APPLICATION OF DEEP LEARNING TECHNIQUES FOR LITHOFACIES CLASSIFICATION USING WELL LOG DATA OF TARANAKI BASIN

A DISSERTATION

Submitted in partial fulfillment of the

Requirements for the award of the degree

of

INTEGRATED MASTER OF TECHNOLOGY

in

GEOPHYSICAL TECHNOLOGY

by

MAYANK VERMA

DEPARTMENT OF EARTH SCIENCES

INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

ROORKEE-247667 (INDIA)

MAY, 2019

INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

ROORKEE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the dissertation entitled "**APPLICATION OF DEEP LEARNING TECHNIQUES FOR IDENTIFICATION OF LITHOFACIES CLASSIFICATION USING WELL LOG DATA OF TARANAKI BASIN**" in partial fulfilment of the requirements for the award of the Degree of Integrated Master of Technology in Geophysical Technology and submitted in the Department of Earth Sciences of the Indian Institute of Technology Roorkee, Roorkee is an authentic record of my own work carried out during a period from May 2018 to May, 2019 under the supervision of Dr. Ravi Sharma, Asst. Professor, Department of Earth Sciences, Indian Institute of Technology Roorkee, Roorkee.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other Institution.

(MAYANK VERMA)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

(Dr. Ravi Sharma)

Supervisor

Date

Acknowledgement

I am indebted to a number of personalities who at various stages have helped me in completing this dissertation work. Firstly, I would like to extend my sincere gratitude, soulful appreciations and a big thanks to my mentor **Dr. Ravi Sharma,** Assistant Professor, Department of Earth Sciences, IIT Roorkee for his valuable guidance, suggestions, immense knowledge, constant encouragement and engagement throughout the whole course of the dissertation work. Without his personal attention and interest, it would have been very difficult for me to work on this dissertation work in an effective manner.

My humble gratitude to **Prof. Sunil Bajpai,** the Head of the Department, Department of Earth Sciences, IIT Roorkee, for providing all the necessary facilities and fast administrative help necessary to complete this work. I would also like to thank professor-in-charge, **Dr. Anand Joshi** under whom I got this excellent opportunity.

I feel short of words to express my sentiments towards my loving family for their unduly love and fondle, for their everlasting moral support, for the sacrifices they have made on my behalf. I also would like to thank my friends for all the support at every stage of this dissertation work.

ABSTRACT

Classification of different lithofacies is crucial in seismic interpretation because different rocks have different permeability and fluid saturation for a given porosity. The ideal sources for lithofacies classification are core samples of rocks extracted from wells. Nevertheless, core samples cannot always be obtained due to associated costs. The conventional classification method is based on manually assigning lithofacies by human interpreters and is a very tedious and time consuming process.

I aimed at automating this classification process through the use of machine learning and deep learning methods. I selected wells from the region which have the mud log present with them. Machine learning algorithm, Support vector machines (SVM) was employed to build an automatic lithofacies classifier. The algorithm was trained on the dataset that has lithofacies marked from the mud log. The accuracy of the model was validated on a well with unlabeled lithofacies and accuracy of 0.63 was achieved.

In the next phase, a Deep learning (DL) model based on Convolutional Neural Network was developed and training was carried out using the labeled dataset (same dataset as in case of SVM algorithm). It achieved an classification accuracy of 0.71 on blind dataset. The model performed fairly good at classifying facies which have a larger number of training examples. Due to the skewed nature of the dataset, the validation accuracy of the model showed a stark drop when compared with training accuracy. This major drop in accuracy occurs while classifying those facies which have limited number of training example.

The accuracy could be further improved by incorporating adjacent lithofacies in classification task, which was the limitation of the target dataset.

CONTENTS

LIST OF FIGURES AND TABLES

FIGURES

TABLES

Chapter 1: Introduction

1.1 Importance of Lithofacies Classification

Facies are used by geologists to group together body of rocks with similar characteristics in order to facilitate the study of a basin of interest. In the case of Oil & Gas reservoirs, porosity and permeability are critical properties to determine since they give indications about the potential volume of fluids that might be stored in a rock and how they will flow during production. We can therefore expect that grains size, shape and density as well as the depositional and compaction history of the rocks will be a dominant factor for the categorization. While the main source of information for defining those facies comes from the observation of core samples under visible and x-ray light, we also have a variety of well log recordings at our disposal. By measuring the acoustic and electrical responses as well as the nuclear radiations of the drilled medium, we can infer properties about its rock matrix and fluid content and indirectly relate them to the porosity, permeability or fluid saturation of the rocks.

1.2 Motivation for automated Lithofacies classification

Facies (i.e., lithofacies) classification consists in assigning a rock kind or class to a particular sample on the basis of measured features. Classification of different lithofacies is crucial in seismic interpretation because different rocks have different permeability and fluid saturation for a given porosity. The ideal sources for lithofacies classification are core samples of rocks extracted from wells. Nevertheless, core samples cannot always be obtained due to associated costs. Therefore, a method for classifying facies from indirect quantification (e.g., wireline logs.) is necessary. The conventional method consists in manually assigning lithofacies by human interpreters and is a very tedious and time consuming process. Therefore, several alternative approaches to the issue of facies classification from well data have been proposed.

1.3 Objectives

This thesis had the following objectives:

1. Study the effectiveness of machine learning Models (SVM) in automated prediction of prominent lithofacies in a well.
2. Study the effectiveness of deep learning Models (ConvNet) in automated prediction of prominent lithofacies in a well.
3. Prediction of lithofacies in a well using a classification model trained over the labeled dataset.

1.4 Thesis Outline

The disposition of the thesis work is as follows:

- The first chapter is the introduction. This chapter deals with the motivation, objective and outline of the thesis work.
- The second chapter, Background knowledge gives a brief introduction about various algorithms used during the course of this thesis work.
- The third chapter, Literature review gives an insight about the development of different statistical, machine and deep learning techniques for classification of Lithofacies.
- The fourth chapter Materials and methods gives an overlook of the dataset available. It also covers the implementation of Support Vector Machines (SVM) and Convoluted Neural Networks (ConvNet) for facies classification.
- The fifth chapter Results deals with the prediction outcome from SVM and ConvNet. It further describes about the parameter tuning approach used to further refine the results.
- The six and the final chapter reiterates over the conclusion drawn from this thesis work and prospects of the future work.

Chapter 2: Background Knowledge

2.1 Support Vector Machines (SVM)

SVM is a form of binary classifier, classifying a set of data into two classes, like malignant or benign for tumors. In SVM, the aim is to find the line gives the best data separation. In Figure 2.1, the line can’t be too close to the blue dots or too close to the red dots because it might overfit on the training data and perform poorly on the actual testing data. In other words, the objective here is to maximize the margins. The way the algorithm actually works is by solving a set of equations using a technique called quadratic programming.

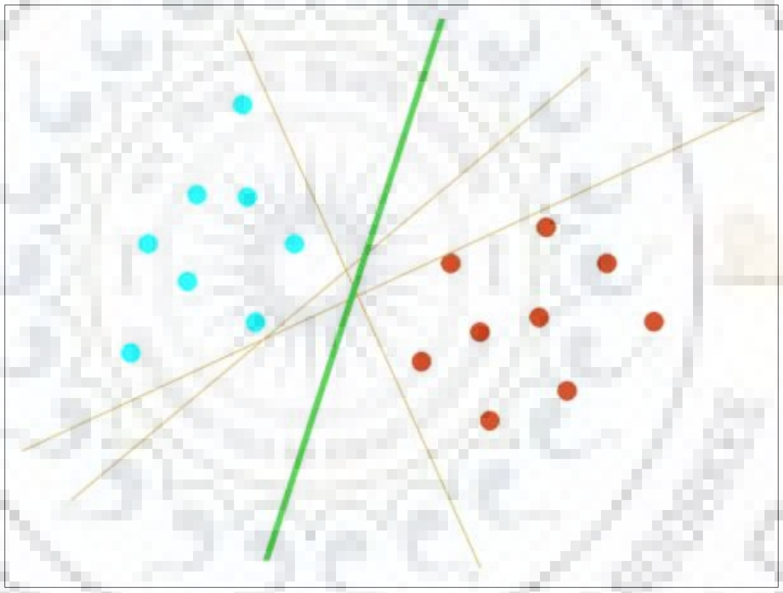


Figure 2.1: Optimal Separating Hyperplane
Source: MIT Edu

- **Kernel Trick**:

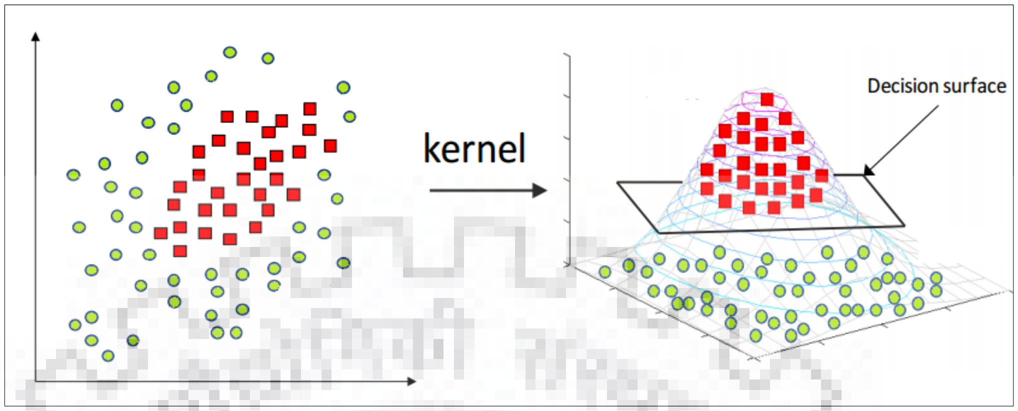
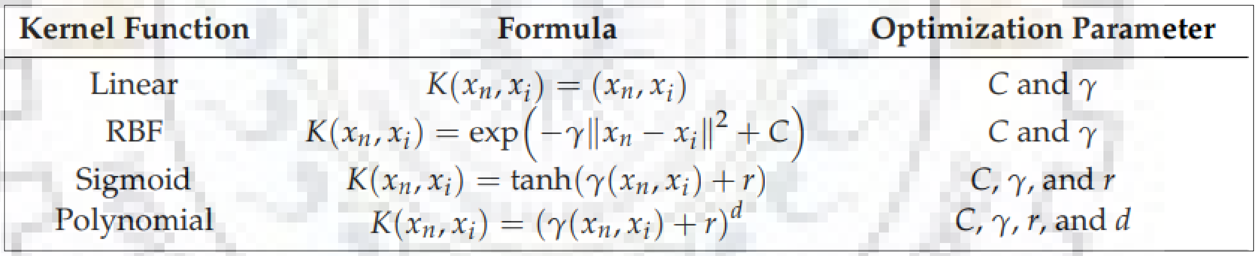


Figure 2.2: Kernel Trick
Source: MIT Edu

The kernel trick is a technique in ML in order to avert some rigorous computation that are involved in algorithm's. This trick helps in making few calculation goes from computationally improbable to probable.

Kernel functions:



Kernel Function	Formula	Optimization Parameter
Linear	$K(x_n, x_i) = (x_n, x_i)$	C and γ
RBF	$K(x_n, x_i) = \exp\left(-\gamma \lVert x_n - x_i \rVert^2 + C\right)$	C and γ
Sigmoid	$K(x_n, x_i) = \tanh(\gamma(x_n, x_i) + r)$	C, γ, and r
Polynomial	$K(x_n, x_i) = (\gamma(x_n, x_i) + r)^d$	C, γ, r, and d

Figure 2.3: Kernel functions commonly used in SVM
Source: Nanda et.al., 2017

- **Tuning parameters:**
  - Regularization parameter (C)
  - Gamma
  - Margin

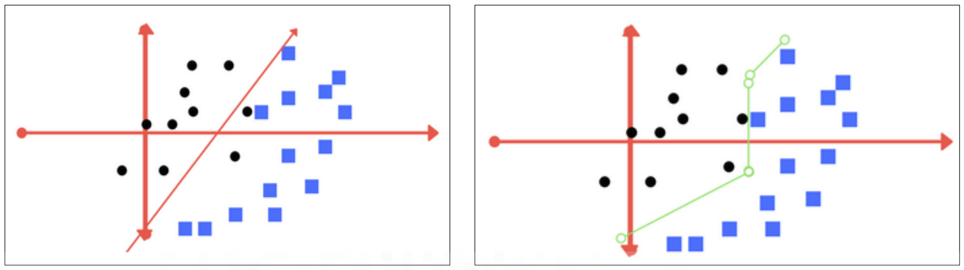


Figure 2.4: Left: low regularization value, right: high regularization value

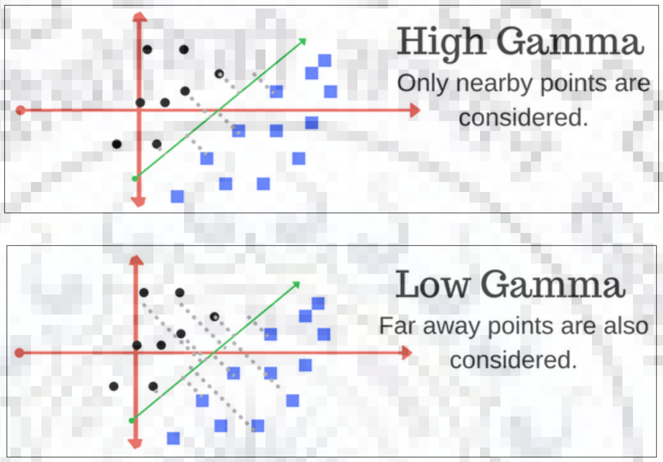

High Gamma
Only nearby points are considered.
Low Gamma
Far away points are also considered.

Figure 2.5: Top: High Gamma, Bottom: Low Gamma

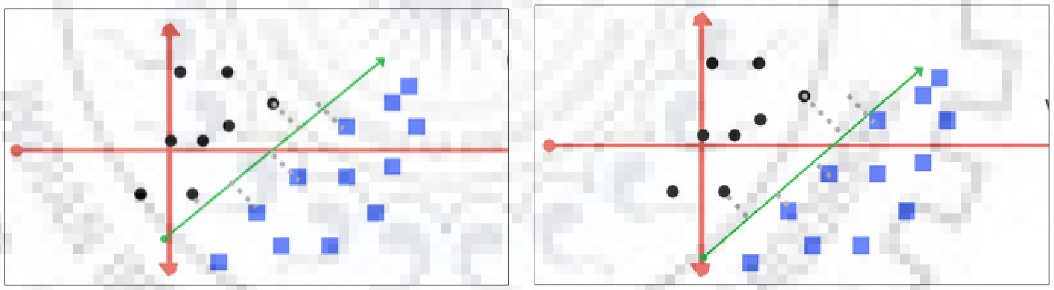


Figure 2.6: Left: Good margin, Right: Bad margin

2.2 Convolutional Neural Networks (ConvNet)

A convolutional neural network works by splitting the input into smaller chunks, or scanning over it in piecemeal sizes via some rule, and then passes that to the next layer who does the same thing with other rules.

- **Architecture of ConvNet**

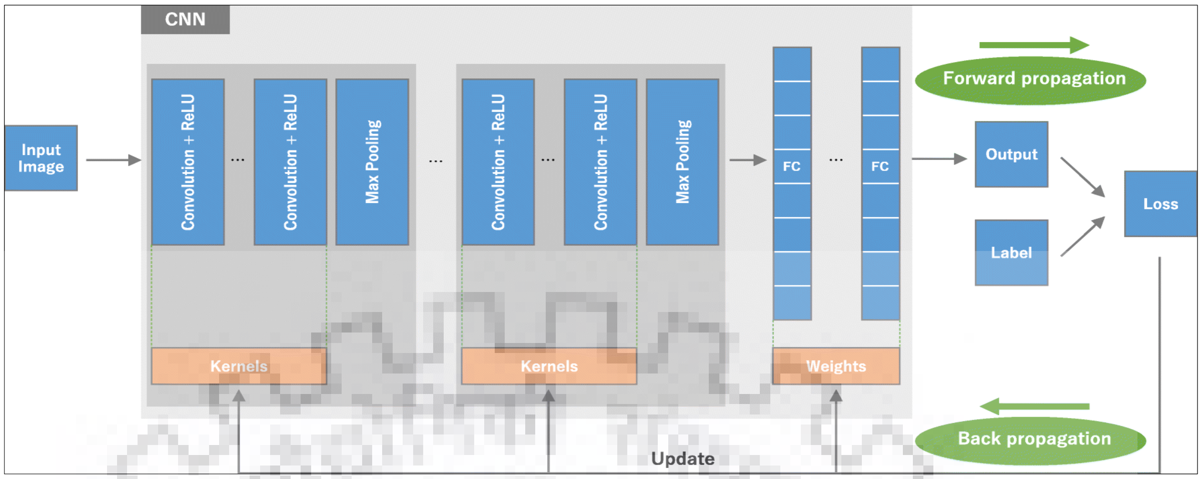


Figure 2.7: CNN architecture and sequence
Source: R Yamashita et al., 2018

Terminology:

- **Input Image**

  Holds the raw pixel values of the image.

- **Feature Learning**
  - **Convolution Layer - The Kernel**

    This layer helps in extracting the high-level feature(s) from the input image.

  - **Activation Layer**

    This layer introduces non-linearity in the network. Otherwise, it will act as a perceptron i.e. output can be predicted using the linear combination of input variables.

  - **Padding**

    Reduces dimensionality and helps in counter the border effect problem while convolution step.

  - **Pooling**

    Down sampling technique to identify the most important feature(s).

- **Classification**

    - **Fully connected Layer**

      Learns about non-learning combinations of high-level feature(s).

