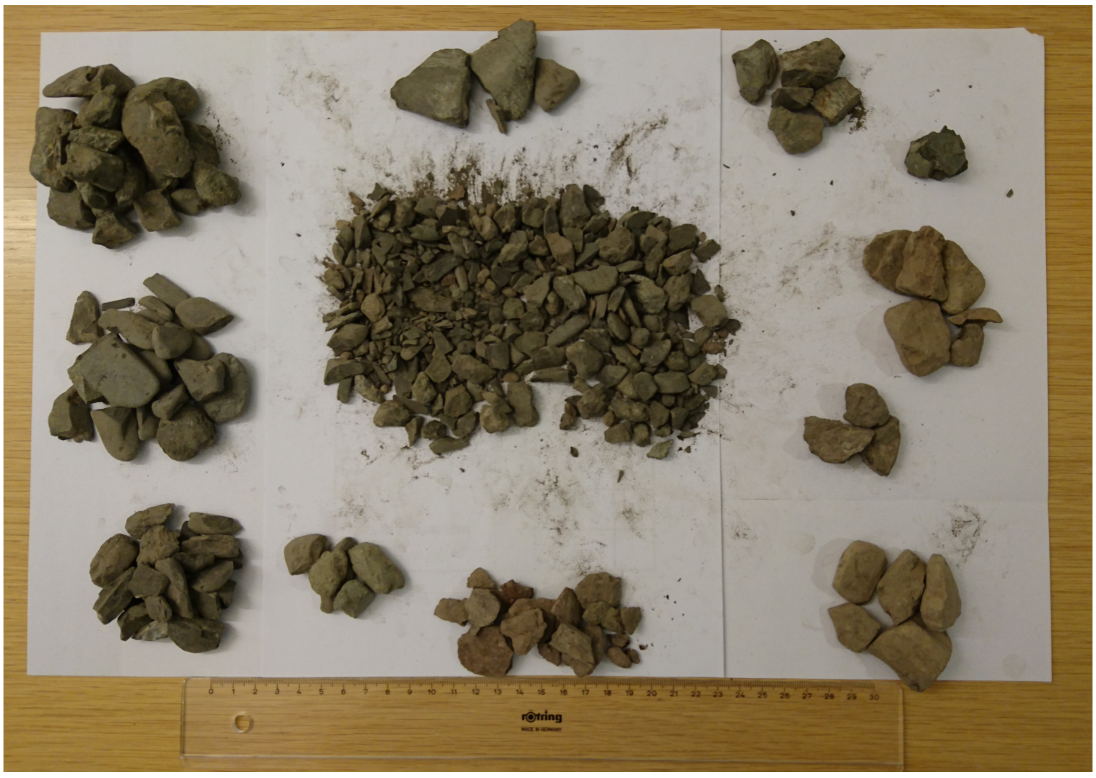


Figure 3.1: Received caving samples

Caving samples are not desired during drilling operations, however their presence does not necessarily mean that the well is experiencing problems. The available cavings for the project were provided from different wells, Figure 3.1 shows how the cavings range from a few millimeters to a few centimeters, having all of the a distinctive brown color due to the drilling mud. To the naked eye, the cavings present a natural color that can be linked to its chemistry and rock type they belong to; to be able to clearly distinguish these colors, the samples need to be washed with olive oil to avoid any chemical reaction if water were to be used.

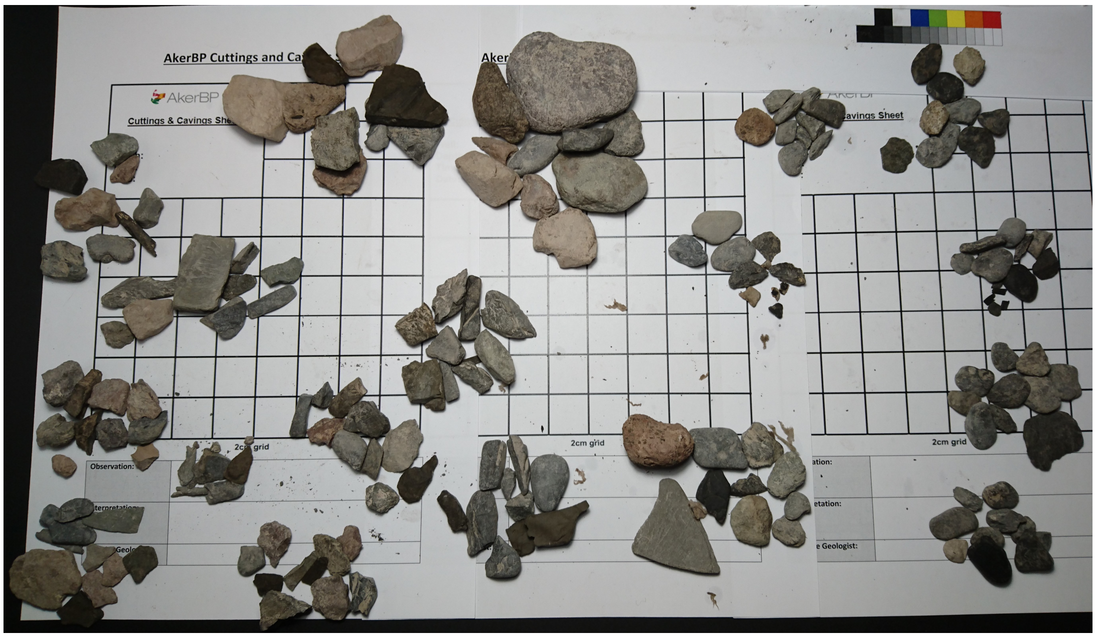


Figure 3.2: Clean caving samples

This decision will alter the information that the picture will store; it is different to use a dataset with clean samples than samples with mud. In a real situation the cavings would be covered by fresh drilling mud and the pictures would be taken that way, in this specific case the mud is now dry and it was decided to remove it so the real color of the caving can be enhanced and analyzed with computer vision.

Figure 3.2 shows the samples after cleaning the drilling mud with olive oil. It is now more evident that the cavings come from different formations and that the different shapes can have different colors and sizes. The oil is evaporated at room temperature and dry clean cavings are ready to be photographed. This procedure is done by hand and can take long hours depending on the number of samples, there are commercial options in the market to automate this step [Rockwash, 2019] [Samplewash, 2019] that can be considered for a future project.

3.2 The RAW Dataset

After all the cavings are cleaned and photographed with the template, a database of 1,064 cavings was developed. As it can be seen in Figure 3.3, the cavings are photographed from both sides to capture the subtle differences, since there are 1,064 in one single image, zoom-in circles has been added to appreciate some sections and have a look on how different the cavings are. It is also noted that the background have been removed and made transparent, making it possible to change it to a color where they can contrast with for aesthetic reasons, black. This is only possible giving the fact that there is no shadow.

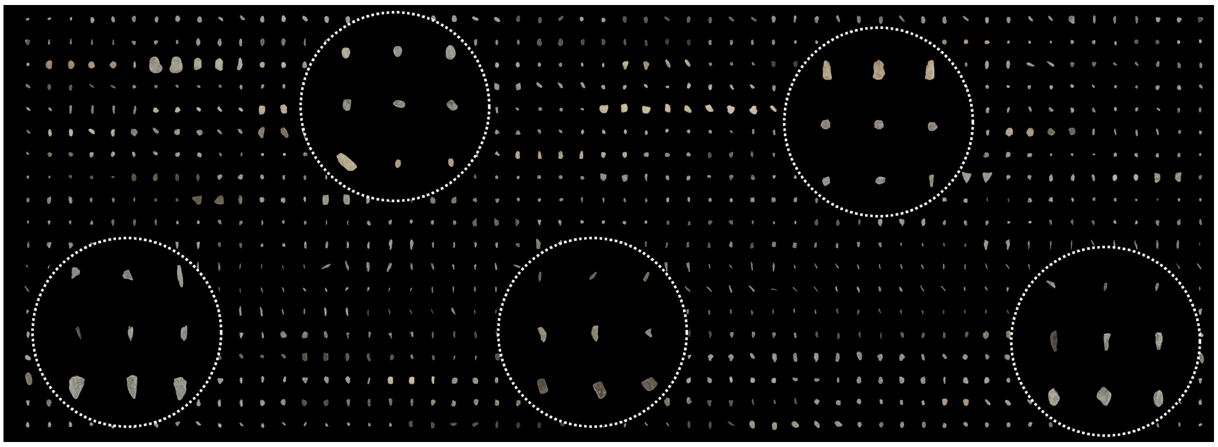


Figure 3.3: Complete cavings dataset

It is important also to notice that none of the cavings have been re-sized and that the RAW images contain the scale at the left hand side, making it possible to measure width and length. The dataset is limited by two factors: Availability of samples & Difference in shape; both have already been addressed in chapter 1. As one of the key aspects of the project is shape recognition, it is important to have a balanced dataset, which means that all the classes should have more or less the same number of samples to prevent overfitting over one class in particular.

If the dataset is categorized thinking about the shape of the cavings, it will have the following distribution:

#	Shape	Samples (number)	Percentage (%)
1	Concave-Convex	152	14.29
2	Irregular	152	14.29
3	Triangular / Pyramidal	152	14.29
4	Elongated / Splintery	304	28.57
5	Blocky / Tabular	304	28.57
	TOTAL	**1,064**	**100**

Table 3.1: Cavings distribution according to its shape

Since the dataset presents a clear imbalance there are three options to make it even among the classes:

1. Find and document more cavings from the lacking classes.
2. Subtract the excess of cavings in the two classes: Elongated and Blocky.
3. Split the classes Elongated and Blocky into two each and increase the classes from 5 to 7.

Option number 1 is difficult to achieve as the specific shape cavings cannot be ordered, it will be a matter of luck to find the samples among more samples. Option number 2 is not the most adequate for the project since it already have a small dataset; however, by using option number 3, it will allow to retain the number of samples and also analyze the new classes for different drilling problems.

The selected method is size-depending, according to the length of the object and it will be split into: short or long. For elongated cavings, this threshold is 2.48 which represents the ratio between the length of the caving and the width, the same for Blocky is 1.44. The code that does the splitting can be found in Appendix B.2 - Listing B.1.

The splitting of these two categories allows the dataset to become balanced and also to change the previous labels and create two new ones. The number of samples, results from a manual classification of the cavings getting a number of samples between 150 and 157, since most of the cavings shapes can be found within 152 samples per class, this number was chosen to be the same among all classes (some samples were left out of the database and some had to be asked in order to complete the class). The representation of the split done by Listing B.1 can be seen in Figure 3.4. Now the balanced dataset is:

#	Shape	Samples (number)	Percentage (%)
1	Concave-Convex	152	14.29
2	Irregular	152	14.29
3	Triangular / Pyramidal	152	14.29
4	Long Elongated	152	14.29
5	Short Elongated	152	14.29
6	Rectangular Blocky	152	14.29
7	Squared Blocky	152	14.29
	TOTAL	**1,064**	**100**

Table 3.2: New cavings distribution after splitting

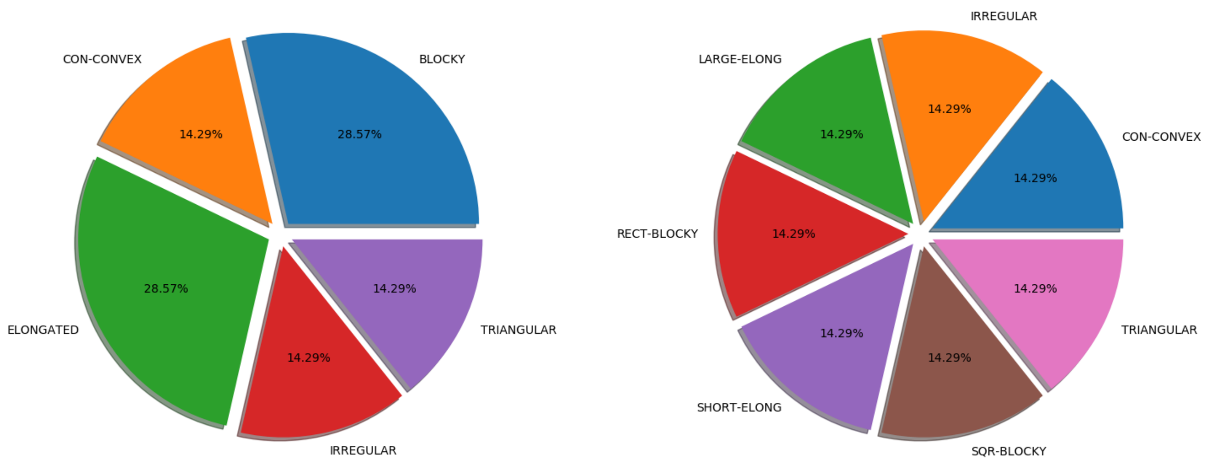


Figure 3.4: Dataset before and after splitting the Elongated and Blocky categories

3.3 The ROTATED Dataset

It is important to consider the position of the caving in the picture, so far the samples has been placed in one and only one position. For example, for a Pyramidal caving pointing north, the algorithms will learn that this type of caving should always be placed in that position in order to be recognized. Just as the orientation, the position of the caving in the picture is important; at the moment the code will be fed with samples placed in the center of the image. These two aspects have to be considered when using any algorithm for machine learning as it is very important when a new sample is provided as a test.

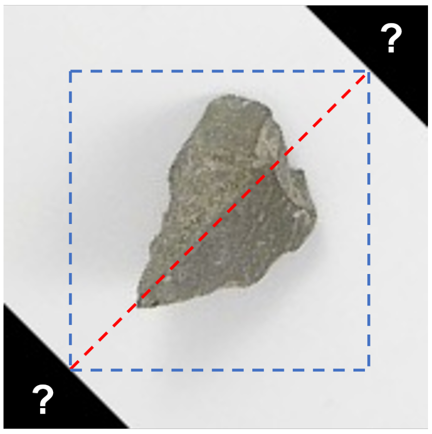


Figure 3.5: Black areas on the edges, the image needs to be cropped

In the case that the user needs to feed an image with a caving in any position and orientation, the ML algorithms needs to be trained with images telling it that these possibilities exist in real life. For the RAW dataset, the caving is already placed in the center but regarding the orientation, the image can be rotated using computer vision, but this approach needs to be addressed carefully. As seen in Figure 3.5 by rotating the image, it creates black triangles on the corners depending on the orientation the caving is rotated.

The correct way to rotate the image is to crop the minimum area defined by these corners, the length and width of the new area is always defined by the largest possible diagonal between picture. This area is also known as the Region of Interest (ROI), the square with blue dashed-lines which will never change its size regardless of the rotation.

Using the code in B.2 - Listing B.2, it rotates the caving in Figure 3.5 by 45 degrees clockwise until it completes a full rotation, the generated images are shown in Figure 3.6. This means that the same picture can create 7 more and if the angle is reduced it will contemplate more positions in which the caving is placed in the picture. This allows to increase the dataset and create the ROTATED Dataset by rotating each picture 15 degrees clockwise for a total of 25,536 images, 24 pictures per original picture.

This rotation is chosen thinking in 6 different positions per cuadrant, also less degrees would give too few positions that the caving can be in the training dataset and too many options during a test with a random position of a new sample and more degrees of rotation would simply be too much for computer power. This can be also called, as a manual Data Augmentation, as the process has been implemented out of the frame of the ML algorithms.

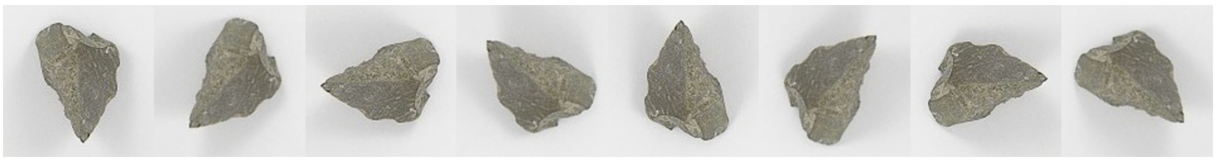


Figure 3.6: Rotation of cavings

Chapter 4

Feature Extraction and Machine Learning

4.1 The Features: Shape, Roundness, Size & Colour - Why?

The study of the properties of the cavings is important to determine: origin, volume, correlation with the bit, transport time and more importantly the reason they are produced in the borehole. There is two aspects that are needed to be study in order to have a clear picture of physical phenomenon in the borehole: the information from the rock and the information from the rig.

Shape	Most Probable Cause*
Concave-Convex	Elevated Pore Pressure
Long Elongated	Bedding Plane Failure
Short Elongated	Weak Rock Fabric
Irregular	Fault Zones
Triangular / Pyramidal	Stress Anisotropy
Rectangular Blocky	Fractured Formation
Squared Blocky	Mud Support

Table 4.1: Cavings shapes & most probable reason (*it should always be linked with the drilling parameters and other characteristics of the well)

The study of its morphology can be related with the effect of pressure and stresses in the wellbore, an elongated or concave convex shape is an indication of formation over-pressure while a pyramidal shape is more related with stress anisotropy. There are other factors involved but to simplify the problem at this stage each shape will be assigned a failure mechanism, the most probable (this should not be taken as a law as the study of cavings need to be linked with the drilling parameters and events before, during and after the presence of them at the shale shakers).

Roundness is a synonym for transport, it is well known that the more rounded the more the caving spent time inside the wellbore bouncing around and losing material on its edges. It is the same as in a river, the more rounded rocks have been transported for kilometers and kilometers, making it collide with the stream bed and other rocks. This process will smooth the edges and eventually it will tend to produce a spherical rock. In the wellbore occurs the same phenomena, the caving stays inside until it can be lifted and seen at the shale shaker. This characteristic can tell the required size that the mud and hole geometry can lift and also how long it is staying downhole. Bridging problems can occur when the material is not evacuated and it can endanger the BHA and the drillpipe.

Size is important to measure in order to obtain an approximate value of the volume of material that is being returned to surface, that material is now widening the hole and creating a more complicated situation: washouts. This is very important for cementing calculations and fluid losses. If this problem can be reduced or mitigated after the first arrivals of cavings, it will definitely optimize drilling and completion operations.

Colour can always be related to the lithology of the rock, the mudlogger should always know the dominant lithology of the formation the driller is currently drilling, and if in the returns there is a lithology that does not correspond with the depth of the bit, it means that a different formation is caving-in and the mud parameters are allowing it to happen. Thus, remedial actions need to be taken.

There are more features that can be extracted from the caving: effect of the bit, mud invasion and plasticity; also from the well: angle, ROP and mud chemistry. This list does not limit the characteristics that should be analyzed or the parameters that need to be linked, and it only emphasizes the need of studying drilling operations and the close relationship with cavings.

4.2 The MNIST Approach

4.2.1 Shape

Pattern recognition by means of machine learning has been addressed by the MNIST challenge [LeCun, 1998] where the dataset consist of handwritten digits on images which has been resized and centered in 28x28 pixels gray-scale image. The dataset consists of 60,000 samples for training and 10,000 samples for testing and it is a clear example of how different each person can write the same number.

The value of each pixel is treated as a feature for the ML algorithms, 784 features per image; the different combination of values and the position in the matrix is what defines how a digit (class) is build. The dataset that is being used in this project consist of only 1,064 samples (see Figure 4.1 for reference of the basic original image) which is less than ideal for pattern recognition but challenging in terms of how the algorithms can address this problem.

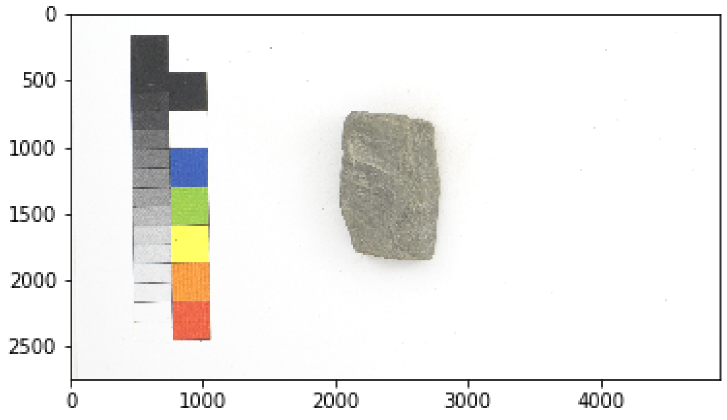
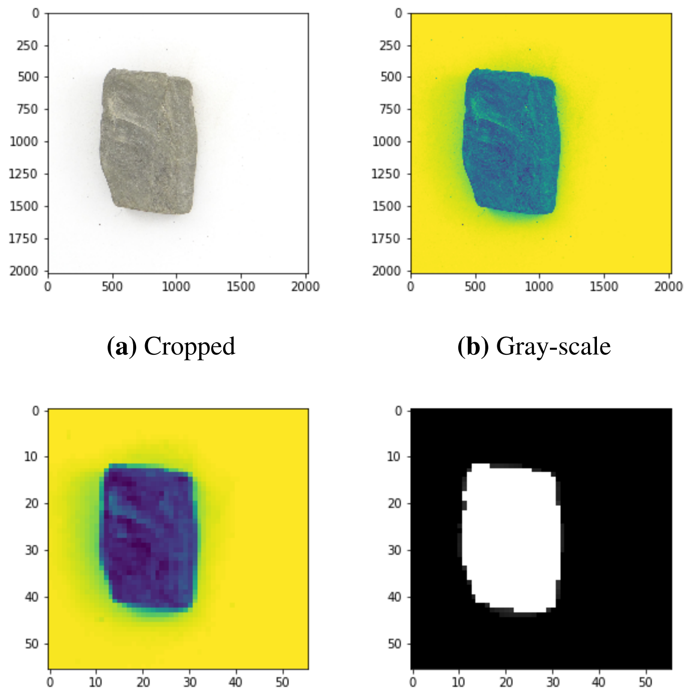


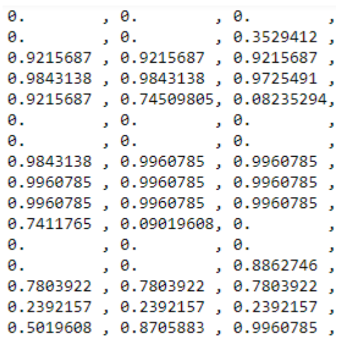
Figure 4.1: Original image

Using the same approach and methodology to build the MNIST dataset, the Raw and Rotated datasets will be transformed into a matrix of vectors containing values between 0-255 for each pixel in a gray-scale image. Figure 4.2 shows the process from the cropped image to the final image composed of values in a matrix. The matrix depends on the size of the picture that is being analyzed.



(a) Cropped **(b)** Gray-scale

(c) Re-sized **(d)** Normalized



```
0.        , 0.        , 0.        ,
0.        , 0.        , 0.3529412 ,
0.9215687 , 0.9215687 , 0.9215687 ,
0.9843138 , 0.9843138 , 0.9725491 ,
0.9215687 , 0.74509805, 0.08235294,
0.        , 0.        , 0.        ,
0.        , 0.        , 0.        ,
0.9843138 , 0.9960785 , 0.9960785 ,
0.9960785 , 0.9960785 , 0.9960785 ,
0.9960785 , 0.9960785 , 0.9960785 ,
0.7411765 , 0.09019608, 0.        ,
0.        , 0.        , 0.        ,
0.        , 0.        , 0.8862746 ,
0.7803922 , 0.7803922 , 0.7803922 ,
0.2392157 , 0.2392157 , 0.2392157 ,
0.5019608 , 0.8705883 , 0.9960785 ,
```

(e) Vectors

Figure 4.2: Dataset transformation from images to vectors, 56x56 pixels

The first step is to crop the image as it posses the scale at the left hand-side, the ROI is a squared area of 2,020x2,020 pixels (Figure 4.2a) which can fit all the cavings in the dataset; at this stage the information for one image is 12,241,200 features in the form of a matrix:

2,020x2,020x3, due to the image dimensions and the three channels from the RGB. The ROI is changed from an RGB image to monochrome image which means that the three channels of colors are converted into one (Figure 4.2b); this is done to simplify the problem and have one value per pixel, the image now contains 4,804,000 features.

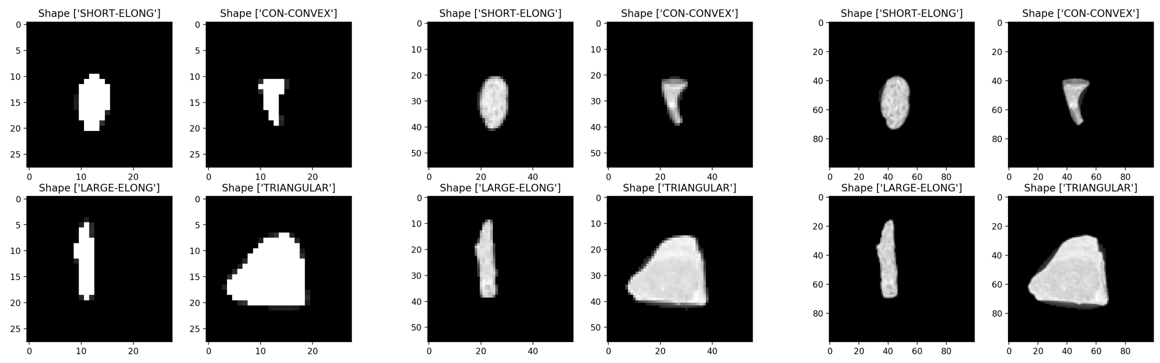
To simplify it more, the image is re-sized to 28x28 pixels (Figure 4.2c) and with this, the features in the matrix are reduced to 784 (28x28), the same as MNIST. The last step is to normalize the data and between 0-1 being zero complete black and 1 a complete white pixel (Figure 4.2d), this is the representation of the vectors showed in Figure 4.2e (only a small number of vectors are showed). All this pre-processing is done in Appendix B.2 - Listing B.3.

At this stage, the database has been changed from pictures to digits, making it a table that depends on the number of samples and the number of features. Starting with the example of 28x28 pixels the database has the form described in Table 4.2. Each value of pixel ranges between 0 to 1; it is obvious that in the corners the values will be zero and the values in the center would tend to be white, 1. If the dataset uses a higher density of pixels, the features will change and the columns of the matrix of vectors will reflect the new number of vectors, eg: 100x100, 10,000 pixels or features.

Sample	**Pixel1**	**Pixel2**	**Pixel3**	**...**	**Pixel392**	**...**	**Pixel783**	**Pixel784**
1	0	0	0	...	0.99	...	0	0
2	0	0	0	...	1	...	0	0
3	0	0	0	...	0.92	...	0	0
.	.	.	.	...	.	...	.	.
.	.	.	.	...	.	...	.	.
.	.	.	.	...	.	...	.	.
1,063	0	0	0	...	0.99	...	0	0
1,064	0	0	0	...	1	...	0	0

Table 4.2: Performance of Classifiers RAW Dataset

There is uncertainty regarding the effect of the pixel density in terms of accuracy of the algorithms for shape detection. Is it better to use more density of pixels at the cost of more computer power or is it as good as using the simplest 28x28 pixel images?. Figure 4.3 repre-



(a) 28x28 **(b)** 56x56 **(c)** 100x100

Figure 4.3: Pixel size effect on the cavings matrix

sents the change in density of data from 25x28 to 100x100, this is from 784 features to 10,000 features. From different methods to classify images using the databases, the first iteration will be done using the RAW dataset with 28x28 pixels to test the performance of the different classifiers, each with its particularities, pros and cons, to summarize a few:

- Logistic Regression
- Random Forest
- Naive Bayes
- RBF Support Vector Machines (SVM)
- K-Nearest Neighbors (KNN)
- Linear SVM
- Decision Tree
- AdaBoost
- QDA

The performance of each classifier is summarized in Table 4.3 where it becomes clear that the best classifiers are Logistic Regression and K-NN. If the same classifiers are used with the ROTATED dataset, there is some improvement in K-NN which means that the more samples, the better. Some classifiers perform worse in the light of more samples and it strictly depends on how the algorithm handles the data, rotation equivariance to be precise.

The analysis about why other algorithms present a poor behavior will not be discussed in this project, all the effort will focused on improving the results of the most successful algorithms. The results for the ROTATED dataset are showed in Table 4.4.

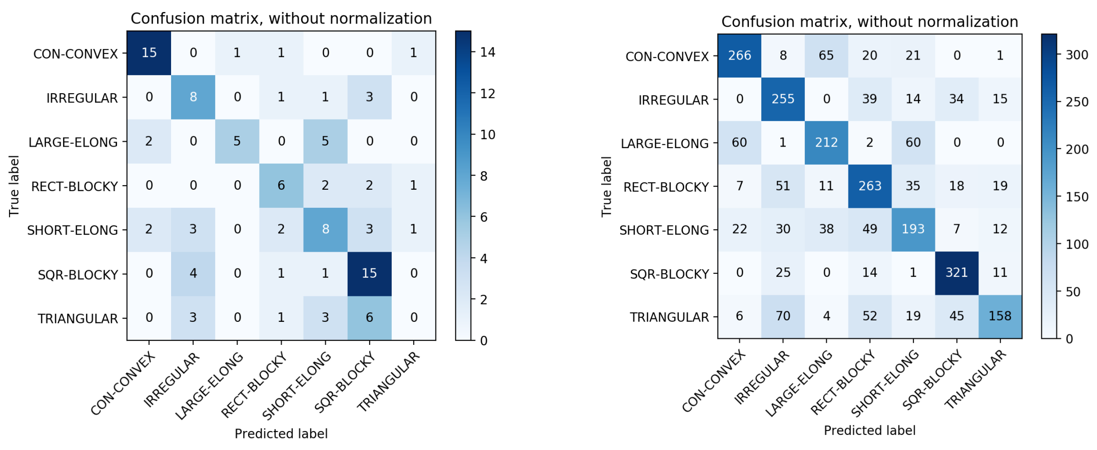
Classifier	**Precision**	**Recall**	**f1-score**
Logistic Regression	0.46	0.46	0.46
Random Forest	0.36	0.35	0.31
Naive Bayes	0.25	0.18	0.11
RBF Support Vector Machines	0.01	0.11	0.02
K-Nearest Neighbors (KNN)	0.46	0.46	0.45
Linear SVM	0.25	0.31	0.27
Decision Tree	0.39	0.4	0.37
AdaBoost	0.41	0.4	0.4
QDA	0.14	0.17	0.1

Table 4.3: Performance of Classifiers RAW Dataset

Classifier	**Precision**	**Recall**	**f1-score**
Logistic Regression	0.26	0.28	0.27
Random Forest	0.28	0.27	0.22
Naive Bayes	0.25	0.17	0.08
RBF Support Vector Machines	0.29	0.28	0.24
K-Nearest Neighbors (KNN)	0.52	0.51	0.49
Linear SVM	0.27	0.29	0.27
Decision Tree	0.24	0.31	0.26
AdaBoost	0.24	0.26	0.25
QDA	0.17	0.16	0.07

Table 4.4: Performance of Classifiers ROTATED Dataset

From the obtained results, it becomes clear that the best results for the cavings dataset is obtained by using K-NN as the selected classifier and if the database is increased it will increase its accuracy. Figure 4.4 shows the confusion matrix for the two datasets, the increment in samples results in a better accuracy during the testing period. It is no surprise that K-NN is the best classifier, according to [Pedregosa et al., 2011b] (figure 4.5) the number of samples and the labels play a role at the moment of choosing the optimum classifier. For the number of labeled samples in the RAW dataset 1,064 (less than 100K) and since our data are digits and not text, it is better to use K-NN over Linear SVM. If K-NN fails, SVC and ensemble classifiers can be used.



(a) RAW **(b)** ROTATED

Figure 4.4: Confusion matrix for RAW & ROTATED dataset

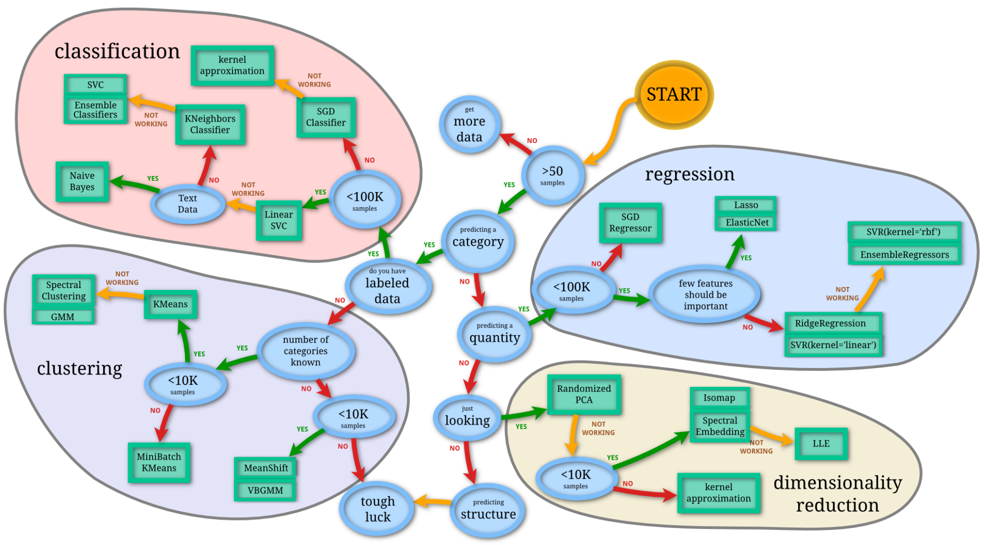


Figure 4.5: Choosing the right estimator taken from [Pedregosa et al., 2011b]

The next step is to use neural networks (NN) to improve the results, both datasets will be used to evaluate its performance and the complete list of classifiers is:

- K-Nearest Neighbors
- Perceptron
- Logistic Regression
- Random Forest

- Multi-Layer Perceptron
- Dense Layers: 1 input - 1 output
- Dense Layers: 1 Hidden Layer
- Dense Layers: 2 Hidden Layers
- CNN: Model 1
- CNN: Model 2
- InceptionV3

The best classifiers from supervised learning are added to compare the results with the rest of the NN, Figure 4.6 shows how it competes with the different architectures of NN. It is noted that Random Forest is also included and its performance with the ROTATED dataset increases thanks to a optimization of the algorithm with cross validation and the bootstrap method.