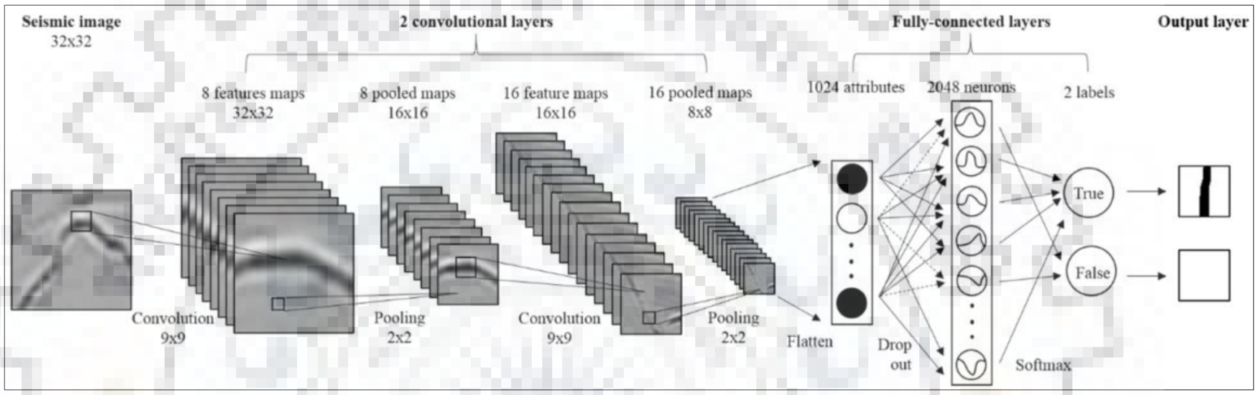


Figure 2.8: ConvNet used in study for salt body delineation
Source: Di et al., 2018

- **Hyper-parameters of ConvNet**:
    - Number of Hidden Layers and units
    - Dropout
    - Network Weight Initialization
    - Activation function

Chapter 3: Literature Review

3.1 Facies

Facies, in geology, is basically a way to differentiate rock bodies into units which are mappable, in terms of composition, characteristics, formation or several other attributes. Facies are mostly used by chemical, biological, or physical means to identify distinct units of rock bodies from adjacent lying units within a contiguous rock body. Facies generates a succession when compiled together that gives insight into processes and systems that might have acted within or on the region and rock record.

3.2 Lithofacies

The interpretation done on sedimentary sections are based on lithofacies units. While differentiating the sequences from each other, we take certain factors into consideration which include grain size, biogenic and physical sedimentary structure(s), lithologies, as these are directly related to the environment of deposition.

3.3 Related Work

Several types of conventional techniques including use of mud logs, well logs are essential in inferring subsurface lithology. Considering the importance of well log data in oil industry, the borehole data is used to infer the lithological changes taking the well logs as an accurate source of subsurface lithologies. Any change detected in the well log data is interpreted to be a geological and lithological change.

A number of quantitative and qualitative methods for lithology interpretation are already employed that combine various metrics.

Qualitatively, we inefficiently evaluate measurements from logging operations along with Photo electric (P_e) factor analyses, gamma ray evaluations done for identifying shale (Gardner and Dumanoir (1980), Serra et al. (1985), Dewan (1986), or using multiple logs. This proves insufficient as several lithologies pose complexity which requires larger set of information than just provided through these.

Within time, lithology interpretation has expanded and starts to consider the usage of quantitative methods, such as Crossplots, statistical analysis, and neural network.

Lithology interpretation has come a long way since, employing the use of methods quantitatively, including cross plots, statistical and neural networks analysis.

Burke et al. (1969) was the first one to introduce Crossplots based quantitative analysis which is commonly used over neutron, P_e, density and sonic. It involves simultaneous plots of two or more log data points. Even Clavier and Rust (1976) studied its application in quantitative analyses. These methods have a limitation that they require manual human intervention and cannot be automated for the task.

Statistical Analysis, proven to have applications for lithological identification by Delfiner et al. (1987), laid way for combining wireline measurement done these days, to generate automated lithological description. Statistical Classification methods, like linear regression, kernel estimation, discrimination analysis etc. have greater dependence on the data characteristics and hence leads to efficient modeling based on interpretation. Probability based classification methods that returns a probability which predicts the belonging of member.

Busch et al. (1987) demonstrated lithological prediction was possible using statistical analysis. It proved the importance of Bayesian rule for probability and discriminant analysis in lithological classification. But, this method has a limitation in the criteria for the data to be of normal distribution based (limits the geophysical dataset). Hence, it was deprecated because of its lack of flexibility in application to non-parametric distributions.

Kernel Density Estimator, which analyses the multimodal nature of the data, is a method that estimates the probability density function (PDF) for a non-parametric distribution. Mwenifumbo (1993) demonstrated the usage of the estimator for borehole geophysical data, by applying it for univariate and bivariate data, thus identifying sulfide mineralization by analyzing it.

Chapter 4: Material and Methods

4.1 Data Source

The dataset is from Taranaki basin which is situated on the west coast of New Zealand. It is an onshore-offshore rift basin.

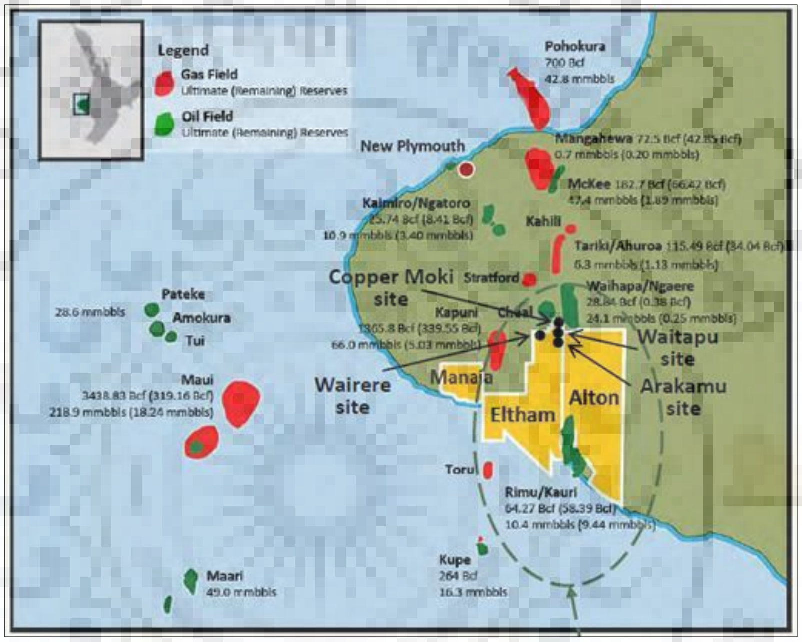


Figure 4.1: Taranaki basin
Source: energy-pedia

4.2 Target Data

The target wells selected for this thesis work were from KAURI region of the Taranaki basin. The selection criteria were the type of wireline logging data presented in any particular well. The ideal log for the purpose of lithological classification should be:

- Most affected by rock properties
- Least affected by fluid properties

The available logs in the Kauri regions based on above criteria were:

- Caliper (CALI)
- Gamma ray (GR)
- Formation density (DENS)

- Photoelectric absorption (PEF)
- Neutron porosity (NEUT)
- Sonic log (DTC)
- Resistivity log deep (RESD)

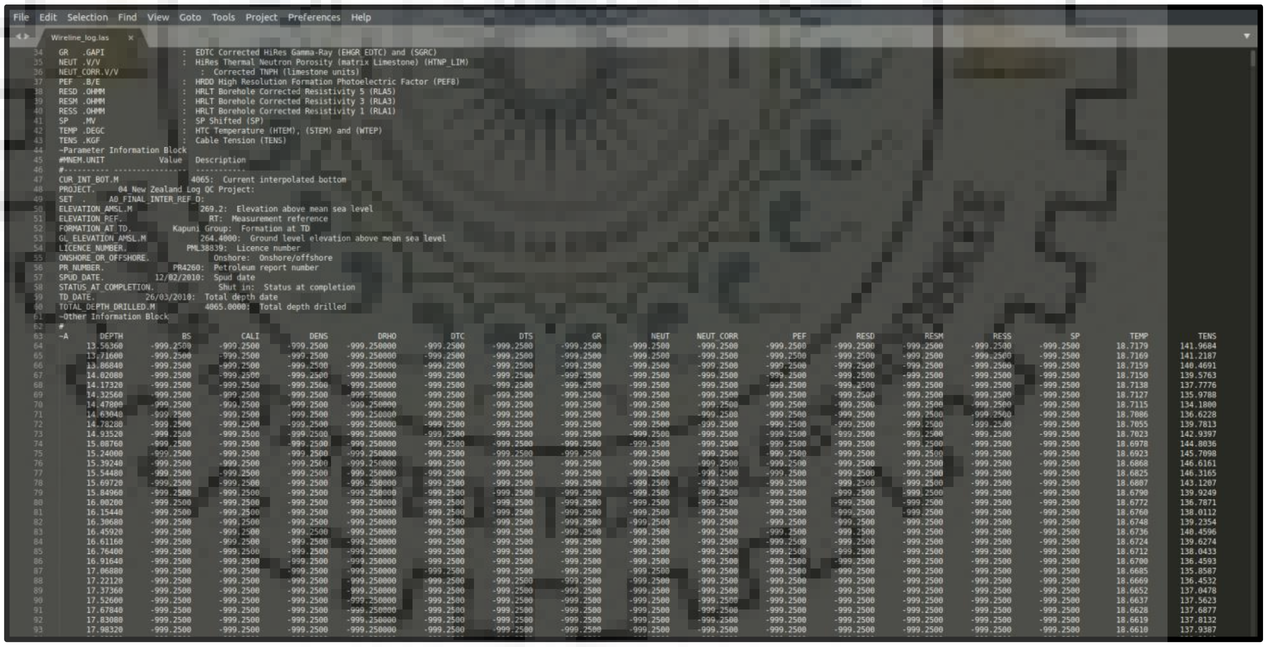
4.3 Data Preparation:

- **Selection of wells**

  In the first step, the selection of 8 wells from Kauri region was done out of nearly 600 available wells having appropriate wireline logs and mud log information.

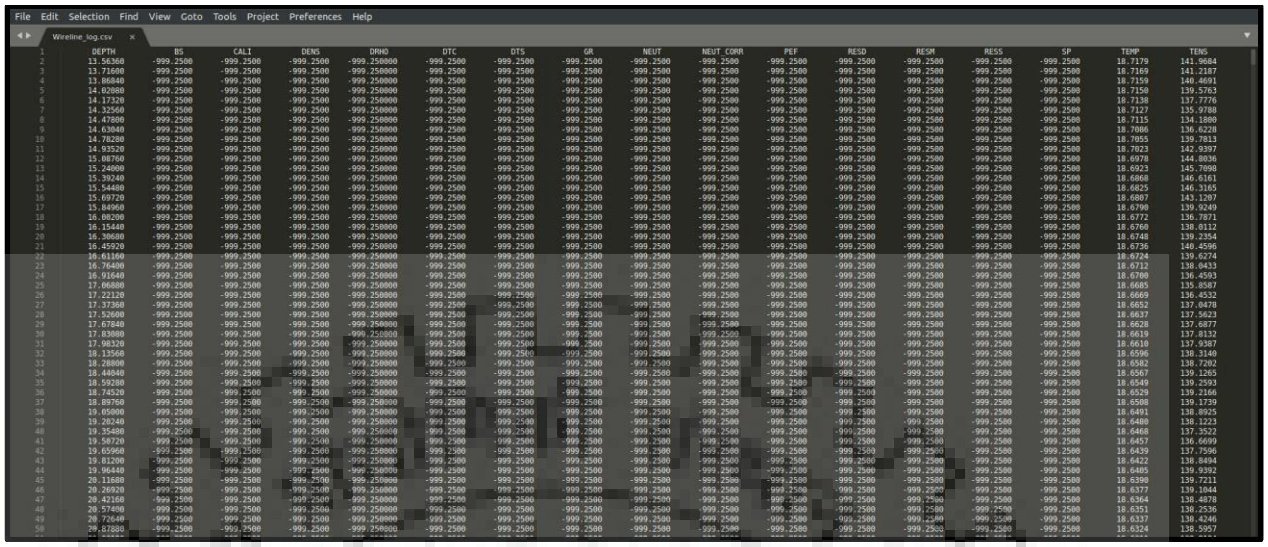
- **LAS log files to TXT files**

  Converting wireline data in LAS file format to text file format using Python. Code is given in Appendix 1.


File Edit Selection Find View Goto Tools Project Preferences Help
Wireline_log.las

```
GR   .GAPI              :  EDTC Corrected HiRes Gamma-Ray (EHGR_EDTC) and (SGRC)
NEUT .V/V               :  HiRes Thermal Neutron Porosity (matrix Limestone) (HTNP_LIM)
NEUT_CORR.V/V             :  Corrected TNPH (limestone units)
PEF  .B/E               :  HRDD High Resolution Formation Photoelectric Factor (PEF8)
RESD .OHMM              :  HRLT Borehole Corrected Resistivity 5 (RLA5)
RESM .OHMM              :  HRLT Borehole Corrected Resistivity 3 (RLA3)
RESS .OHMM              :  HRLT Borehole Corrected Resistivity 1 (RLA1)
SP   .MV                :  SP Shifted (SP)
TEMP .DEGC              :  HTC Temperature (HTEM), (STEM) and (WTEP)
TENS .KGF               :  Cable Tension (TENS)
~Parameter Information Block
#MNEM.UNIT           Value   Description
#---------- --------------   -----------
CUR_INT_BOT.M            4065:  Current interpolated bottom
PROJECT.     04_New Zealand Log QC Project:
SET   .     A0_FINAL_INTER_REF_D:
ELEVATION_AMSL.M           269.2:  Elevation above mean sea level
ELEVATION_REF.                RT:  Measurement reference
FORMATION_AT_TD.       Kapuni Group:  Formation at TD
GL_ELEVATION_AMSL.M        264.4000:  Ground level elevation above mean sea level
LICENCE_NUMBER.          PML38839:  Licence number
ONSHORE_OR_OFFSHORE.          Onshore:  Onshore/offshore
PR_NUMBER.             PR4260:  Petroleum report number
SPUD_DATE.         12/02/2010:  Spud date
STATUS_AT_COMPLETION.          Shut in:  Status at completion
TD_DATE.         26/03/2010:  Total depth date
TOTAL_DEPTH_DRILLED.M        4065.0000:  Total depth drilled
~Other Information Block
#
~A      DEPTH         BS       CALI       DENS        DRHO        DTC        DTS         GR       NEUT  NEUT_CORR        PEF       RESD       RESM       RESS         SP       TEMP       TENS
     13.56360  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.7179   141.9684
     13.71600  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.7169   141.2187
     13.86840  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.7159   140.4691
     14.02080  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.7150   139.5763
     14.17320  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.7138   137.7776
     14.32560  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.7127   135.9788
     14.47800  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.7115   134.1800
     14.63040  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.7086   136.6228
     14.78280  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.7055   139.7813
     14.93520  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.7023   142.9397
     15.08760  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6978   144.8036
     15.24000  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6923   145.7098
     15.39240  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6868   146.6161
     15.54480  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6825   146.3165
     15.69720  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6807   143.1207
     15.84960  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6790   139.9249
     16.00200  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6772   136.7871
     16.15440  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6760   138.0112
     16.30680  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6748   139.2354
     16.45920  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6736   140.4596
     16.61160  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6724   139.6274
     16.76400  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6712   138.0433
     16.91640  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6700   136.4593
     17.06880  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6685   135.8587
     17.22120  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6669   136.4532
     17.37360  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6652   137.0478
     17.52600  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6637   137.5623
     17.67840  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6628   137.6877
     17.83080  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6619   137.8132
     17.98320  -999.2500  -999.2500  -999.2500 -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500    18.6610   137.9387
```

Figure 4.2: Wireline log file in .las format



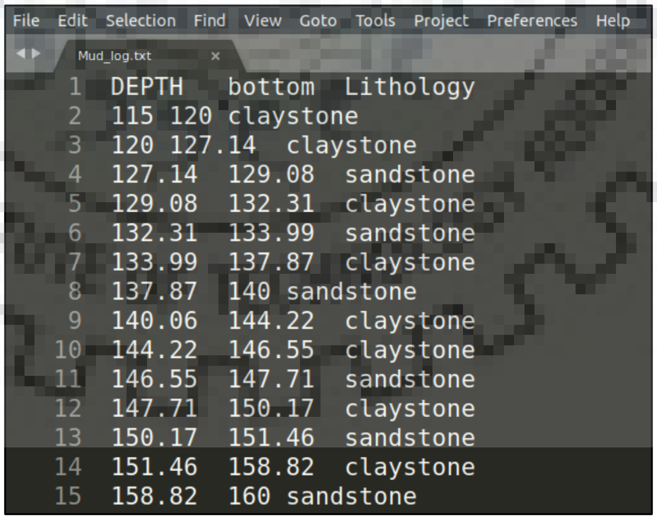
File Edit Selection Find View Goto Tools Project Preferences Help
Wireline_log.csv

```
DEPTH       BS         CALI       DENS       DRHO         DTC        DTS        GR         NEUT       NEUT_CORR  PEF        RESD       RESM       RESS       SP         TEMP     TENS
13.56360   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.7179  141.9684
13.71600   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.7169  141.2187
13.86840   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.7159  140.4691
14.02080   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.7150  139.5763
14.17320   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.7138  137.7776
14.32560   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.7127  135.9788
14.47800   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.7115  134.1800
14.63040   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.7086  136.6228
14.78280   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.7055  139.7813
14.93520   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.7023  142.9397
15.08760   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6978  144.8036
15.24000   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6923  145.7098
15.39240   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6868  146.6161
15.54480   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6825  146.3165
15.69720   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6807  143.1207
15.84960   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6790  139.9249
16.00200   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6772  136.7871
16.15440   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6760  138.0112
16.30680   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6748  139.2354
16.45920   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6736  140.4596
16.61160   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6724  139.6274
16.76400   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6712  138.0433
16.91640   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6700  136.4593
17.06880   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6685  135.8587
17.22120   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6669  136.4532
17.37360   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6652  137.0478
17.52600   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6637  137.5623
17.67840   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6628  137.6877
17.83080   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6619  137.8132
17.98320   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6610  137.9387
18.13560   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6596  138.3140
18.28800   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6582  138.7202
18.44040   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6567  139.1265
18.59280   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6549  139.2593
18.74520   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6529  139.2166
18.89760   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6508  139.1739
19.05000   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6491  138.8925
19.20240   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6480  138.1223
19.35480   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6468  137.3522
19.50720   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6457  136.6699
19.65960   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6439  137.7596
19.81200   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6422  138.8494
19.96440   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6405  139.9392
20.11680   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6390  139.7211
20.26920   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6377  139.1044
20.42160   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6364  138.4878
20.57400   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6351  138.2536
20.72640   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6337  138.4246
20.87880   -999.2500  -999.2500  -999.2500  -999.250000  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  -999.2500  18.6324  138.5957
```

Figure 4.3: Wireline log file in .txt format

- **Parsing the Mud log**

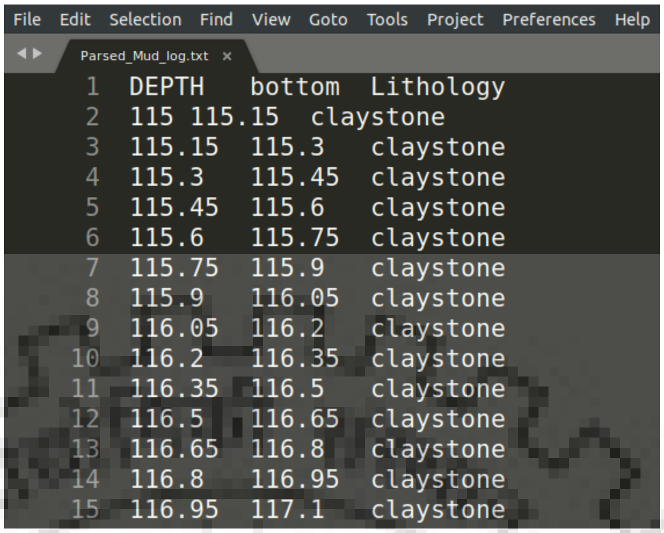
  Since the resolution of data were not same for wireline and mud logs tagging corresponding lithology to a depth point was not possible. To accomplish this, we parsed the mud logs and sampled them at a depth interval of 0.15 meters. Parsing was done using C++. Code is given in Appendix 2.