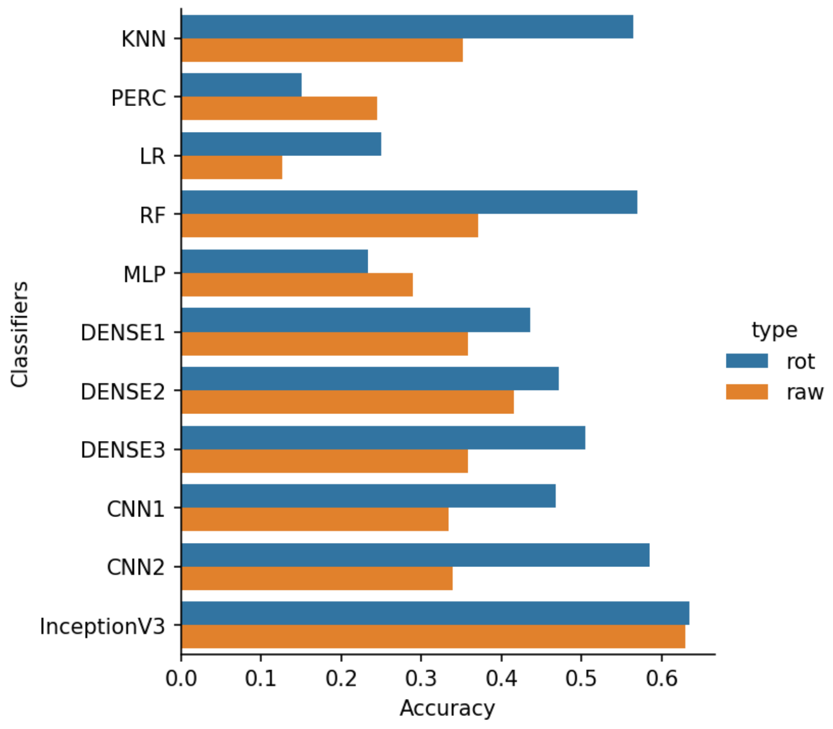


Figure 4.6: Accuracy of Neural Networks

InceptionV3 is a state-of-the-art NN for image classification which gives the best results for both datasets and the reason is because the NN uses image augmentation when it performs the training of the model. This makes manual rotation of the dataset unnecessary, yielding the same results as both datasets have a common origin; the idea of using a stronger NN designed for image classification will not only improve the accuracy results obtained from K-NN and InceptionV3 but it will increase the possibilities of the applicability of the trained model.

From the previous work the project can yield some conclusions:

- The MNIST approach is good for the steps into the right direction, however the dataset lacks quantity of samples.
- The use of NN is the right path, InceptionV3 proves the concept that even with a small dataset, accuracy can go above 60%.
- To improve accuracy and make the model more practical, Computer Vision with Neural Networks need to be used.

4.3 The Computer Vision Approach

4.3.1 Size

For size calculation, it is being used a technique which involves CV processes in order to measure the size of objects in an image, in this case cavings. This procedure can be comparable with computing the distance from a camera to an object; as a result, it is necessary to define a ratio so that this can be a tool to measure the number of pixel per a given metric.

First, calibration is done which means that an object of references is being used to do this. There are some important properties that the reference object should have:

- The dimensions of the reference object should be known in advance. These dimension could be width or height, only one is necessary, and a measurable unit should be used to describe this dimension such as centimeters, millimeters, inches, etc...
- The reference object should be easily found in an image. For that reason, it is important to place this in the top-left corner of an image. Also, appearance is a good approach to find the reference objects, and this can be done using a distinctive color or shape;

consequently, our reference should different from all other objects in the image in some manner.

To determine cavings size, a color scale is used which has a known width of 2 centimeters (Figure 4.7), and it is located on the left side of the image. This was determined to standardize our dataset. By guaranteeing the color scale is located on the left side, it is possible to sort the cavings contours from left to right. The color scale will be the first object to be analyzed in relation to the contours, and this is used to define the pixels per metric, which is:

$$pixels\_per\_metric = \frac{object\_width}{known\_width} \tag{4.1}$$

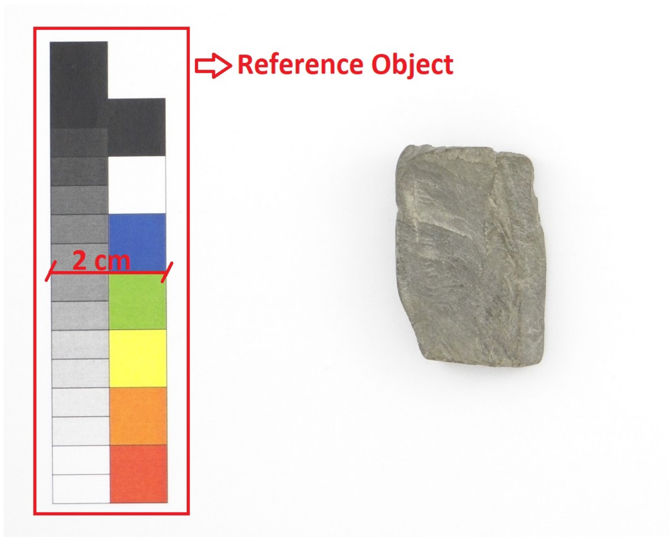


Figure 4.7: Color scale as a reference object

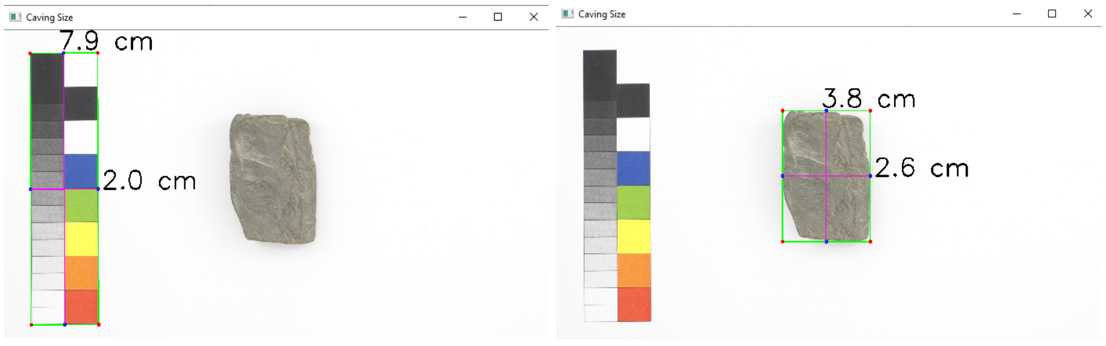
For example, the colour scale has a known width of 2 centimeters, and assuming that our object_width, measured in pixels, is computed 160 pixels. As a result, the pixel per metric can be calculated using the equation 4.1, and the result is 80 px. Thus, it can be implied that there are approximately 80 px per every 2 centimeters in the image. Using this ratio, it is possible to calculate the size of cavings in our dataset.

To develop this algorithm, OpenCV library [Bradski, 2000] was widely used along with Imutils package [Rosebrock, 2015], which has multiple functions to perform image processing. First, A "midpoint" function was define to compute two sets of (x,y) coordinates. Loading a caving image using the sheet template with the color scale is the first input information for the algorithm.

Second, an image process is performed which includes converting the image to grayscale, smoothing the image using Gaussian filter. Then, a edge detection is done along with erosion + dilation so that gaps between edges can be closed; consequently, the contours of the image can be found, and this is sorted from left to right, so the reference object can be extracted. Working on the individual contours is the next step.

Using the bounding boxes allows us to calculate the midpoint of these boxes. Also, the other important input information is providing our width of our reference object which is 2 centimeters. Furthermore, Euclidean distance is computed between the sets of midpoints to extract *height* and *width* from objects.

In conclusion, applying all the steps mentioned before, the size of the cavings is calculated as can be seen in Figure 4.8. See Appendix B.4.



(a) Reference Object Measurement **(b)** Caving Measurement

Figure 4.8: Caving Size Measurement

4.3.2 Colour

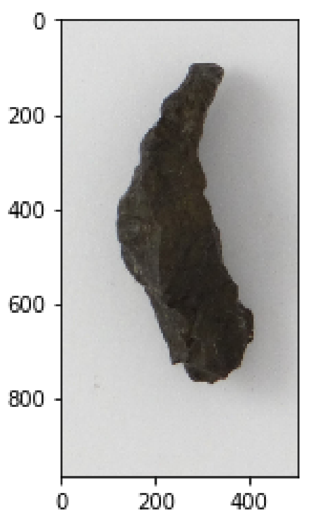
The approach used to calculate colour feature involves OpenCV [Bradski, 2000], Imutils package [Rosebrock, 2015] and Scikit Learn library [Pedregosa et al., 2011a]. Even though colour feature is not decisive to identify potential wellbore instability problems, it is an important feature to characterize cavings and have traceability of the produced cavings. Also, there are potential applications to use this colour information, and one of them is applying ML algorithms for rock classification (see Chapter 6.4.4) or help ML models for this purpose [Bestagini et al., 2017].

This process involves multiples steps. The first step is to load a caving image using OpenCV, and then it is necessary to segment out the caving image; as a result, a function called *"extract-Skin"* was created. It is important to have in mind the following points:

- OpenCV works on color images in BGR color space which is the reverse of RGB color space.
- Caving segmentation is achieved by applying threshold parameters in the HSV (Hue, Saturation, Value) color space. Using thresholds allow us to create a binary image by filtering out pixel based on this limits which were applied in the previous function. In other words, this function analyzes each pixel, and if the value of the pixel is in the threshold range, the pixel is converted into white. Otherwise, the pixel is made black, and there is another function to remove black pixels so that cavings color pixels are analyzed. In this project, the threshold values will be HSV values denoting the of the caving dataset colours.
- The caving image is subjected to image processing such as Gaussian filter.

After applying thresholds, the second step is to create a function to remove black pixels from extracted caving image (Figure 4.9). The following step consists in applying a function which has to extract the colour information. The basic of this function is to count for the occurrence of each colour; also, it generates an index which contains colour, colour percentage and cluster index.

Last, there is a function called *"extractDominantColor"* which calls all the above functions. This uses an unsupervised clustering algorithm KMeans Clustering [Hastie et al., 2013] in order


(a) Original Caving Image

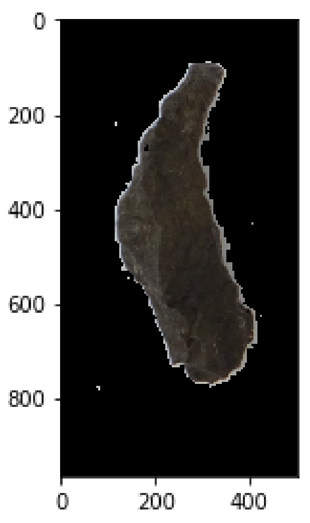
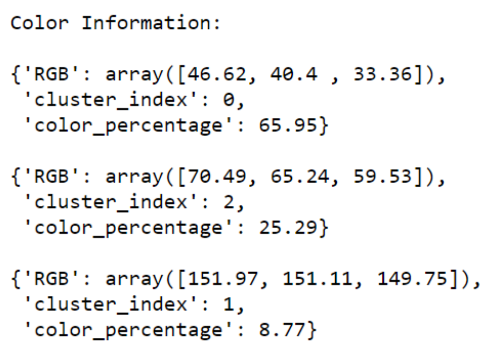

(b) Mask Applied to Caving Image

Figure 4.9: Caving Colour Process

to cluster the caving image pixels. It is important to highlight that the number of cluster established for this project is K=3. This was done according to the types of cavings in the dataset since 3 colours are relevant as information for futures analysis and characterization. However, it is possible to change this in the source code so that more colours will be extracted based on future purposes.

To sum up, the most dominant colours are extracted from a caving image as can be seen in Figure 4.10. See Appendix B.5



```
Color Information:

{'RGB': array([46.62, 40.4 , 33.36]),
 'cluster_index': 0,
 'color_percentage': 65.95}

{'RGB': array([70.49, 65.24, 59.53]),
 'cluster_index': 2,
 'color_percentage': 25.29}

{'RGB': array([151.97, 151.11, 149.75]),
 'cluster_index': 1,
 'color_percentage': 8.77}
```

(a) Colour Information

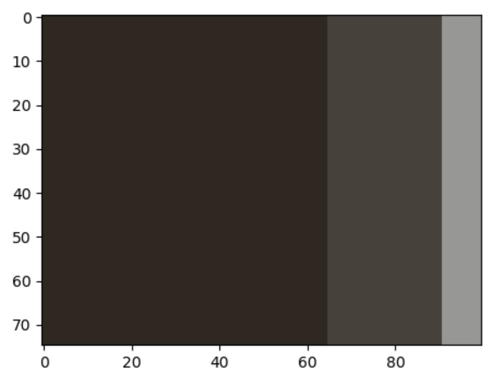
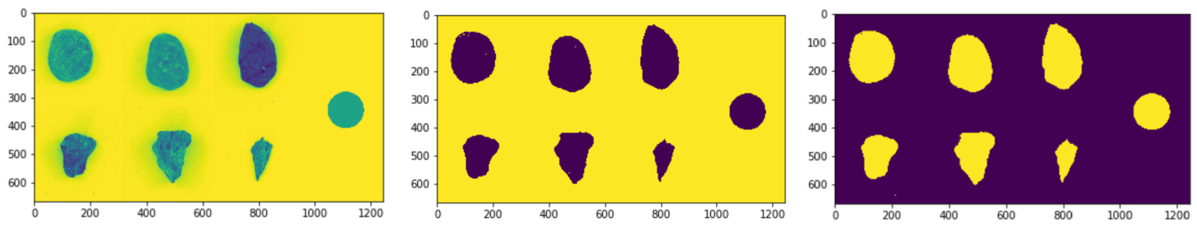

(b) Colour Bar

Figure 4.10: Colour Results

4.3.3 Edge Definition

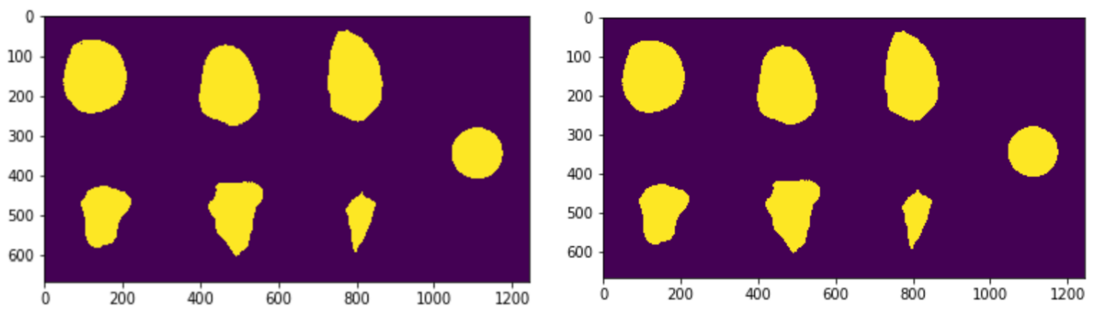
Edge definition or Roundness is a significant parameter since it gives an idea how much time the caving spent in the well. For this feature, a mathematical approach is used in order to calculate approximate roundness values of cavings. This approach is related to circularity of figures. It is important to highlight that values around 1 are related to a circle; conversely, values around 0 are related to very angular forms. Also, when the term roundness or edge definition is used in this project, it is referred to circularity due to the approach which is used for this feature calculation. It was previously mentioned that edge definition refers to how smooth the caving edges are.



(a) Grayscale Image **(b)** Binary Image **(c)** Inverted Binary Image

Figure 4.11: Caving Roundness Process

The algorithm for edge definition performs different image processes and mathematical operations. First, it is fundamental to work on the image using OpenCV library, and converting the image to a binary black and white image with Otsu thresholding will give an output to be able to work on area and perimeter of cavings. Also, a command to invert the binary image is used due to the sheet template which is white as can be seen in rigure 4.11.



(a) Noisy Image **(b)** Clean Image

Figure 4.12: Removing Noise Process

Second, removing noise from images is done using morphology functions; as a result, some pixels, which are not part of the cavings, are removed (figure 4.12). This cannot be perceptible in the figure 4.12a because the removed pixels are really small, but this is necessary to avoid miscalculation. Next, the heavy work in relation to edge definition is performed calculating values to show the roundness of cavings. In order to do this task, the following equations are being used based on [Wirth, 2001] presentation:

$$Roundness = \frac{4 * \pi * area}{convex\_perimeter^2} \tag{4.2}$$

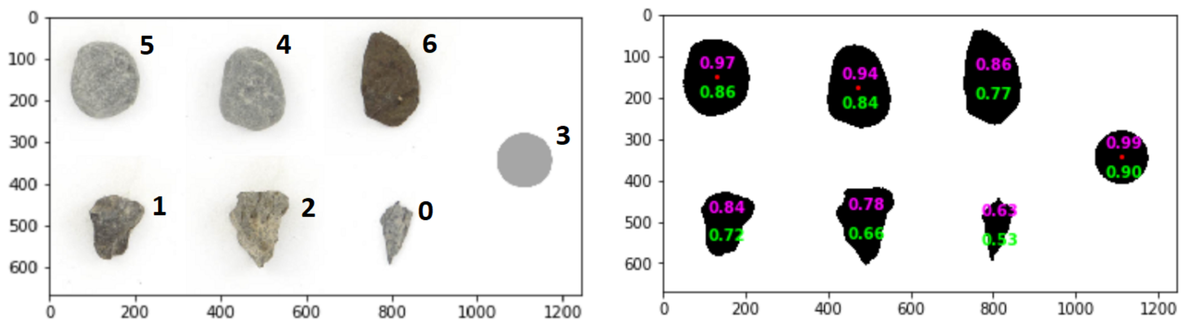
$$Roundness = \frac{convex\_perimeter^2}{4 * \pi * area} \tag{4.3}$$

Equation 4.2 is valid for smooth shape which does not take into account local irregularities. Roundness values equal to 1 for circular shape which would be the highest roundness level, and values less than 1 mean to be less rounded until angular. Besides, equation 4.3 is used for objects which have elliptical shapes, being roundness values equal to 1 for a circular shape, and values higher than 1 are referred to angular shapes which has irregular boundaries. Furthermore, it is important to remember the difference between perimeter and convex perimeter, and have in mind the concept of convex hull. All of this terms were previously explained in section 2.3.

In conclusion, to develop this algorithm, a sample from each type of cavings according their roundness was taken, which are: very rounded, rounded, sub-rounded, sub-angular, angular and very angular.The results obtained from roundness algorithm can be seen in Table 4.5. It is observed in Figure 4.13 a visual representation of the results where the magenta color represents the roundness values using convex perimeter, and the lime color represents the roundness values using perimeter.

Object	Area	Perimeter	PerimeterHull	Roundness	RoundnessHull	InvertRoundness
0	6,394	387.61	358.522	0.5348	0.62505	1.59987
1	13,705	490.07	452.686	0.7171	0.84042	1.18989
2	16,786	564.70	519.236	0.6615	0.78240	1.27812
3	13,054	426.96	406.875	0.8999	0.99091	1.00918
4	24,068	599.39	568.256	0.8419	0.93662	1.06767
5	23,698	588.70	554.151	0.8593	0.96976	1.03118
6	24,285	629.19	594.827	0.7709	0.86251	1.15940

Table 4.5: Roundness Results



(a) Original Image **(b)** Roundness Values

Figure 4.13: Roundness Results

4.3.4 Shape - Revisited

Improving the results on shape recognition means using a state-of-the-art NN that can differentiate between the small differences in the shapes of the cavings. The selected architecture is from YoloV3 [Redmon and Farhadi, 2018] [Redmon, 2016] that has proven to be more accurate and faster than previous iterations and other architectures.

Figure 4.14 shows the complexity of the NN and the number of different convolutional layers it requires to identify an object in a picture; the important layers for the present study are the YOLO layers, number 82, 94 and 106. Each of them is in charge of detection of objects of different sizes in the picture and the activation of the three of them means that it can detect an object that is very close and far at the same time.

For the cavings dataset, there are 7 different classes associated with the wellbore stability phenomena rather than the shape the caving possess; YoloV3 will only print the result of the most probable outcome of the image detection, this change is done to get more value from the

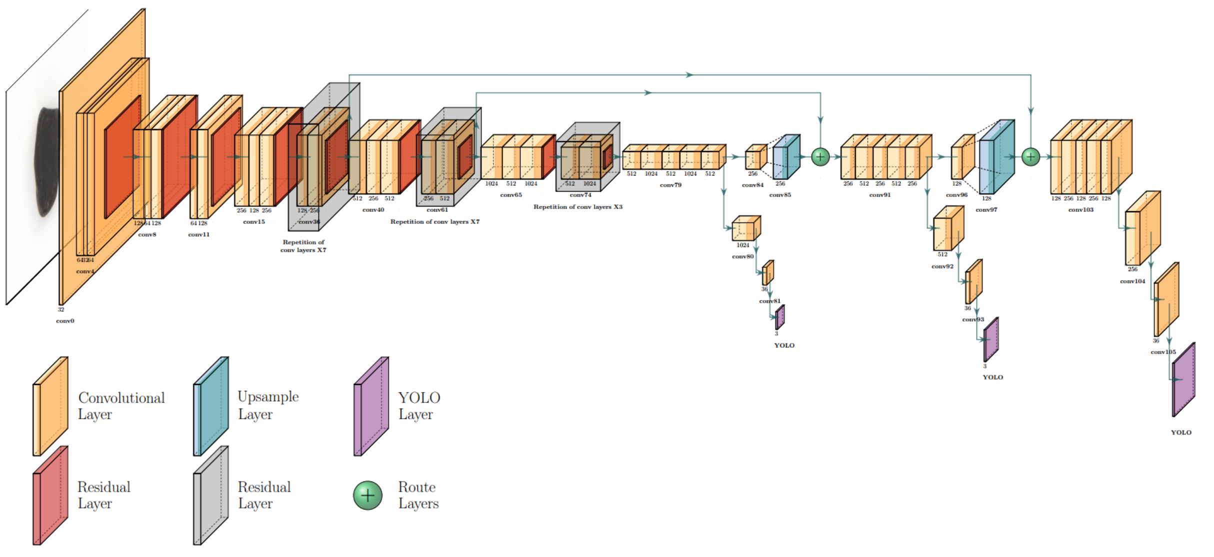


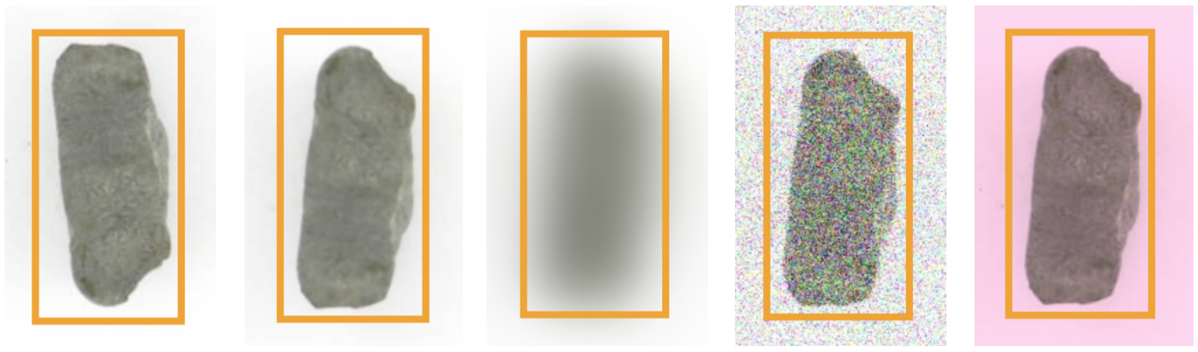
Figure 4.14: YoloV3 Architecture

same information.

YoloV3 has been built using the Linux environment and thus, to re-purpose the NN and use the cavings dataset, it is better to keep using this environment and also increase the computer power as the training can take from a few minutes to several hours. It is important to note that using the GPU possess a great advantage while using this NN, it will allow to speed up the computing and reduce the load on the CPU.

At this stage, image augmentation needs to be used to increase the size of the dataset and the chance of detecting the specific shape of the caving in the image. Figure 4.15 show a few examples of the different transformations that the image can have but all of them need to be related with the bounding box. By performing this augmentation, the RAW dataset is transformed into the Augmented Dataset with 21,280 samples or 3,040 samples per class; this means that every picture is transformed into 20. The complete transformations done to the dataset is shown in Appendix C.1.

The complete Augmented Dataset needs to be shuffled and labeled according to the percentage that will be used for training and the remaining for validation. In this project training has been done using 80% for the MNIST approach but with the Computer Vision approach 90%



(a) Original image (b) Flipped image (c) Blur (d) Noise (e) Mask color

Figure 4.15: Bounding box and image augmentation

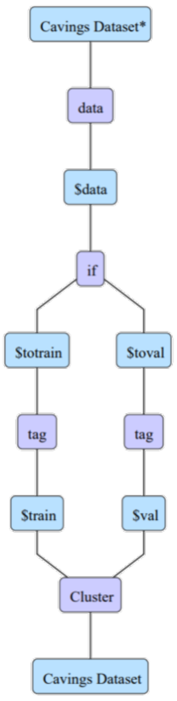


Figure 4.16: Labeling workflow & splitting of the dataset

will be used to give more training samples to the NN. This process cannot be done manually, thus a simple workflow was implemented to tag the images and then building the deck as desired (see Figure 4.16).

Training using the workstation provided by AkerBP took around 28 hours. Figure 4.17 shows the loss function through the epochs, it quickly decreases during the first 10 epochs and after the 20th it becomes more stable. This means that by extracting the weights from the lowest points of the evaluation period, the NN will be able to detect new images.

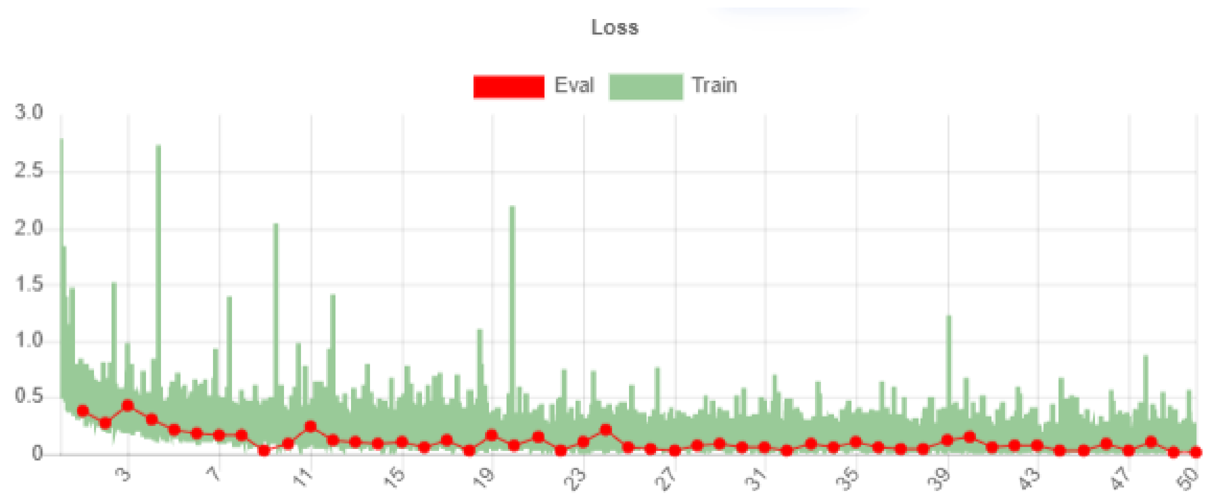


Figure 4.17: Loss function, 50 epochs

The versatility of using computer vision can not only extend to images but also video and live-stream video. YoloV3 is a NN designed to work with high FPS [Redmon and Farhadi, 2018] making it a one of the best tools for processing video. Figure 4.18 show the results of YoloV3 in images where it can correctly detect the most probable failure mechanism by looking at the shape of the caving. More examples of image detection can be found in Appendix C.2.

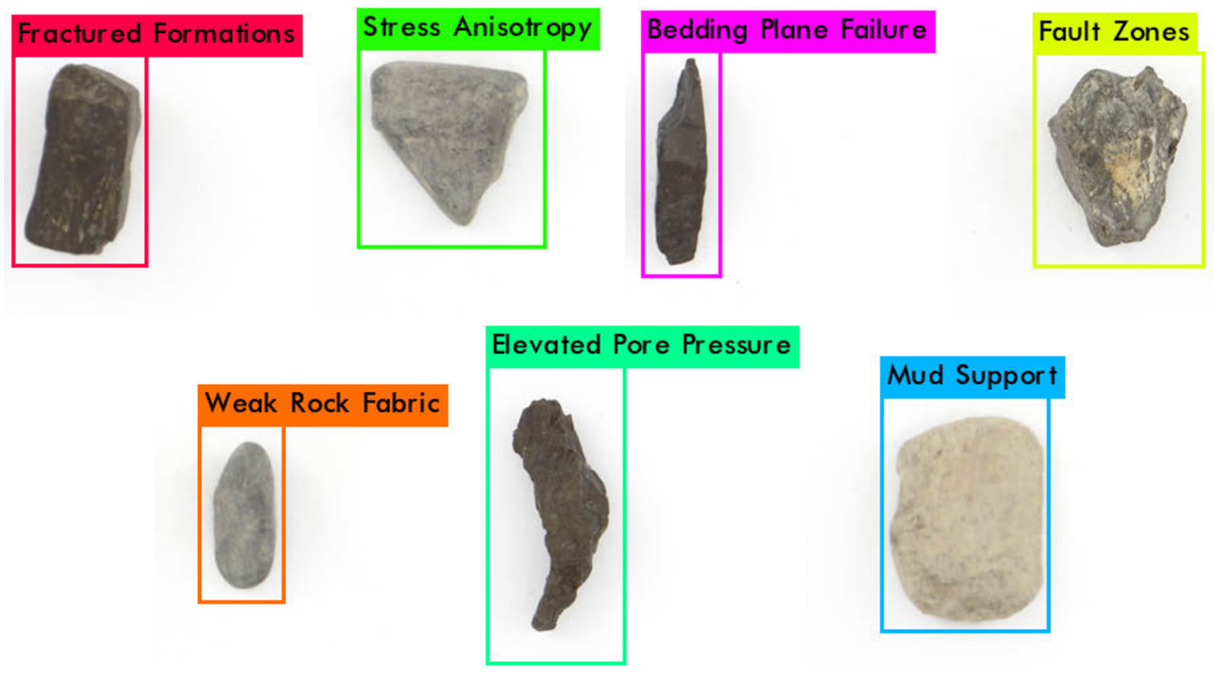
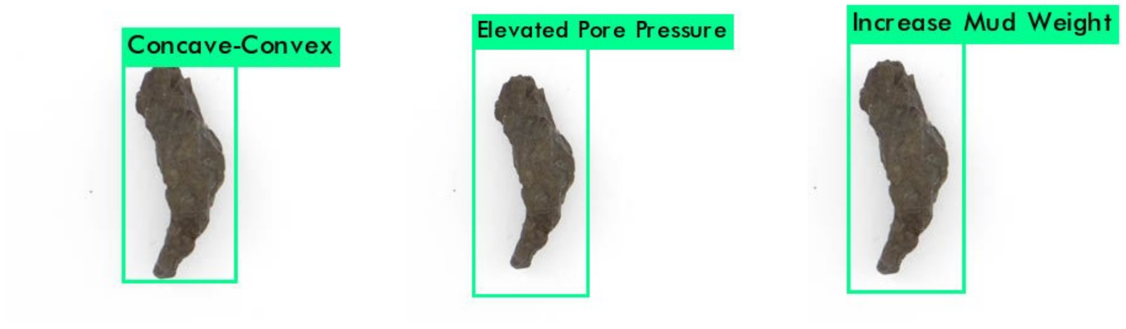


Figure 4.18: Shape associated with the most probable failure mechanism

Once the model is trained and tested, the Label that it used to detect the shape can be anything, from simple information like the shape of the caving or the formation that belongs to, it can be an alert telling that there is a physical phenomenon in the wellbore that requires the attention from the drilling crew. With enough confidence and using an evaluation of the different drilling parameters, it can be a suggestion to increase the mud weight or check the mud chemistry. Figure 4.19 shows how the label can be changed based on the interest and level of automation of the task.



(**a**) Informative (**b**) Alertive (**c**) Suggestive

Figure 4.19: Types of labels for the same shape

Chapter 5

Result Analysis

5.1 Defining the Perfect Picture

5.1.1 Camera Calibration

In order to build a standard dataset for ML and CV implementation, camera calibration is important to achieve this goal. There are many parameters which affects this calibration, and they are camera settings, photography setup, angle and distance between camera and caving. Caving features like size, colour and roundness might be seen affected by this situation. Even though the results are approximate values, optimizing these parameters helps to obtain an accurate outcome.

First, camera settings were tested many times during this project until obtaining the desired results. Table 5.1 shows the chosen settings for building the dataset of this project. One of the most important property is ISO speed, which is 320, so that adopting this value allows us to reduce the noise in the images; hence, not only horizontal but also vertical resolution values of 350 dpi were obtained where most of the caving details were captured, and this is really important for edge definition and size calculation accuracy. This was achieved also by the design of the white-box and light conditions (see appendix A.1) because this setups do not allow to generate shadows. These shadows can affect and generate miscalculations in terms of area and perimeter of cavings. Besides, position of the camera in front of cavings is fundamental to avoid radial distortion. For that reason, all of the objects to be measured need to be co-planar (in the same plane) with the reference object, and the camera must be at as near as possible 90 degrees to that plane (top view image is optimum).

Property	Values
Camera maker	Sony
Camera model	DSC-HX200V
F-stop	f/4.5
Exposure time	1/30sec.
ISO speed	320
Focal length	14mm.
Max aperture	4.33

Table 5.1: Camera Settings

5.1.2 Colour Analysis

To be able to obtain colour information from caving images, lower and upper HSV thresholds were established. These thresholds were set based on complete dataset of cavings colours, and most of the cavings has lower BGR values which means dark colours. Also, light colours of some cavings were taken into account for these boundaries. An example of this is the figure 5.1.

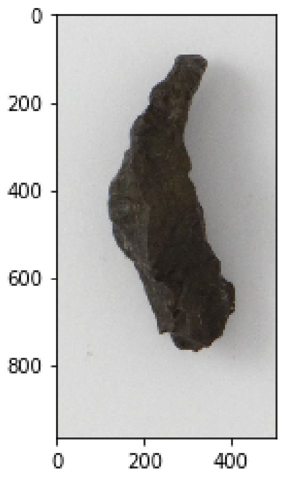
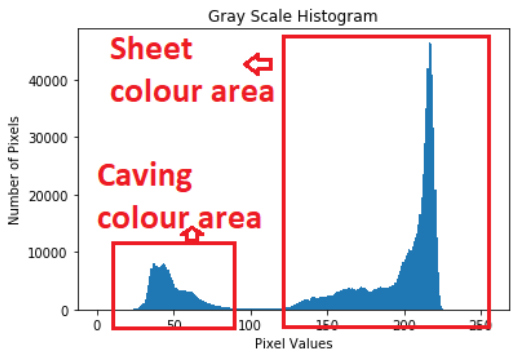


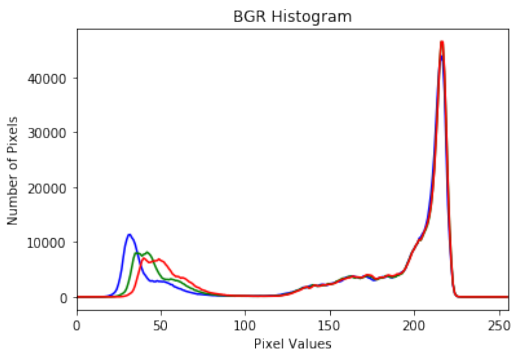
Figure 5.1: Elongated Caving

Analysis were done to realize what the boundaries should be adopted. A grayscale histogram (Figure 5.2a) shows the distribution of the colours from the caving image; consequently, it clearly looks that lower values are associated with the cavings, and higher values are related to the sheet colour. The same analysis were performed in BGR space (Figure 5.2b).

In order to keep improving the model of this project in relation to the different features that need to be extracted, more caving images will be needed for this purpose. As a result, it is highly important analyze the new dataset colour so that the threshold should be updated to maintain or increase the performance of the model.



(a) Grayscale Histogram



(b) BGR Histogram

Figure 5.2: Colour Histograms

5.1.3 Roundness Analysis

There are many research on how to determine roundness of particles as it is explained in section 6.2. For that reason, as it is mentioned previously, a mathematical approach is used which is circularity of objects, in this case referred to cavings. Different analysis were performed. These show image resolution impact on perimeter and area values using CV, and how the roundness values are distributed according to their shapes.

For example, a caving sample was taken to perform how the perimeter varied based on different resolution. For that reason, perimeter was calculated using tools from OpenCV library, and a ratio between perimeter and image size was obtained as can be seen in Figure 5.3. Based on the results seen in Figure 5.4 and table 5.2, it is observed that perimeter values increase while image resolution also increases. Given these points, it was determined that 50x50 pixels is the lowest image size to be used without losing much information about edge definitions of cavings. Nevertheless, if it is possible to use high resolution, results will be much better. This depends on computer power availability.

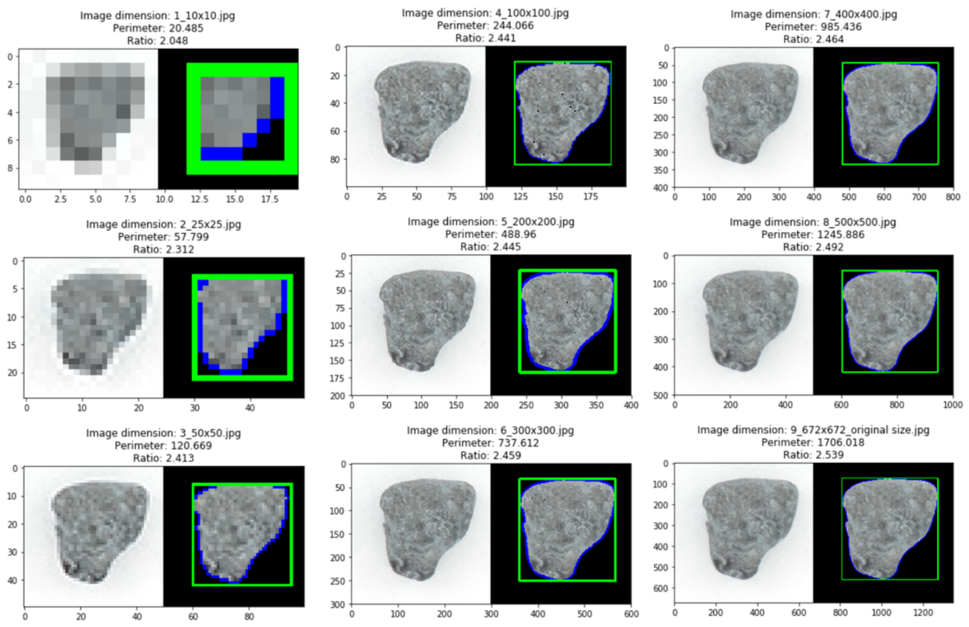


Image dimension: 1_10x10.jpg
Perimeter: 20.485
Ratio: 2.048
Image dimension: 4_100x100.jpg
Perimeter: 244.066
Ratio: 2.441
Image dimension: 7_400x400.jpg
Perimeter: 985.436
Ratio: 2.464
Image dimension: 2_25x25.jpg
Perimeter: 57.799
Ratio: 2.312
Image dimension: 5_200x200.jpg
Perimeter: 488.96
Ratio: 2.445
Image dimension: 8_500x500.jpg
Perimeter: 1245.886
Ratio: 2.492
Image dimension: 3_50x50.jpg
Perimeter: 120.669
Ratio: 2.413
Image dimension: 6_300x300.jpg
Perimeter: 737.612
Ratio: 2.459
Image dimension: 9_672x672_original size.jpg
Perimeter: 1706.018
Ratio: 2.539

Figure 5.3: Resolution Impact on Perimeter

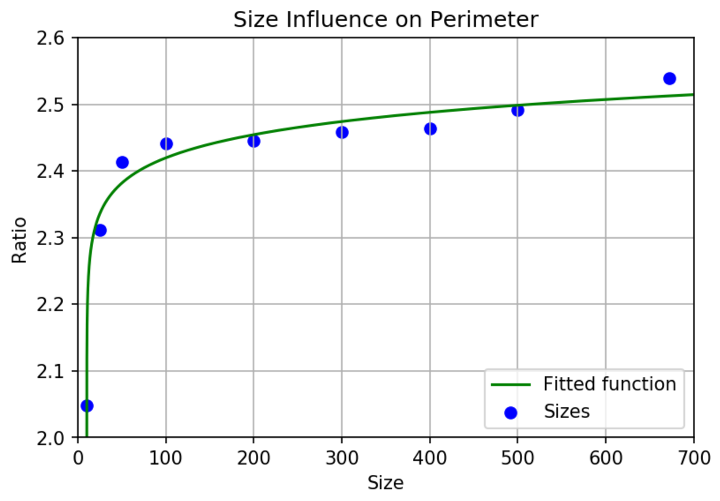


Figure 5.4: Perimeter Variation Based on Resolution

A sensitivity analysis was performed to show the variation of area, and two types of perimeters values. In Figure 5.5a, it expected to see an increasing in area values while size image increases. For this parameter, a ratio calculation was not performed because the concept of this was showed analyzing perimeter variation based on image size (Figure 5.4). Also, in order

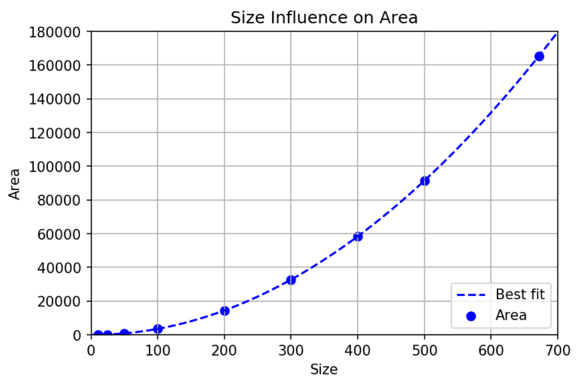
Dimension (pixels)	Ratio
10x10	2.048
25x25	2.312
50x50	2.413
100x100	2.441
200x200	2.445
300x300	2.459
400x400	2.464
500x500	2.492
672x672	2539
Average Ratio	**2.401**

Table 5.2: Results of Resolution Impact on Perimeter

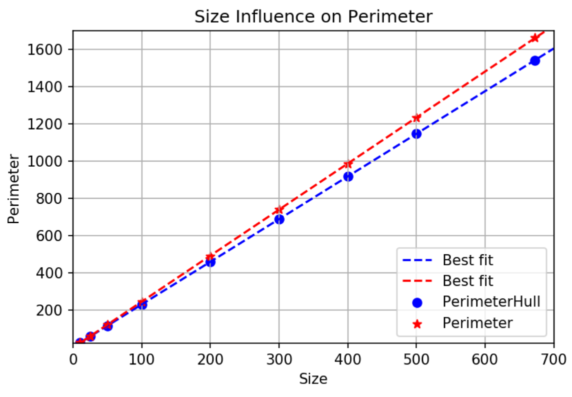
to compute perimeter of cavings, there are some parameters that need to be known, and they are perimeter and convex perimeter. To calculate convex perimeter, OpenCV library provides a tool called convex hull which was already explained in section 2.3. The difference between perimeter and convex perimeter is fundamental to perform a correct computing of edge definition. Thus, it is observed in Figure 5.5b that there is a difference in relation to **Perimeter** and **PerimeterHull**, which is calculated using convex hull tool. Also, the same difference can be seen in Figure 5.5c related to roundness values. Using convex hull tool of OpenCV, an accurate roundness value is obtained, and this was demonstrated using a circle as a reference as can be seen in 4.13 from Chapter 4, where roundness value is around 0.99, knowing that 1 corresponds for a circle shape.

A histogram analysis was performed to show the complete data base in relation to circularity values, which includes all types of caving shapes. The purpose is to show what the circularity range values are in the database as can be seen in Figure 5.6.To generate this histogram, it was required first not to take into account pictures taken from lateral side, an example of this is figure 5.7. This type of pictures are not suitable for roundness computing since it does not represent the whole caving perimeter. Also, histograms per shape were generated to know the distribution per every single category, and this can be seen in Appendix B.6.4.

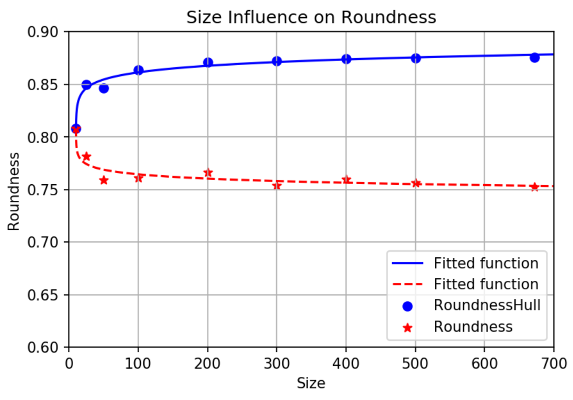
Circularity approach works quite good, but it was noticed that there are some calculations which are not accurate. These calculations are related to cavings which have elongated shape



(a) Size Influence on Area

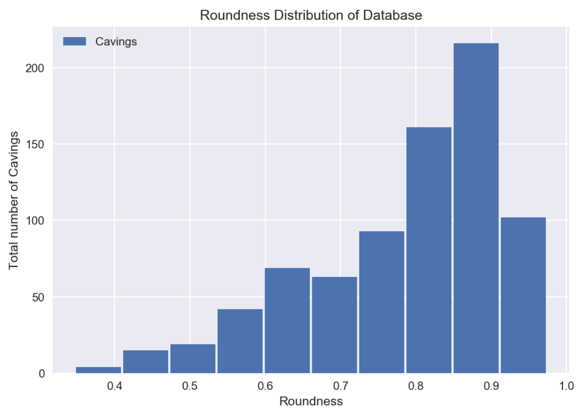


(b) Size Influence on Two Types of Perimeter

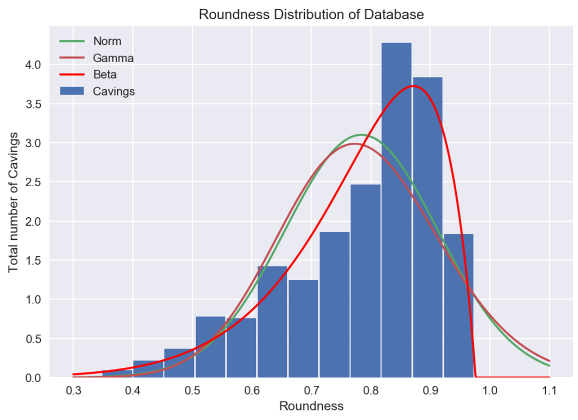


(c) Size Influence on Two Types of Roundness

Figure 5.5: Sensitivity Analysis



(a) Complete Database Distribution



(b) Fitting Distribution

Figure 5.6: Roundness Distribution of Database

(Figure 5.1). Under those circumstances, an analysis was performed to visualize the values ranges for each caving shape. The shapes are *CON-CONVEX*, *IRREGULAR*, *LARGE-ELONG*, *RECT-BLOCKY*, *SHORT-ELONG*, *SQR-BLOCKY* and *TRIANGULAR*.

Three histograms were generated to understand how roundness values are distributed in

Figure 5.7: Caving Image taken from Lateral Side

relation to caving shapes. Figure 5.8 shows the distribution of the complete dataset for roundness purposes. Even though all caving shapes have different values ranges, it is quite difficult to interpret the results. Therefore, a second type of histogram was done (Figure 5.9), which is stack type. This figure clearly shows that values related to *SQR-BLOCKY*, *TRIANGULAR*, *IRREGULAR* and *RECT-BLOCKY* have the highest roundness values. Between 0.6 and 0.8, *SHORT-ELONG* is located, and *CON-CONVEX* and *LARGE-ELONG* show the lowest values from the whole dataset.

To conclude this analysis, figure 5.10 presents a density distribution of the results, and this confirms that cavings with elongated shape tends to show lower values compared to the other types of shapes; as a result, circularity approach do a great job working on blocky, triangular and irregular cavings since the difference between their area and perimeter is not high. On the contrary, elongated cavings have a bigger difference regard their area and perimeter.

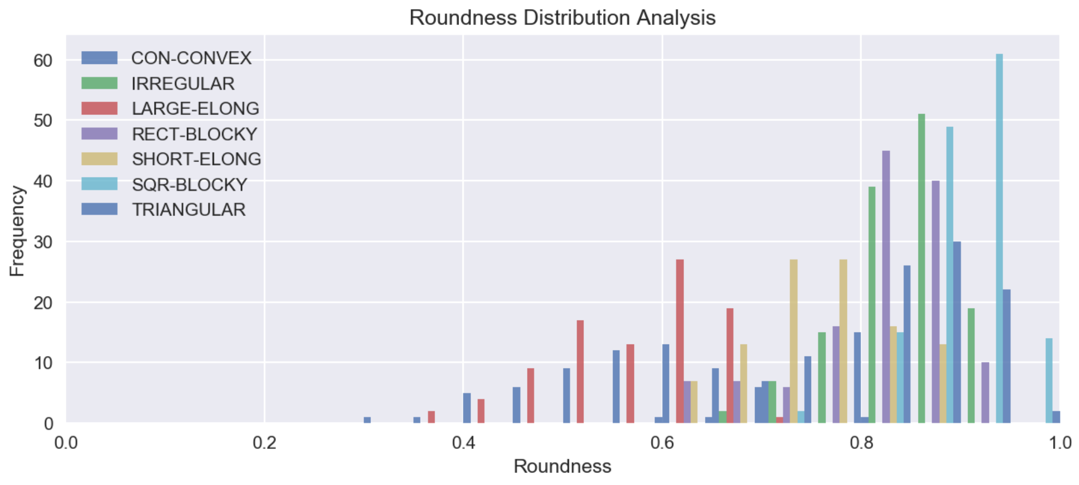


Figure 5.8: Roundness Distribution Analysis

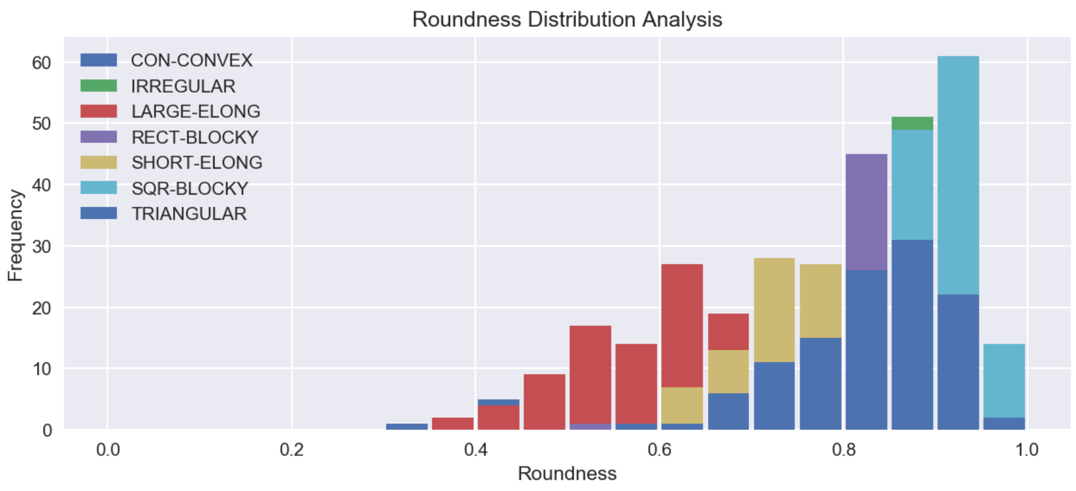


Figure 5.9: Stack Distribution Analysis

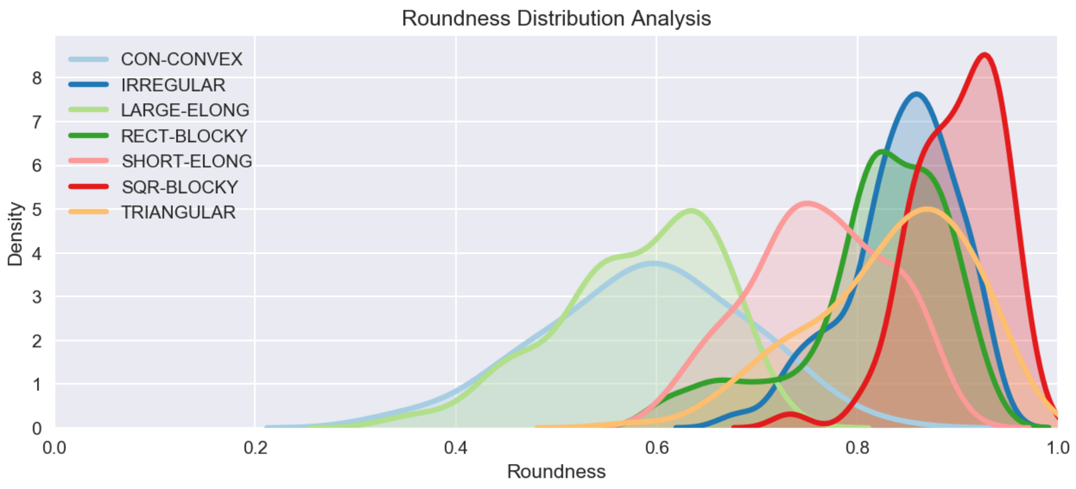


Figure 5.10: Kernel Density Analysis

The results obtained from these analysis are valuable information so that future improvements can be done in order to get better and accurate outcomes. Also, it can be determined how much resolution can be used based on computer power limitations and availability of high computer power to perform these tasks.

5.2 Computer Power Limitations

Pre-processing the data is one of the fundamental parts before using it, and the cavings dataset had to be converted into a matrix of vectors and also rotated; also, image augmentation took place and the training of the algorithms took most of the computing time. Two types of processors were used in this project, listed in Table 5.3. Unfortunately, the first setup was not able to use the advantage of the GPU computing.

Setup	CPU	GPU	RAM
ASUS FX502VM-AS73	Intel i7-7700, 2.80 GHz	-	16 GB DDR4
HP Z840 Workstation	Intel Xeon E5-2637, 3.50 GHz	NVIDIA Quadro FX 5000 4 GB	64 GB DDR3

Table 5.3: Characteristics of the used setups

While converting the images into vectors the first setup was outperformed by the task during the rotated dataset, as a result all the RAM memory was occupied by the process causing it to reboot. During the training phase for supervised learning, different size of pixels were used to identify the impact of having more details in the picture.

The results show that as long as the shape of the caving is clear, no more details are needed; 28x28 pixels are more than enough and the bigger the size the longer the computer process. Figure 5.11 shows the results in accuracy versus pixel size while using K-NN, note that for the ROTATED dataset there is no accuracy at 100x100 as the first setup was not able to finish the training.

After defining the impact of the pixel size in the dataset, it was clear that for shape recognition, only 28x28 pixels is good enough and it should be used for training. However, the performance with K-NN were not satisfactory, and a NN were used to increase the accuracy. The biggest impact of using the GPU capabilities of setup 2 can be seen during the training phase of YoloV3, setup 1 required 5 seconds to compute each time-step and setup 2 took 0.667 seconds. This difference in performance resulted in 28 hours of training against a calculated 209.7 hours, (8.74 days!, see Figure 5.12a) if the setup 1 were to be used. It is also useful when a new image is tested, and the NN gives the results in a fraction of a second, 0.4 seconds in setup 2 while using setup 1 it can take up to 14.67 seconds for a single image (Figure 5.12b).

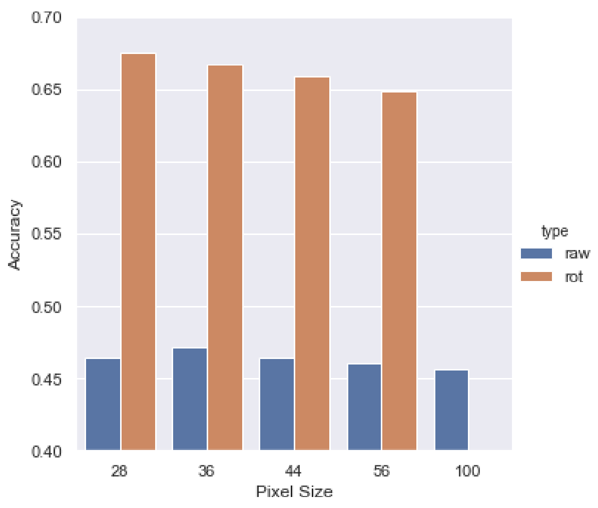
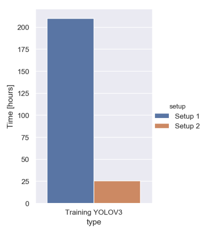
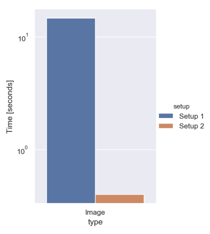


Figure 5.11: Performance in CPU, K-NN in different resolution

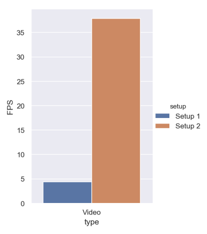
In terms of video processing, it can be done in 37.84 FPS against the 4.38 FPS in the first setup (Figure 5.12c).



(a) YoloV3 training



(b) YoloV3 image processing



(c) YoloV3 video processing

Figure 5.12: Performance of setups

5.3 Supervised Learning vs Neural Networks

When analyzing the shape of the caving, the first approach was to use supervised learning to try the most common classifiers and test their accuracy on the caving dataset. After having the first results (Figure 5.13), it was clear that K-NN was the best option (see Appendix B.3) to follow and continue to investigate with and by using an optimized K-NN for the dataset. The results were improved 52% (in Figure 5.13) to 67% (in Figure 5.11) of accuracy (29% of improvement); however, this last accuracy is not attractive for a commercial or industrial application, so it needed to be improved.

Table 5.4 shows the results of the accuracy on the RAW and ROTATED datasets for supervised learning. By using the same parameters, the accuracy increases in some of the cases: K-NN, RBF & Linear SVM and QDA, but in others it decreases: LR, DT and AB. The difference in the algorithms about how they treat the rotated images is key when it comes to increasing the dataset with data augmentation. K-NN shows the best opportunities to handle a new and larger dataset when it comes and thus, it should be used as a first try of classify the shape of the cavings. Most of the classifiers have a performance between 20% to 30% which were not used in further research in this project.

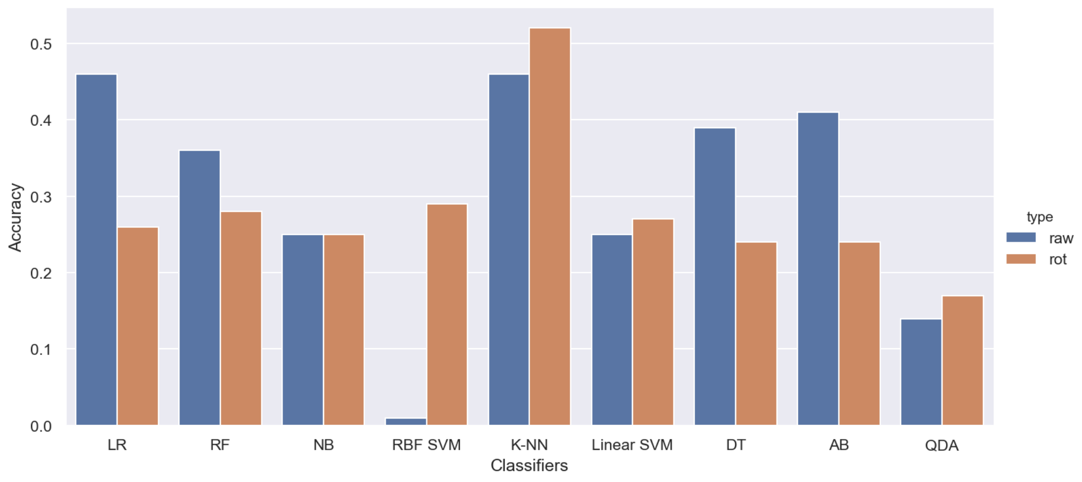


Figure 5.13: Accuracy, RAW vs ROTATED datasets

	Precision		Recall		f1-score	
Classifier	**Raw**	**Rot**	**Raw**	**Rot**	**Raw**	**Rot**
Logistic Regression	0.46	0.26	0.46	0.28	0.46	0.27
Random Forest	0.36	0.28	0.35	0.27	0.31	0.22
Naive Bayes	0.25	0.25	0.18	0.17	0.11	0.08
RBF Support Vector Machines	0.01	0.29	0.11	0.28	0.02	0.24
K-Nearest Neighbors (KNN)	0.46	0.52	0.46	0.51	0.45	0.49
Linear SVM	0.25	0.27	0.31	0.29	0.27	0.27
Decision Tree	0.39	0.24	0.4	0.31	0.37	0.26
AdaBoost	0.41	0.24	0.4	0.26	0.4	0.25
QDA	0.14	0.17	0.17	0.16	0.1	0.07

Table 5.4: Performance of Classifiers Dataset

Figure 5.14 shows the accuracy for the different architectures used and it can be seen that the average accuracy for the last and better architectures are around 30% to 40%. InceptionV3 was used to test a state-of-the-art NN and investigate how image augmentation is handled, resulting in similar values between the RAW and ROTATED dataset. This set the path to use YoloV3 as the definitive solution for analyzing the shape of the caving as it out-performs InceptionV3 and other NN.

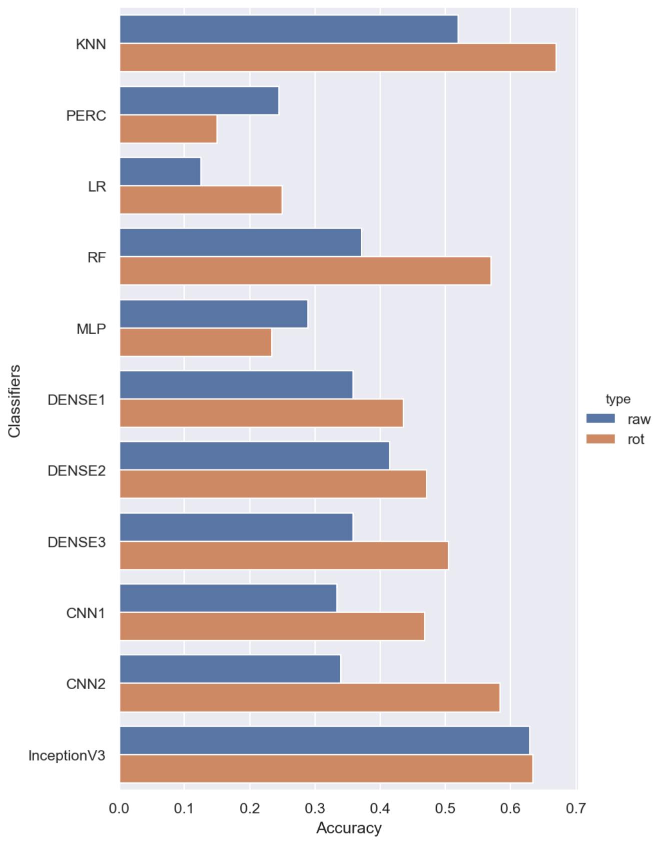


Figure 5.14: Accuracy, NN architectures

YoloV3 allowed to identify the shape of the caving with three (3) different layers: 82, 94 and 106. As seen in Figure 5.15, the caving is detected easily when it occupies a medium or small portion of the image. If the camera is too close to the caving it is difficult to get a probability that there is a caving and even more difficult to tell what kind of caving it is.

For layers 82 and 94, the IoU is above 80% and below this value for layer 106, this represents how secure the NN is to have correctly found the caving and its dimensions in the image. The detected caving will be correctly classified with at least an **accuracy of 98.25%** and for layer 106 it will be correctly classified with an **accuracy of 80.76%**.

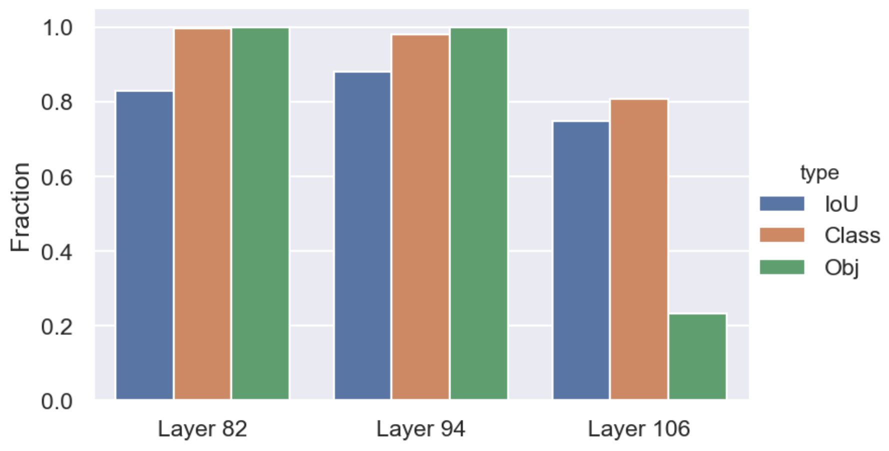


Figure 5.15: Metrics for YoloV3 with cavings dataset

After using YoloV3, new pictures can be used for testing. Figure 5.16 was not part of the training, but it gives positive results under different conditions than the trained thanks to image augmentation. The number of samples that can be detected depends on the confidence thresholds written in the code, in this project YoloV3 will only show cavings that have more than 95% of confidence that there is a caving present in the picture and then it will figure out what is the shape of the caving. Thus, the number of cavings that can be detected can range from none to multiple (more than 10 in Figure 5.16) depending on the confidence threshold. Figure 5.17 is an example of how the position of the caving can play a role during testing, the orientation is changed from Figure 5.17a to 5.17a and the sample on the left in detected. More samples of detection in images can be found in Appendix C.2.