File Edit Selection Find View Goto Tools Project Preferences Help
Mud_log.txt

```
DEPTH   bottom  Lithology
115 120 claystone
120 127.14  claystone
127.14  129.08  sandstone
129.08  132.31  claystone
132.31  133.99  sandstone
133.99  137.87  claystone
137.87  140 sandstone
140.06  144.22  claystone
144.22  146.55  claystone
146.55  147.71  sandstone
147.71  150.17  claystone
150.17  151.46  sandstone
151.46  158.82  claystone
158.82  160 sandstone
```

Figure 4.4: MUD log file in .txt format

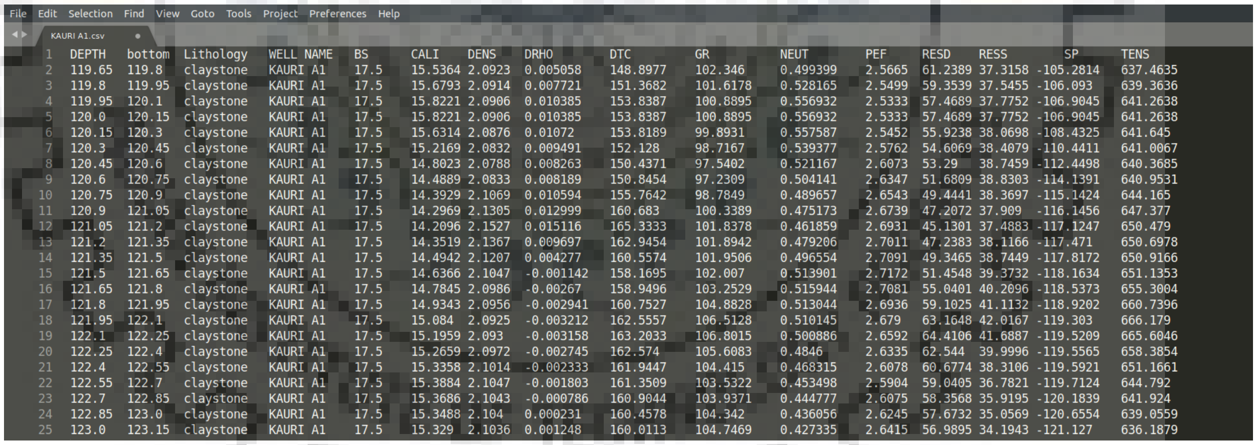

File Edit Selection Find View Goto Tools Project Preferences Help
Parsed_Mud_log.txt

```
DEPTH   bottom  Lithology
115 115.15  claystone
115.15  115.3   claystone
115.3   115.45  claystone
115.45  115.6   claystone
115.6   115.75  claystone
115.75  115.9   claystone
115.9   116.05  claystone
116.05  116.2   claystone
116.2   116.35  claystone
116.35  116.5   claystone
116.5   116.65  claystone
116.65  116.8   claystone
116.8   116.95  claystone
116.95  117.1   claystone
```

Figure 4.5: Parsed MUD log file in .txt format

- **Tagging lithology to corresponding wireline data depth**

  After parsing the mud logs and sampling them at a depth interval of 0.15 meters, the wireline data was merged with mud log on the basis of depth values using Python. Code is given in Appendix 3.


File Edit Selection Find View Goto Tools Project Preferences Help
KAURI A1.csv

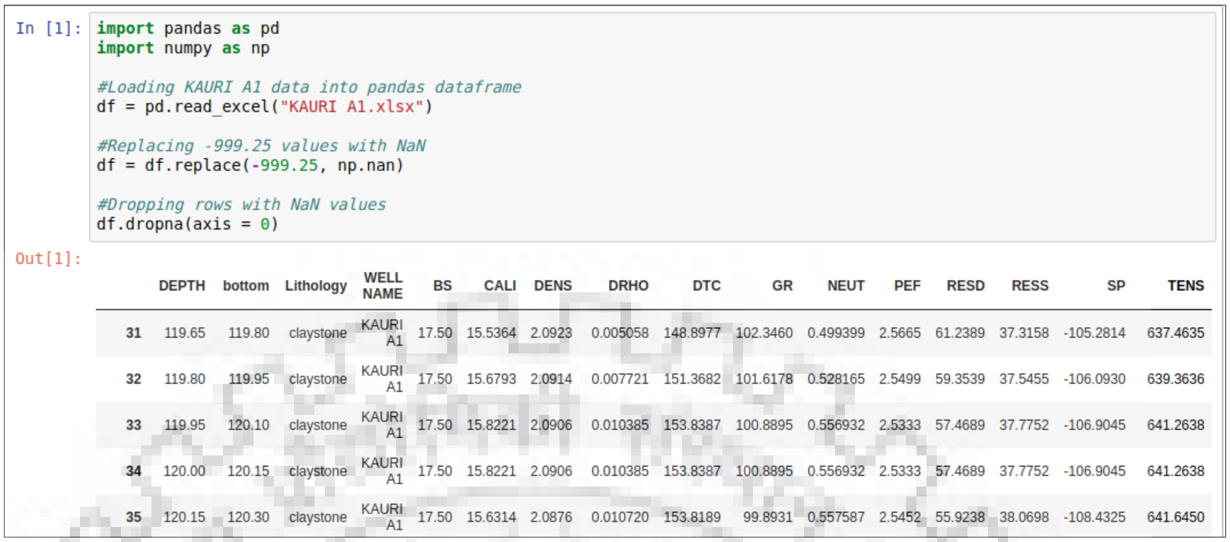
```
DEPTH   bottom  Lithology  WELL NAME  BS    CALI    DENS   DRHO      DTC       GR        NEUT      PEF    RESD    RESS    SP        TENS
119.65  119.8   claystone  KAURI A1   17.5  15.5364 2.0923 0.005058  148.8977  102.346   0.499399  2.5665 61.2389 37.3158 -105.2814 637.4635
119.8   119.95  claystone  KAURI A1   17.5  15.6793 2.0914 0.007721  151.3682  101.6178  0.528165  2.5499 59.3539 37.5455 -106.093  639.3636
119.95  120.1   claystone  KAURI A1   17.5  15.8221 2.0906 0.010385  153.8387  100.8895  0.556932  2.5333 57.4689 37.7752 -106.9045 641.2638
120.0   120.15  claystone  KAURI A1   17.5  15.8221 2.0906 0.010385  153.8387  100.8895  0.556932  2.5333 57.4689 37.7752 -106.9045 641.2638
120.15  120.3   claystone  KAURI A1   17.5  15.6314 2.0876 0.01072   153.8189  99.8931   0.557587  2.5452 55.9238 38.0698 -108.4325 641.645
120.3   120.45  claystone  KAURI A1   17.5  15.2169 2.0832 0.009491  152.128   98.7167   0.539377  2.5762 54.6069 38.4079 -110.4411 641.0067
120.45  120.6   claystone  KAURI A1   17.5  14.8023 2.0788 0.008263  150.4371  97.5402   0.521167  2.6073 53.29   38.7459 -112.4498 640.3685
120.6   120.75  claystone  KAURI A1   17.5  14.4889 2.0833 0.008189  150.8454  97.2309   0.504141  2.6347 51.6809 38.8303 -114.1391 640.9531
120.75  120.9   claystone  KAURI A1   17.5  14.3929 2.1069 0.010594  155.7642  98.7849   0.489657  2.6543 49.4441 38.3697 -115.1424 644.165
120.9   121.05  claystone  KAURI A1   17.5  14.2969 2.1305 0.012999  160.683   100.3389  0.475173  2.6739 47.2072 37.909  -116.1456 647.377
121.05  121.2   claystone  KAURI A1   17.5  14.2096 2.1527 0.015116  165.3333  101.8378  0.461859  2.6931 45.1301 37.4883 -117.1247 650.479
121.2   121.35  claystone  KAURI A1   17.5  14.3519 2.1367 0.009697  162.9454  101.8942  0.479206  2.7011 47.2383 38.1166 -117.471  650.6978
121.35  121.5   claystone  KAURI A1   17.5  14.4942 2.1207 0.004277  160.5574  101.9506  0.496554  2.7091 49.3465 38.7449 -117.8172 650.9166
121.5   121.65  claystone  KAURI A1   17.5  14.6366 2.1047 -0.001142 158.1695  102.007   0.513901  2.7172 51.4548 39.3732 -118.1634 651.1353
121.65  121.8   claystone  KAURI A1   17.5  14.7845 2.0986 -0.00267  158.9496  103.2529  0.515944  2.7081 55.0401 40.2096 -118.5373 655.3004
121.8   121.95  claystone  KAURI A1   17.5  14.9343 2.0956 -0.002941 160.7527  104.8828  0.513044  2.6936 59.1025 41.1132 -118.9202 660.7396
121.95  122.1   claystone  KAURI A1   17.5  15.084  2.0925 -0.003212 162.5557  106.5128  0.510145  2.679  63.1648 42.0167 -119.303  666.179
122.1   122.25  claystone  KAURI A1   17.5  15.1959 2.093  -0.003158 163.2033  106.8015  0.500886  2.6592 64.4106 41.6887 -119.5209 665.6046
122.25  122.4   claystone  KAURI A1   17.5  15.2659 2.0972 -0.002745 162.574   105.6083  0.4846    2.6335 62.544  39.9996 -119.5565 658.3854
122.4   122.55  claystone  KAURI A1   17.5  15.3358 2.1014 -0.002333 161.9447  104.415   0.468315  2.6078 60.6774 38.3106 -119.5921 651.1661
122.55  122.7   claystone  KAURI A1   17.5  15.3884 2.1047 -0.001803 161.3509  103.5322  0.453498  2.5904 59.0405 36.7821 -119.7124 644.792
122.7   122.85  claystone  KAURI A1   17.5  15.3686 2.1043 -0.000786 160.9044  103.9371  0.444777  2.6075 58.3568 35.9195 -120.1839 641.924
122.85  123.0   claystone  KAURI A1   17.5  15.3488 2.104  0.000231  160.4578  104.342   0.436056  2.6245 57.6732 35.0569 -120.6554 639.0559
123.0   123.15  claystone  KAURI A1   17.5  15.329  2.1036 0.001248  160.0113  104.7469  0.427335  2.6415 56.9895 34.1943 -121.127  636.1879
```

Figure 4.6: Wireline dataset tagged with lithology information

4.4 Data Preprocessing:

- **Assumptions, Delimitations and Limitations**
  - Uncertainties in logging measurements.
  - Most well logs include basic corrections through tool calibrations and environmental corrections for known systematic errors. Because of the statistical nature of most measurements and the complexity of the borehole environment, there will remain some uncertainty, especially as the corrections applied cannot be quantified to eliminate all the errors completely.
  - Problems with acquisition in wireline logging, such as total or partial failure of tools, bad borehole conditions, and poor choice of logging suites, either due to the availability of tools or acquisition costs, may affect logging runs and cause uncertainty in logging data.

- **Handling NULL values**
  - There are always few null values in any real-world dataset. Whether it's a regression, classification or any other kind of problem, it doesn't really matter, no model can handle these NULL or NaN values alone, so a solution for this need to be applied.
  - After merging the wireline and mud log dataframe, we discarded the depth intervals during which tool was not recording i.e. have -999.25 value in the dataset.
  - Mostly these NAN value intervals lies at the beginning or towards the end of log operation.



```
In [1]: import pandas as pd
        import numpy as np

        #Loading KAURI A1 data into pandas dataframe
        df = pd.read_excel("KAURI A1.xlsx")

        #Replacing -999.25 values with NaN
        df = df.replace(-999.25, np.nan)

        #Dropping rows with NaN values
        df.dropna(axis = 0)
```

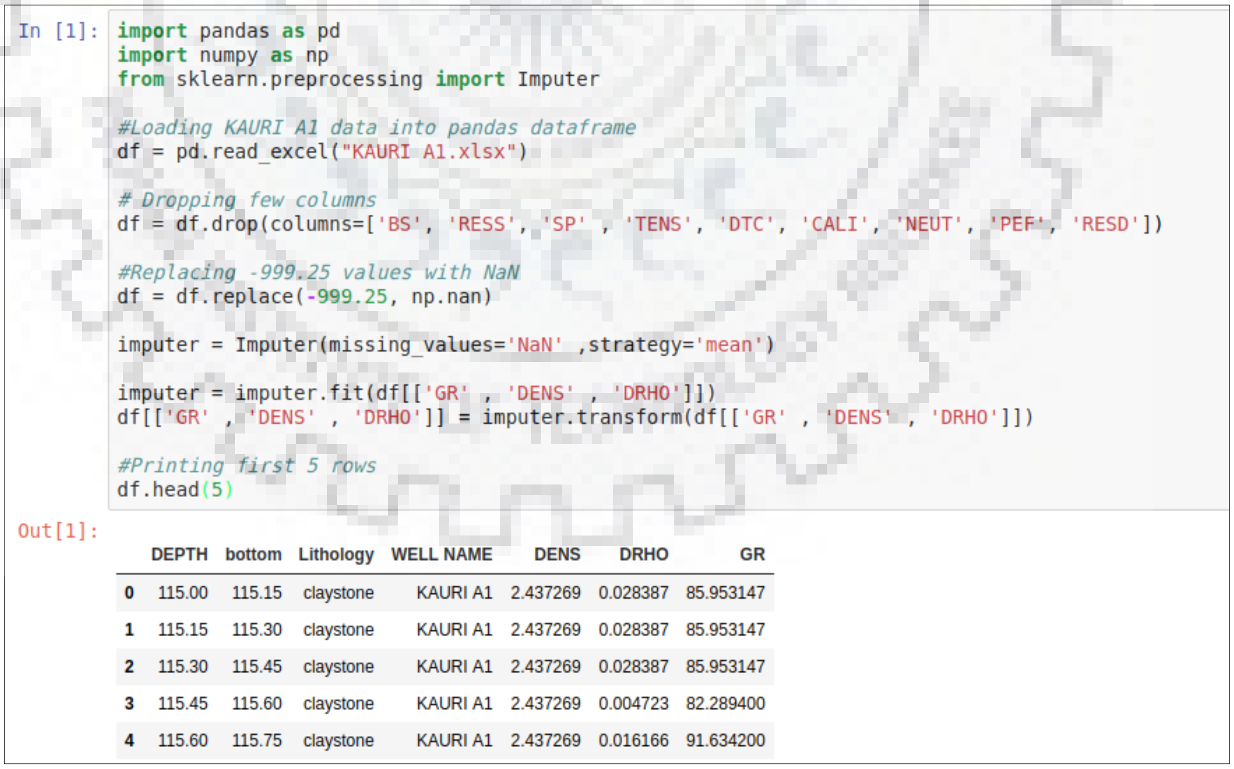
Out[1]:

	DEPTH	bottom	Lithology	WELL NAME	BS	CALI	DENS	DRHO	DTC	GR	NEUT	PEF	RESD	RESS	SP	TENS
31	119.65	119.80	claystone	KAURI A1	17.50	15.5364	2.0923	0.005058	148.8977	102.3460	0.499399	2.5665	61.2389	37.3158	-105.2814	637.4635
32	119.80	119.95	claystone	KAURI A1	17.50	15.6793	2.0914	0.007721	151.3682	101.6178	0.528165	2.5499	59.3539	37.5455	-106.0930	639.3636
33	119.95	120.10	claystone	KAURI A1	17.50	15.8221	2.0906	0.010385	153.8387	100.8895	0.556932	2.5333	57.4689	37.7752	-106.9045	641.2638
34	120.00	120.15	claystone	KAURI A1	17.50	15.8221	2.0906	0.010385	153.8387	100.8895	0.556932	2.5333	57.4689	37.7752	-106.9045	641.2638
35	120.15	120.30	claystone	KAURI A1	17.50	15.6314	2.0876	0.010720	153.8189	99.8931	0.557587	2.5452	55.9238	38.0698	-108.4325	641.6450

Figure 4.7: Handling NULL values in the dataset

- ## Imputation

  Imputation is simply the process of substituting the missing values of our datasets. It can be achieved either by defining own customized function or simple imputing technique implementation using the sklearn Imputer class.



```
In [1]: import pandas as pd
        import numpy as np
        from sklearn.preprocessing import Imputer

        #Loading KAURI A1 data into pandas dataframe
        df = pd.read_excel("KAURI A1.xlsx")

        # Dropping few columns
        df = df.drop(columns=['BS', 'RESS', 'SP' , 'TENS', 'DTC', 'CALI', 'NEUT', 'PEF', 'RESD'])

        #Replacing -999.25 values with NaN
        df = df.replace(-999.25, np.nan)

        imputer = Imputer(missing_values='NaN' ,strategy='mean')

        imputer = imputer.fit(df[['GR' , 'DENS' , 'DRHO']])
        df[['GR' , 'DENS' , 'DRHO']] = imputer.transform(df[['GR' , 'DENS' , 'DRHO']])

        #Printing first 5 rows
        df.head(5)
```

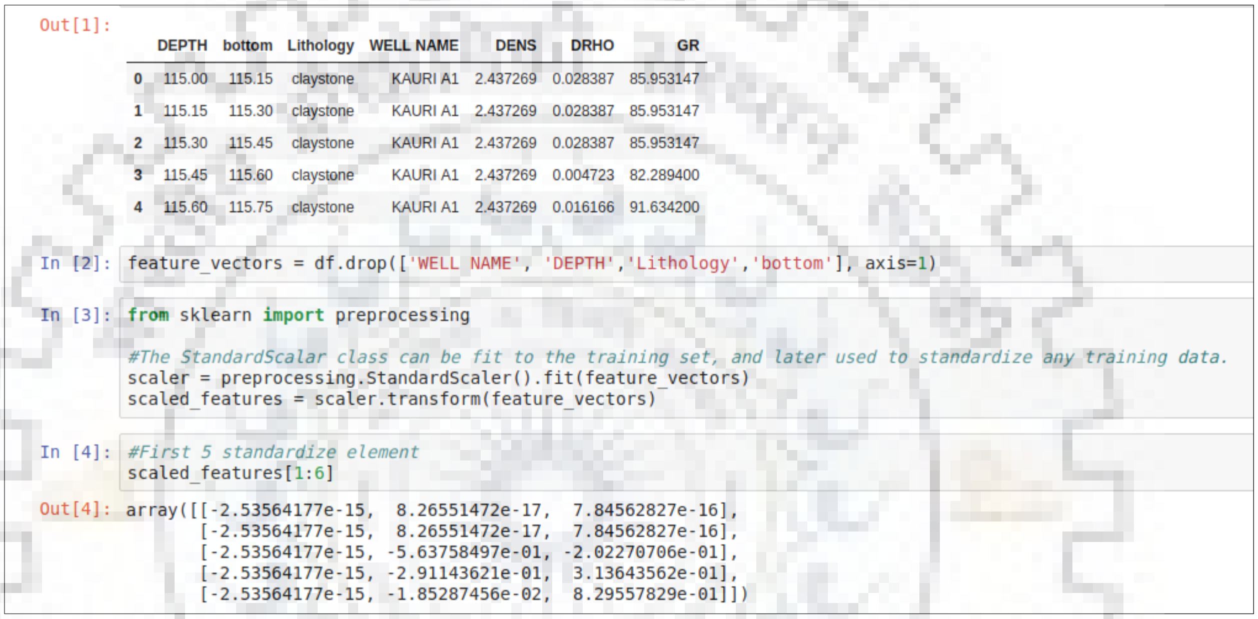
Out[1]:

	DEPTH	bottom	Lithology	WELL NAME	DENS	DRHO	GR
0	115.00	115.15	claystone	KAURI A1	2.437269	0.028387	85.953147
1	115.15	115.30	claystone	KAURI A1	2.437269	0.028387	85.953147
2	115.30	115.45	claystone	KAURI A1	2.437269	0.028387	85.953147
3	115.45	115.60	claystone	KAURI A1	2.437269	0.004723	82.289400
4	115.60	115.75	claystone	KAURI A1	2.437269	0.016166	91.634200

Figure 4.8: Applying data Imputation technique to the dataset

- **Standardization**

  Standardization transform the dataset values in such a way that the mean value (μ) is 0 and the standard deviation(SD) is 1. It can be achieved by calculating the mean and SD of the values, then subtracting the mean from each data point and then dividing it by standard deviation. Sklearn library provides a function called **StandardScaler.**



```
Out[1]:
```

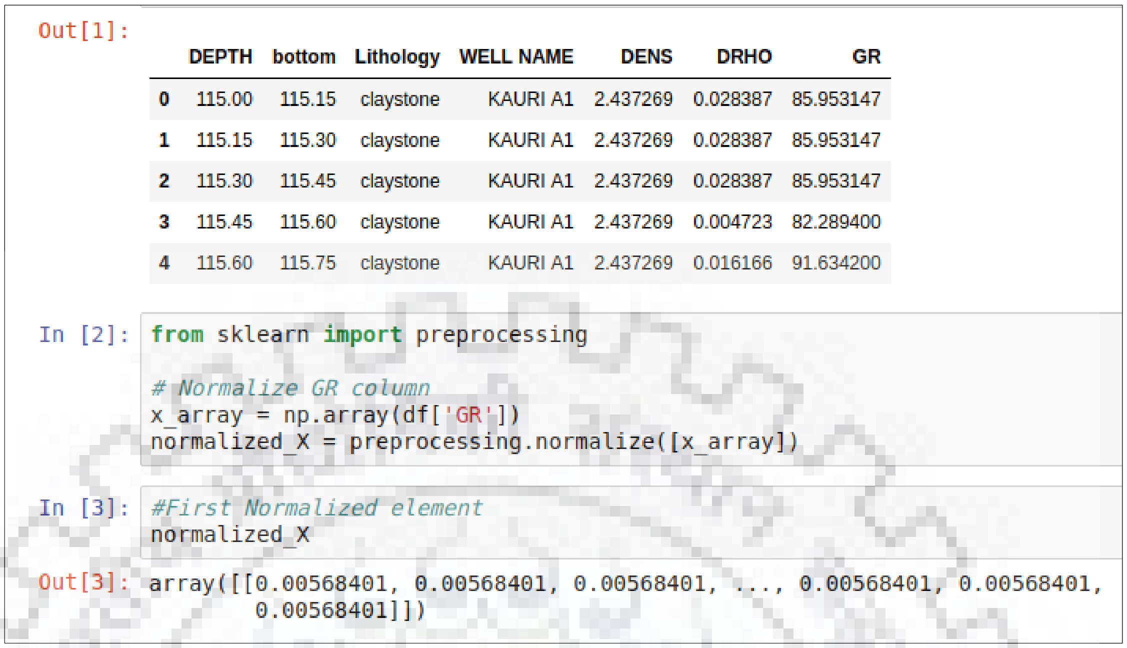
	DEPTH	bottom	Lithology	WELL NAME	DENS	DRHO	GR
0	115.00	115.15	claystone	KAURI A1	2.437269	0.028387	85.953147
1	115.15	115.30	claystone	KAURI A1	2.437269	0.028387	85.953147
2	115.30	115.45	claystone	KAURI A1	2.437269	0.028387	85.953147
3	115.45	115.60	claystone	KAURI A1	2.437269	0.004723	82.289400
4	115.60	115.75	claystone	KAURI A1	2.437269	0.016166	91.634200

```
In [2]: feature_vectors = df.drop(['WELL NAME', 'DEPTH','Lithology','bottom'], axis=1)

In [3]: from sklearn import preprocessing

        #The StandardScalar class can be fit to the training set, and later used to standardize any training data.
        scaler = preprocessing.StandardScaler().fit(feature_vectors)
        scaled_features = scaler.transform(feature_vectors)

In [4]: #First 5 standardize element
        scaled_features[1:6]

Out[4]: array([[-2.53564177e-15,  8.26551472e-17,  7.84562827e-16],
               [-2.53564177e-15,  8.26551472e-17,  7.84562827e-16],
               [-2.53564177e-15, -5.63758497e-01, -2.02270706e-01],
               [-2.53564177e-15, -2.91143621e-01,  3.13643562e-01],
               [-2.53564177e-15, -1.85287456e-02,  8.29557829e-01]])
```

Figure 4.9: Applying data Standardization technique to the dataset

- **Normalization**

  Standardization also often simply called Min-Max scaling essentially shrinks the data range so that the range is set between 0 and 1 (or -1 to 1 if negative values are present). It works better than standardization technique for the case where the distribution isn’t Gaussian or the SD is very small.


```
Out[1]:
```

	DEPTH	bottom	Lithology	WELL NAME	DENS	DRHO	GR
0	115.00	115.15	claystone	KAURI A1	2.437269	0.028387	85.953147
1	115.15	115.30	claystone	KAURI A1	2.437269	0.028387	85.953147
2	115.30	115.45	claystone	KAURI A1	2.437269	0.028387	85.953147
3	115.45	115.60	claystone	KAURI A1	2.437269	0.004723	82.289400
4	115.60	115.75	claystone	KAURI A1	2.437269	0.016166	91.634200

```
In [2]: from sklearn import preprocessing

        # Normalize GR column
        x_array = np.array(df['GR'])
        normalized_X = preprocessing.normalize([x_array])

In [3]: #First Normalized element
        normalized_X

Out[3]: array([[0.00568401, 0.00568401, 0.00568401, ..., 0.00568401, 0.00568401,
                0.00568401]])
```

Figure 4.10: Applying data Normalization technique to the dataset

4.5 Final Dataset

The final dataset has in total 8 wells from the Kauri region of Taranaki Basin. Out of these 8, seven wells were used for training purpose and 1 well was used for validation purpose.

- Training wells: A1, A2, C1, E3, E6, E7, E8
- Testing well: E9

In Figure 4.11 to Figure 4.18, the wireline logs are plotted alongside the lithofacies associated with them for that particular depth.