The results obtained with YoloV3 are based on the dataset, for a caving to be recognized it would be easier if it is placed alone in a white background. in Figure 5.16 some of the cavings are too close to others or some of it is out of the picture, thus making it more difficult for the NN to detect it.

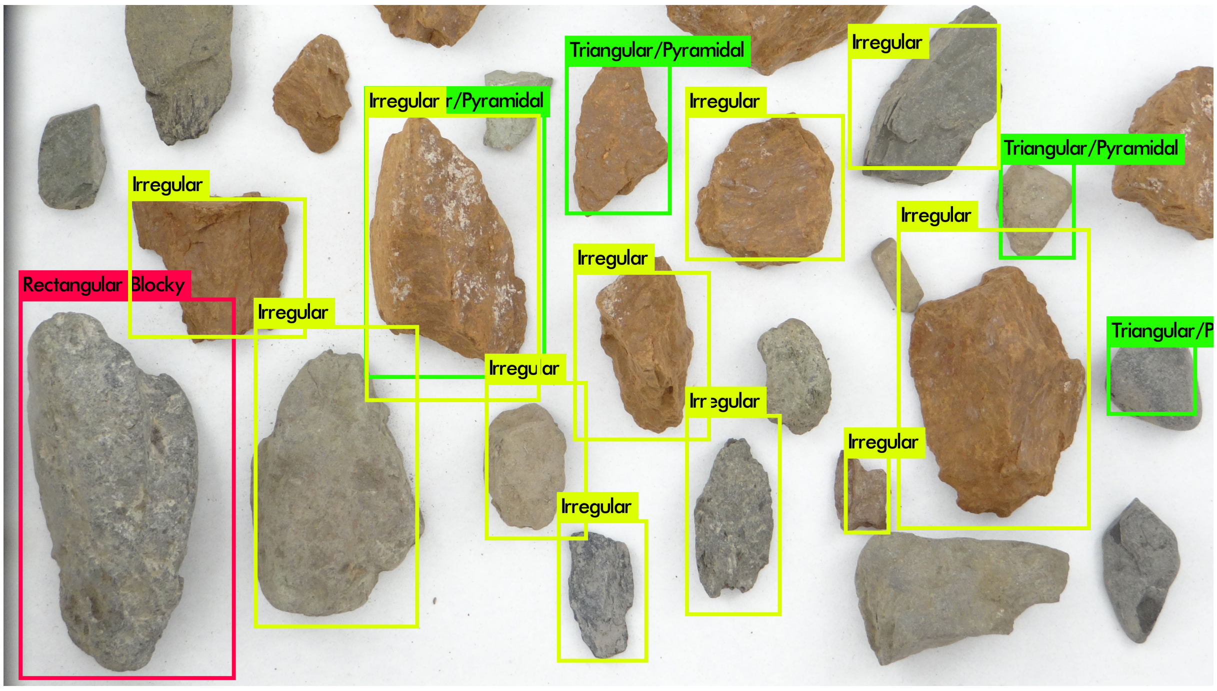
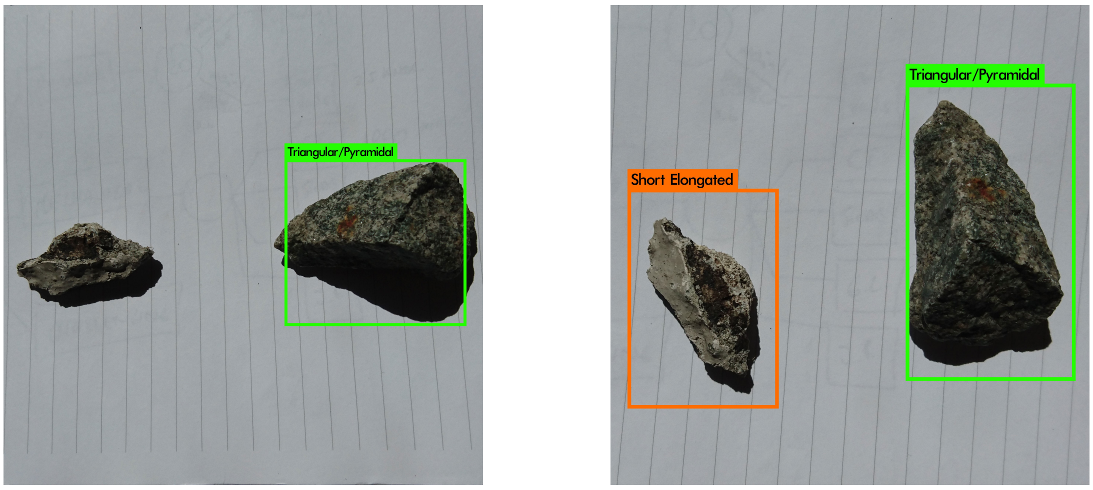


Figure 5.16: Cavings detection with new samples



(a) cavings placed horizontally **(b)** Cavings placed vertically

Figure 5.17: Performance of cavings detection

Chapter 6

Conclusions and Future Work

6.1 Conclusions

- Feature extraction for shape, size, colour and roundness was successfully done on the cavings dataset, either using the RAW, ROTATED or the AUGMENTED dataset. This was done using packages written in Python, it eased handling the volume of data and keep the data organized. Size was extracted to get an idea of the volume of rocks, the roundness score is related with transport of cutting in a wellbore, color has been extracted to match it with the formation being drilled with the information from well-logs and telemetry (LWD and MWD) and shape was successfully identified with object detection to link it with the failure mechanisms.
- There are four main algorithms in relation to shape, size, color and roundness features. Each algorithm works independently. In addition, other algorithms were developed to perform classification, sensitivity analysis and other operations, which were explained in this document.
- Size was computed using a tool which in this case was a color scale as reference object. The measurement was done in centimeters. First, the reference object was analyzed since its width measurement was known, and then the caving size feature is extracted with high accuracy.
- Colour feature extraction was performed using thresholds based on the colours from the

different samples in the complete database. It is important to mention that only 3 dominant colours are being extracted for this project. However, the algorithm is able to extract more colours depending on the necessity or for future studies. Also, the results of this feature are RGB codes, and percentage of each dominant colours.

- Edge definition is affected by the resolution of the image as it was explained before. If an image has a higher ppi, its perimeter would be high; consequently, a significant result will be obtained if images with high resolution are used. The number of pixels per inch plays a tremendous role in the calculation of this feature. It was also determined when the term *"Edge Definition"* or *"Roundness"* is used. It is referred to Circularity which is the used approach to compute this feature. It is important to highlight that the database was reduce to perform this analysis because lateral images were eliminated. These lateral pictures are not representatives regarding area and perimeter of cavings.

- Seven different classes were trained and validated using several classifiers, from supervised and unsupervised learning, among the ones that yielded the best results: K-Nearest Neighbors, InceptionV3 and YoloV3. The most important for the project is YoloV3 as it allows to detect objects in video and real time video; by using python it was also possible to access the camera of an android device and analyze the live-stream video.

- GPU made a great impact on training Neural Networks using Linux. It is always advised to use it when available . This will enable fast computing and reproductibility of the steps taken to process the data, train the model and deploy it for testing it with new samples.

6.2 What Can Be Improved?

The main concern of the investigation is the dataset, all the well-know challenges datasets like Dogs vs Cats or the MNIST dataset contains more than 10,000 samples per category. To improve the results, the main goal is to increase the cavings dataset to have at least 1,000 samples per category.

Regarding the edge definition or roundness, a mathematical approach was used to calculate circularity of cavings. Even though this is a valid technique because most of the used methods involve visual characterization, an improvement for this feature calculation can be adopted using computer vision based on an existing ISO 9276-6 [ISO, 2008]. Also, sphericity is an important parameter which needs to be calculated and taken into account in order to get better results in relation to edge definition. There are many shape definitions and research to calculate or define roundness as accurate as possible (see table 6.1).

Name	**Author**	**Year**	**Based on**
Roundness	Pentland	1927	Area
Roundness	Cox	1927	Area-perimeter
Roundness	Tickell	1931	Area
Circularity	Horton	1932	Drainage basin
Outline circularity	Wadell	1935	Circle diameter
Degree of circularity	Wadell	1935	Perimeter
Inscribed circle sphericity	Riley	1941	Circle diameter
Circularity	Krumbein and Sloss	1963	Chart
-	Janoo	1998	Area-perimeter
Shape factor	Sukumaran	2001	Segmentation of particle and angles

Table 6.1: Chronological summary of particle roundness and sphericity for 2D particles adapted from [Rodriguez et al., 2013]

Since the data base contains short and large elongate cavings as well short and large blocky cavings, sphericity has an important role in order to implement a scale for roundness classification. Power's scale [Powers, 1953] is one of most used in the industry (see table 6.2). This scale shows 6 categories where sphericity is considered as can be seen in figure 6.1.

Grade Terms	**Class Intervals**	**Geometric Means**
Very angular	0.12-0.17	.14
Angular	0.17-0.25	.21
Subangular	0.25-0.35	.30
Subrounded	0.35-0.49	.41
Rounded	0.49-0.70	.59
Well rounded	0.7-1.00	.84

Table 6.2: Roundness Grades adapted from [Powers, 1953]

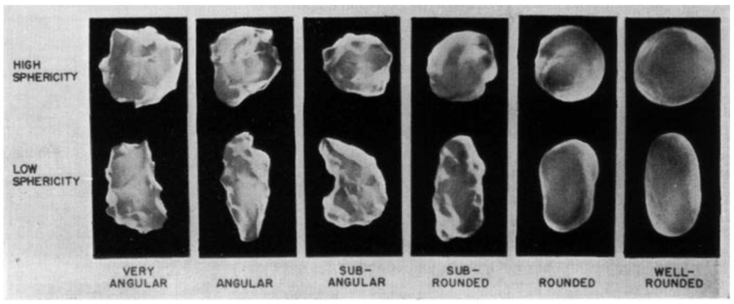


Figure 6.1: Roundness Scale taken from [Powers, 1953]

Nowadays, the power of computer vision is endless, and this will continue improving in the future; as a result, a recommendation to get accurate measurements is to use Wadell's method [Wadell, 1935] which consists in dividing the average of the radius of the corners of the particle by the radius of the maximum inscribed circle in the particle, in this case a caving. This method can be seen in figure 6.2 where on the right side explains Wadell's approach.

This method is the most accurate yet, but it is complex to implement; however, computer vision tools can use it in combination with Power's scale [Powers, 1953], and it can be compared to the ISO 9276-6 [ISO, 2008] to obtain reliable and accurate results.

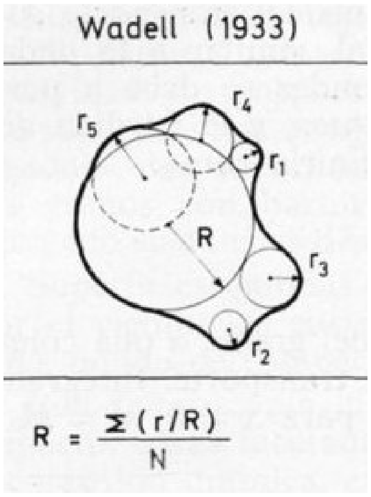


Figure 6.2: Wadell's formula taken from [Corrales Zarauza, 1977] where R is the maximum inscribed circle, r is the radius of curvature of particle corners, and n is the number of particle corners measured.

6.3 The Present Development

Looking at the dataset as it is at the time, it needs to incorporate rotation for YoloV3. Two approaches have been taken with similar but not definitive results:

- Use of the RAW dataset and image augmentation, however rotation has been addressed only for multiples of 90 degrees, it cannot be done automatically due to a restriction on the calculation of the bounding box, it needs to be straight on every picture regardless of the orientation of the sample. This is a limitation at the moment that requires a closer look; python can be useful to calculate the new bounding boxes after the caving has been rotated.
- The ROTATED dataset has already rotated the cavings but the creation of the labels for YoloV3 is a manual procedure. This means that 25,536 images needs to be manually labelled and tagged with the bounding box. It requires time but after having this dataset ready for image augmentation it will enhance the detection of the cavings.

Training was done using a single caving per image, with the available samples a new dataset can be created by placing different shapes close to others and mix the samples in the picture. This will enable recognition of more complex patterns that will have on-rig applicability.

6.4 The Future Development

6.4.1 Going from 2D to 3D and Live Data

All the samples so far have been 2D pictures of clean cavings, perfectly taken pictures with no shadows in a white-box. When it comes to think about real conditions in a rig, this cannot match with the ones presented in the dataset. The next step is to train the algorithms with pictures taken from unwashed cavings placed in a white surface or even at the shale shakers. This is obviously a more difficult task as it requires the personnel to be present on site and expect to have the drilling problems in order to have the specific related cavings. As with any other dataset, the more the better and the dataset needs to be balanced to prevent over-fitting of a specific caving.

As previously discussed, if we want to consider a volume calculation with the amount of cavings that are coming from the well, a 3D digitalization of the caving needs to be done. One of the alternatives is using the shadow as a distortion of the third dimension of the caving; in a more elaborated scheme is to use laser scanners and gather data to give the measurement of depth. There are important approaches to work on 3D digitalization of cavings which are:

- **Laser Technology:** The principle of this technology which consists in scan and analyzing real objects to collect information such as shape, colour and depth so that this technology using point clouds [Qi et al., 2017] can recreate a 3D digital model of the object. This approach is able to quantify drilling cuttings volume, detect presence of cavings in shale shakers and build 3D model of caving shapes [Han et al., 2018]. Also, the great advantage is that this can be applied in real time to improve drilling efficiency. Even though applications of this technology in relation to drilling operations are prototypes, laser equipment has a promising future.

- **Multiple Cameras:** This technique derives or use a 3D mesh from multiple static pictures or images. It is possible to build a 3D model using only two images, but it is better to have more images in order to have a better model. Also, this can be achieved using one camera, but it requires to have a turn table; as a result, the object will be turning around the camera, and multiple images will be captured to build the 3D model. This method is also applicable in real-time [Petit et al., 2010].

- **Camera with Mirror:** This method uses one camera and a mirror or multiple mirror to generate multiple images from the same object. The depth of an object can be extracted using mirrors located at specific position for a good 3D object reconstruction [Hu et al., 2005]. The great advantage is a cheaper option compared to the other techniques previously mentioned.

6.4.2 Robotic Arm

The main goal in terms of digitalization in drilling operations is automatize operations and tasks as much as possible. For that reason, a combination of the model created in this project with a robotic arm can help to speed up decision-making and analysis process with high efficiency and accuracy. Nowadays, robotic arm technology is being used for different applications in multiples industries such as sampling [Fleischer et al., 2016], welding, material handling, palletizing, assembling, etc.. This equipment can be used for caving sampling. A system needs to be build in conjunction with the model for recognizing shape, edge definition, colour and size.

It is important to highlight that shape can be described with caving covered with mud. However, if a complete description of cavings is needed, a more robust system needs to be built which consists in a sampling system using a robotic arm, a cleaning system for identification of other caving features, and cameras for capturing images and videos with the purpose of real time application. To implement this system, there are important aspects to be taken into account, and one of this is the available space in a rig to implement this equipment.

In summary, the complete process to be implemented will consist in a robotic arm to sample relevant cavings in the shale shakers. Also, cavings need to be recognized using this project model in live stream. Once the caving has been collected, the robotic arm takes it to the cleaning and drying system to run the model for edge definition, colour and size features. This potential system needs to be tested before deploying in real operations, but it could be a breakthrough to identify wellbore instability problems and provide remedial actions in real time operations.

6.4.3 Linking to Drilling Parameters

Being able to recognize features from caving images and videos is a remarkable achievement to pursue a safer, faster and more economically drilling environment. However, it is fundamental to link these caving features with drilling parameters, which are complementary evidence, so that causal mechanism related to wellbore instability problems can be identified. Caving analysis needs to be complemented by drilling parameters such as torque, drag, hydraulics, wellbore inclination, etc. to be able to identify the most likely causal mechanism and to adopt a suggested remedial action. Otherwise, misunderstanding of the situation may lead to further aggravation of the problem.

Further analysis with delay issues will need to be added into the research, the cavings that are being shown in the shale shakers spent some time inside the wellbore and this time is directly related with the cuttings transport theory, the minimum transport velocity and hole cleaning. The cavings should also be linked with the log data and downhole data from the LWD and MWD as it must match the drilled formations, making a synergy between computer vision, well logs and drilling parameters. All of the different characteristics should be looked at once to have a clear and complete picture of the well and its behavior.

The next step is to built a more robust model which consists in adding drilling parameters to caving features model; proposed by [Ashrafi et al., 2019] the drilling parameters and well-logs have been used to predict the ROP, this can be used as a first step to link different parameter and the features of the cavings. The Figure 6.3 shows the future model that needs to be added to the Caving Features Model.

As it was explained in figure 2.5 and appendix E.1, drilling parameters are fundamental to build the final model in order to identify causal mechanism and propose a remedial action. Extracting features from cavings is the first step to accomplish the final goal which is to build a complete and robust model for wellbore instability problems based on cavings information. Drilling Engineering and Geology fields are essential to reach this goal. The model of this project which is based in ML, CV and Neural Networks has to be used to extract information from cavings when they arrive on the shale shakers, and this information needs to be analyzed in conjunction to drilling parameters at the time that cavings were produced. Drilling parameters would be the input information, and a model for this needs to be built. According to the chord diagram previously explained, there are many parameters which are involved in order to establish what the most likely causal mechanism is, and what the most appropriate remedial action is. Figure 6.3 shows the potential future model involving the link between caving features and drilling parameters (see Appendix D.1 for Caving Features Model Flowchart, and see Appendix D.2 for future model flowchart)

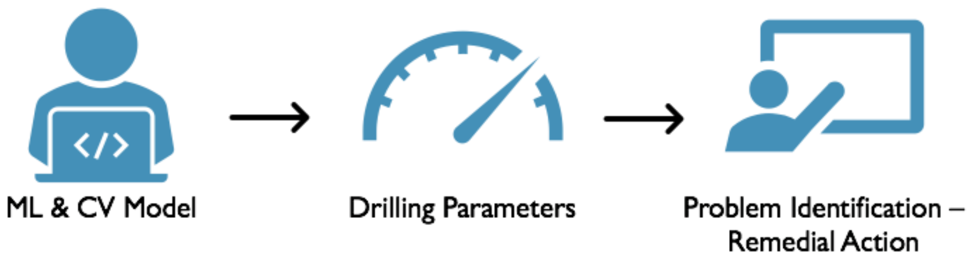


Figure 6.3: Caving Analysis Future Model

To understand the process of linking caving features with drilling parameters, an example will be explained. Based on figure E.2 in appendix E.1, this chord diagram shows an example to identify ***Pore Pressure*** problems. It can be observed that shape, size and edge definition are important to correlate which problem might be occurring in the wellbore. However, it cannot be determined what the problem is without knowing other parameters associated to drilling operations. For this example, any type of edge definition can lead to this problem, and only elongated and concavo-convex caving shapes are associated to this situation; also, structures related to pore pressure problems are conchoidal, flame and fissibility. In addition, size (<10mm, 11-20mm, 21-40mm, >41mm), dominant lithology (mudstone, marl, claystone), drag trend (normal, increasing, decreasing, overpull, jarring, etc.), torque trend, wellbore inclination and other parameters (see more details in figure E.2). If those parameters are observed during the operation, there is a high probabilities that the most likely causal mechanism is ***Pore Pressure***, and the potential remedial action would be ***Raise Mud Weight***. This is just an example of how to link caving features with drilling parameters, and it does not represent an absolute truth.

6.4.4 Machine Learning for Rock Type Identification

A potential application or further development involves colour feature. Since the model of this project extracts colour information, it is also able to store RGB codes of the mos dominant colour present in a caving sample in this case 3 colours. However, manipulating K-means algorithm allows us to increase the number of colours to extract from a caving sample depending on the scope of the study. This future study will enable an alternative to complement existing models regarding facies predictions [Bestagini et al., 2017], and this will increase the accuracy of them. A great advantage of having a history of the colours of the rock type allows us to correlate with formations which were unstable and presented problems previously in other drilling operations; consequently, precautionary actions can be implemented.

References

[Aadnoy et al., 1987] Aadnoy, B., Chenevert, M., et al. (1987). Stability of highly inclined boreholes (includes associated papers 18596 and 18736). *SPE Drilling Engineering*, 2(04):364–374.

[Ashrafi et al., 2019] Ashrafi, S. B., Anemangely, M., Sabah, M., and Ameri, M. J. (2019). Application of hybrid artificial neural networks for predicting rate of penetration (rop): A case study from marun oil field. *Journal of Petroleum Science and Engineering*, 175:604–623.

[Bestagini et al., 2017] Bestagini, P., Lipari, V., and Tubaro, S. (2017). A machine learning approach to facies classification using well logs. In *SEG Technical Program Expanded Abstracts 2017*, pages 2137–2142. Society of Exploration Geophysicists.

[Bradley, 1979] Bradley, W. (1979). Failure of inclined boreholes. *Journal of Energy Resources Technology*, 101(4):232–239.

[Bradski, 2000] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

[Caenn, 2015] Caenn, R. (2015). Drilling fluids technology. https://www.aogr.com/magazine/cover-story/new-solutions-practices-redefining-cutting-edge-in-fluids-technology. Accessed: 2019-05-23.

[Cao et al., 2016] Cao, Q., Banerjee, R., Gupta, S., Li, J., Zhou, W., Jeyachandra, B., et al. (2016). Data driven production forecasting using machine learning. In *SPE Argentina Exploration and Production of Unconventional Resources Symposium*. Society of Petroleum Engineers.

[Cheatham Jr et al., 1984] Cheatham Jr, J. et al. (1984). Wellbore stability. *Journal of petroleum technology*, 36(06):889–896.

[Chen et al., 2003] Chen, G., Chenevert, M. E., Sharma, M. M., and Yu, M. (2003). A study of wellbore stability in shales including poroelastic, chemical, and thermal effects. *Journal of Petroleum Science and Engineering*, 38(3-4):167–176.

[Corrales Zarauza, 1977] Corrales Zarauza, I. (1977). Estratigrafía. Technical report.

[Fleischer et al., 2016] Fleischer, H., Drews, R. R., Janson, J., Chinna Patlolla, B. R., Chu, X., Klos, M., and Thurow, K. (2016). Application of a dual-arm robot in complex sample preparation and measurement processes. *Journal of laboratory automation*, 21(5):671–681.

[Galea and Capelo, 2018] Galea, A. and Capelo, L. (2018). *Applied Deep Learning with Python: Use scikit-learn, TensorFlow, and Keras to create intelligent systems and machine learning solutions*. Packt Publishing Ltd.

[Galvis et al., 2014] Galvis, C., Viviana, L., Corzo Rueda, R., and Arguello Fuentes, H. (2014). Caving depth classification by feature extraction in cuttings images. *Earth Sciences Research Journal*, 18(2):157–163.

[Gurney, 2014] Gurney, K. (2014). *An introduction to neural networks*. CRC press.

[Han et al., 2018] Han, R., Ashok, P., Pryor, M., van Oort, E., et al. (2018). Real-time 3d computer vision shape analysis of cuttings and cavings. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers.

[Hastie et al., 2013] Hastie, T., James, G., Tibshirani, R., and Witten, D. (2013). An introduction to statistical learning with applications in r.

[Hegde et al., 2015] Hegde, C., Wallace, S., Gray, K., et al. (2015). Real time prediction and classification of torque and drag during drilling using statistical learning methods. In *SPE Eastern Regional Meeting*. Society of Petroleum Engineers.

[Hernández-García and König, 2018] Hernández-García, A. and König, P. (2018). Do deep nets really need weight decay and dropout? *arXiv preprint arXiv:1802.07042*.

[Hu et al., 2005] Hu, B., Brown, C., and Nelson, R. (2005). Multiple-view 3-d reconstruction using a mirror.

[Hughes, 2019] Hughes, B. (2019). International rig count. `http://phx.corporate-ir.net/phoenix.zhtml?c=79687&p=irol-rigcountsintl`. Accesed: 2019-03-30.

[ISO, 2008] ISO, I. (2008). 9276-6 representation of results of particle size analysis—part 6: Descriptive and quantitative representation of particle shape and morphology.

[Itseez, 2014] Itseez (2014). *The OpenCV Reference Manual*. Itseez, 2.4.9.0 edition.

[Kantardzic, 2011] Kantardzic, M. (2011). *Data mining: concepts, models, methods, and algorithms*. John Wiley & Sons.

[Khojasteh et al., 2015] Khojasteh, P., Ahmadyfard, A., Tokhmechi, B., and Mirmahdavi, S. A. (2015). Automatic detection of formations using images of oil well drilling cuttings. *Journal of Petroleum Science and Engineering*, 125:67–74.

[Kluyver et al., 2016] Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J. B., Grout, J., Corlay, S., et al. (2016). Jupyter notebooks-a publishing format for reproducible computational workflows. In *ELPUB*, pages 87–90.

[LeCun, 1998] LeCun, Y. (1998). The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

[Li et al., 2017] Li, Y., Han, Y., et al. (2017). Decline curve analysis for production forecasting based on machine learning. In *SPE Symposium: Production Enhancement and Cost Optimisation*. Society of Petroleum Engineers.

[MacQueen et al., 1967] MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

[Mishra and Datta-Gupta, 2017] Mishra, S. and Datta-Gupta, A. (2017). *Applied Statistical Modeling and Data Analytics: A Practical Guide for the Petroleum Geosciences*. Elsevier.

[Moos et al., 2003] Moos, D., Peska, P., Finkbeiner, T., and Zoback, M. (2003). Comprehensive wellbore stability analysis utilizing quantitative risk assessment. *Journal of Petroleum Science and Engineering*, 38(3-4):97–109.

[Parloff, 2016] Parloff, R. (2016). Why deep learning is suddenly changing your life. `http://fortune.com/ai-artificial-intelligence-deep-machine-learning/`. Accessed: 2019-05-19.

[Parsian, 2015] Parsian, M. (2015). *Data Algorithms: Recipes for Scaling Up with Hadoop and Spark*. " O'Reilly Media, Inc.".

[Pedregosa et al., 2011a] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011a). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

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

[Perner, 2007] Perner, P. (2007). *Machine Learning and Data Mining in Pattern Recognition: 5th International Conference, MLDM 2007, Leipzig, Germany, July 18-20, 2007, Proceedings*, volume 4571. Springer.

[Peterson, 2016] Peterson, B. (2016). *Understanding exposure: how to shoot great photographs with any camera*. AmPhoto books.

[Petit et al., 2010] Petit, B., Lesage, J.-D., Menier, C., Allard, J., Franco, J.-S., Raffin, B., Boyer, E., and Faure, F. (2010). Multicamera real-time 3d modeling for telepresence and remote collaboration. *International journal of digital multimedia broadcasting*, 2010.

[Powers, 1953] Powers, M. C. (1953). A new roundness scale for sedimentary particles. *Journal of Sedimentary Research*, 23(2):117–119.

[Qi et al., 2017] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660.

[Redmon, 2016] Redmon, J. (2013–2016). Darknet: Open source neural networks in c. http://pjreddie.com/darknet/.

[Redmon and Farhadi, 2018] Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

[Rockwash, 2019] Rockwash (2019). Rockwash. http://www.rockwash.co.uk/sample-cleaning/. Accessed: 2019-05-19.

[Rodriguez et al., 2013] Rodriguez, J., Edeskär, T., and Knutsson, S. (2013). Particle shape quantities and measurement techniques: a review. *The Electronic journal of geotechnical engineering*, 18:169–198.

[Rosebrock, 2015] Rosebrock, A. (2015). Imutils. https://github.com/jrosebr1/imutils. Accessed: 2019-05-20.

[Rosebrock, 2017a] Rosebrock, A. (2017a). *Deep Learning for Computer Vision with Python: ImageNet Bundle*. PyImageSearch.

[Rosebrock, 2017b] Rosebrock, A. (2017b). *Deep Learning for Computer Vision with Python: Practitioner Bundle*. PyImageSearch.

[Samplewash, 2019] Samplewash (2019). Samplewash. http://www.samplewash.com. Accessed: 2019-05-19.

[Sidahmed et al., 2017] Sidahmed, M., Roy, A., Sayed, A., et al. (2017). Streamline rock facies classification with deep learning cognitive process. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers.

[Singh, 2010] Singh, A. (2010). Machine learning. http://www.cs.cmu.edu/~aarti/Class/10701/hws.html. Accessed: 2019-06-06.

[Skea et al., 2018] Skea, C., Rezagholilou, A., Far, P. B., Gholami, R., and Sarmadivleh, M. (2018). An approach for wellbore failure analysis using rock cavings and image processing. *Journal of Rock Mechanics and Geotechnical Engineering*, 10(5):865–878.

[Sneed, 2017] Sneed, J. (2017). Predicting esp lifespan with machine learning. In *Unconventional Resources Technology Conference, Austin, Texas, 24-26 July 2017*, pages 863–869. Society of Exploration Geophysicists, American Association of Petroleum ....

[Solem, 2012] Solem, J. E. (2012). *Programming Computer Vision with Python: Tools and algorithms for analyzing images*. " O'Reilly Media, Inc.".

[SPE, 2019] SPE (2019). Onepetro spe. https://www.onepetro.org. Accessed: 2019-01-25.

[Tandon et al., 2019] Tandon, S. et al. (2019). Integrating machine learning in identifying sweet spots in unconventional formations. In *SPE Western Regional Meeting*. Society of Petroleum Engineers.

[Unrau et al., 2017] Unrau, S., Torrione, P., Hibbard, M., Smith, R., Olesen, L., Watson, J., et al. (2017). Machine learning algorithms applied to detection of well control events. In *SPE Kingdom of Saudi Arabia Annual Technical Symposium and Exhibition*. Society of Petroleum Engineers.

[Van Rossum and Drake, 2011] Van Rossum, G. and Drake, F. L. (2011). *The python language reference manual*. Network Theory Ltd.

[Wadell, 1935] Wadell, H. (1935). Volume, shape, and roundness of quartz particles. *The Journal of Geology*, 43(3):250–280.

[Wallace et al., 2015] Wallace, S. P., Hegde, C. M., Gray, K., et al. (2015). A system for real-time drilling performance optimization and automation based on statistical learning methods. In *SPE Middle East Intelligent Oil and Gas Conference and Exhibition*. Society of Petroleum Engineers.

[Wirth, 2001] Wirth, M. A. (2001). Shape analysis and measurement. *University of Guelph. CIS*, 6320.

[Yellowbrick, 2018] Yellowbrick (2018). Confusion matrix. https://www.scikit-yb.org/en/latest/api/classifier/confusion_matrix.html. Accessed: 2019-05-25.

[York et al., 2009] York, P. L., Prichard, D. M., Dodson, J. K., Dodson, T., Rosenberg, S. M., Gala, D., Utama, B., et al. (2009). Eliminating non-productive time associated with drilling through trouble zones. In *Offshore Technology Conference*. Offshore Technology Conference.

[Zhang et al., 2008] Zhang, J., Standifird, W. B., Lenamond, C., et al. (2008). Casing ultradeep, ultralong salt sections in deep water: A case study for failure diagnosis and risk mitigation in record-depth well. In *SPE annual technical conference and exhibition*. Society of Petroleum Engineers.

[Zhu et al., 2017] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*.

Appendices

Appendix A

White-box Setup and Blueprints

A.1 Photography Setup, The White-box

The dataset of cavings will have a well defined picture under standard conditions, making every photo comparable with the rest of them. The easiest way to capture the caving with smooth light conditions, eliminating shadows and enhancing the details in its texture is by using a white-box.

The white-box was built from scratch but there were different setups that were tested before in order to capture all important details from the cavings and learn from about eliminating environmental conditions of daylight. Figure A.1 shows the blueprint of the first setup that was built using a tripod, a camera and one lamp. The distances of the elements are important in relation to camera settings. This setup would be the simplest but it will create a problem with the shadow to the far end of the caving from the light and it will also be affected by the conditions of light from the room as there is no way to isolate the sample.

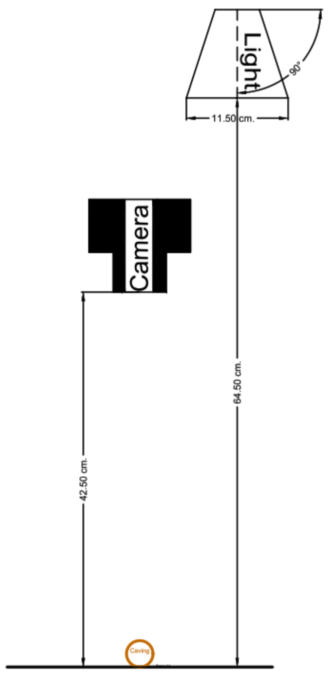
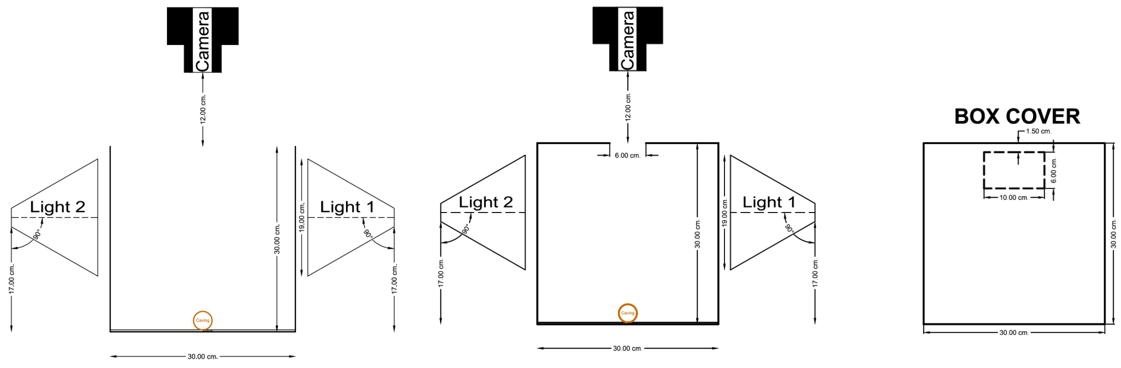


Figure A.1: Setup 1

Figure A.2a shows the second setup which consists of a white-box with two windows to the sides, the light from two lamps will be pointed through the them and smoothed by two pieces of thin and almost transparent fabric; when the lights are pointed with the same angle, the shadow will be eliminated by the beam of light of the other. The lights are controlled by a remote dial that allows full control over the brightness, thus allowing to lower the ISO from the camera. This setup is better than the previous, however it still is affected by the conditions of light in the room.