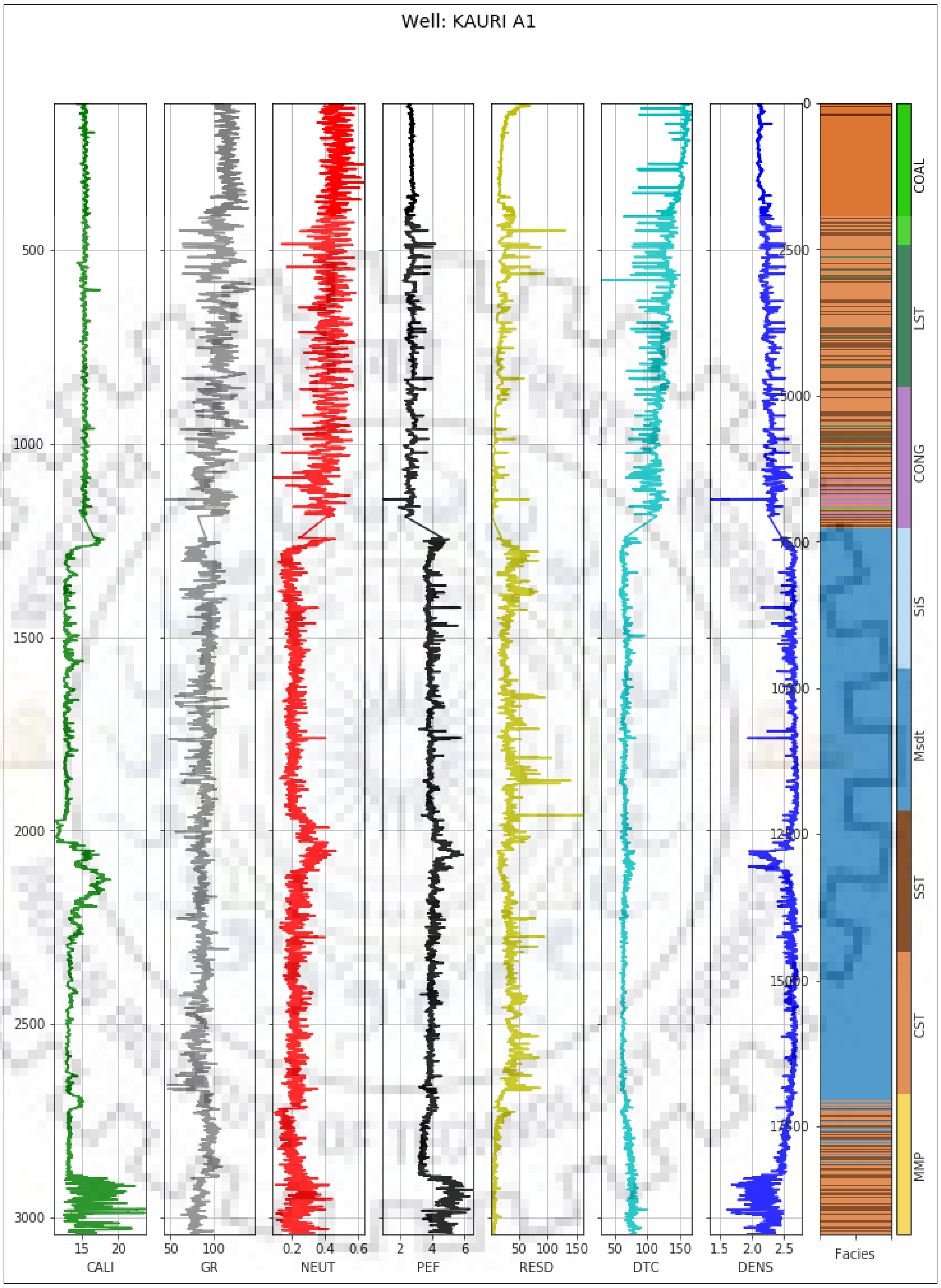


Figure 4.11: Well KAURI A1 having wireline and lithology information

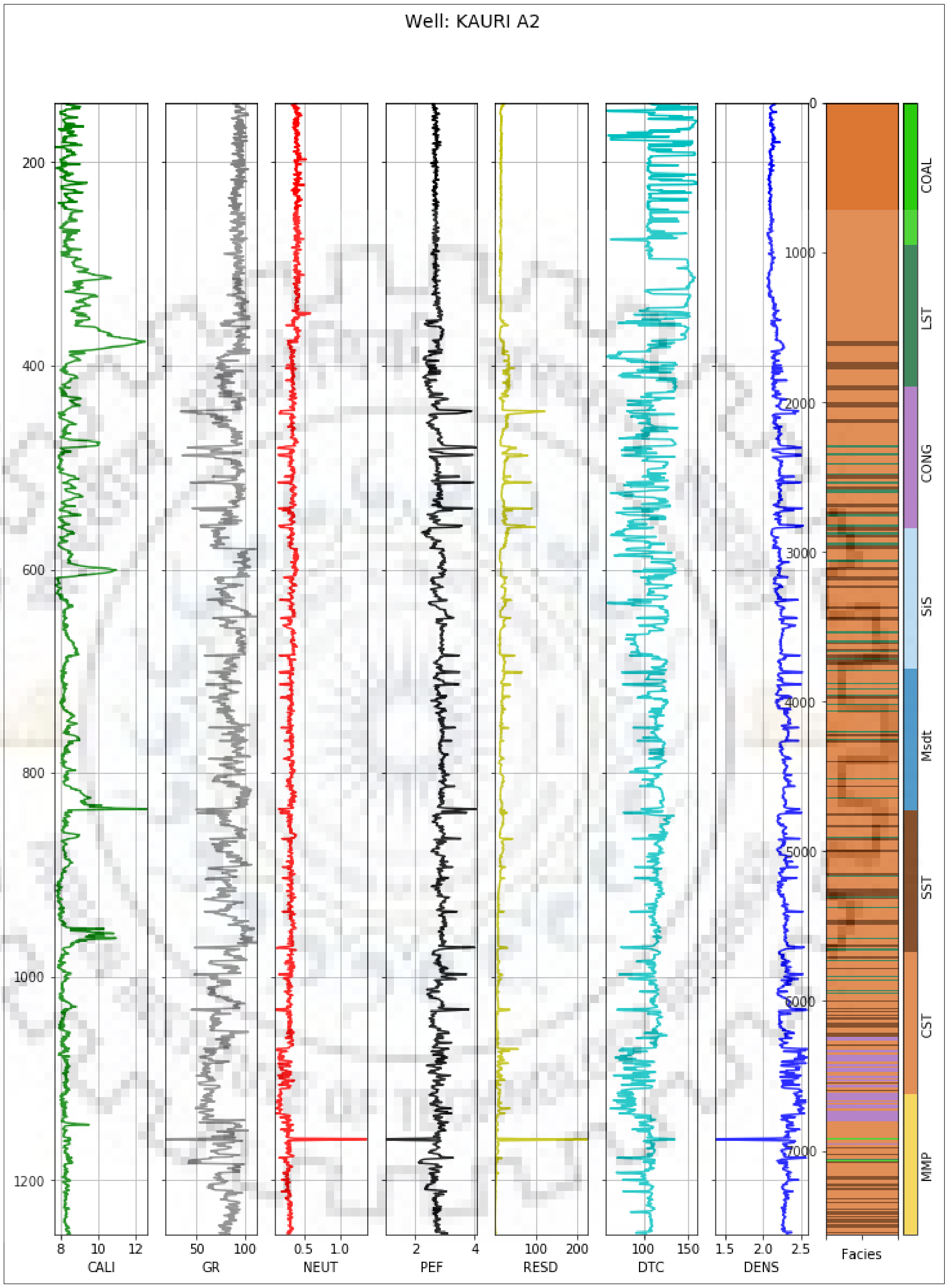


Figure 4.12: Well KAURI A2 having wireline and lithology information

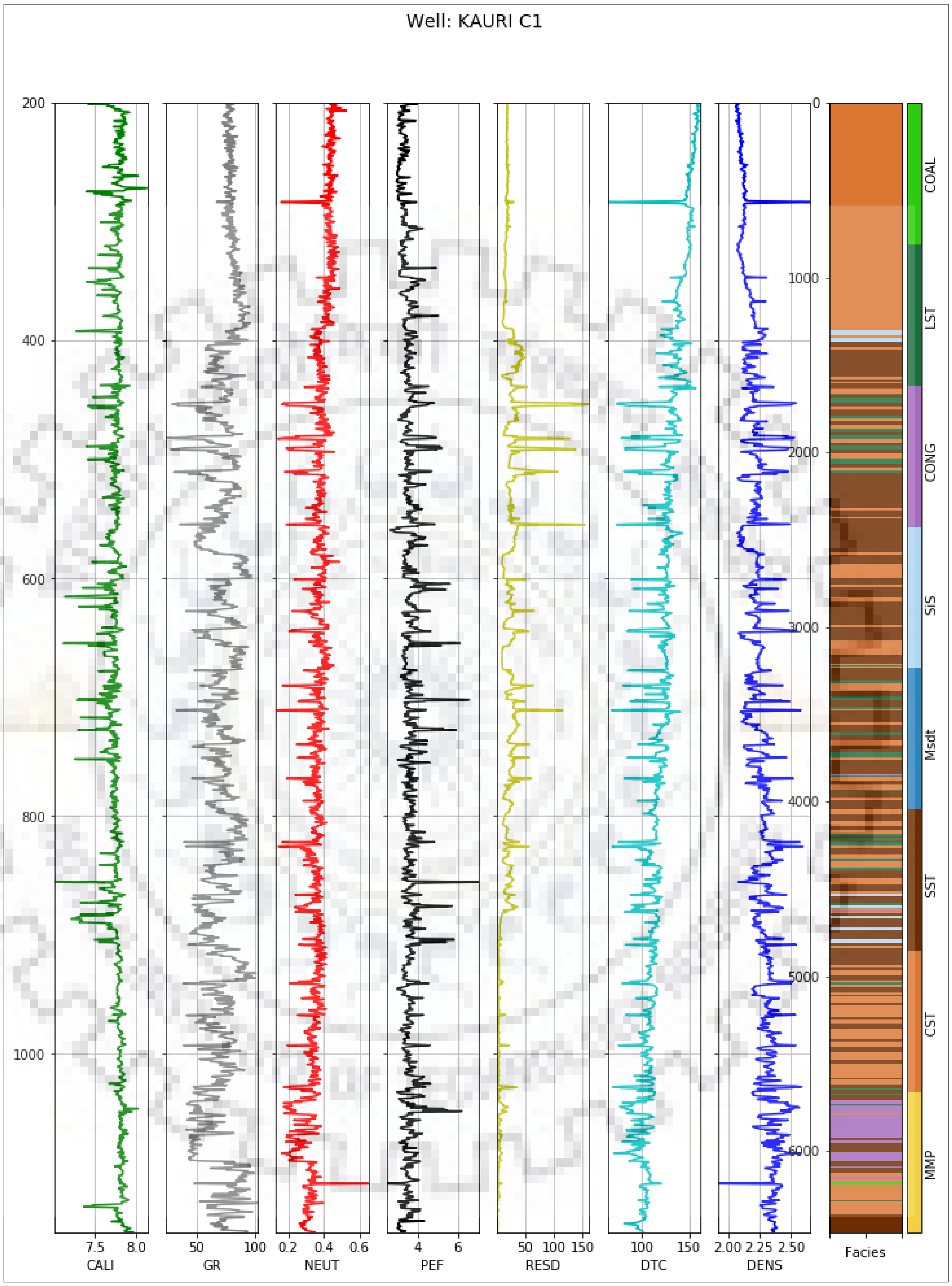


Figure 4.13: Well KAURI C1 having wireline and lithology information

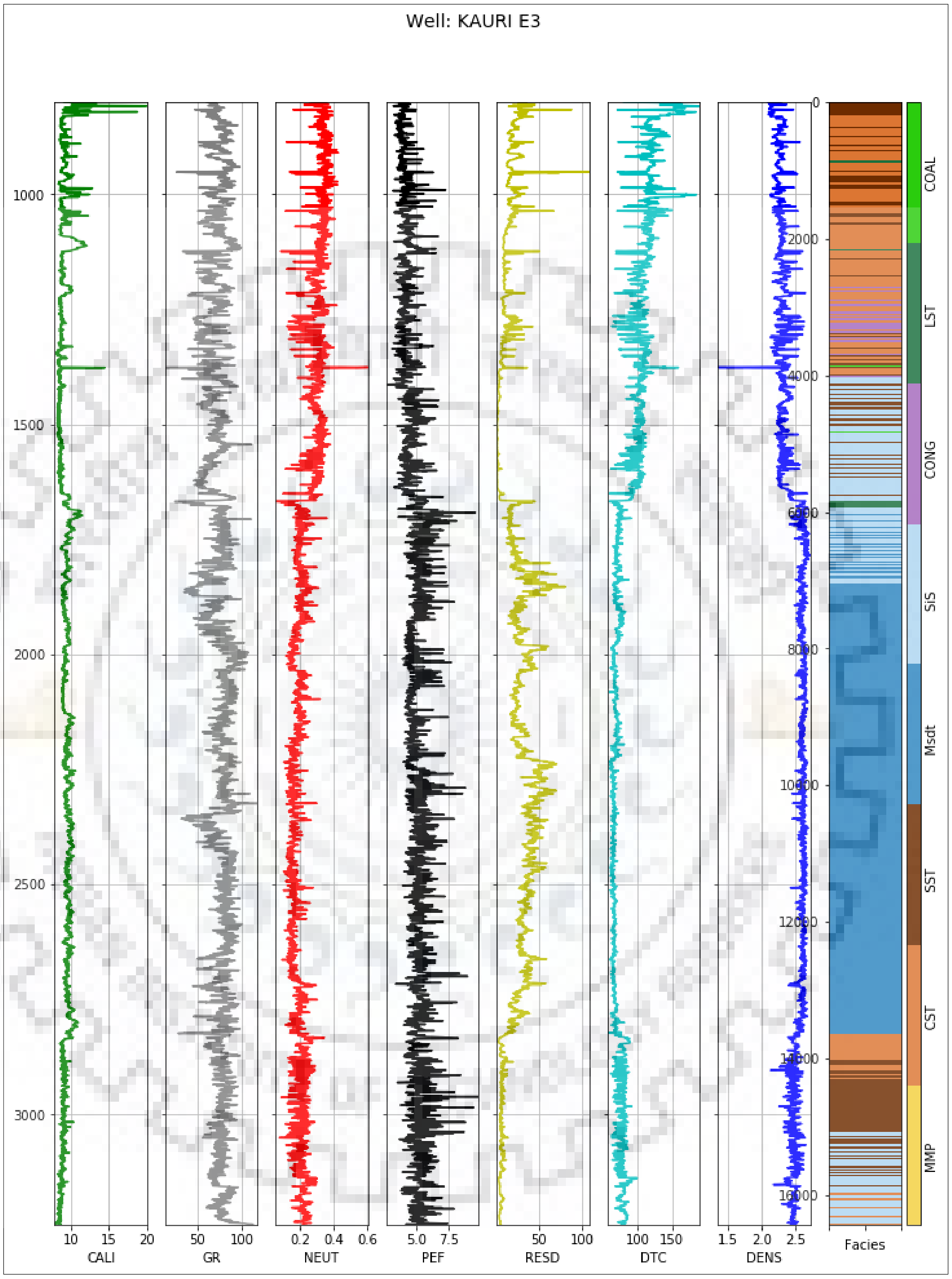


Figure 4.14: Well KAURI E3 having wireline and lithology information

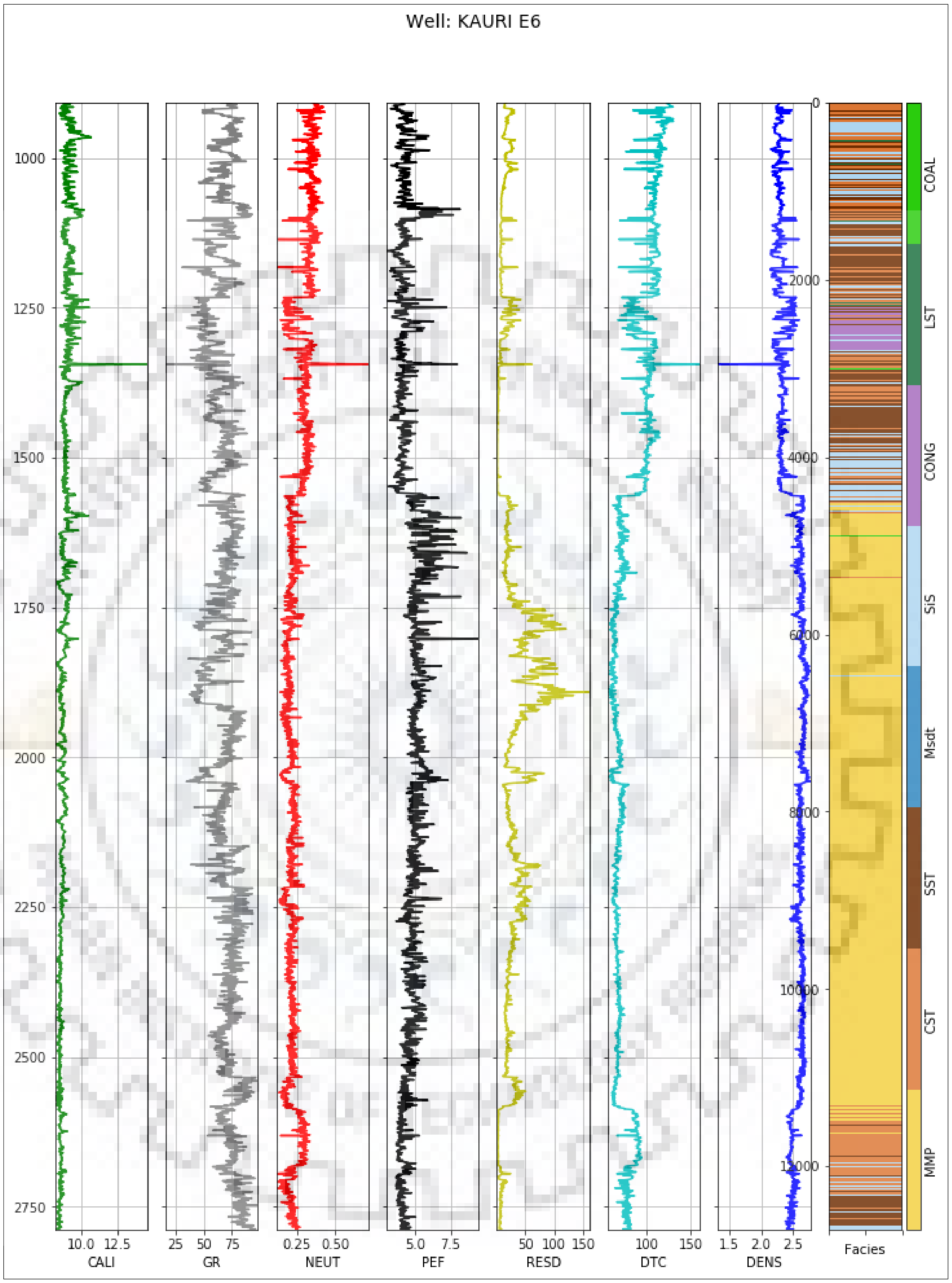


Figure 4.15: Well KAURI E6 having wireline and lithology information

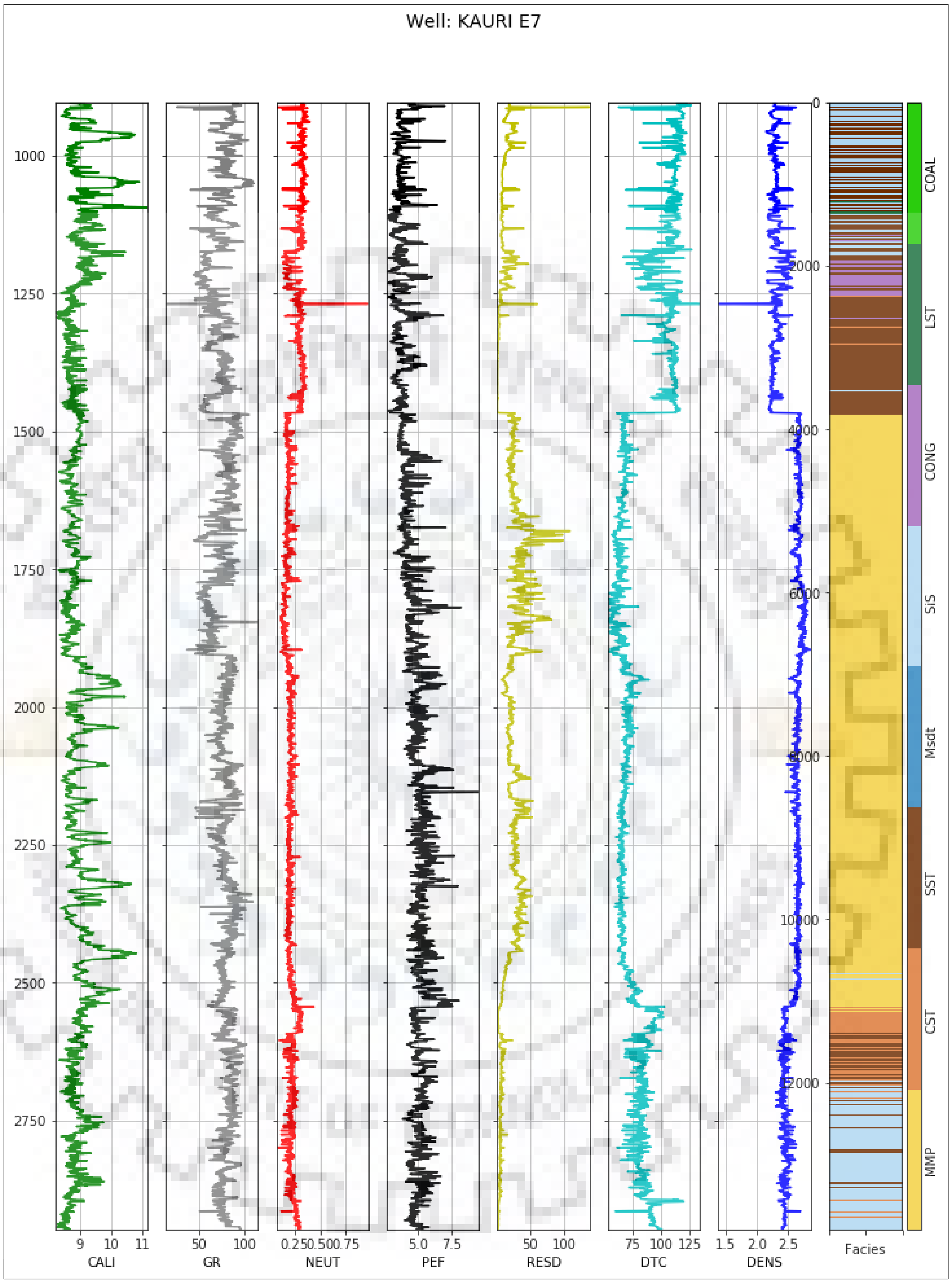


Figure 4.16: Well KAURI E7 having wireline and lithology information

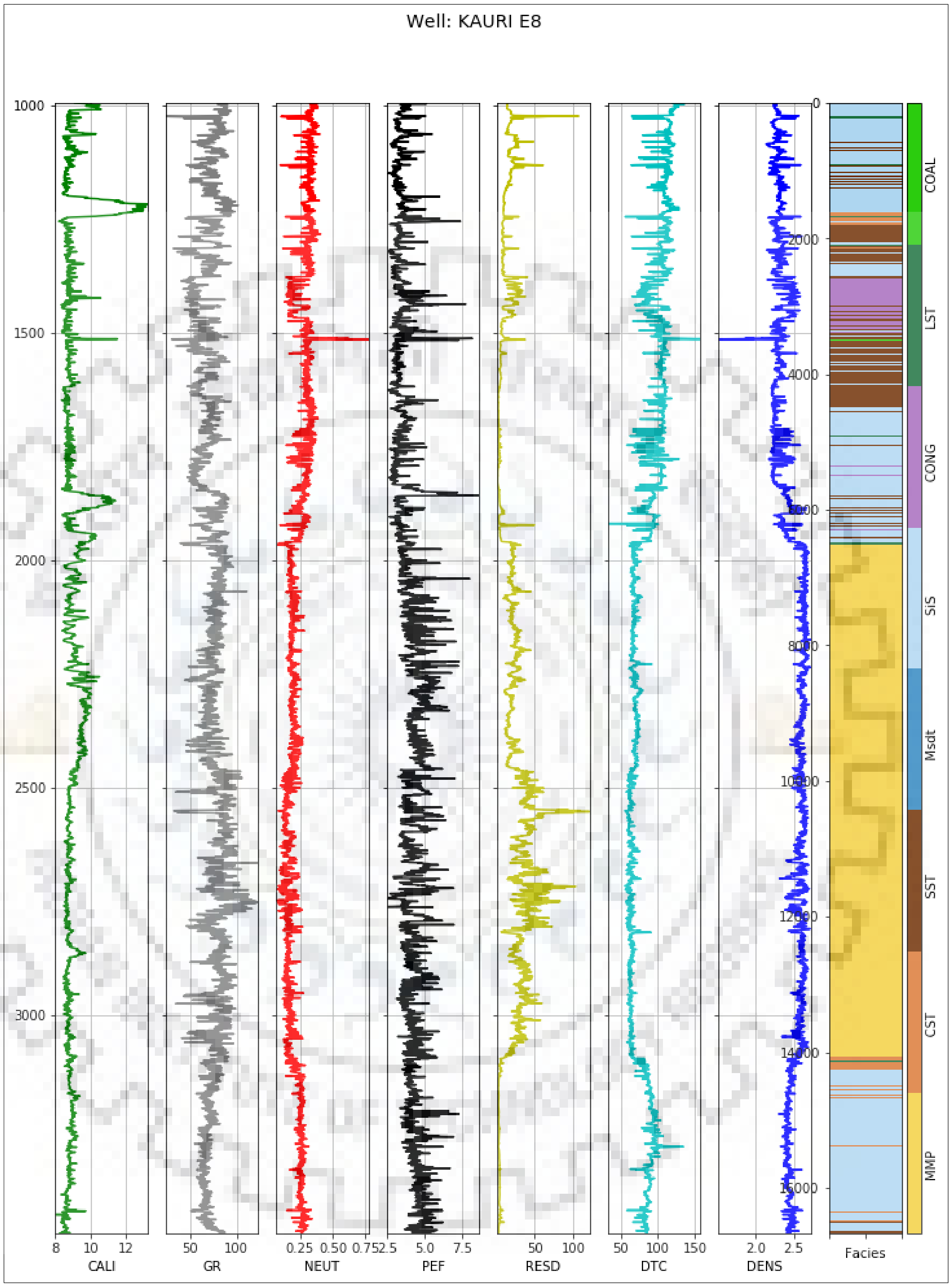


Figure 4.17: Well KAURI E8 having wireline and lithology information

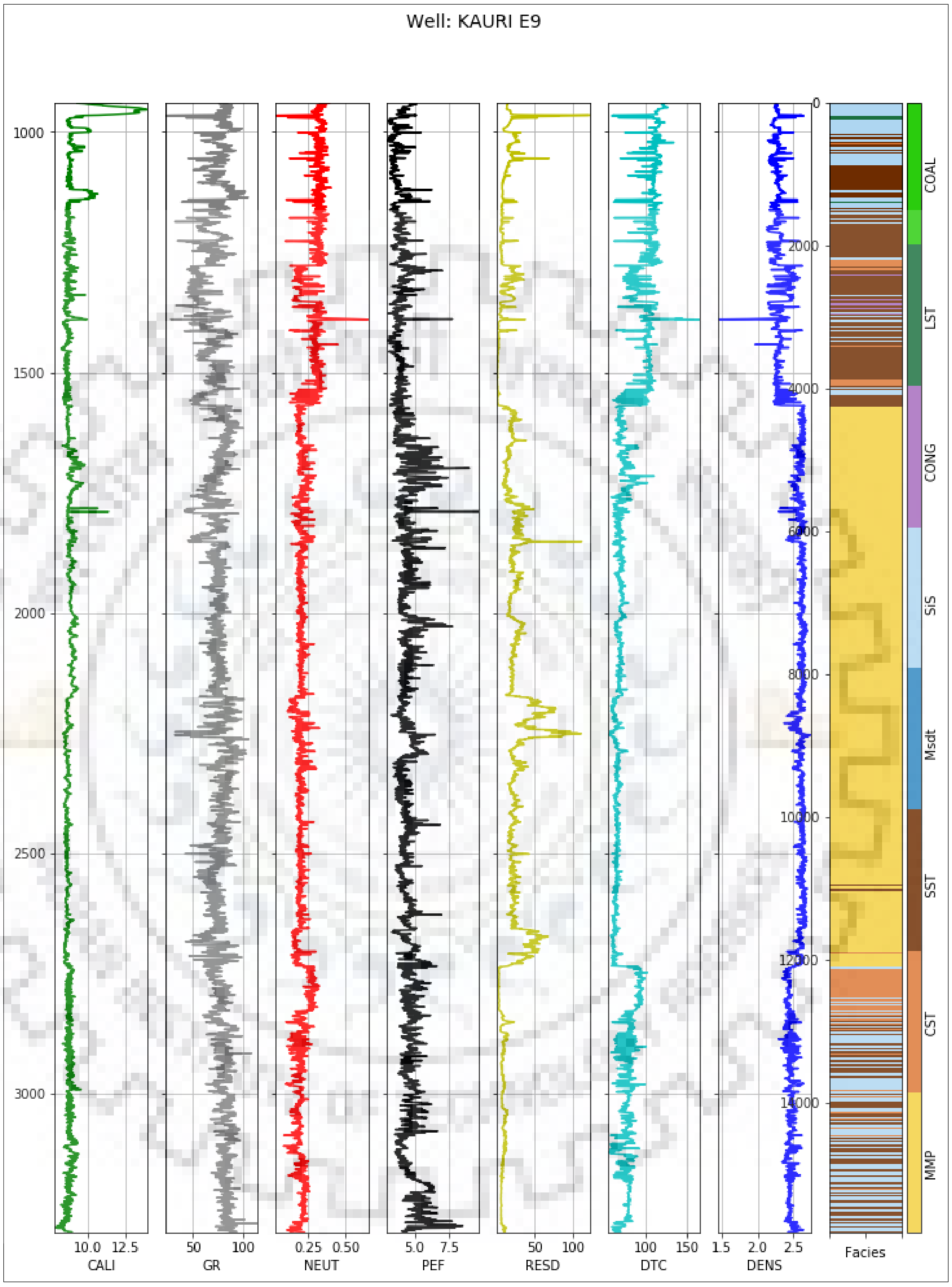


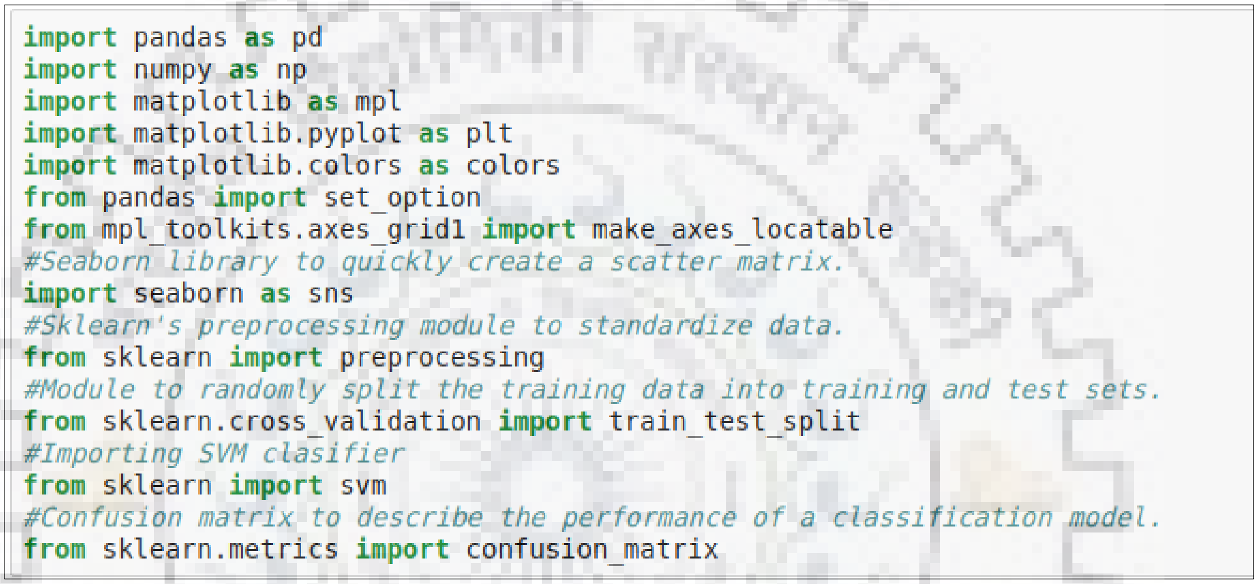
Figure 4.18: Well KAURI E9 having wireline and lithology information

Chapter 5: Methodology

5.1 Support Vector Machines (SVM)

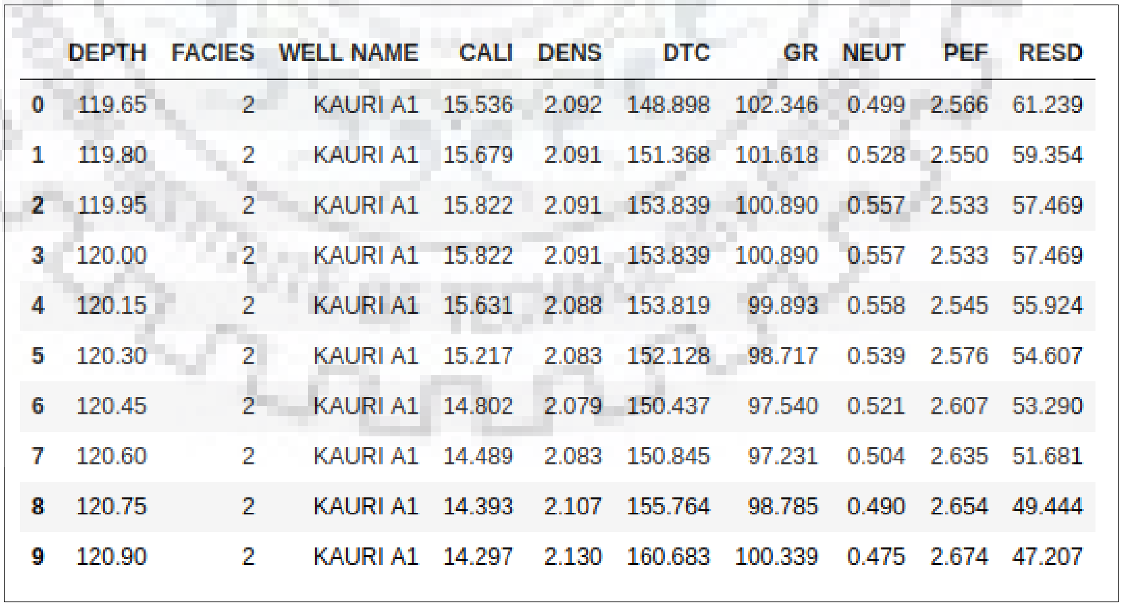
- **Exploring the Dataset**

  First, Import the python libraries



```python
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.colors as colors
from pandas import set_option
from mpl_toolkits.axes_grid1 import make_axes_locatable
#Seaborn library to quickly create a scatter matrix.
import seaborn as sns
#Sklearn's preprocessing module to standardize data.
from sklearn import preprocessing
#Module to randomly split the training data into training and test sets.
from sklearn.cross_validation import train_test_split
#Importing SVM clasifier
from sklearn import svm
#Confusion matrix to describe the performance of a classification model.
from sklearn.metrics import confusion_matrix
```

Figure 5.1: Importing python libraries for SVM classification model



	DEPTH	FACIES	WELL NAME	CALI	DENS	DTC	GR	NEUT	PEF	RESD
0	119.65	2	KAURI A1	15.536	2.092	148.898	102.346	0.499	2.566	61.239
1	119.80	2	KAURI A1	15.679	2.091	151.368	101.618	0.528	2.550	59.354
2	119.95	2	KAURI A1	15.822	2.091	153.839	100.890	0.557	2.533	57.469
3	120.00	2	KAURI A1	15.822	2.091	153.839	100.890	0.557	2.533	57.469
4	120.15	2	KAURI A1	15.631	2.088	153.819	99.893	0.558	2.545	55.924
5	120.30	2	KAURI A1	15.217	2.083	152.128	98.717	0.539	2.576	54.607
6	120.45	2	KAURI A1	14.802	2.079	150.437	97.540	0.521	2.607	53.290
7	120.60	2	KAURI A1	14.489	2.083	150.845	97.231	0.504	2.635	51.681
8	120.75	2	KAURI A1	14.393	2.107	155.764	98.785	0.490	2.654	49.444
9	120.90	2	KAURI A1	14.297	2.130	160.683	100.339	0.475	2.674	47.207

Figure 5.2: Training dataset

● Predictor variables:

The variables which will be used to define the model. Based on the values of these predictor variables, SVM assign a class label i.e. facies type to a particular data point.

- **Caliper_log (CALI)**
- **Density_log (DENS)**
- **Sonic_log (DTC)**
- **Gamma_ray_log (GR)**
- **Neutron_log (NEUT)**