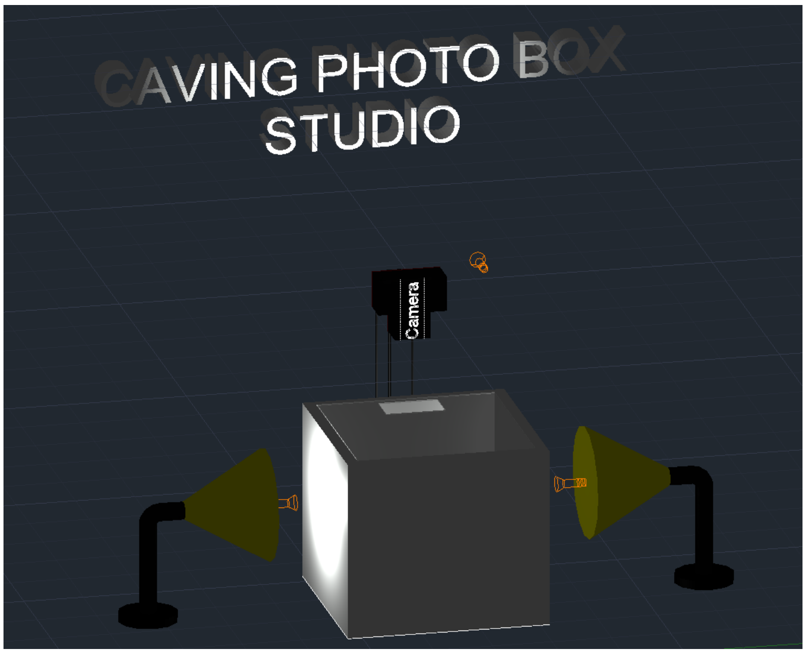
(**a**) Setup 2 (**b**) Setup 3

Figure A.2: Setups

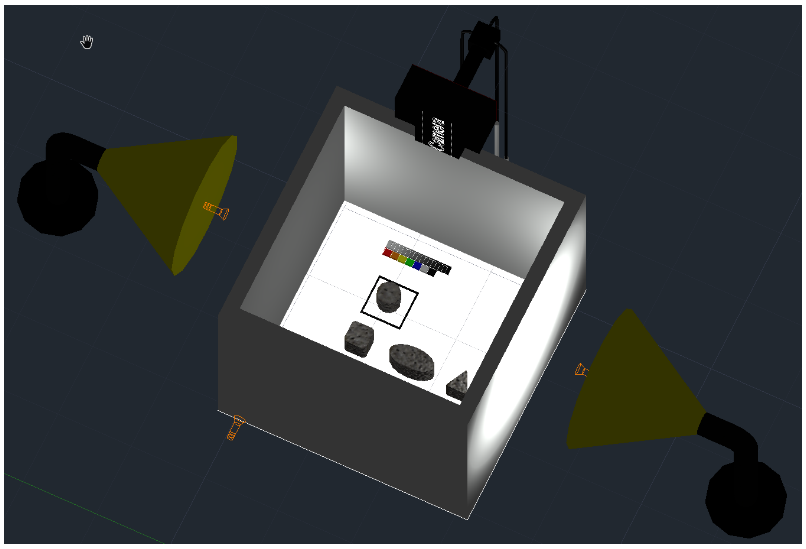
Figure A.2b shows the third and final setup. It is important to notice that there is a difference with the previous setup which is the cover since this last element has a big impact on the quality of the caving pictures.

The box has a cover with a small window for the camera lenses to see the caving inside of the box. This detail will improve the quality of the picture by bouncing the light inside the box, shadows are eliminated and the noise from external lights are minimized. The drawback is that the cover needs to be removed every time a new caving needs to be photograph making the process too repetitive and manual. As a recommendation for futures boxes it would be better to include an access door from the front for quick and easy access.

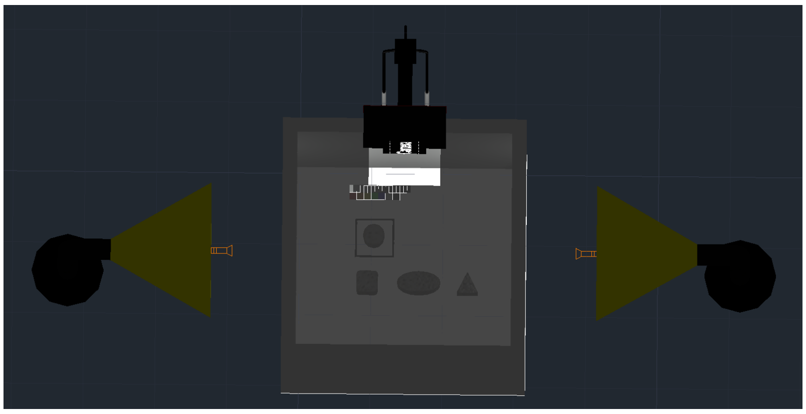
All the white-box blueprints (see Appendix A.2 for full scale blueprints) show the distances between the objects and the angle of the lamps which is fundamental so that the optimal light conditions can be reached for getting a standard dataset. Figure A.3 shows a 3D simulation of the final setup made in AutoCAD. This was done with the objective of having a preview of what the final setup would look like.



(a) Lateral View



(b) Top view without cover



(c) Top view with cover

Figure A.3: Photo-Box 3D Simulation

After trying different setups, optimal light conditions and good quality of pictures were obtained and the final result of white-box design can be seen in Figure A.4. Furthermore, an example of the final design of the template for pictures is observed in Figure A.5 where a color scale was established, each squared box has a dimension of 1 squared centimeter as an object of reference and a also as a reference to the original colors of the caving, the ones in the scale are the primary colors, plus: green, orange, 100% white and 100% black. To the left there is a 16-tone gray scale bar, from (255,255,255) to (0,0,0) or in other words from white to black.

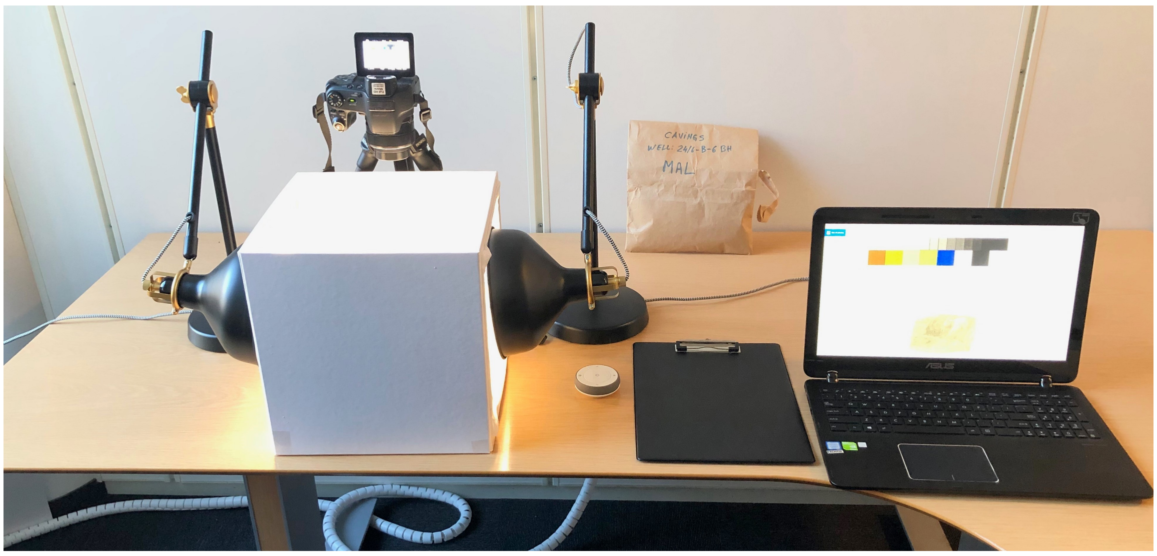


Figure A.4: The photography set-up

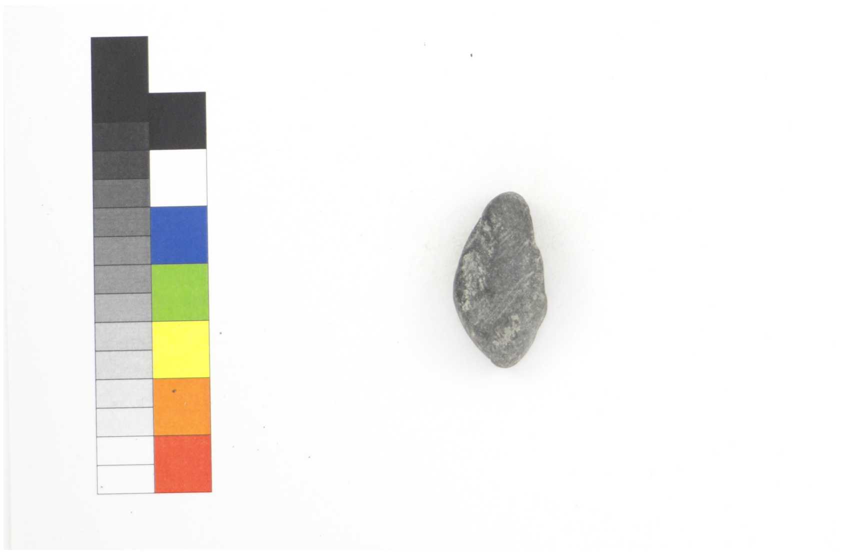


Figure A.5: Template with scale

A.2 Setups

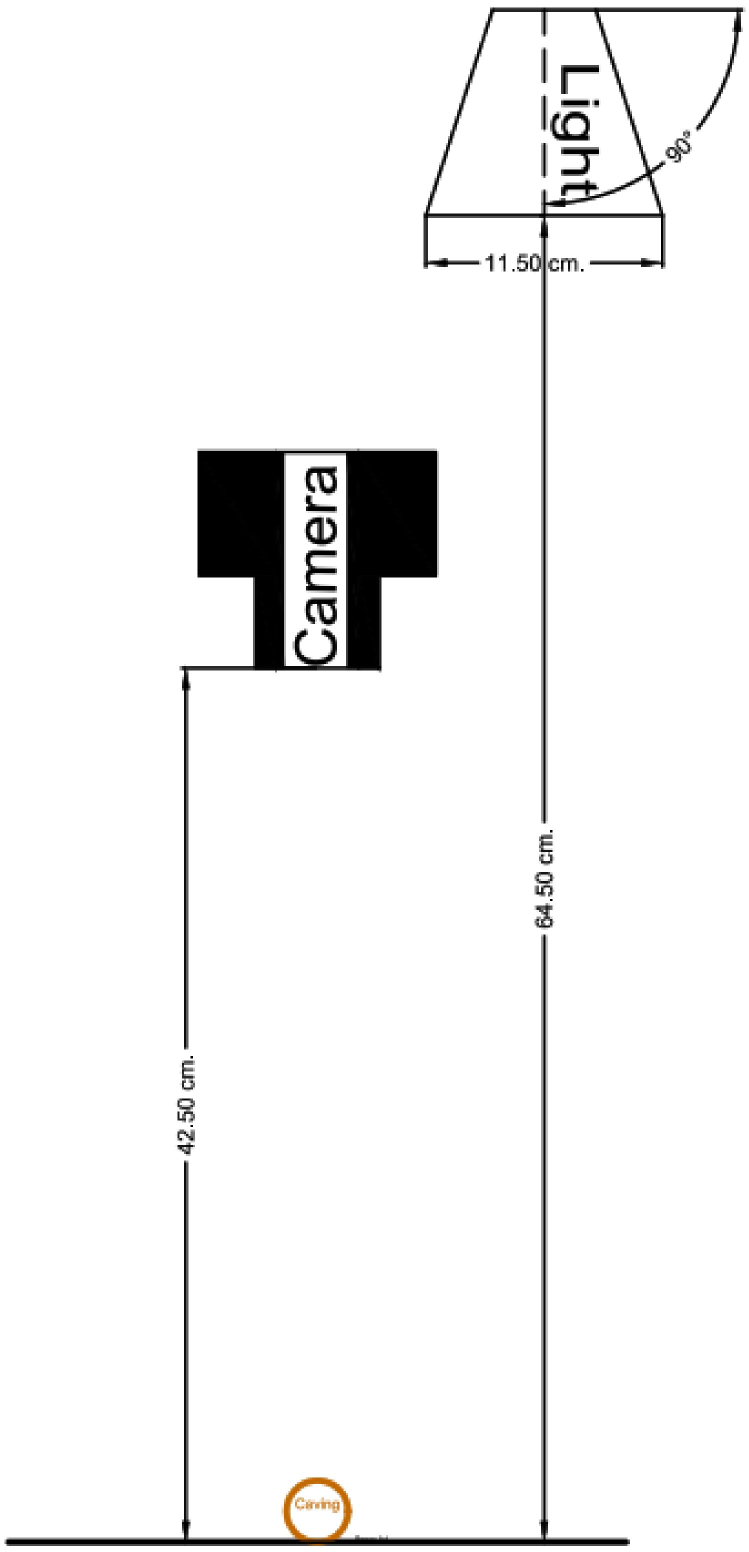


Figure A.6: Setup 1

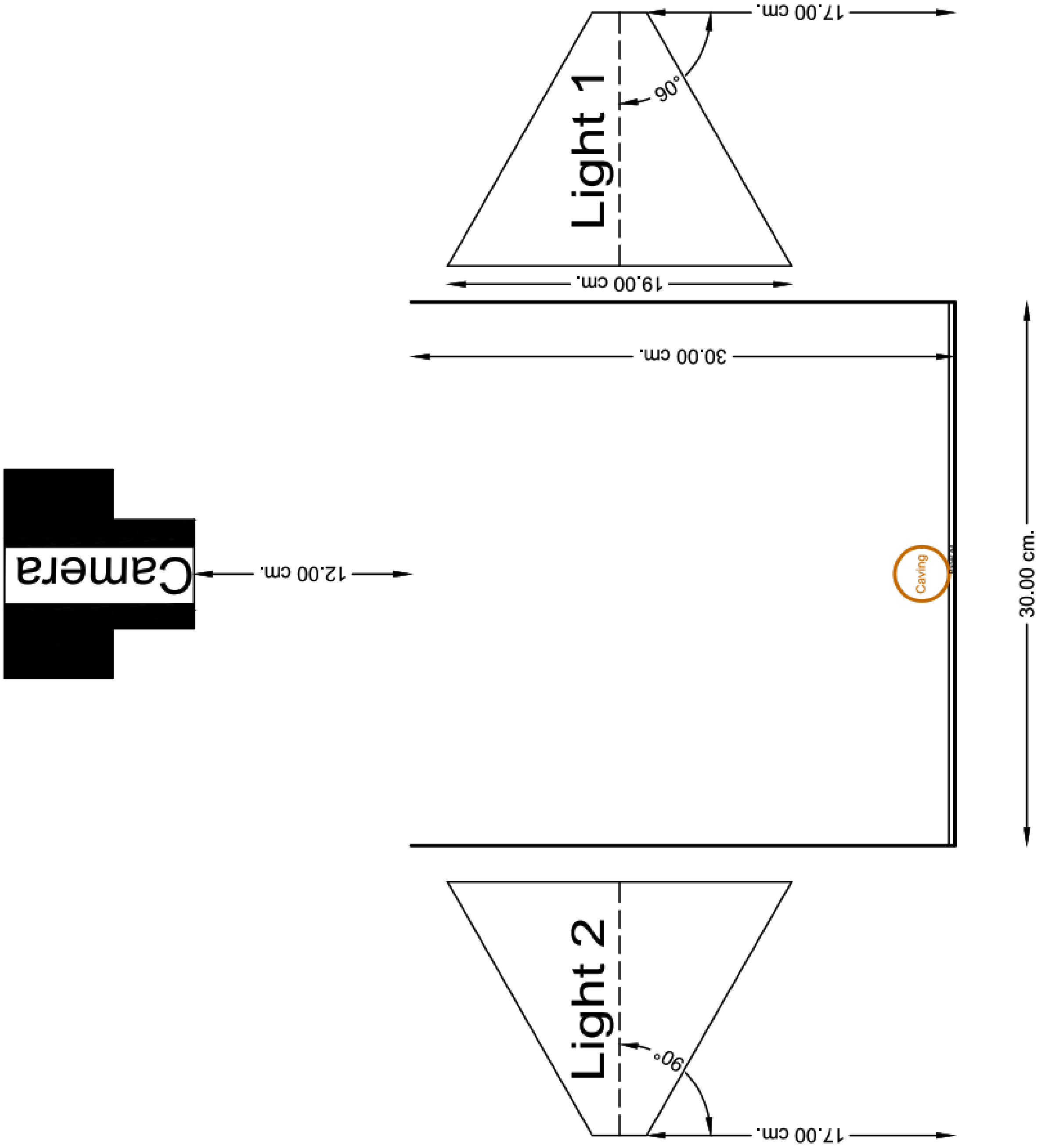


Figure A.7: Setup 2

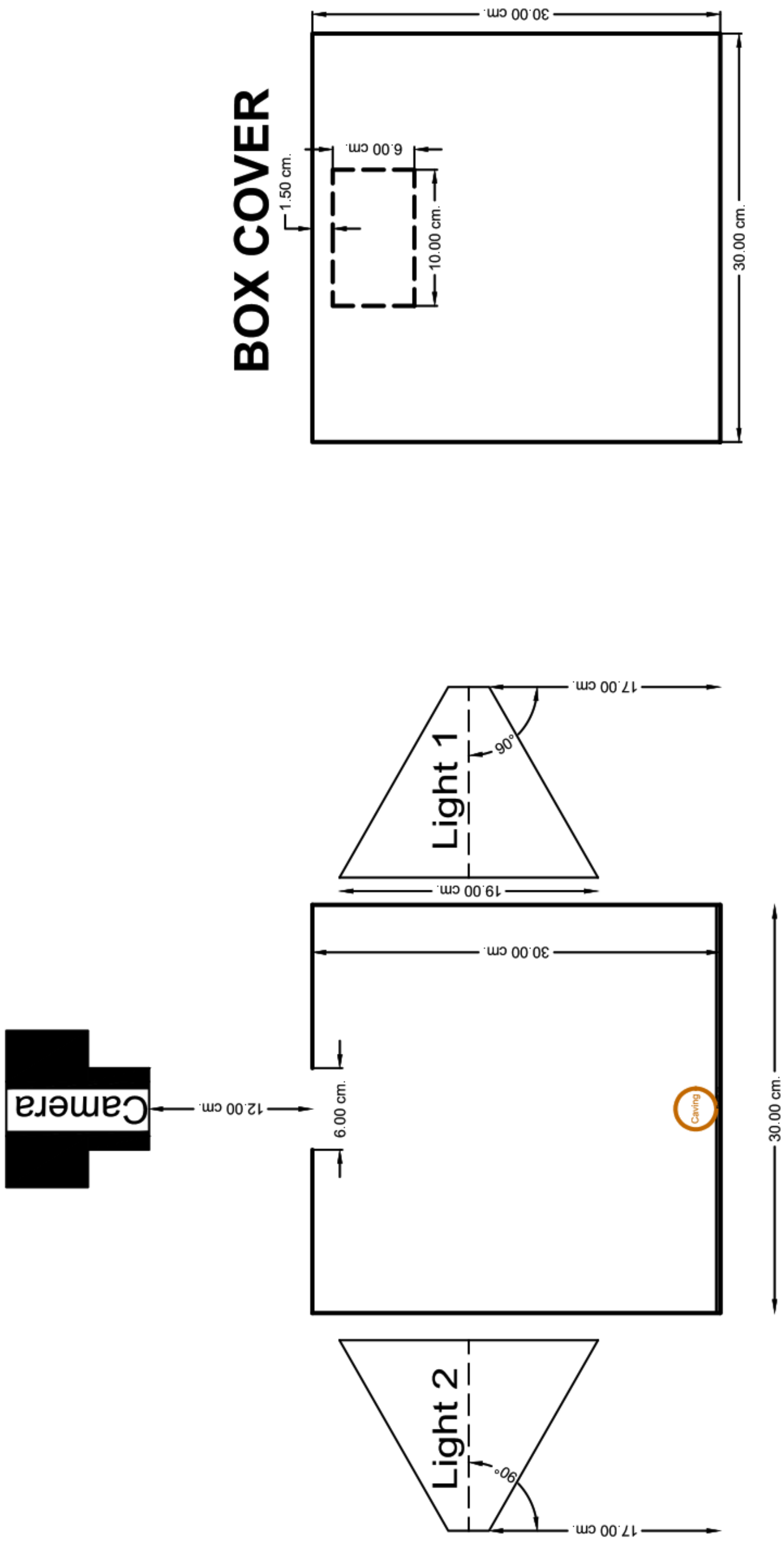


Figure A.8: Setup 3

Appendix B

Python Code

B.1 Installed Packages

Name	Version	Name	Version
_tflow_190_select	0.0.1	isort	4.3.4
_tflow_select	2.1.0	jedi	0.12.1
absl-py	0.4.1	jinja2	2.1
alabaster	0.7.11	jpeg	9b
appdirs	1.4.3	json	0.1.1
asn1crypto	0.24.0	jsonschema	2.6.0
astor	0.7.1	jupyter_client	5.2.3
astroid	2.0.4	jupyter_core	4.4.0
attrs	18.2.0	jupyterthemes	0.20.0
automat	0.7.0	keras	2.2.2
babel	2.6.0	keras-applications	1.0.4
backcall	0.1.0	keras-base	2.2.2
backports	1.0	keras-preprocessing	1.0.2
backports.weakref	1.0.post1	keyring	13.2.1
blas	1.0	kivy	1.10.1
bleach	2.1.3	kivy-garden	0.1.4
ca-certificates	2018.03.07	kiwisolver	1.0.1
certifi	2018.8.24	lazy-object-proxy	1.3.1
cffi	1.11.5	lesscpy	0.13.0
chardet	3.0.4	libgpuarray	0.7.6
cloudpickle	0.7.0	libopencv	3.4.2
colorama	0.3.9	libpng	1.6.34
constantly	15.1.0	libprotobuf	3.5.2
cryptography	1.0.2	libsodium	1.0.16
cryptography-vectors	2.3	libtiff	4.0.9
cudatoolkit	9.0	m2w64-gcc-libgfortran	5.3.0
cudnn	7.3.1	m2w64-gcc-libs	5.3.0
cycler	0.10.0	m2w64-gcc-libs-core	5.3.0
cython	0.28.5	m2w64-gmp	6.1.0
dask-core	1.1.1	m2w64-libwinpthread-git	5.0.0.4634.697f757
decorator	4.3.0	make	3.82
docutils	0.14	mako	1.0.7
entrypoints	0.2.3	markdown	2.6.11
ffmpeg	3.2.4	markupsafe	1.0
freetype	2.8.1	matplotlib	2.2.2
gast	0.2.0	mccabe	0.6.1
git	2.18.0	mistune	0.8.3
glew	2.0.0	mkl	2018.0.3
grpcio	1.12.1	mkl_fft	1.0.4
h5py	2.8.0	mkl_random	1.0.1
hdf5	1.10.2	mock	2.0.0
html5lib	1.0.1	msys2-conda-epoch	20160418
hyperlink	17.1.1	nbconvert	5.3.1
icc_rt	2017.0.4	nbformat	4.4.0
icu	58.2	networkx	2.1
idna	2.7	notebook	5.6.0
imageio	2.3.0	numpy	1.15.0
imagesize	1.0.0	numpy-base	1.15.0
imutils	0.5.2	numpydoc	0.8.0
incremental	17.5.0	olefile	0.45.1
intel-openmp	2018.0.3	opencv	3.4.2
ipykernel	4.8.2	opencv3	3.1.0
ipython	6.5.0	openssl	1.0.2p

ipython_genutils	0.2.0	packaging	17.1
pandas	0.23.4	seaborn	0.9.0
pandoc	2.2.1	selenium	3.14.0
pandocfilters	1.4.2	send2trash	1.5.0
parso	0.3.1	service_identity	17.0.0
path.py	11.0.1	setuptools	40.0.0
patsy	0.5.0	simplegeneric	0.8.1
pbr	5.1.3	singledispatch	3.4.0.3
pickleshare	0.7.4	sip	4.19.8
pillow	5.2.0	six	1.11.0
pip	10.0.1	smpeg2	2.0.0
ply	3.11	snowballstemmer	1.2.1
prometheus_client	0.3.1	sphinx	1.7.6
prompt_toolkit	1.0.15	sphinxcontrib	1.0
protobuf	3.5.2	sphinxcontrib-websupport	1.1.0
psutil	5.4.6	spyder	3.3.0
pyasn1	0.1.9	spyder-kernels	0.2.4
pyasn1-modules	0.0.8	sqlite	3.24.0
pycodestyle	2.4.0	statsmodels	0.9.0
pycparser	2.18	tensorboard	1.9.0
pyflakes	2.0.0	tensorflow	1.9.0
pygments	2.2.0	tensorflow-base	1.9.0
pygpu	0.7.6	tensorflow-gpu	1.9.0
pylint	2.1.1	termcolor	1.1.0
py-opencv	3.4.2	terminado	0.8.1
pyopenssl	18.0.0	testpath	0.3.1
pyparsing	2.2.0	theano	1.0.2
pyqt	5.9.2	tk	8.6.7
pyreadline	2.1	toolz	0.9.0
pysocks	1.6.8	tornado	5.1
python	3.5.5	traitlets	4.3.2
python-dateutil	2.7.3	twisted	17.5.0
pytorch	0.4.1	typed-ast	1.1.0
pytz	2018.5	typing	3.6.4
pywavelets	1.0.0	urllib3	1.23
pywin32	223	vc	14.1
pywin32-ctypes	0.2.0	vs2015_runtime	15.5.2
pywinpty	0.5.4	wcwidth	0.1.7
pyyaml	3.13	webencodings	0.5.1
pyzmq	17.1.0	werkzeug	0.14.1
qt	5.9.6	wheel	0.31.1
qtawesome	0.4.4	win_inet_pton	1.0.1
qtconsole	4.3.1	win_unicode_console	0.5
qtpy	1.4.2	wincertstore	0.2
qutil	3.2.1	winpty	0.4.3
requests	2.19.1	wrapt	1.10.11
requests_download	0.1.2	xz	5.2.4
rope	0.10.7	yaml	0.1.7
scikit-image	0.14.0	zeromq	4.2.5
scikit-learn	0.20.0	zlib	1.2.11
scipy	1.1.0	zope	1.0
sdl2	2.0.8	zope.interface	4.5.0
sdl2_image	2.0.2	zstd	1.3.7
sdl2_mixer	2.0.1		
sdl2_ttf	2.0.14		

B.2 Code

```python
cavings = ['ELONGATED', 'BLOCKY']

limit_B = 1.44 # length/width for Blocky
limit_E = 2.48 # length/width for Elongated

if cavings[c]=='BLOCKY':
    if np.float(size_cav[0,2]) >= limit_B:
        # For a larger sample, it will be assigned RECTANGULAR
        size_cav[0,3] = 'RECT-BLOCKY'
        label[i,1] = 'RECT-BLOCKY'
    else:
        # For a shorter sample, it will be assigned SQUARED
        size_cav[0,3] = 'SQR-BLOCKY'
        label[i,1] = 'SQR-BLOCKY'

elif cavings[c]=='ELONGATED':
    if np.float(size_cav[0,2]) >= limit_E:
        # For a larger sample, it will be assigned LARGE
        size_cav[0,3] = 'LARGE-ELONG'
        label[i,1] = 'LARGE-ELONG'

    else:
        # For a shorter sample, it will be assigned SHORT
        size_cav[0,3] = 'SHORT-ELONG'
        label[i,1] = 'SHORT-ELONG'

```

Listing B.1: Splitting of dataset (only the relevant part of the code is presented)

```python
def rotate_bound(image, angle):
    # grab the dimensions of the image and then determine the
    # center
    (h, w) = image.shape[:2]
    (cX, cY) = (w // 2, h // 2)

    # grab the rotation matrix (applying the negative of the
    # angle to rotate clockwise), then grab the sine and cosine
    # (i.e., the rotation components of the matrix)
    M = cv2.getRotationMatrix2D((cX, cY), -angle, 1.0)
    cos = np.abs(M[0, 0])
    sin = np.abs(M[0, 1])

    # compute the new bounding dimensions of the image
    nW = int((h * sin) + (w * cos))
    nH = int((h * cos) + (w * sin))

    # adjust the rotation matrix to take into account translation
    M[0, 2] += (nW / 2) - cX
    M[1, 2] += (nH / 2) - cY

    # perform the actual rotation and return the image
    return cv2.warpAffine(image, M, (nW, nH))

incr = 45
(h, w) = image.shape[:2]
for angle in np.arange(0, 360, incr):

    rotated = imutils.rotate_bound(image, angle)

    (nH, nW) = rotated.shape[:2]
    (cX, cY) = (nW // 2, nH // 2)
    crop_rotated = rotated[cY-int((h/math.sqrt(2))/2):cY+int((h/math.sqrt(2))/2), cX-int((
h/math.sqrt(2))/2):cX+int((h/math.sqrt(2))/2)]
    cv2.imwrite("1-" + str(angle) + ".jpg", crop_rotated)

```

Listing B.2: Rotation of image (only the relevant part of the code is presented)

```python
x1 = 1630
y1 = 300
dis = 2020
pix = 28
size = pix,pix
feat = pix*pix

rootdir = r'''C:\CavingPics'''

for subdir, dirs, files in os.walk(rootdir):
for file in files:
    image= Image.open(os.path.join(subdir, file))
    image = image.crop((x1, y1, x1 + dis, y1 + dis))
    image = image.convert('L')
    image.thumbnail(size)
    vectorizedImage = list(image.getdata()) #get pixel values
    finalImage = [ (255-x)*1.0/255.0 for x in vectorizedImage]
    finalImage = [ 0 if x <0.1 else x for x in finalImage]
    finalImage = np.array(finalImage).reshape(1,feat)
    i+=1
    if i ==1:
        data = finalImage
        label = np.array([[os.path.basename(os.path.dirname(os.path.join(subdir, file)))
]])

    else:
        data = np.concatenate((data,finalImage),axis=0)
        label = np.concatenate((label,np.array([[os.path.basename(os.path.dirname(os.path.
join(subdir, file)))]])),axis=0)

```

Listing B.3: Dataset pre-processing (only the relevant part of the code is presented)

```python
dataset = np.hstack((data,label_num))
np.random.seed(100)
np.random.shuffle(dataset)

X = dataset[:,0:feat].astype(np.float64)
y = dataset[:,feat].astype(np.int)
n_samples = len(y)
n_features = len(X[0])

split_point = int(n_samples * 0.8)

# using 0.8 for training
# and 0.2 for testing

y_train = y[:split_point]
X_train = X[:split_point]

y_test = y[split_point:]
X_test = X[split_point:]

clf1 = LogisticRegression(random_state=1, solver='newton-cg', multi_class='multinomial')
clf2 = RandomForestClassifier(random_state=1, max_depth=5, n_estimators=100, max_features
=1)
clf3 = GaussianNB()
clf4 = SVC(gamma='auto')
clf5 = KNeighborsClassifier(n_neighbors=(k-1), weights='uniform', algorithm='auto',
leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None)
clf6 = SVC(kernel="linear", C=0.025)
clf7 = DecisionTreeClassifier(max_depth=7)
clf8 = AdaBoostClassifier()
clf9 = QDA()

labels = ['Logistic Regression','Random Forest', 'Naive Bayes',                  'RBF SVM',
  'Nearest Neighbors', 'Linear SVM', 'Decision Tree',               'AdaBoost', 'QDA']

for clf, lab in zip[clf1, clf2, clf3, clf4, clf5, clf6, clf7, clf8,
  clf9],labels):
    # Learning Phase
    clf.fit(X_train, y_train)

    # Accuracy
    score = clf.score(X_train, y_train)

    # Predict Test Set
    y_pred = clf.predict(X_test)

```

Listing B.4: Supervised Learning (only the relevant part of the code is presented)

```
mode = "commit"
#

if mode == "edit" :
    nr_samples = 574

if mode == "commit" :
    nr_samples = 12942

y_train=y[:nr_samples]
X_train=X[:nr_samples]
start_ix_val = nr_samples
end_ix_val = nr_samples + int(nr_samples/3)
y_val=y[start_ix_val:end_ix_val]
X_val=X[start_ix_val:end_ix_val]

#Perceptron
from sklearn.linear_model import Perceptron
clf_Perceptron = Perceptron(random_state=0)
param_grid = { 'penalty': ['l1'], 'tol': [0.1] }
GridCV_Perceptron = GridSearchCV(clf_Perceptron, param_grid, verbose=1, cv=5)
GridCV_Perceptron.fit(X_train,y_train)
score_grid_Perceptron = get_best_score(GridCV_Perceptron)
pred_val_perc = GridCV_Perceptron.predict(X_val)

#Logistic Regression
from sklearn.linear_model import LogisticRegression
clf_LR = LogisticRegression(random_state=0)
param_grid = {'C': [0.014,0.012], 'multi_class': ['multinomial'],
              'penalty': ['l1'],'solver': ['saga'], 'tol': [0.1] }
GridCV_LR = GridSearchCV(clf_LR, param_grid, verbose=1, cv=5)
GridCV_LR.fit(X_train,y_train)
score_grid_LR = get_best_score(GridCV_LR)
pred_val_lr = GridCV_LR.predict(X_val)

#KNN
from sklearn.neighbors import KNeighborsClassifier
clf_knn = KNeighborsClassifier(n_neighbors=10)
clf_knn.fit(X_train,y_train)
pred_val_knn = clf_knn.predict(X_val)

#Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
clf_RF = RandomForestClassifier(random_state=0)
param_grid = {'max_depth': [15], 'max_features': [100],
              'min_samples_split': [5],'n_estimators' : [50] }
GridCV_RF = GridSearchCV(clf_RF, param_grid, verbose=1, cv=5)
GridCV_RF.fit(X_train,y_train)
score_grid_RF = get_best_score(GridCV_RF)
pred_val_rf = GridCV_RF.predict(X_val)

#Multi Layer Perceptron
from sklearn.neural_network import MLPClassifier

clf_mlp = MLPClassifier(activation = "logistic", hidden_layer_sizes=(200,), random_state
=0)
param_grid = { 'batch_size' : [batchsize] , 'max_iter': [400], 'alpha': [1e-4],
               'solver': ['sgd'], 'learning_rate_init': [0.05,0.06],'tol': [1e-4] }

GridCV_MLP = GridSearchCV(clf_mlp, param_grid, verbose=1, cv=3)
GridCV_MLP.fit(X_train,y_train)
score_grid_MLP = get_best_score(GridCV_MLP)
```

```
    pred_val_mlp = GridCV_MLP.predict(X_val)

    #Keras: only input and output layer
    def dense_model_0():
        model = Sequential()
        model.add(Dense(10, input_dim=784, activation='softmax'))
        model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
        return model

    model_dense_0 = dense_model_0()
    model_dense_0.fit(X_train, y_train, epochs=50, batch_size=batchsize)
    pred_val_dense0 = model_dense_0.predict_classes(X_val)

    #Keras: 1 hidden layer
    def dense_model_1():
        model = Sequential()
        model.add(Dense(100, input_dim=784, activation='relu'))
        model.add(Dense(10, activation='softmax'))
        model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
        return model
    model_dense_1 = dense_model_1()
    history_dense_1 = model_dense_1.fit(X_train, y_train,
      validation_data=(X_val,y_val_10),
                              epochs=50, batch_size=batchsize)
    pred_val_dense1 = model_dense_1.predict_classes(X_val)

    #Keras: 2 hidden layers
    def dense_model_2():
        model = Sequential()
        model.add(Dense(100, input_dim=784, activation='relu'))
        model.add(Dense(200, activation='relu'))
        model.add(Dense(10, activation='softmax'))
        model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
        return model
    model_dense_2 = dense_model_2()
    history_dense_2 = model_dense_2.fit(X_train, y_train,
    validation_data=(X_val,y_val_10),
                epochs=50, batch_size=batchsize)
    pred_val_dense2 = model_dense_2.predict_classes(X_val)

    #Keras: 3 hidden layers
    def dense_model_3():

        model = Sequential()
        model.add(Dense(100, activation='relu', input_dim=784))
        model.add(Dense(200, activation='relu'))
        model.add(Dense(100, activation='relu'))
        model.add(Dense(10, activation='softmax'))

        model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
        return model

    model_dense_3 = dense_model_3()
    history_dense_3 = model_dense_3.fit(X_train, y_train, validation_data=(
                                          X_val,y_val_10), epochs=50,
                                        batch_size=batchsize)

    pred_val_dense3 = model_dense_3.predict_classes(X_val)


    #CNN 1
    X_train = X_train.values.reshape(X_train.shape[0],img_rows,img_cols,1)
    X_val = X_val.values.reshape(X_val.shape[0],img_rows,img_cols,1)

    input_shape = (img_rows, img_cols, 1)
    batchsize = 128
    epochs = 12

    activation = 'relu'
    adadelta = Adadelta()
    loss = categorical_crossentropy

    def cnn_model_1(activation):
        model = Sequential()
        model.add(Conv2D(32, kernel_size=(3, 3), activation=activation,
          input_shape=input_shape))
        model.add(Conv2D(64, (3, 3), activation=activation))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.25))
        model.add(Flatten())
        model.add(Dense(128, activation=activation))
        model.add(Dropout(0.5))
        model.add(Dense(10, activation='softmax'))
        model.compile(loss=loss, optimizer=adadelta, metrics=['accuracy'])
        return model
    model_cnn_1 = cnn_model_1(activation)
    history_cnn_1 = model_cnn_1.fit(X_train, y_train, validation_data=(X_val,y_val_10),epochs=
    epochs,batch_size=batchsize, verbose=1)
    pred_val_cnn1 = model_cnn_1.predict_classes(X_val)

    #CNN 2
    batch_size=90
    epochs=30
    def cnn_model_2(activation):

        model = Sequential()
        model.add(Conv2D(32, (3, 3), padding = 'Same', activation="relu", input_shape=
    input_shape ))
        model.add(MaxPooling2D(pool_size = (2, 2)))
        model.add(Conv2D(32, (3, 3), activation="relu"))
        model.add(MaxPooling2D(pool_size = (2, 2)))
        model.add(Flatten())
        model.add(Dense(256, activation=activation))
        model.add(Dense(10, activation='softmax'))
        model.compile(optimizer = adadelta, loss = loss, metrics =              ['accuracy'])
    # adam
        return model
    model_cnn_2 = cnn_model_2('relu')
    history_cnn_2 = model_cnn_2.fit(X_train, y_train, validation_data=(X_val,y_val_10),epochs=
    epochs,batch_size=batchsize, verbose=1)
    pred_val_cnn2 = model_cnn_2.predict_classes(X_val)

```

Listing B.5: Unsupervised Learning (only the relevant part of the code is presented)

B.3 Shape - MNIST Approach

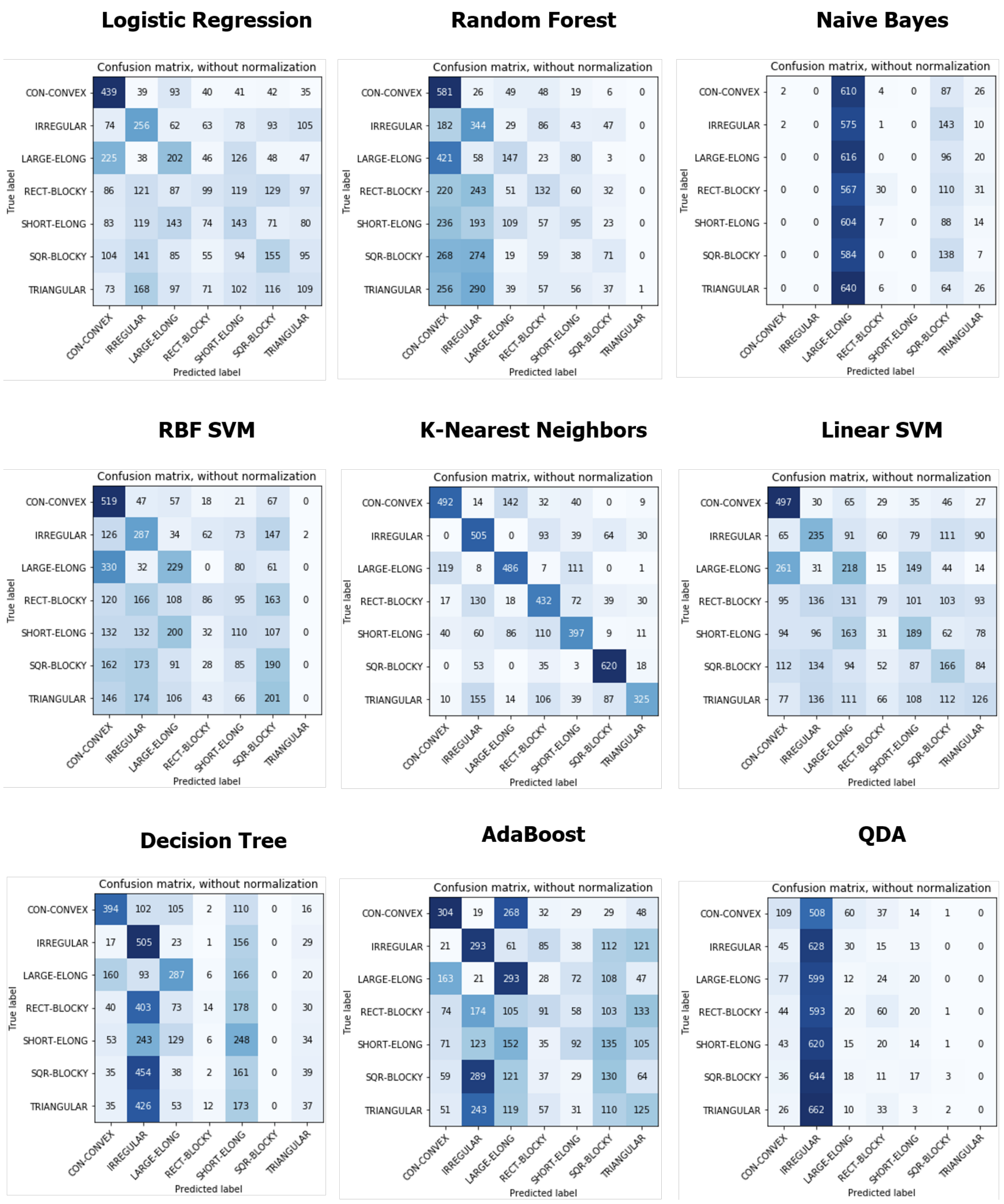


Figure B.1: Confusion Matrix for Supervised Learning - Rotated Dataset

B.4 Size

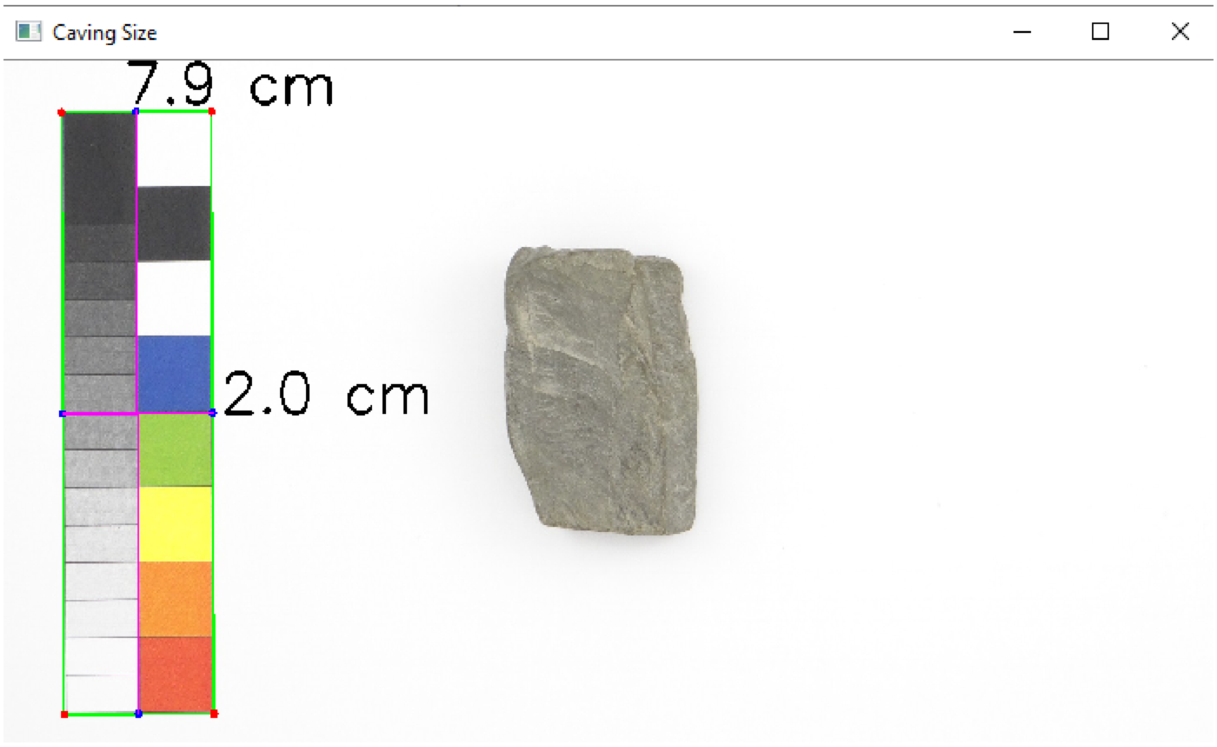


Figure B.2: Measurement of Reference Object

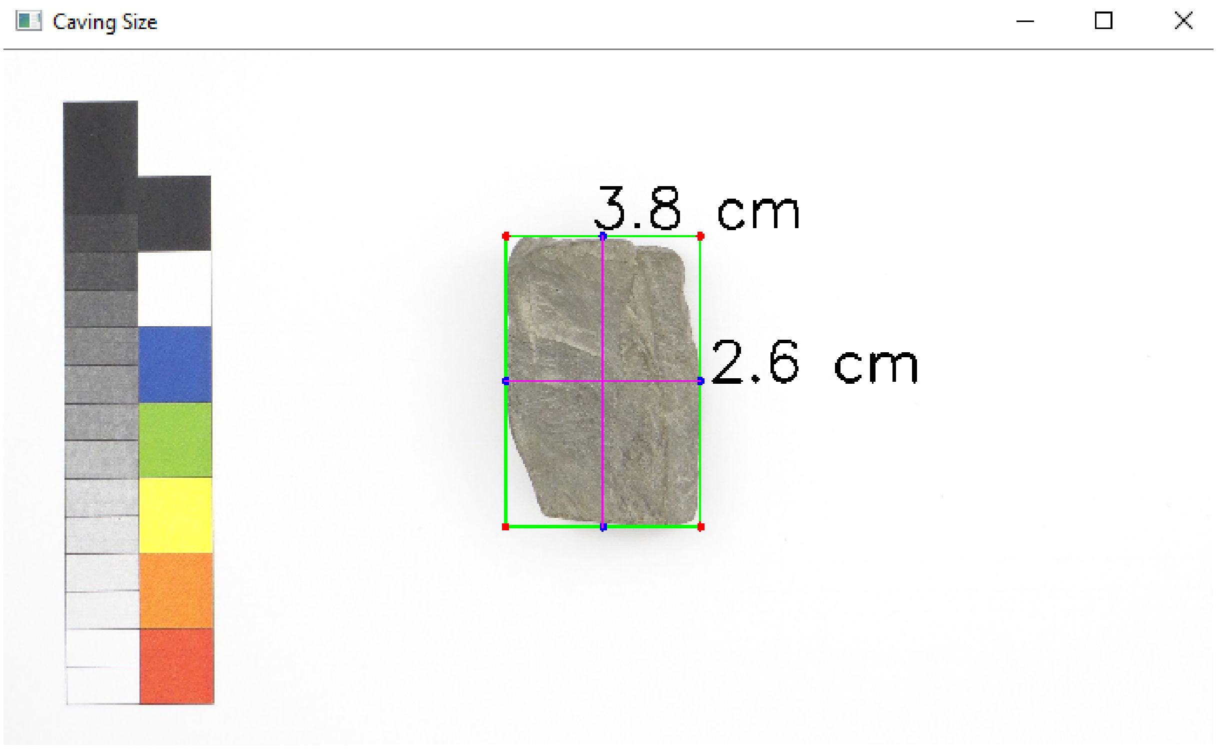


Figure B.3: Measurement of Caving

B.5 Colour

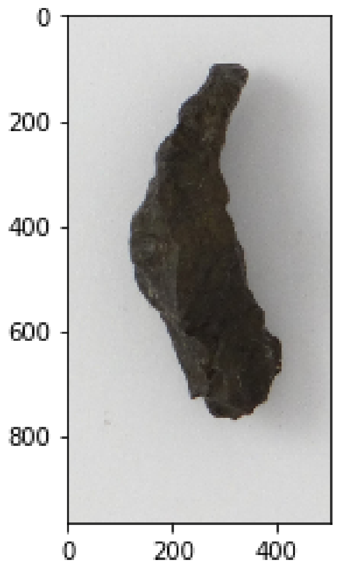


Figure B.4: Original Image

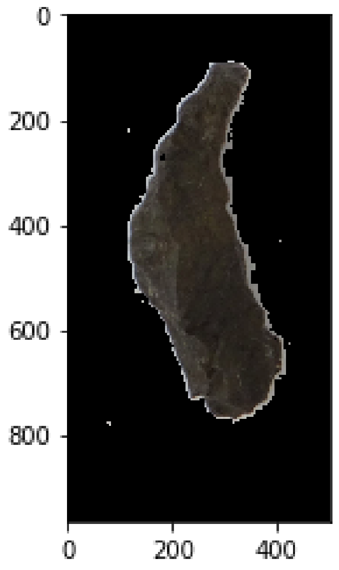


Figure B.5: Mask Applied to an Elongated Caving

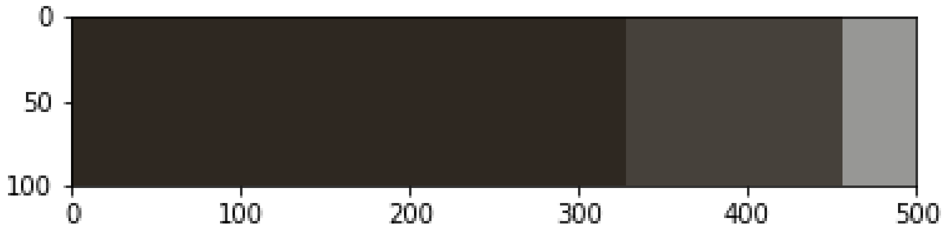
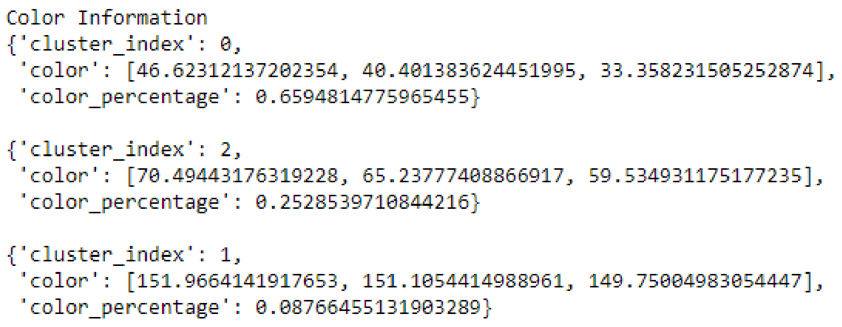


Figure B.6: Colour Bar



```
Color Information
{'cluster_index': 0,
 'color': [46.62312137202354, 40.401383624451995, 33.358231505252874],
 'color_percentage': 0.6594814775965455}

{'cluster_index': 2,
 'color': [70.49443176319228, 65.23777408866917, 59.534931175177235],
 'color_percentage': 0.2528539710844216}

{'cluster_index': 1,
 'color': [151.9664141917653, 151.1054414988961, 149.75004983054447],
 'color_percentage': 0.08766455131903289}
```