- **Photoelectric_log (PEF)**
- **Resistivity_log (RESD)**

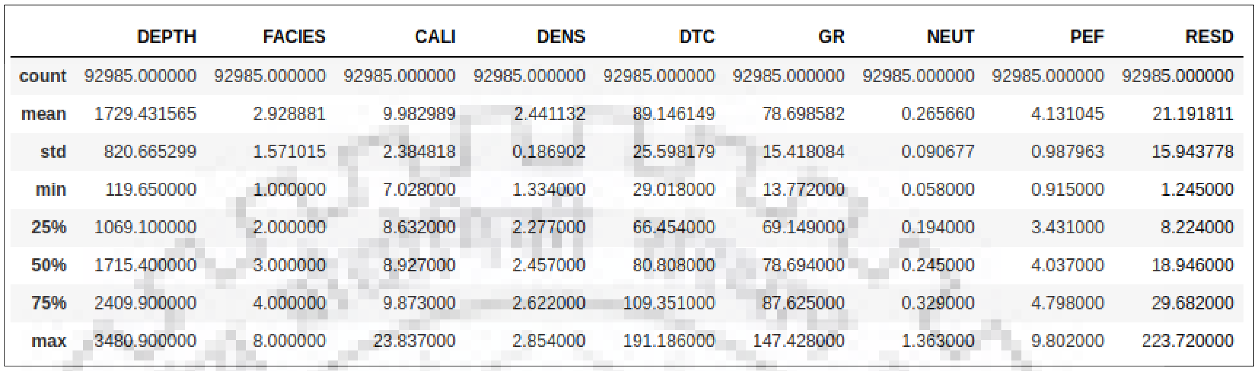
● Facies classes and their label:

Here the dataset has 8 lithofacies type and the label will be assigned to each facies type in order to use it as a predicted variable.

Facies	**Label**	**Abbreviation**
1	METAMORPHIC	MMP
2	CLAYSTONE	CST
3	SANDSTONE	SST
4	METASEDIMENT	Msdt
5	SILTSTONE	SiS
6	CONGLOMERATE	CONG
7	LIMESTONE	LST
8	COAL	COAL

Table 5.1: Facies and their corresponding labels

Figure 5.3 describe the statistical distribution of the training dataset. Here, it can be visualized that no NAN value is present in the database, maximum and minimum value of each log, their mean and standard distribution.



	DEPTH	FACIES	CALI	DENS	DTC	GR	NEUT	PEF	RESD
count	92985.000000	92985.000000	92985.000000	92985.000000	92985.000000	92985.000000	92985.000000	92985.000000	92985.000000
mean	1729.431565	2.928881	9.982989	2.441132	89.146149	78.698582	0.265660	4.131045	21.191811
std	820.665299	1.571015	2.384818	0.186902	25.598179	15.418084	0.090677	0.987963	15.943778
min	119.650000	1.000000	7.028000	1.334000	29.018000	13.772000	0.058000	0.915000	1.245000
25%	1069.100000	2.000000	8.632000	2.277000	66.454000	69.149000	0.194000	3.431000	8.224000
50%	1715.400000	3.000000	8.927000	2.457000	80.808000	78.694000	0.245000	4.037000	18.946000
75%	2409.900000	4.000000	9.873000	2.622000	109.351000	87.625000	0.329000	4.798000	29.682000
max	3480.900000	8.000000	23.837000	2.854000	191.186000	147.428000	1.363000	9.802000	223.720000

Figure 5.3: Statistical Distribution of the Training Data

Figure 5.4 shows the distribution of different facies in the training dataset. It can be easily inferred by visual analysis that the data is skewed and have very few example of COAL lithology as compared with the other lithofacies.

MMP: 21602 , CST: 21213, SST: 16059 , Msdt: 16892,
SiS: 12689, CONG: 2744, LST: 1628, COAL: 158

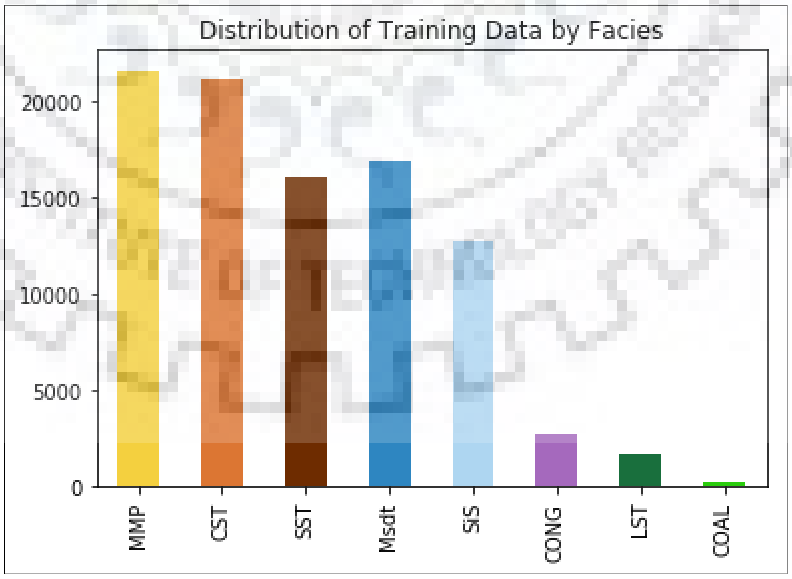


Figure 5.4: Distribution of the Training Data by Facies

Figure 5.5 shows the distribution of different facies in the validation dataset. This dataset has ZERO example of Metasediment (Msdt) and very large number of Metamorphic (MMP) facies example. This type of distribution very heavily affects the prediction ability of the classification model.

MMP: 7771 , CST: 1080, SST: 3824 , Msdt: 0,
SiS: 2804, CONG: 115, LST: 83, COAL: 15

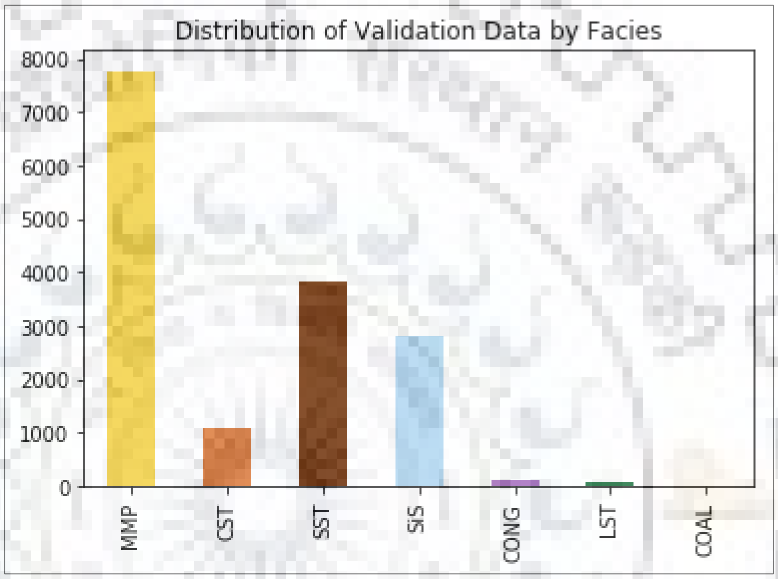


Figure 5.5: Distribution of the Validation Data by Facies

- **Crossplots:**

  Crossplots enable us to visualize variation of two properties with respect to the facies type. This training dataset has 7 log variables, and by plotting them in the form of scatter matrix, it become easy to visualization the variation between any two log values w.r.t. facies.

  Figure 5.6 provide the visualization utility to analyze log value variation with changing lithology.