Figure B.7: Colour Information

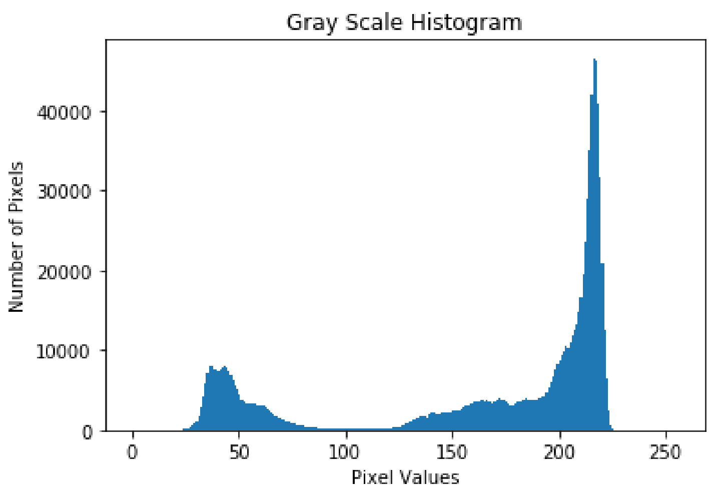


Figure B.8: Grasycale Histogram

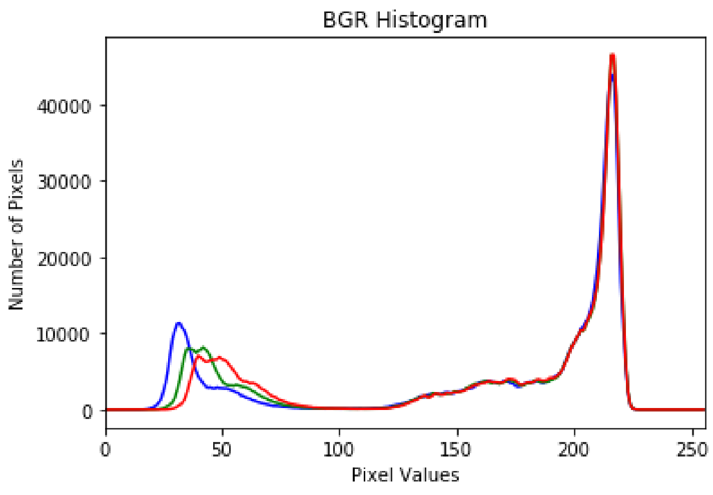


Figure B.9: BGR Histogram

B.6 Edge Definition

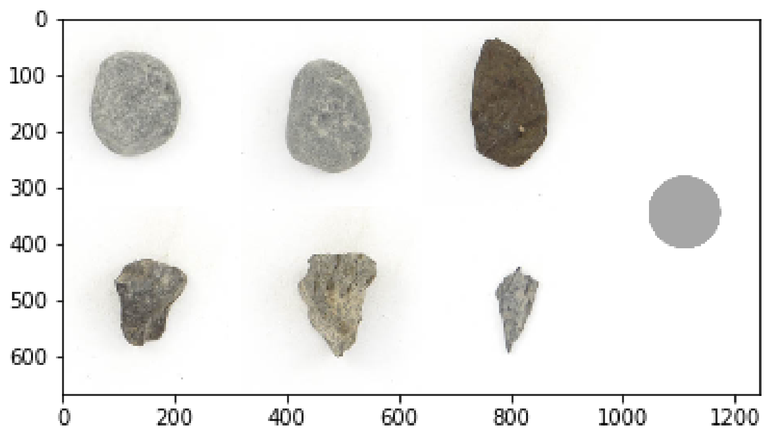


Figure B.10: Original Image

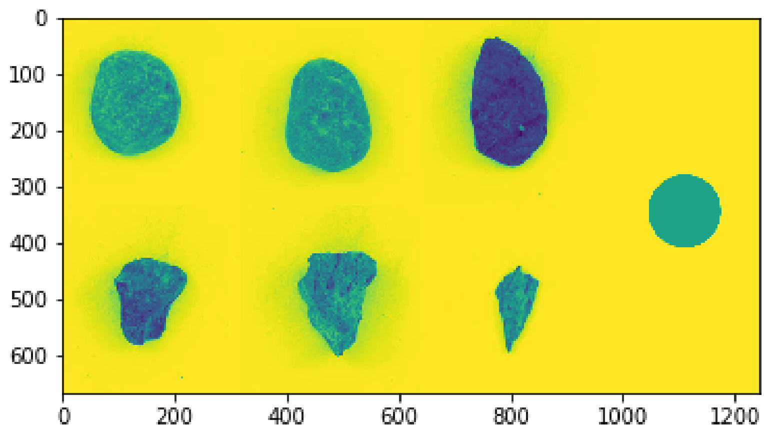


Figure B.11: Grayscale Image

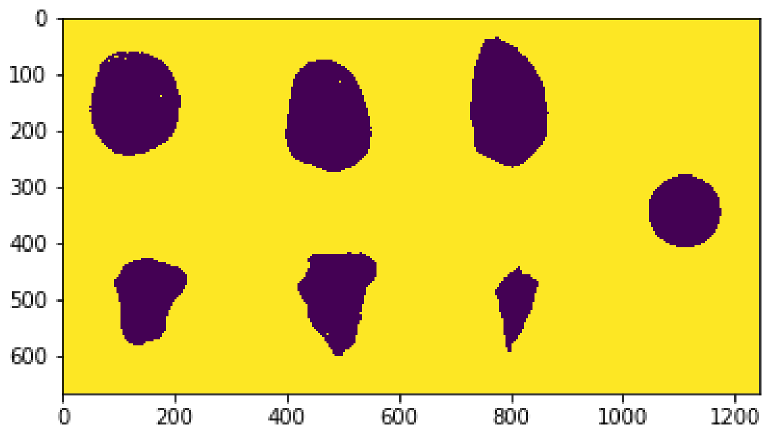


Figure B.12: Binary Image with Noise

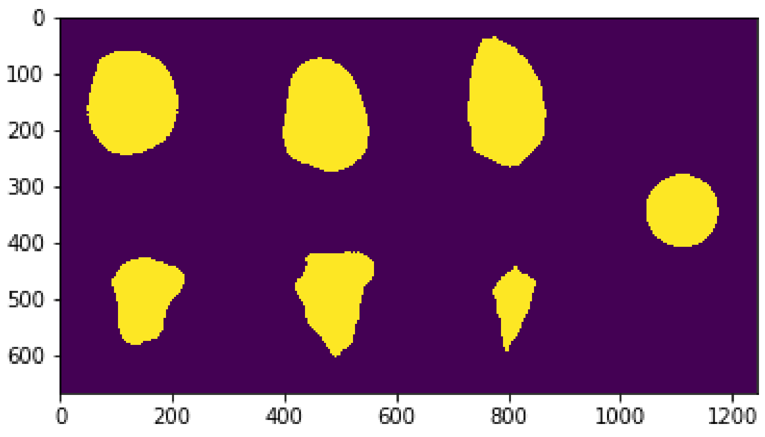


Figure B.13: Inverted Binary Image without Noise

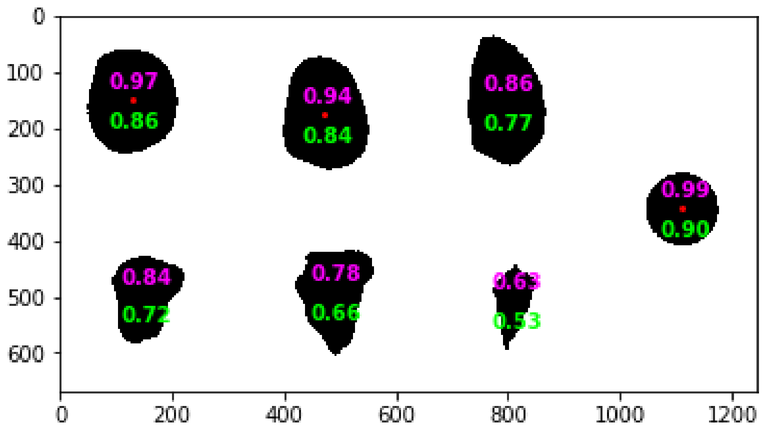


Figure B.14: Roundness Results

B.6.1 Perimeter Analysis

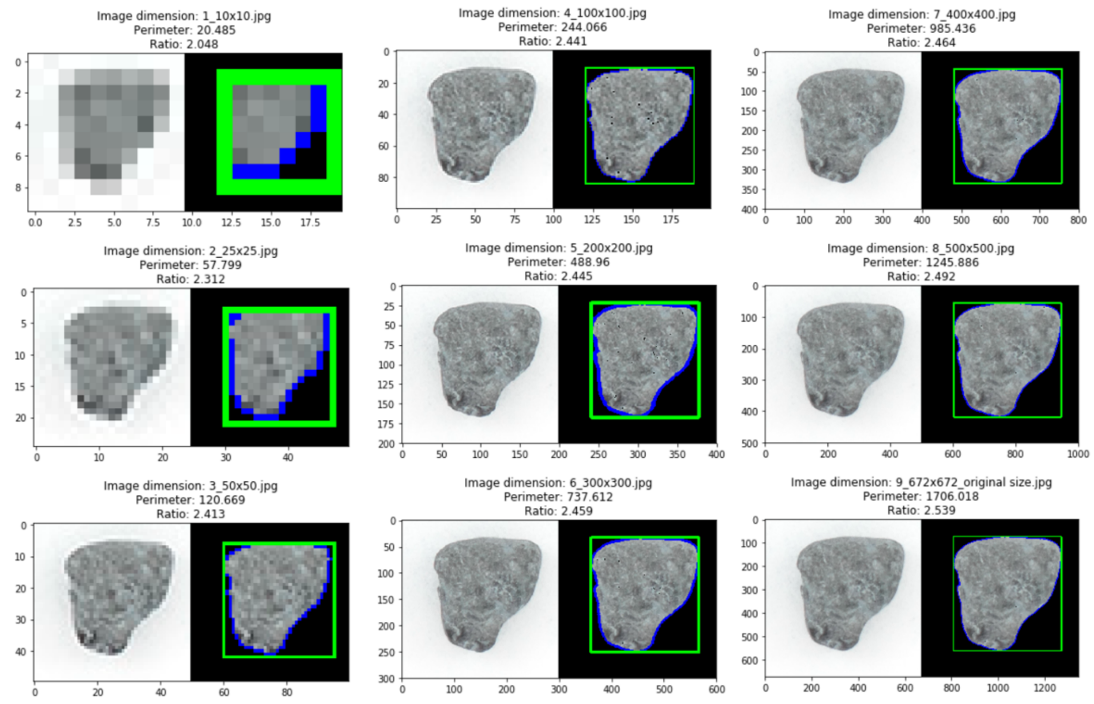


Figure B.15: Perimeter Analysis on Different Resolutions

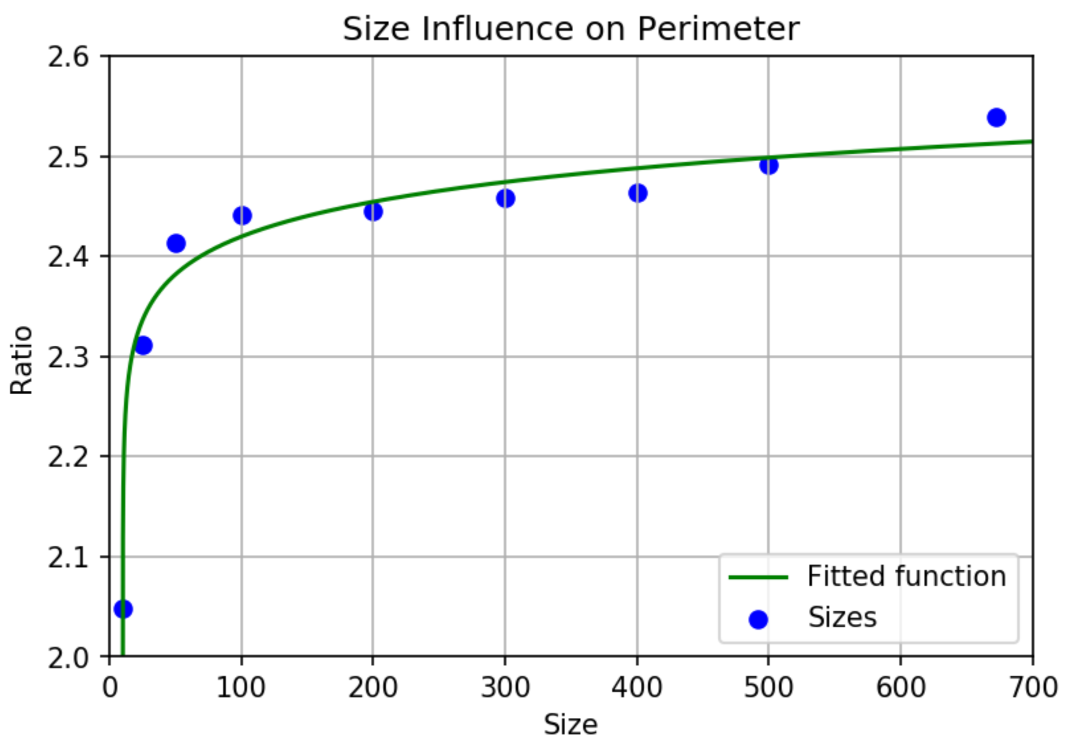


Figure B.16: Resolution Influence on Perimeter

B.6.2 Sensitivity Analysis

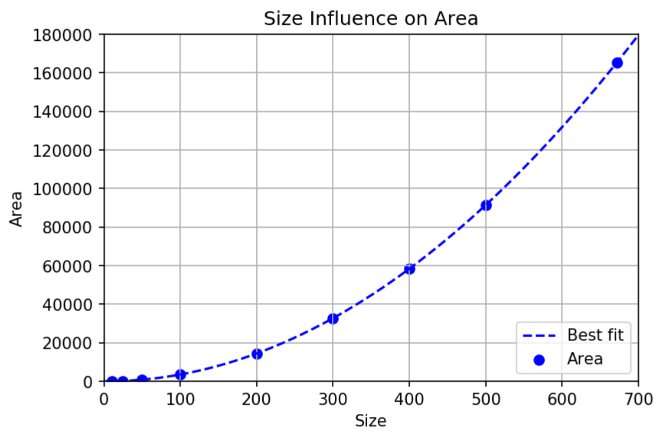


Figure B.17: Size Influence on Area

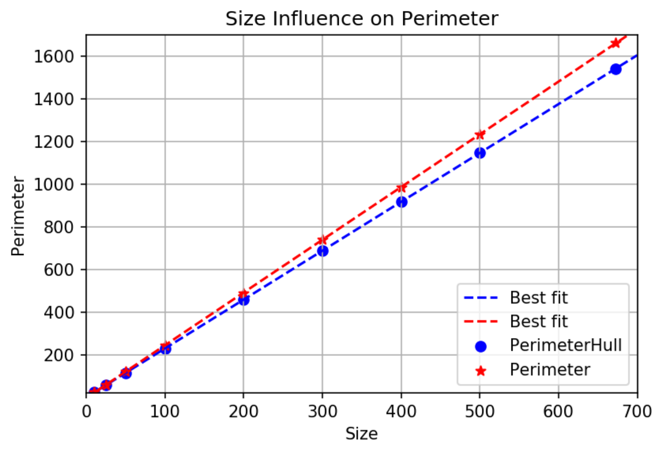


Figure B.18: Size Influence on Perimeter

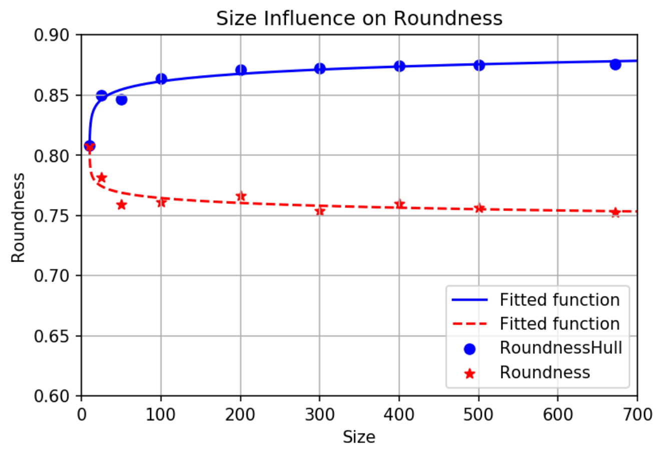


Figure B.19: Size Influence on Roundness

B.6.3 Caving Distribution

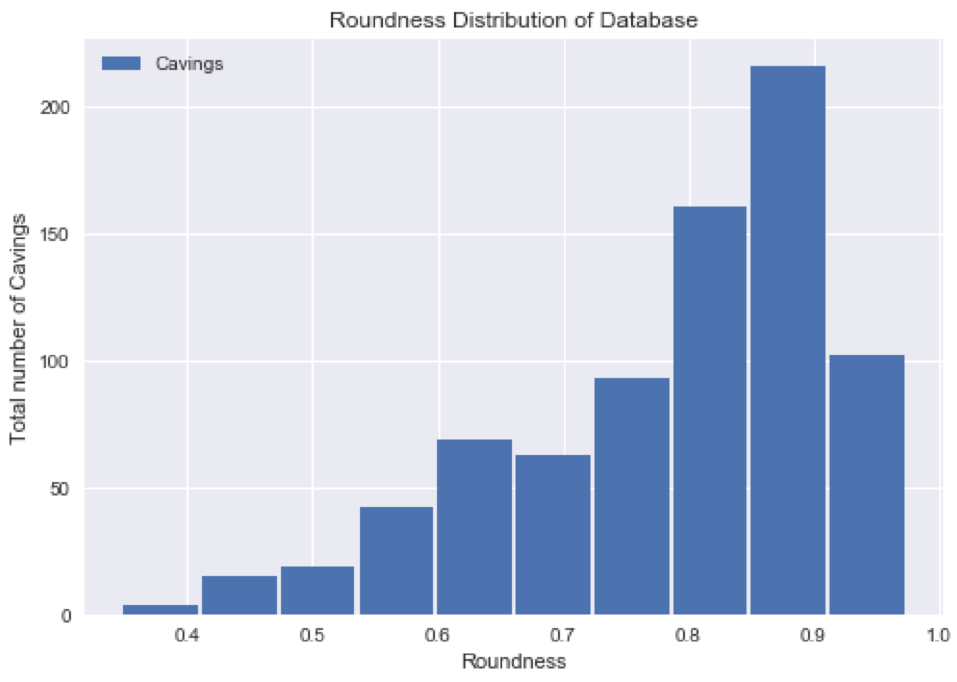


Figure B.20: Roundness Values Distribution of Data Base

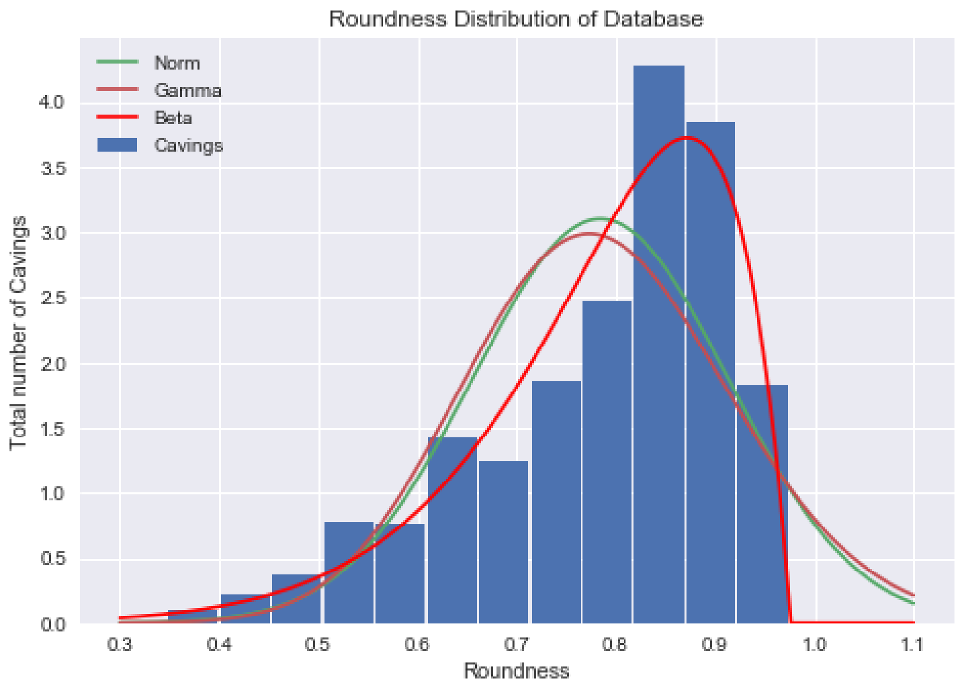
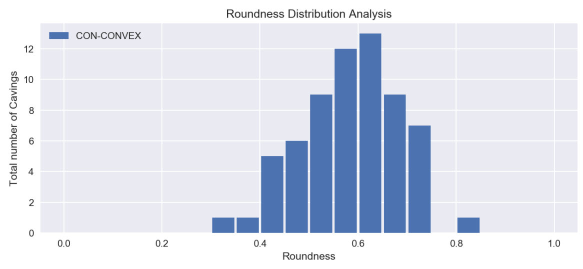
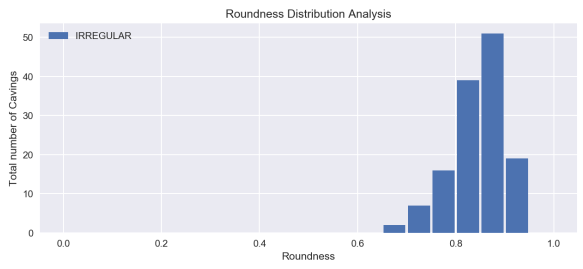


Figure B.21: Fitting Functions

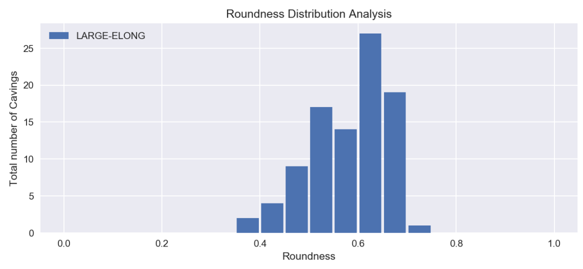
B.6.4 Caving Histograms per Shape



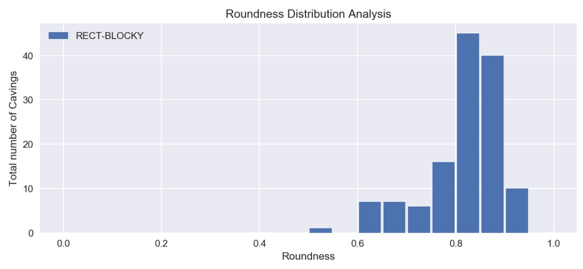
(a) Con-Convex



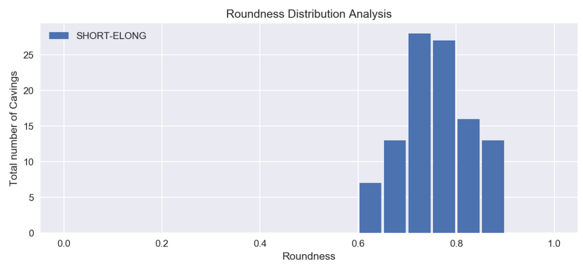
(b) Irregular



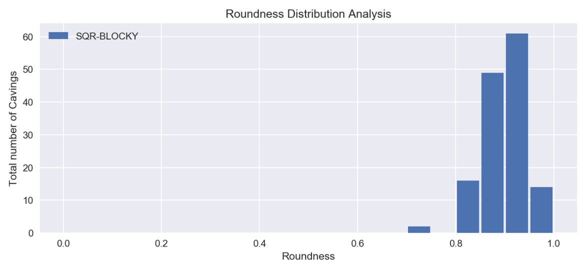
(c) Large-Elong



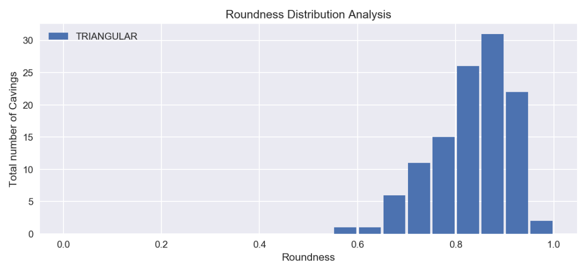
(d) Rect-Blocky



(e) Short-Elong



(f) Sqr-Blocky



(g) Triangular

Figure B.22: Histograms Per Shape

B.6.5 Histogram Analysis

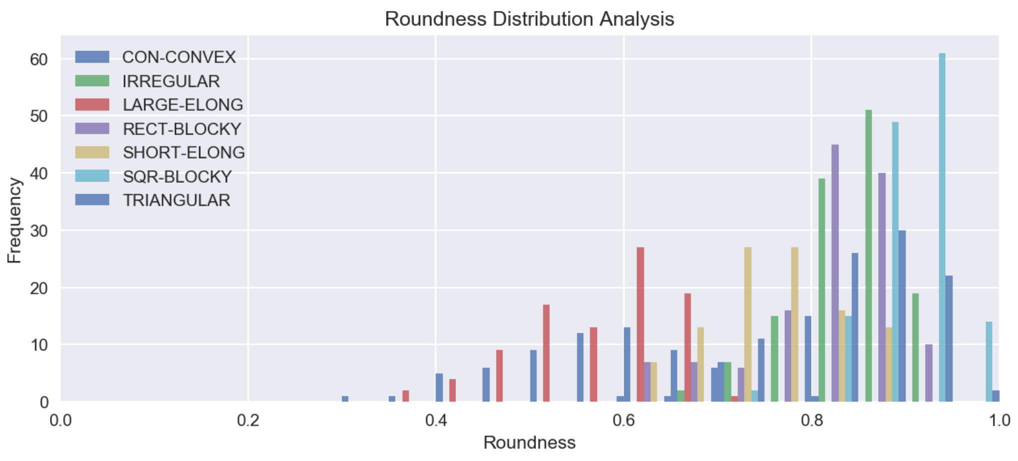


Figure B.23: Histogram per Shapes

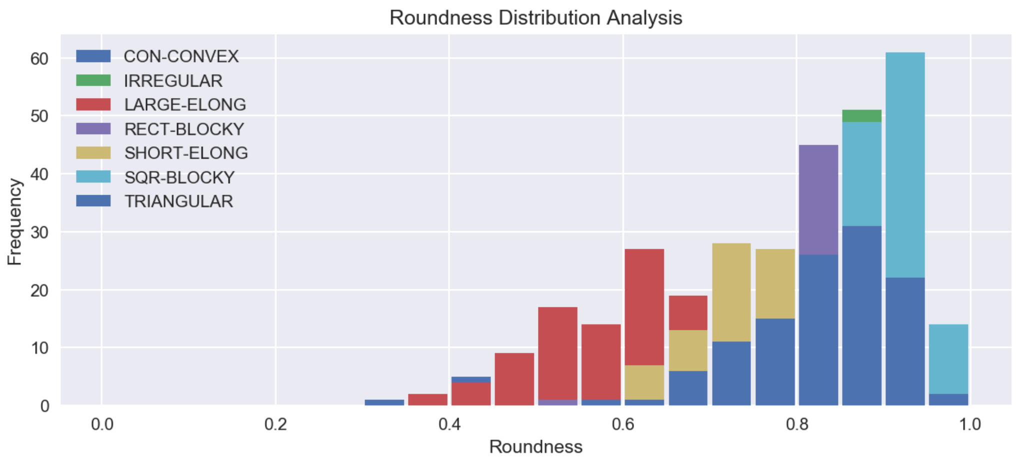


Figure B.24: Stack Histogram

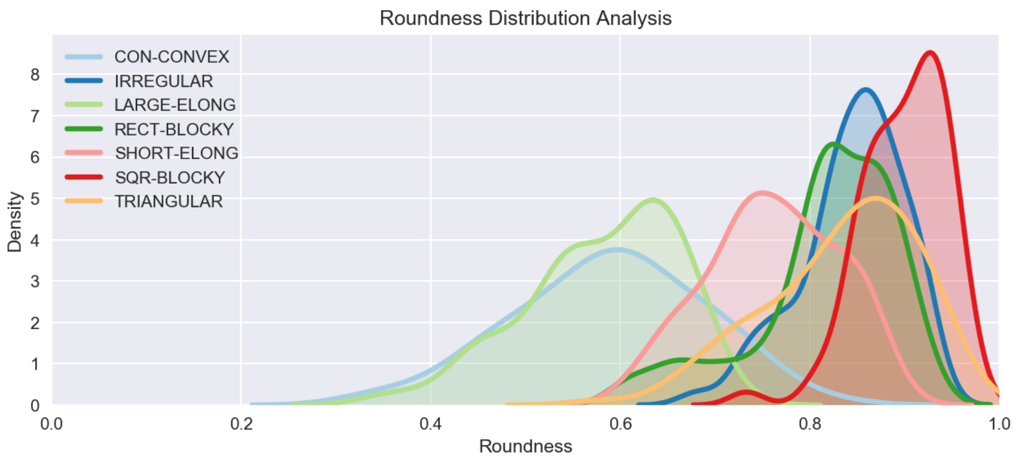


Figure B.25: Kernel Density Analysis

Appendix C

Neural Network YoloV3

C.1 Transformation of the Dataset

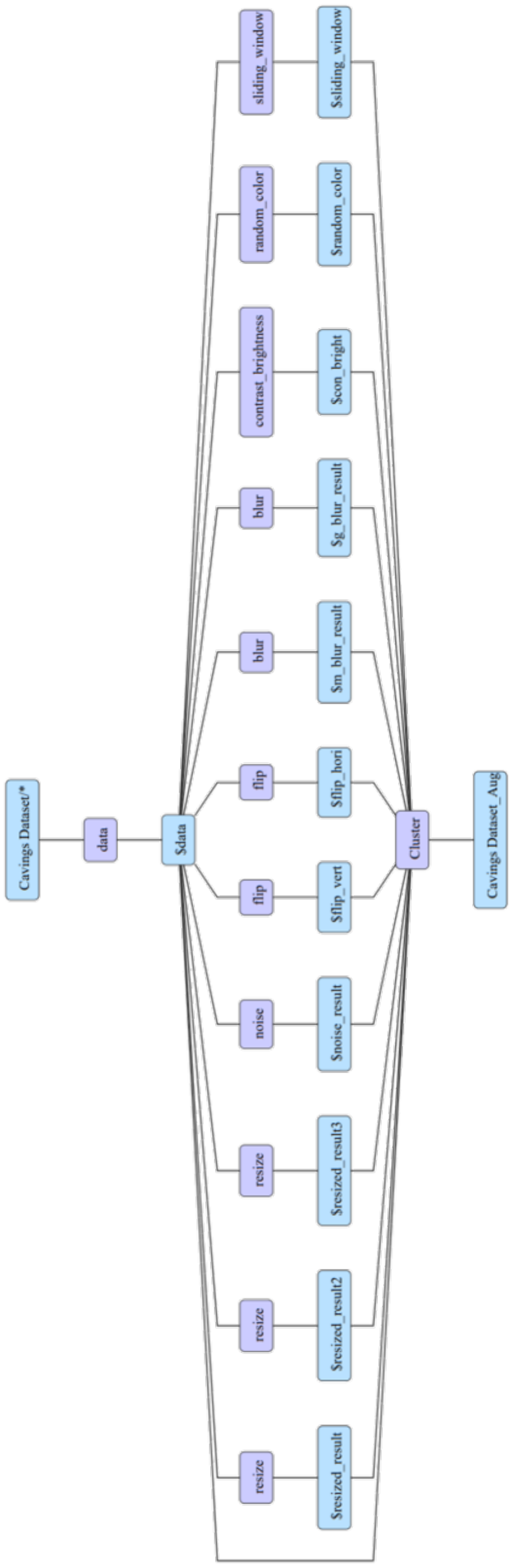


Figure C.1: Data augmentation performed in RAW dataset

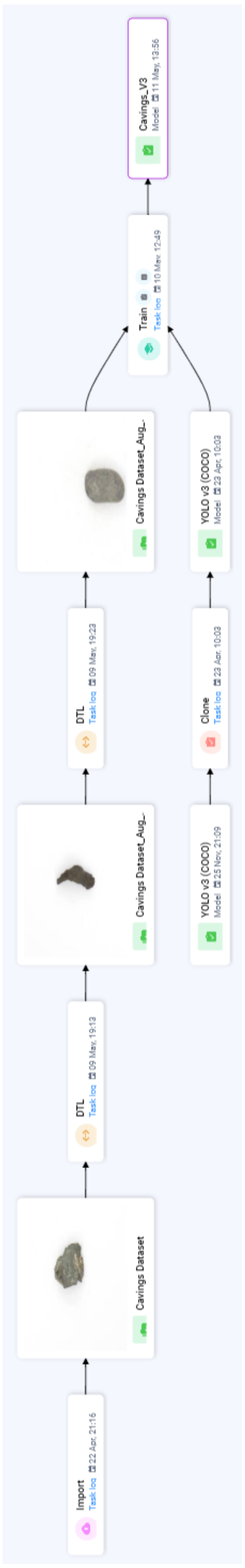
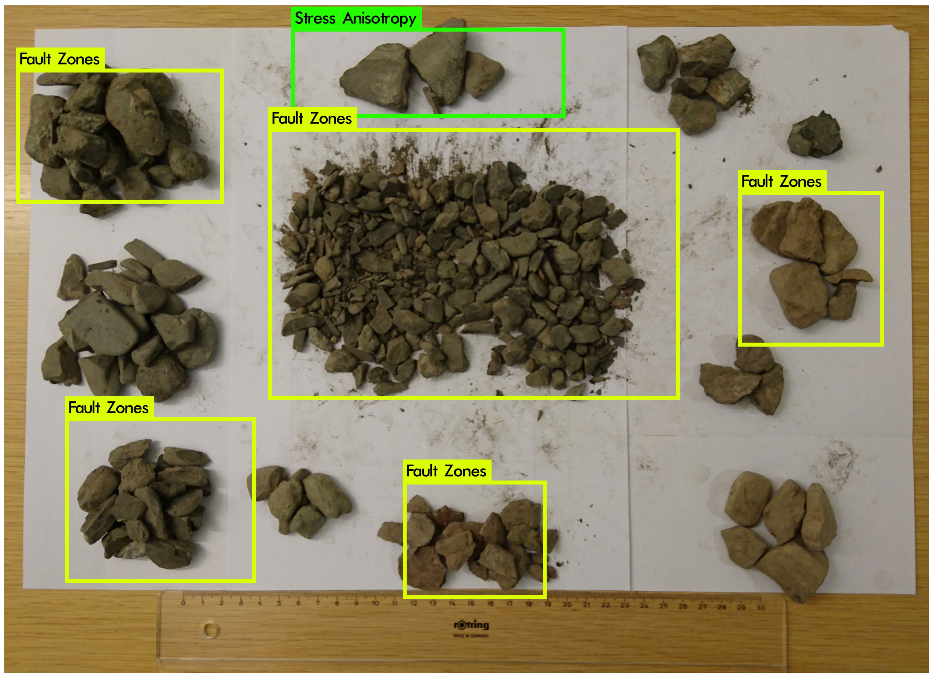
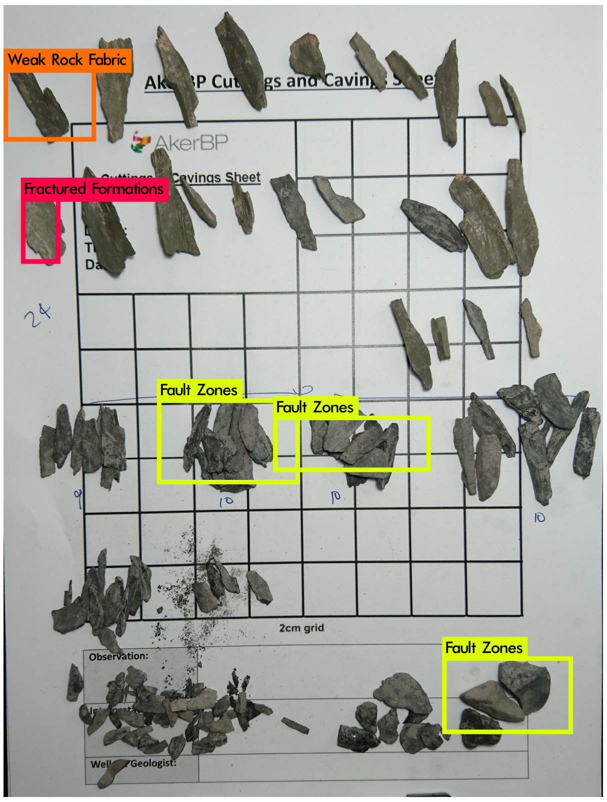


Figure C.2: Workflow to train YOLOv3

C.2 Results

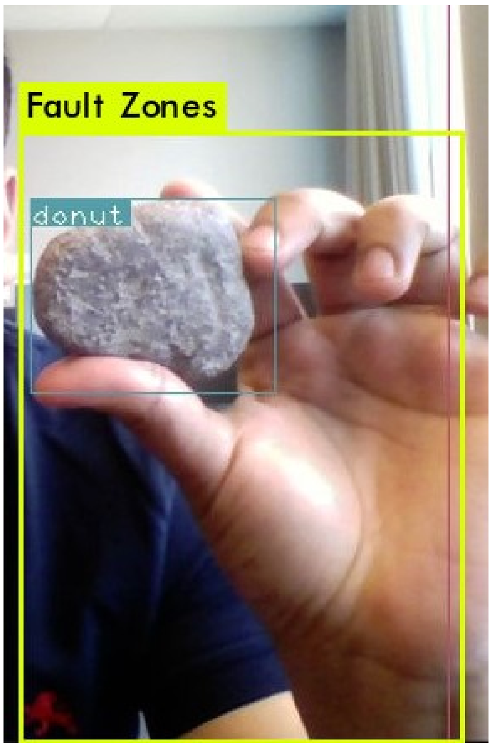


(a) Groups of cavings. It successfully detects the pyramidal cavings at the top as stress anisotropy. Notice how not all the cavings are detected

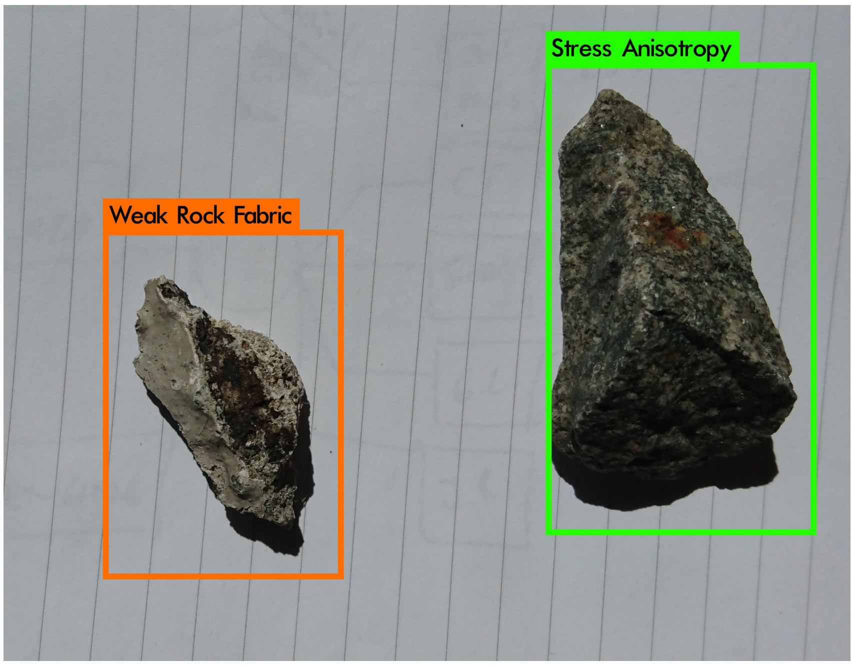


(b) It presents difficulties to detect the cavings that are close to each other. More training needs to be done with images like this. Again, not all the cavigns are detected.

Figure C.3: Caving detection - Groups



(a) YoloV3 was pre-trained and as a result it detects the irregular caving as a Donut, after training it is correctly detected as "Fault Zones" problem



(b) The pyramidal caving is correctly detected as "Stress Anisotropy", however the classification for "Weak Rock Fabric" is arguable

Figure C.4: Caving detection - single samples

Appendix D

Flowchart

D.1 Flowchart of Caving Features Project

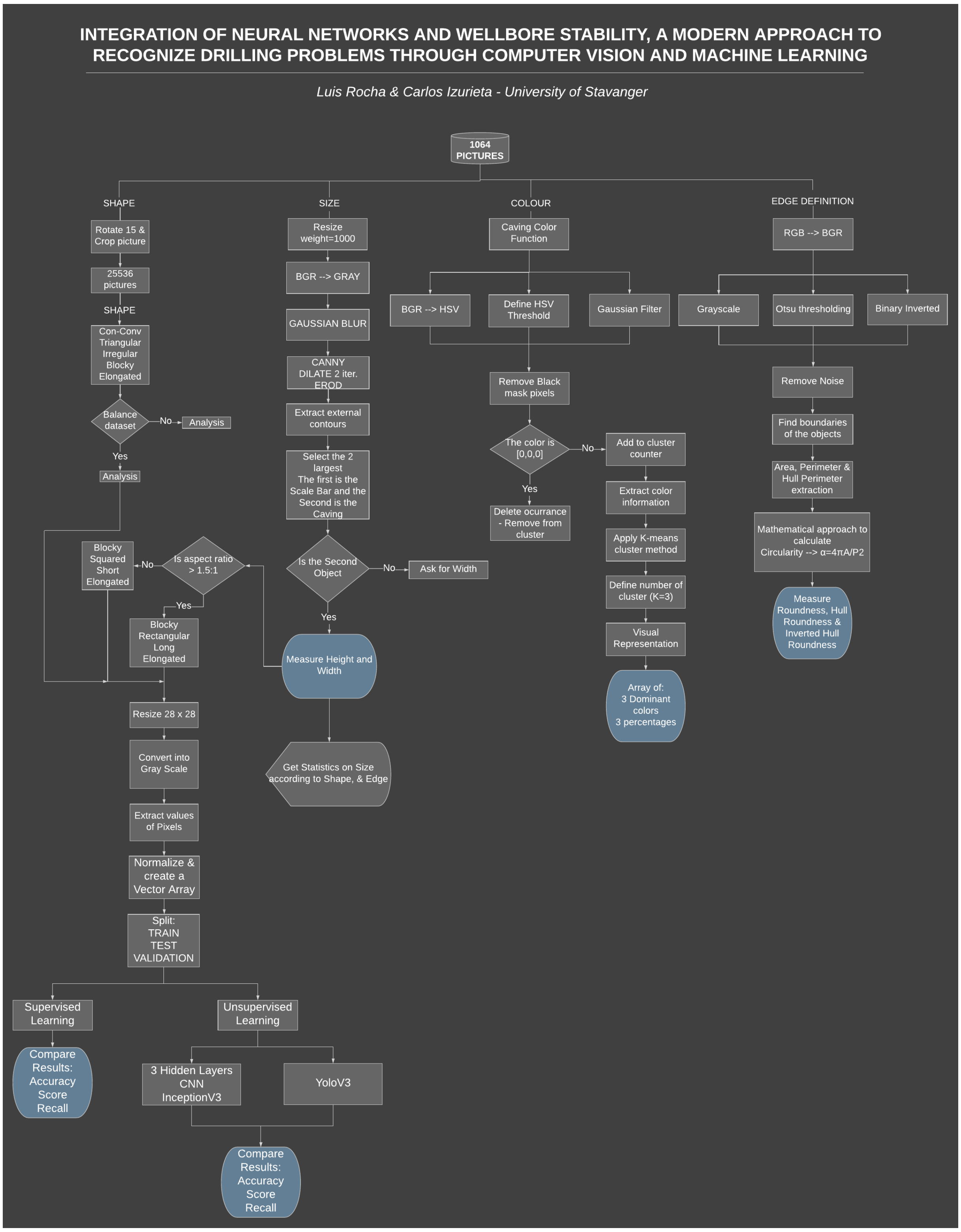

INTEGRATION OF NEURAL NETWORKS AND WELLBORE STABILITY, A MODERN APPROACH TO RECOGNIZE DRILLING PROBLEMS THROUGH COMPUTER VISION AND MACHINE LEARNING
Luis Rocha & Carlos Izurieta - University of Stavanger
1064 PICTURES
SHAPE
SIZE
COLOUR
EDGE DEFINITION
Rotate 15 & Crop picture
25536 pictures
SHAPE
Con-Conv Triangular Irregular Blocky Elongated
Balance dataset
No
Analysis
Yes
Analysis
Blocky Squared Short Elongated
No
Is aspect ratio > 1.5:1
Yes
Blocky Rectangular Long Elongated
Resize 28 x 28
Convert into Gray Scale
Extract values of Pixels
Normalize & create a Vector Array
Split: TRAIN TEST VALIDATION
Supervised Learning
Compare Results: Accuracy Score Recall
Unsupervised Learning
3 Hidden Layers CNN InceptionV3
YoloV3
Compare Results: Accuracy Score Recall
Resize weight=1000
BGR --> GRAY
GAUSSIAN BLUR
CANNY DILATE 2 iter. EROD
Extract external contours
Select the 2 largest The first is the Scale Bar and the Second is the Caving
Is the Second Object
No
Ask for Width
Yes
Measure Height and Width
Get Statistics on Size according to Shape, & Edge
Caving Color Function
BGR --> HSV
Define HSV Threshold
Gaussian Filter
Remove Black mask pixels
The color is [0,0,0]
No
Add to cluster counter
Yes
Delete ocurrance - Remove from cluster
Extract color information
Apply K-means cluster method
Define number of cluster (K=3)
Visual Representation
Array of: 3 Dominant colors 3 percentages
RGB --> BGR
Grayscale
Otsu thresholding
Binary Inverted
Remove Noise
Find boundaries of the objects
Area, Perimeter & Hull Perimeter extraction
Mathematical approach to calculate Circularity --> α=4πA/P2
Measure Roundness, Hull Roundness & Inverted Hull Roundness

Figure D.1: Caving Project Flowchart

D.2 Future Model - Linking Caving Features with Drilling Paramaters

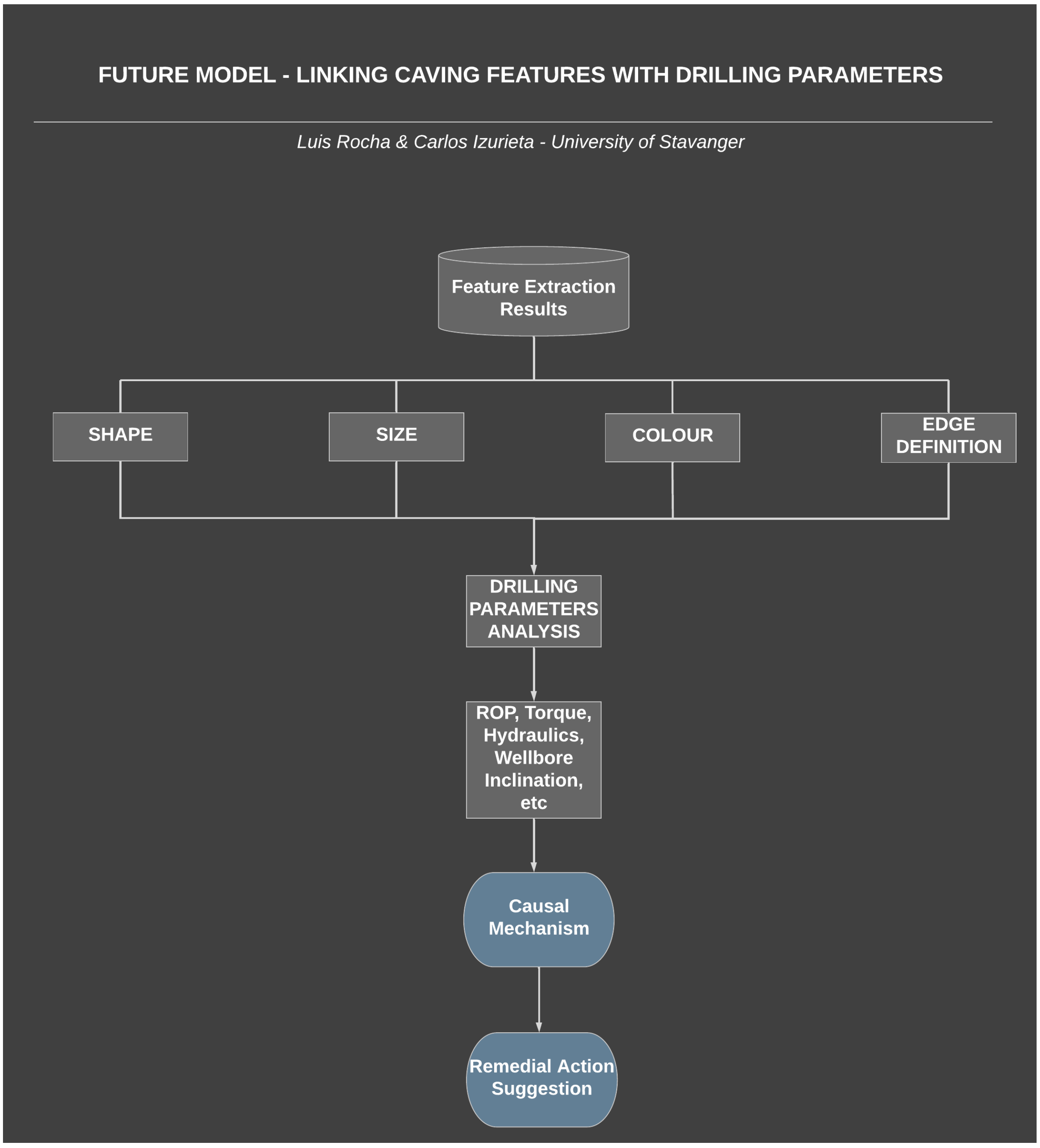

FUTURE MODEL - LINKING CAVING FEATURES WITH DRILLING PARAMETERS
Luis Rocha & Carlos Izurieta - University of Stavanger
Feature Extraction Results
SHAPE
SIZE
COLOUR
EDGE DEFINITION
DRILLING PARAMETERS ANALYSIS
ROP, Torque, Hydraulics, Wellbore Inclination, etc
Causal Mechanism
Remedial Action Suggestion

Figure D.2: Future Model - Linking Caving Features with Drilling Parameters

Appendix E

Drilling Parameters

E.1 Linking Drilling Parameters to Wellbore Stability Problems

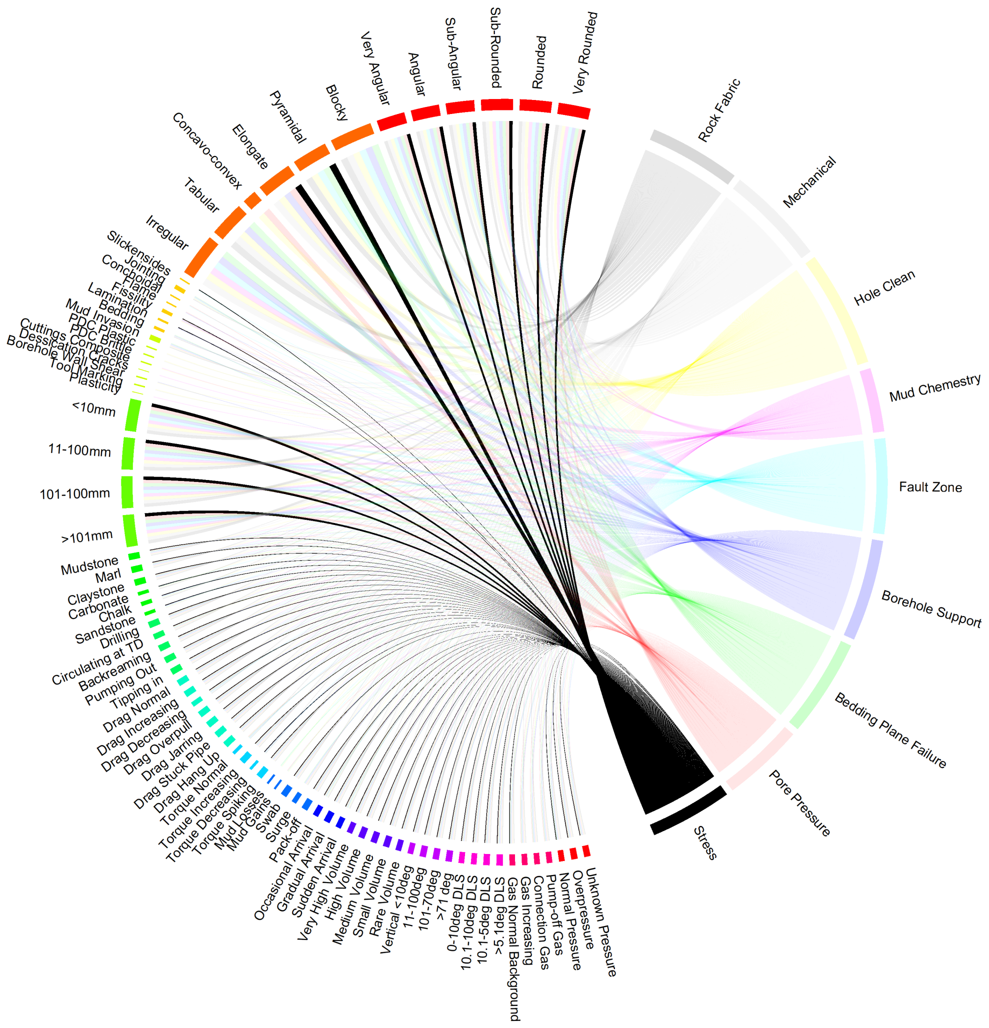


Figure E.1: Stress

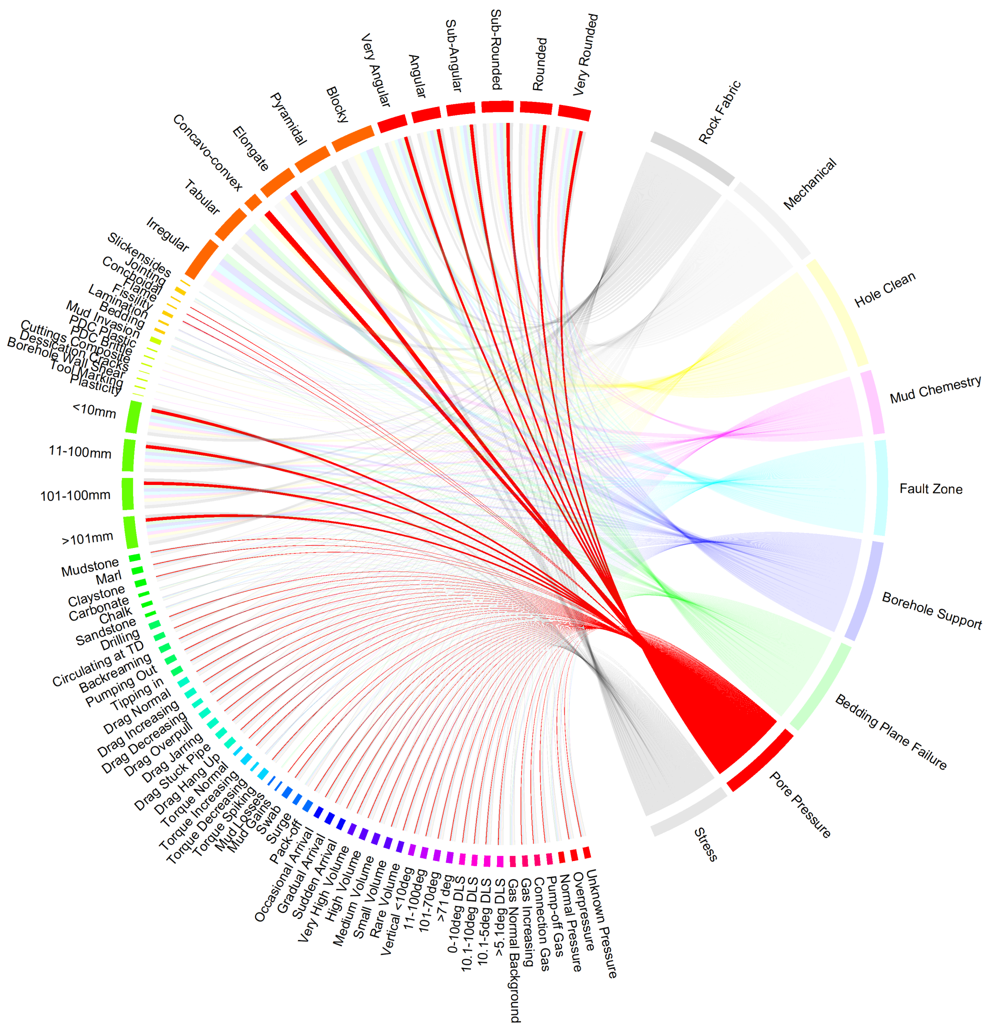


Figure E.2: Pore Pressure

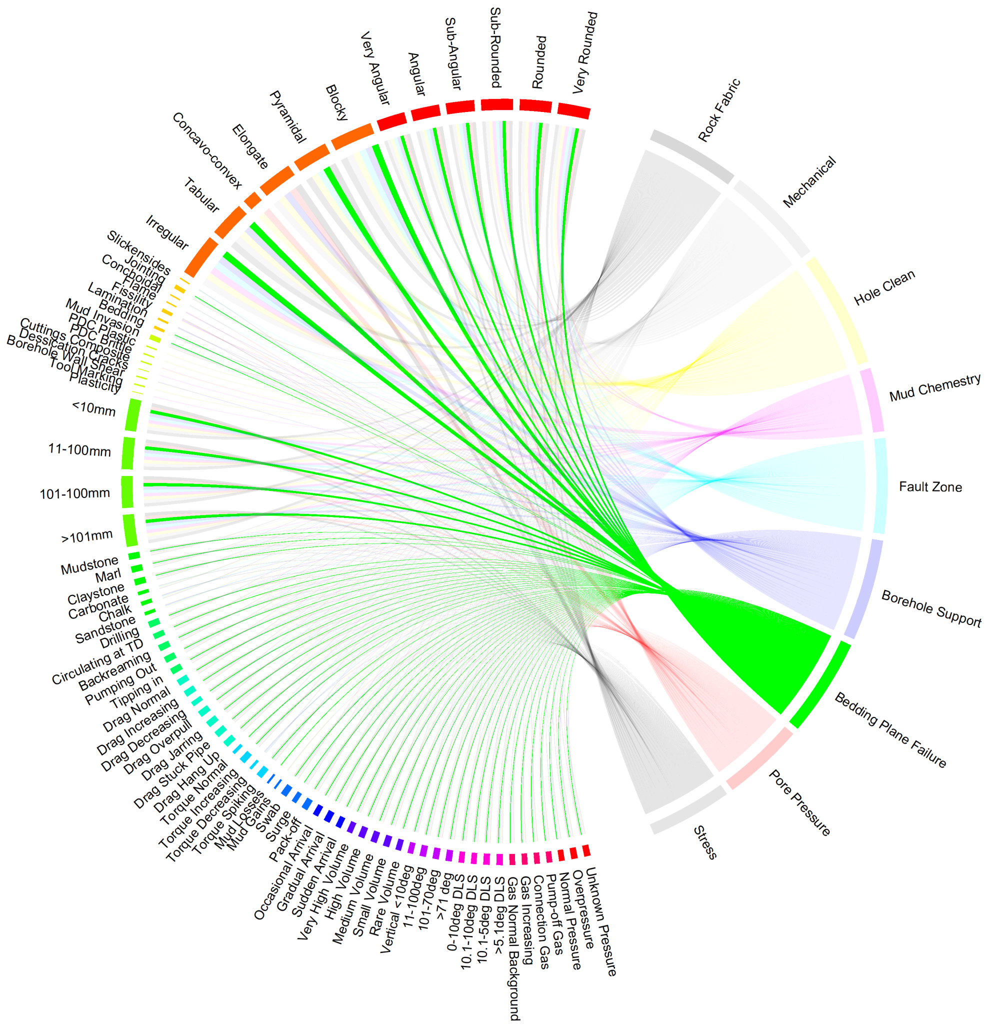


Figure E.3: Bedding Plane Failure

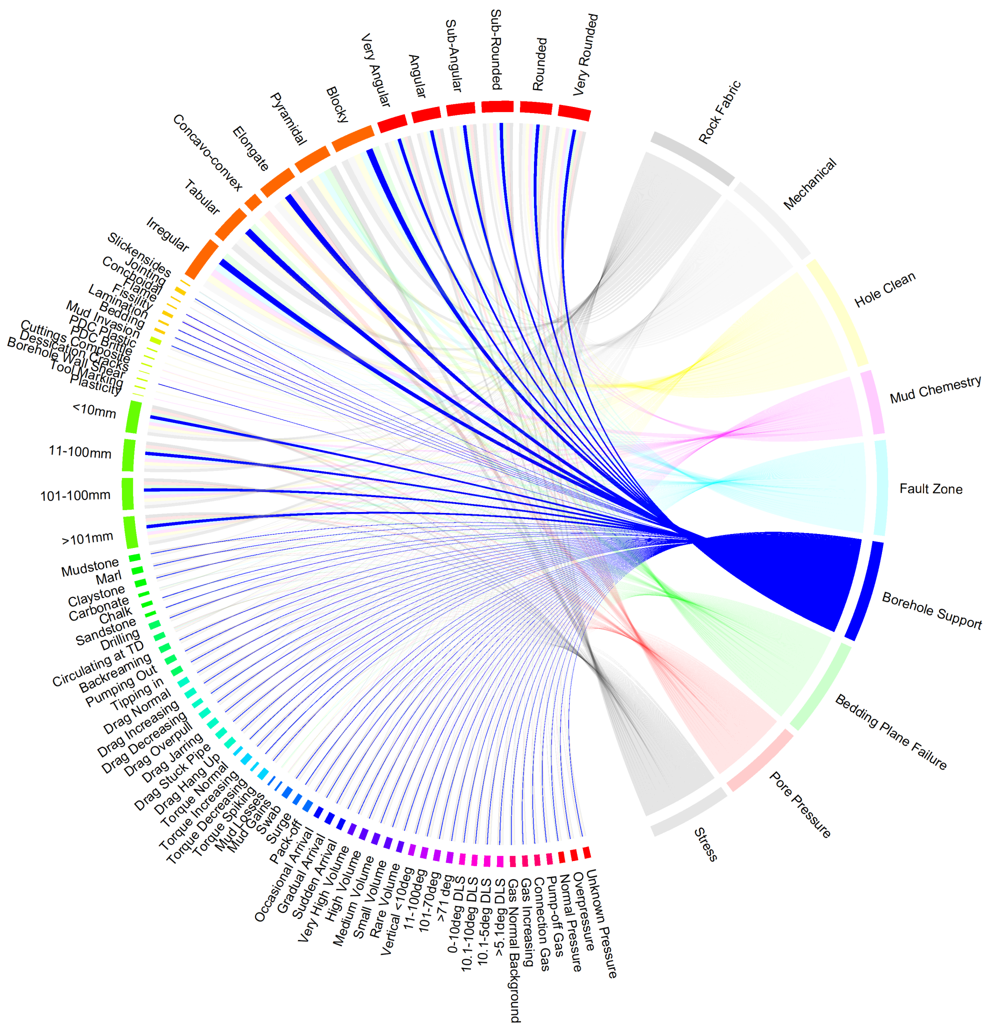


Figure E.4: Borehole Support

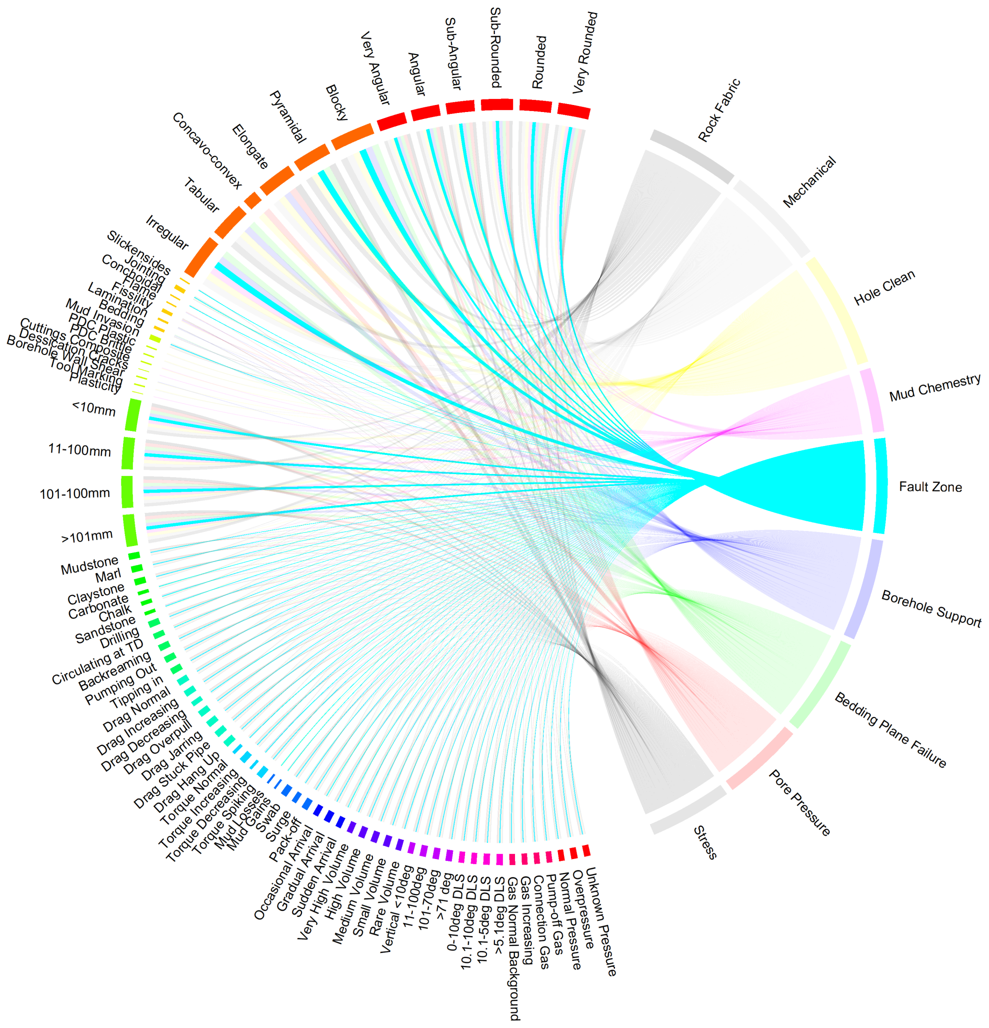


Figure E.5: Fault Zone

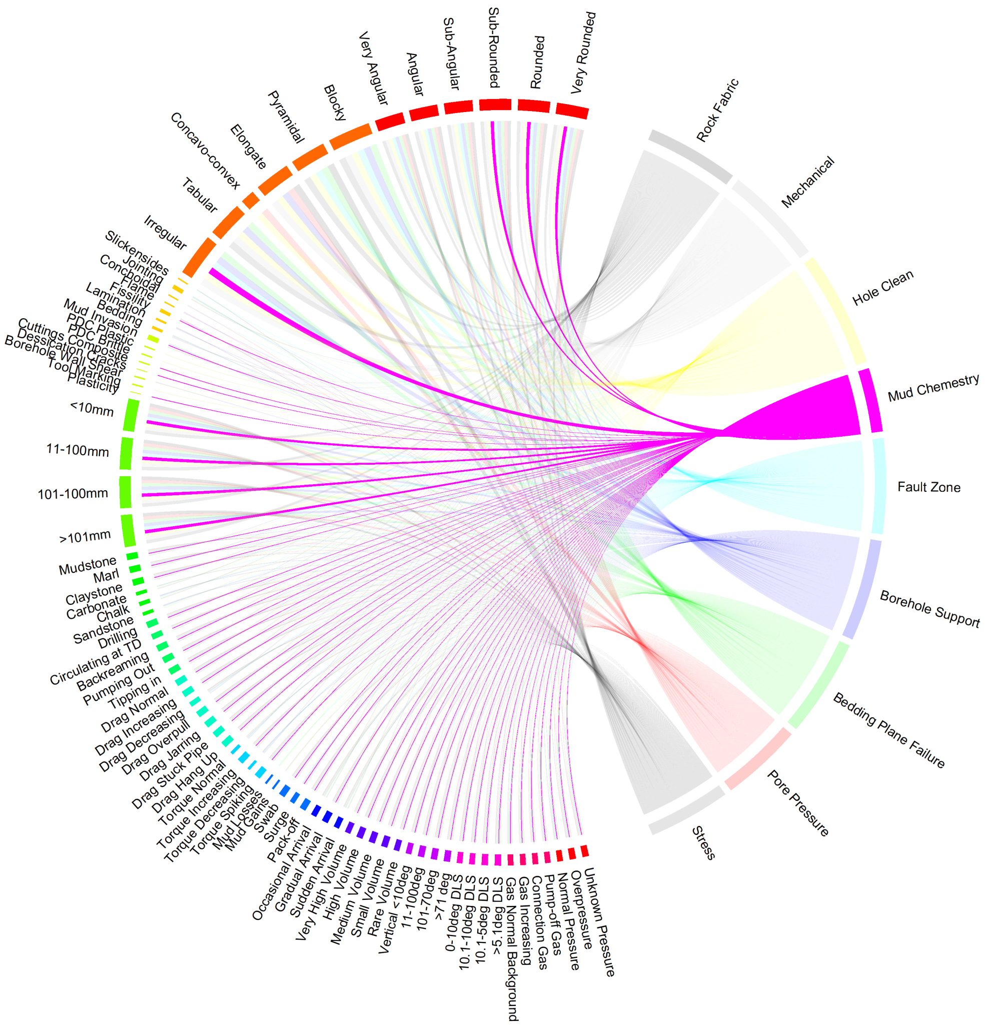


Figure E.6: Mud Chemestry

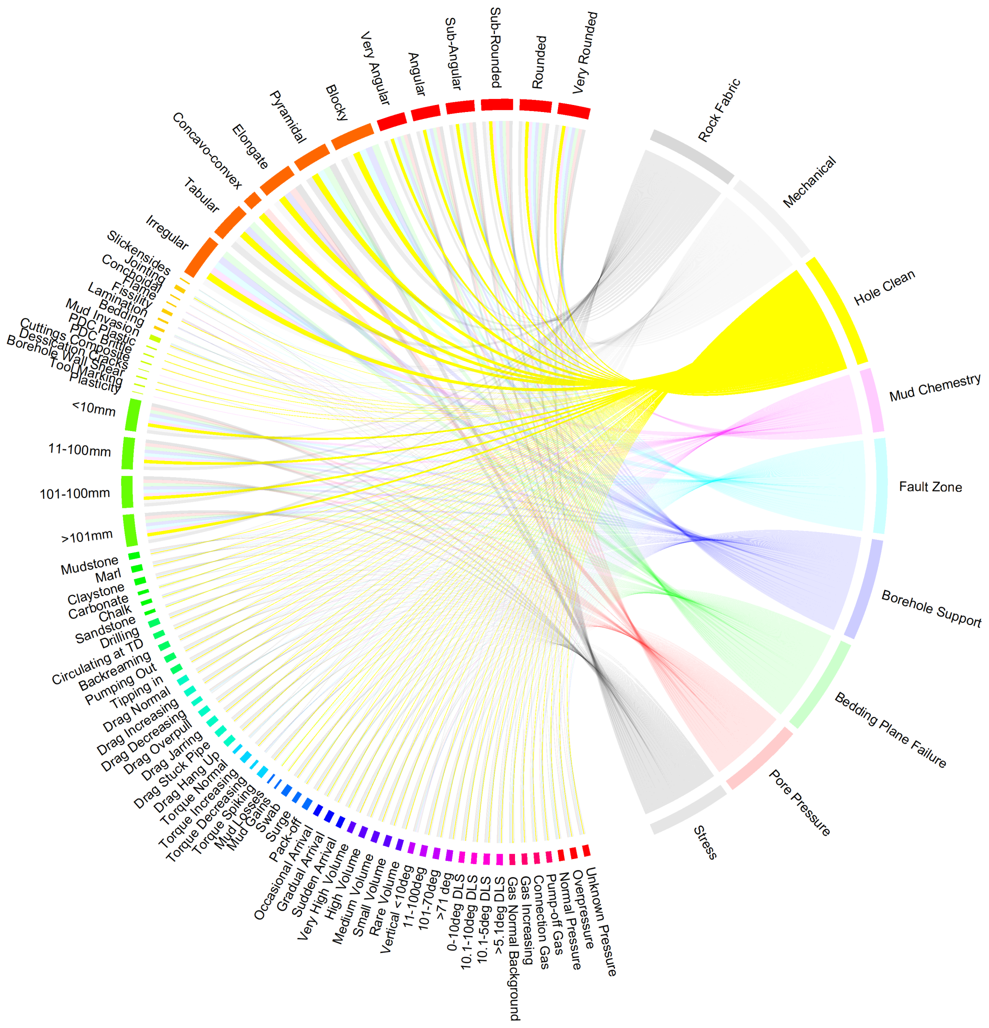


Figure E.7: Hole Clean

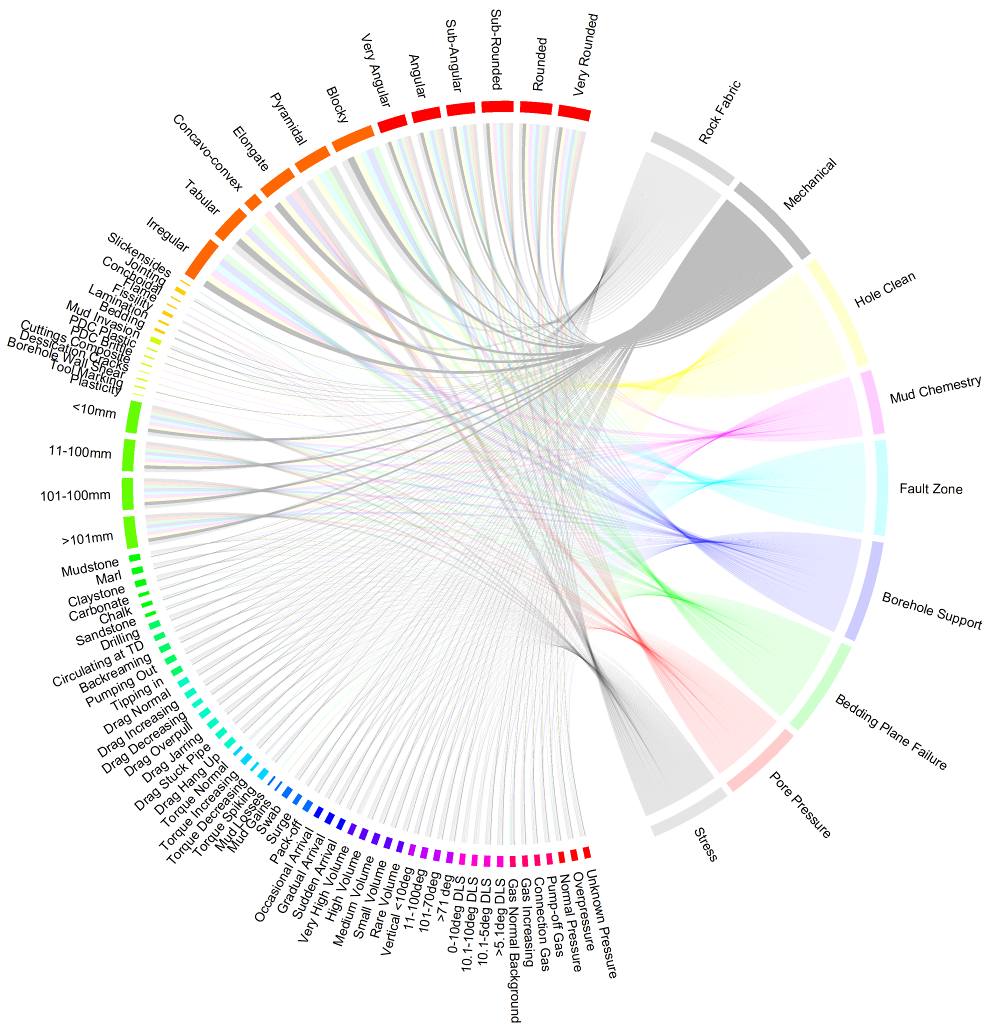


Figure E.8: Mechanical

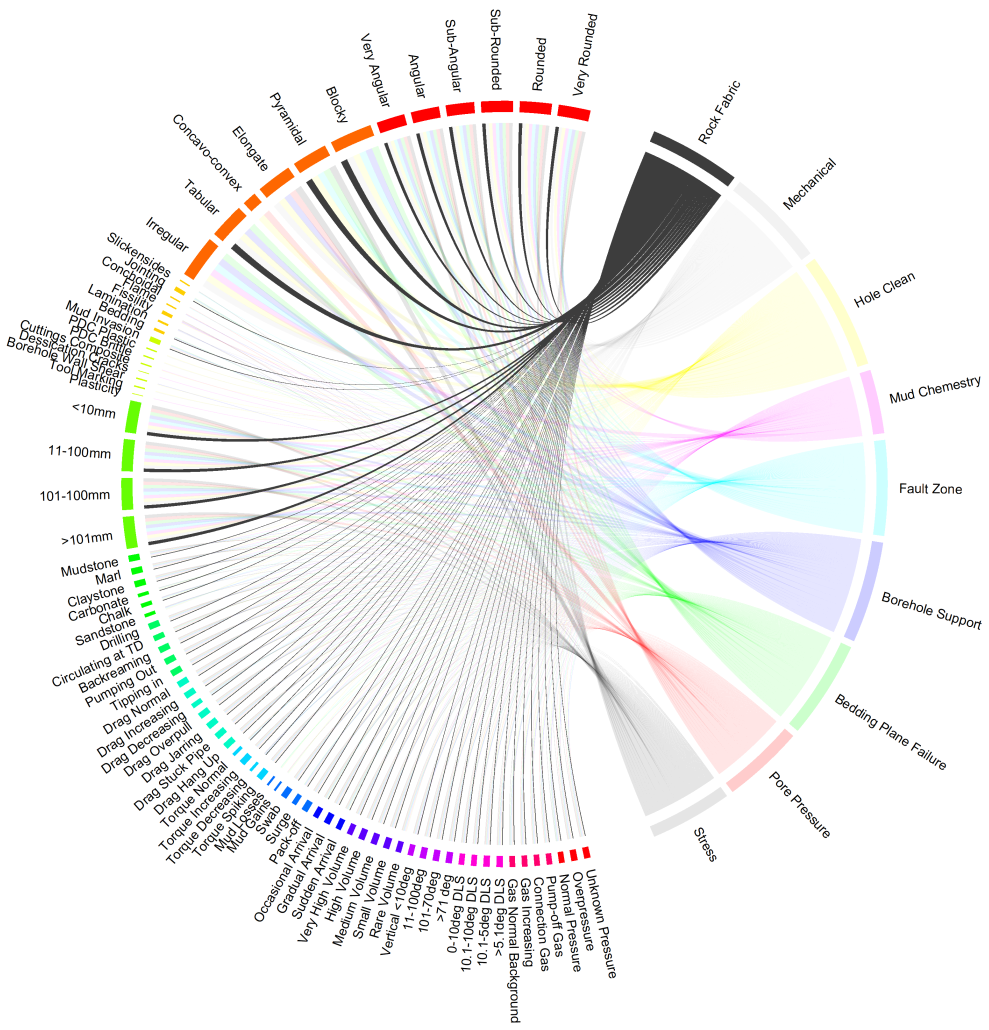


Figure E.9: Rock Fabric