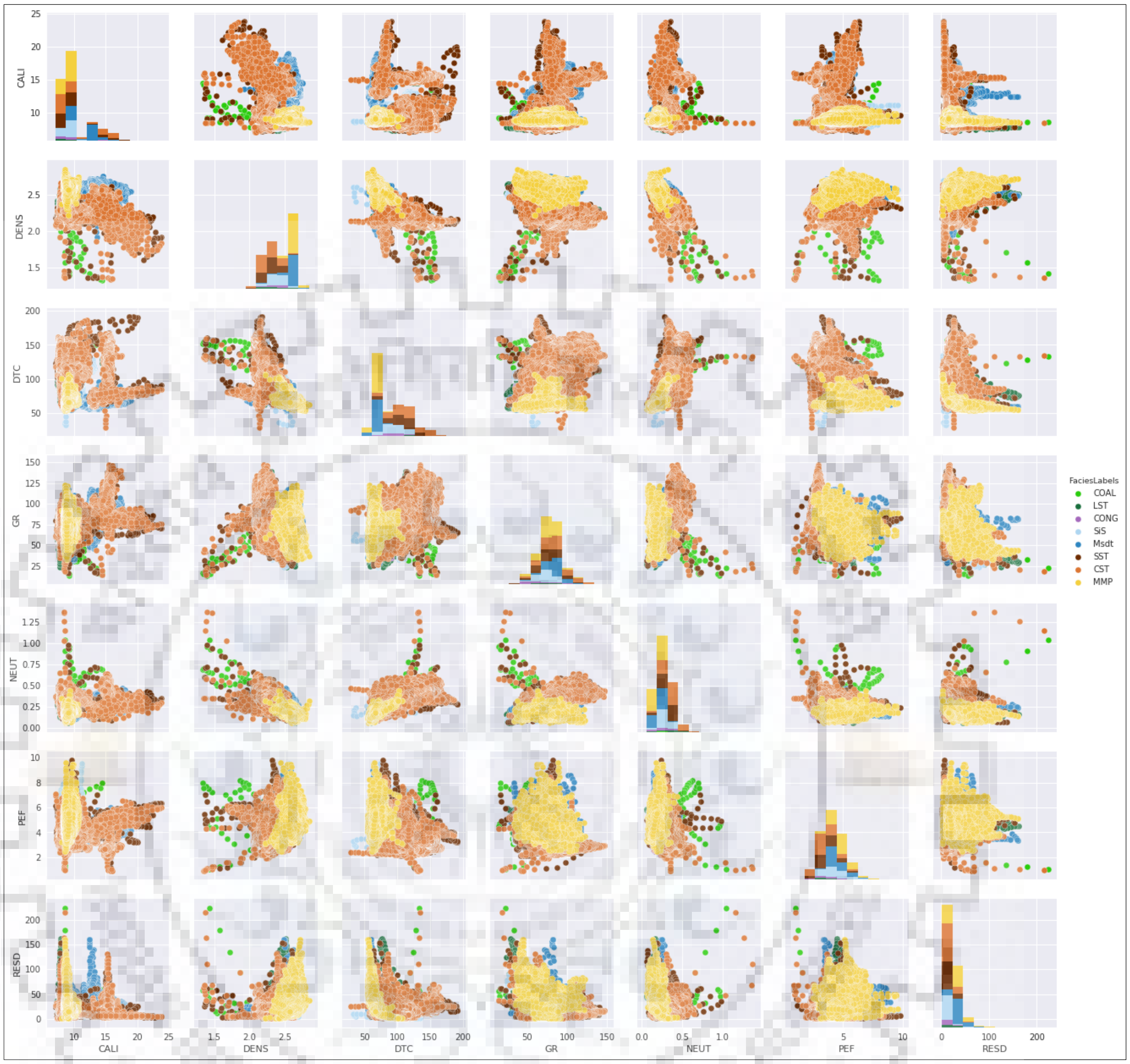
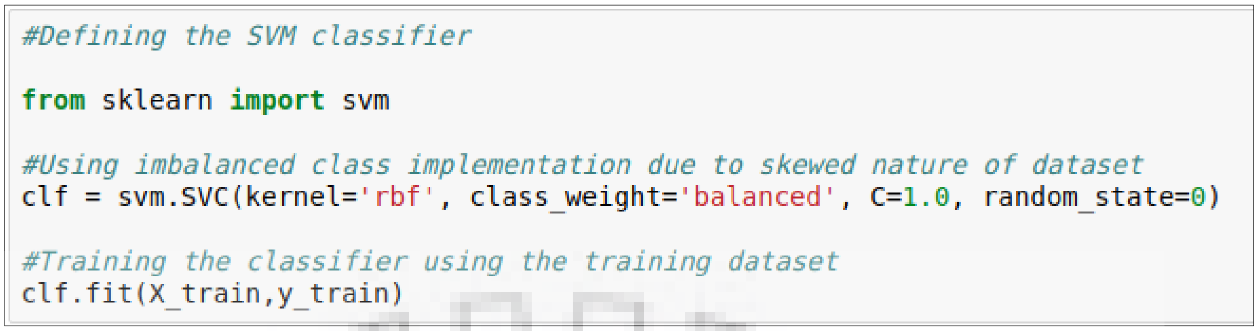


Figure 5.6: Crossplots to demonstrate variation in log values with rock type

- ## Defining and Training the SVM model

In multi-dimensional space, SVM map the feature vectors as points. It helps in clear distinction of one facies from another. Figure 5.7 shows the definition of a SVM classifier with default parameters.

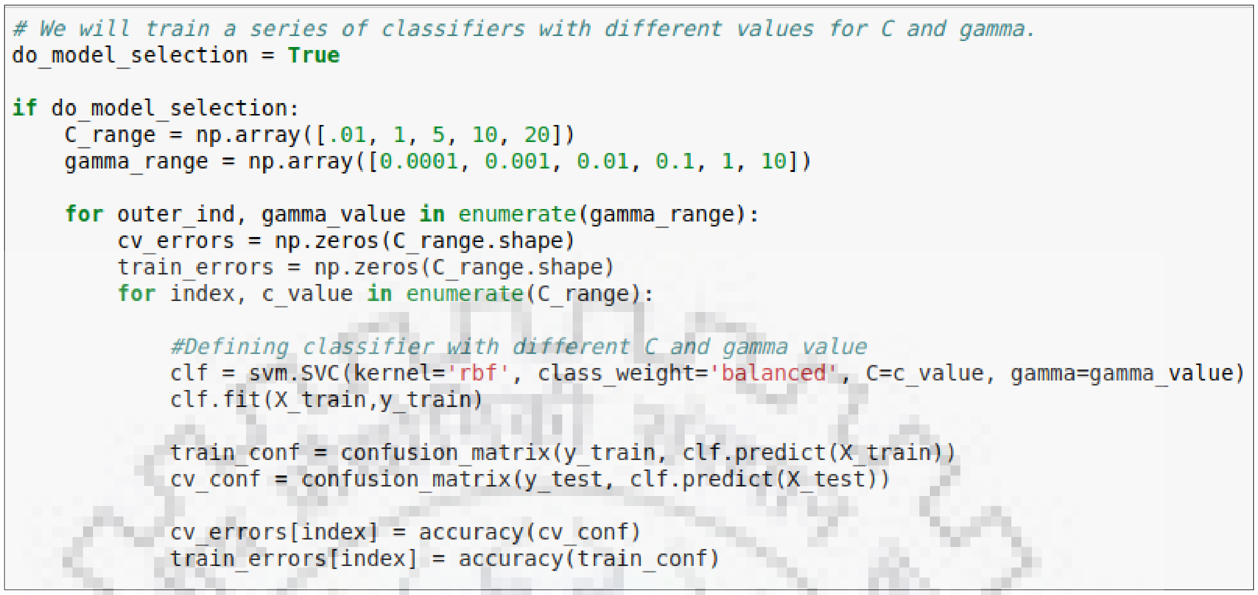


```
#Defining the SVM classifier

from sklearn import svm

#Using imbalanced class implementation due to skewed nature of dataset
clf = svm.SVC(kernel='rbf', class_weight='balanced', C=1.0, random_state=0)

#Training the classifier using the training dataset
clf.fit(X_train,y_train)
```

Figure 5.7: Defining SVM model with default parameter

- **Model Parameter Selection**
  - Initially, the classifier was built with the default parameters. However, in order to get improved classification results, the optimal parameter choices should be made.
  - Now, Considering two parameters C and gamma. The parameter C is a regularization factor and helps in countering the problem of over-fitting.
  - The SVM learning algorithm uses a kernel function to compute the distance between feature vectors. Many kernel functions exist, but in this case we are using the radial basis function rbf kernel (the default).
  - The gamma parameter describes how far away two vectors in the feature space need to be to be considered close.

Figure 5.8 describes the technique of parameter tuning. Different combination of C and gamma are used to describe different classifier. Classifier with good training and testing accuracy will be considered for making prediction on blind dataset.

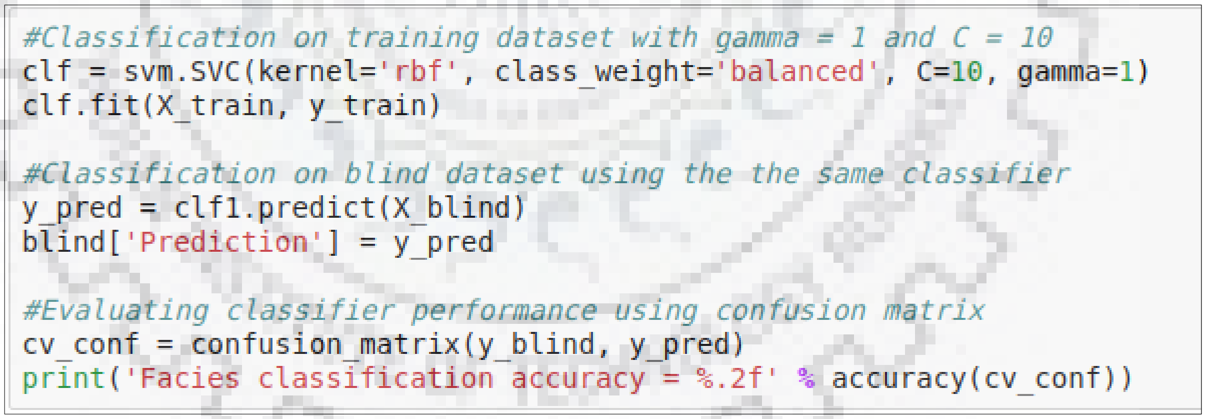


```python
# We will train a series of classifiers with different values for C and gamma.
do_model_selection = True

if do_model_selection:
    C_range = np.array([.01, 1, 5, 10, 20])
    gamma_range = np.array([0.0001, 0.001, 0.01, 0.1, 1, 10])

    for outer_ind, gamma_value in enumerate(gamma_range):
        cv_errors = np.zeros(C_range.shape)
        train_errors = np.zeros(C_range.shape)
        for index, c_value in enumerate(C_range):

            #Defining classifier with different C and gamma value
            clf = svm.SVC(kernel='rbf', class_weight='balanced', C=c_value, gamma=gamma_value)
            clf.fit(X_train,y_train)

            train_conf = confusion_matrix(y_train, clf.predict(X_train))
            cv_conf = confusion_matrix(y_test, clf.predict(X_test))

            cv_errors[index] = accuracy(cv_conf)
            train_errors[index] = accuracy(train_conf)
```

Figure 5.8: Defining SVM model with different parameters (Tuning)

- Applying the classification model to the blind data

In figure 5.9, we used the classification model with best training accuracy to predict the facies on well (KAURI E9).

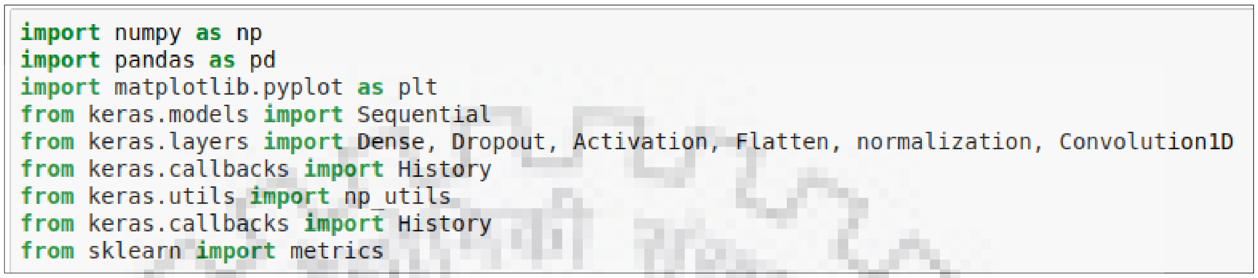


```python
#Classification on training dataset with gamma = 1 and C = 10
clf = svm.SVC(kernel='rbf', class_weight='balanced', C=10, gamma=1)
clf.fit(X_train, y_train)

#Classification on blind dataset using the the same classifier
y_pred = clf1.predict(X_blind)
blind['Prediction'] = y_pred

#Evaluating classifier performance using confusion matrix
cv_conf = confusion_matrix(y_blind, y_pred)
print('Facies classification accuracy = %.2f' % accuracy(cv_conf))
```

Figure 5.9: Classification by SVM model on blind well

5.2 Convolutional Neural Network (ConvNet)

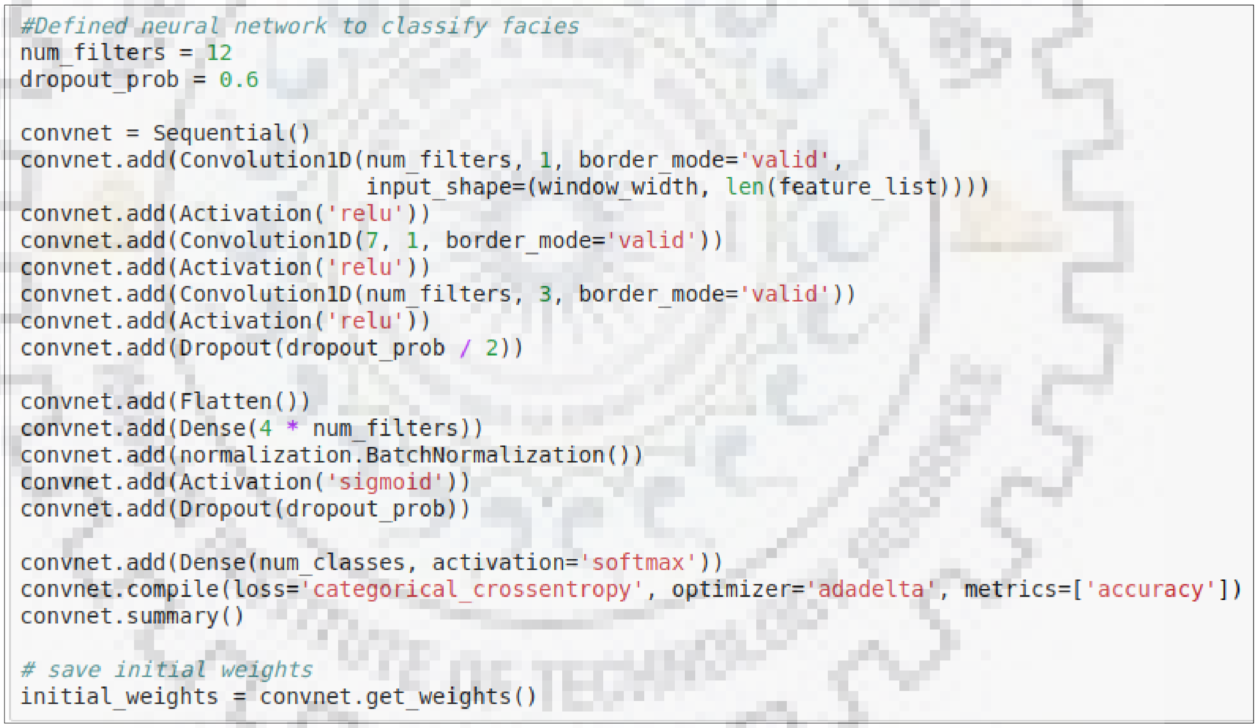
- **Importing Python library**



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten, normalization, Convolution1D
from keras.callbacks import History
from keras.utils import np_utils
from keras.callbacks import History
from sklearn import metrics
```

Figure 5.10: Python libraries for ConvNet

- **Architecture of ConvNet**



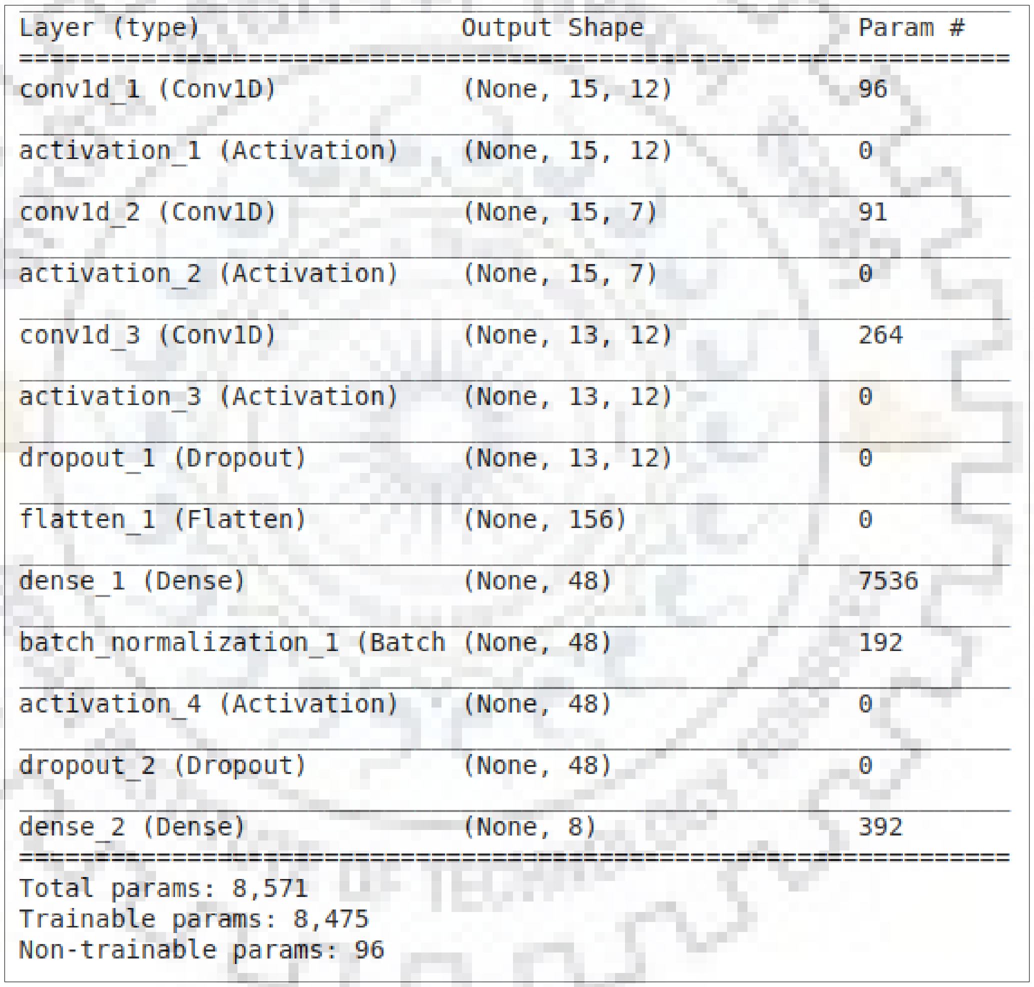
```
#Defined neural network to classify facies
num_filters = 12
dropout_prob = 0.6

convnet = Sequential()
convnet.add(Convolution1D(num_filters, 1, border_mode='valid',
                          input_shape=(window_width, len(feature_list))))
convnet.add(Activation('relu'))
convnet.add(Convolution1D(7, 1, border_mode='valid'))
convnet.add(Activation('relu'))
convnet.add(Convolution1D(num_filters, 3, border_mode='valid'))
convnet.add(Activation('relu'))
convnet.add(Dropout(dropout_prob / 2))

convnet.add(Flatten())
convnet.add(Dense(4 * num_filters))
convnet.add(normalization.BatchNormalization())
convnet.add(Activation('sigmoid'))
convnet.add(Dropout(dropout_prob))

convnet.add(Dense(num_classes, activation='softmax'))
convnet.compile(loss='categorical_crossentropy', optimizer='adadelta', metrics=['accuracy'])
convnet.summary()

# save initial weights
initial_weights = convnet.get_weights()
```

Figure 5.11: Architecture of CovNet

- We build the Convolutional Neural Network using the Sequential( ) model. We included 3 Convolutional layer, 3 activation layer. The non-linearity in the model was introduced by ‘relu’ function.

- In the next step, dropout probability was used before flattening the output. After this a fully connected layer, “Dense” was used. After normalizing the result, “sigmoid” function was used for activation.
- Optimizer: adadelta
- Loss: categorical_crossentropy

Figure 5.12 describe different parameters associated with different layer of ConvNet

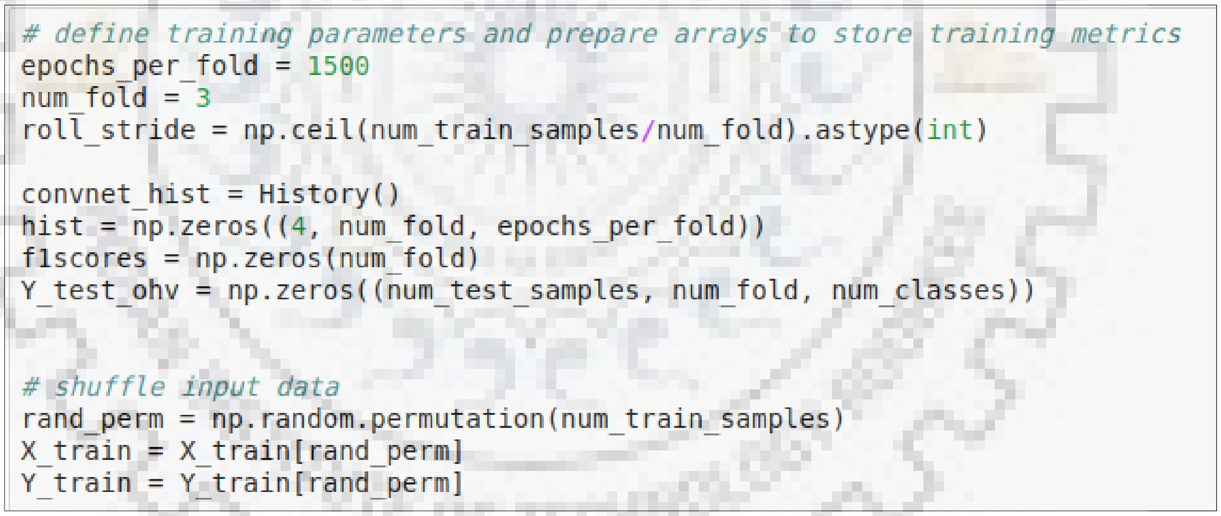


```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 15, 12)            96
_________________________________________________________________
activation_1 (Activation)    (None, 15, 12)            0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 15, 7)             91
_________________________________________________________________
activation_2 (Activation)    (None, 15, 7)             0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 13, 12)            264
_________________________________________________________________
activation_3 (Activation)    (None, 13, 12)            0
_________________________________________________________________
dropout_1 (Dropout)          (None, 13, 12)            0
_________________________________________________________________
flatten_1 (Flatten)          (None, 156)               0
_________________________________________________________________
dense_1 (Dense)              (None, 48)                7536
_________________________________________________________________
batch_normalization_1 (Batch (None, 48)                192
_________________________________________________________________
activation_4 (Activation)    (None, 48)                0
_________________________________________________________________
dropout_2 (Dropout)          (None, 48)                0
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 392
=================================================================
Total params: 8,571
Trainable params: 8,475
Non-trainable params: 96
```

Figure 5.12: Description of Layers and parameters

- **Training Parameters**

  - The parameters involved during the training of ConvNet are: Number of filters (num_filters) , Dropout probability (dropout_prob), Number of fold (num_fold), Epochs per fold (epochs_per_fold).
  - Initially, the model was training was done keeping the values of these parameters as follows:
    num_filters: 12
    dropout_prob: 0.6
    num_fold: 6
    epochs_per_fold: 1500
  - To achieve better prediction accuracy, parameter tuning technique was employed.



```
# define training parameters and prepare arrays to store training metrics
epochs_per_fold = 1500
num_fold = 3
roll_stride = np.ceil(num_train_samples/num_fold).astype(int)

convnet_hist = History()
hist = np.zeros((4, num_fold, epochs_per_fold))
f1scores = np.zeros(num_fold)
Y_test_ohv = np.zeros((num_test_samples, num_fold, num_classes))


# shuffle input data
rand_perm = np.random.permutation(num_train_samples)
X_train = X_train[rand_perm]
Y_train = Y_train[rand_perm]
```

Figure 5.13: Training parameters for ConvNet model

Chapter 6: Results

6.1 SVM Classification Results

- **Hyper parameter tuning**

Figure 4.1 shows the result of hyper parameter tuning. The model has trained a series of classifiers with different values for C and gamma. Two nested loops are used to train a classifier for every possible combination of values in the ranges specified. The classification accuracy is recorded for each combination of parameter values.

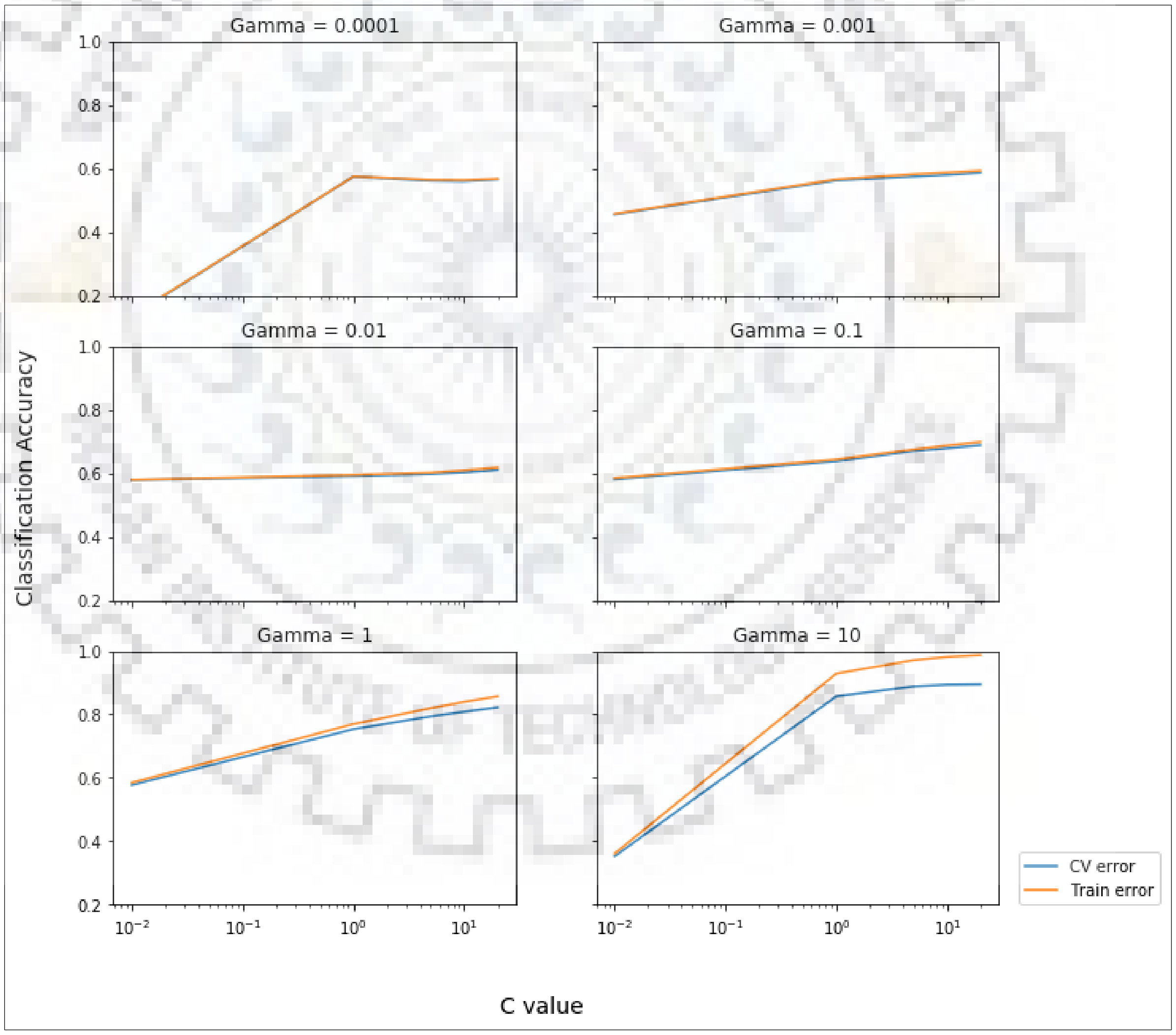


Figure 6.1: Hyper Parameter Tuning

The classifier was evaluated for four different combinations of Gamma and C value. The training and validation accuracy of all these combinations were reported in Table 6.1.

Table 6.1: Hyper parameter selection table

GAMMA	C	Training_Accuracy	Validation_Accuracy
1	1	0.89	0.58
1	10	0.81	0.63
10	1	0.89	0.58
10	10	0.89	0.58

The best validation accuracy was achieved using the classifier having parameter values gamma: 1 and C: 10.

Figure 6.2 shows the result of classification with the model parameters Gamma = 1 and C = 10. The classifier achieved an accuracy of 0.63 in classifying data from well (KAURI E9).

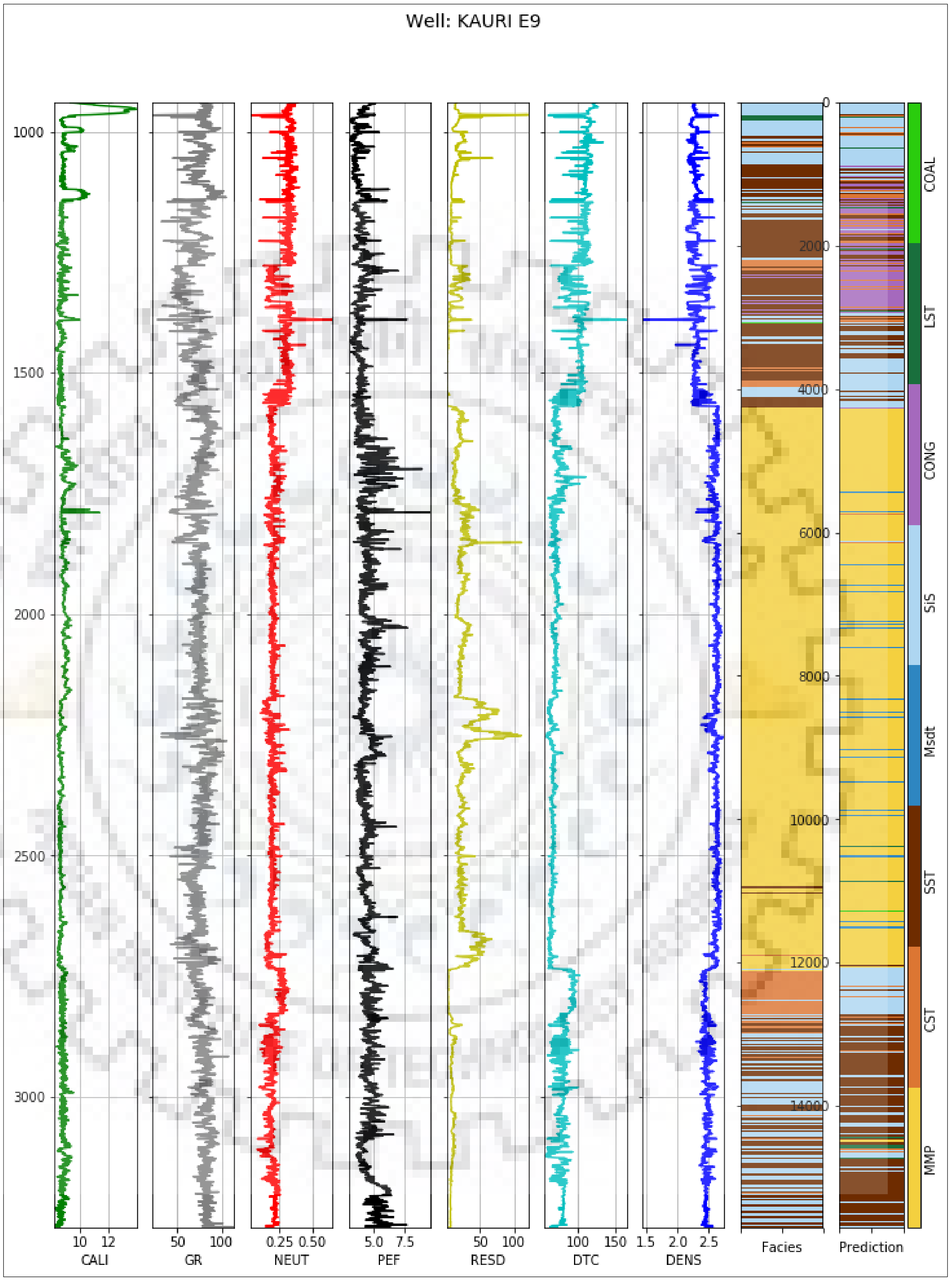
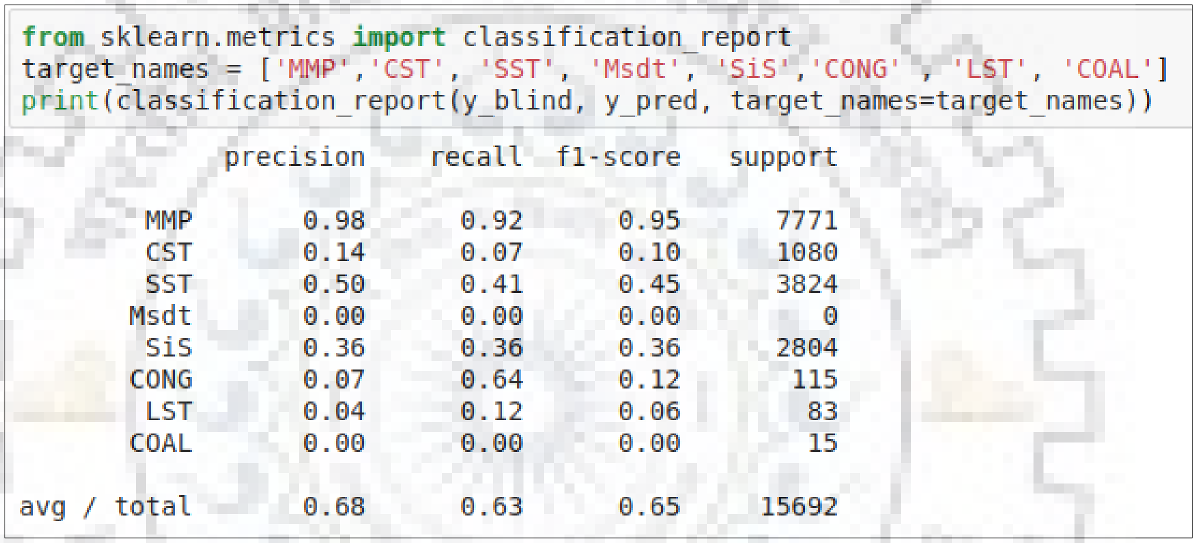


Figure 6.2: Facies prediction result from SVM

- **Classification Report:**
    - Precision and recall probability provide a measure for classifier performance on individual lithofacies.
    - Precision gives the probability of a particular sample to belong to a particular group.
    - Recall measures the accuracy i.e. correct classification probability.
    - Support is the frequency of occurrence of a particular type of sample.



```
from sklearn.metrics import classification_report
target_names = ['MMP','CST', 'SST', 'Msdt', 'SiS','CONG' , 'LST', 'COAL']
print(classification_report(y_blind, y_pred, target_names=target_names))
```

```
             precision    recall  f1-score   support

        MMP       0.98      0.92      0.95      7771
        CST       0.14      0.07      0.10      1080
        SST       0.50      0.41      0.45      3824
       Msdt       0.00      0.00      0.00         0
        SiS       0.36      0.36      0.36      2804
       CONG       0.07      0.64      0.12       115
        LST       0.04      0.12      0.06        83
       COAL       0.00      0.00      0.00        15

avg / total       0.68      0.63      0.65     15692
```

Figure 6.3: Facies prediction result from SVM

Figure 6.3 gives us an idea about the dependency of precision and recall probabilities on the support. Higher the support of a particular facies, higher the f1-score i.e. higher probability of classification model to correctly classify it.

6.2 ConvNet Classification Results

- **Hyper parameter tuning**

Case 1: Drop out: 0.3, Fold: 6, Training_acc: 0.697, Validation_acc: 0.70

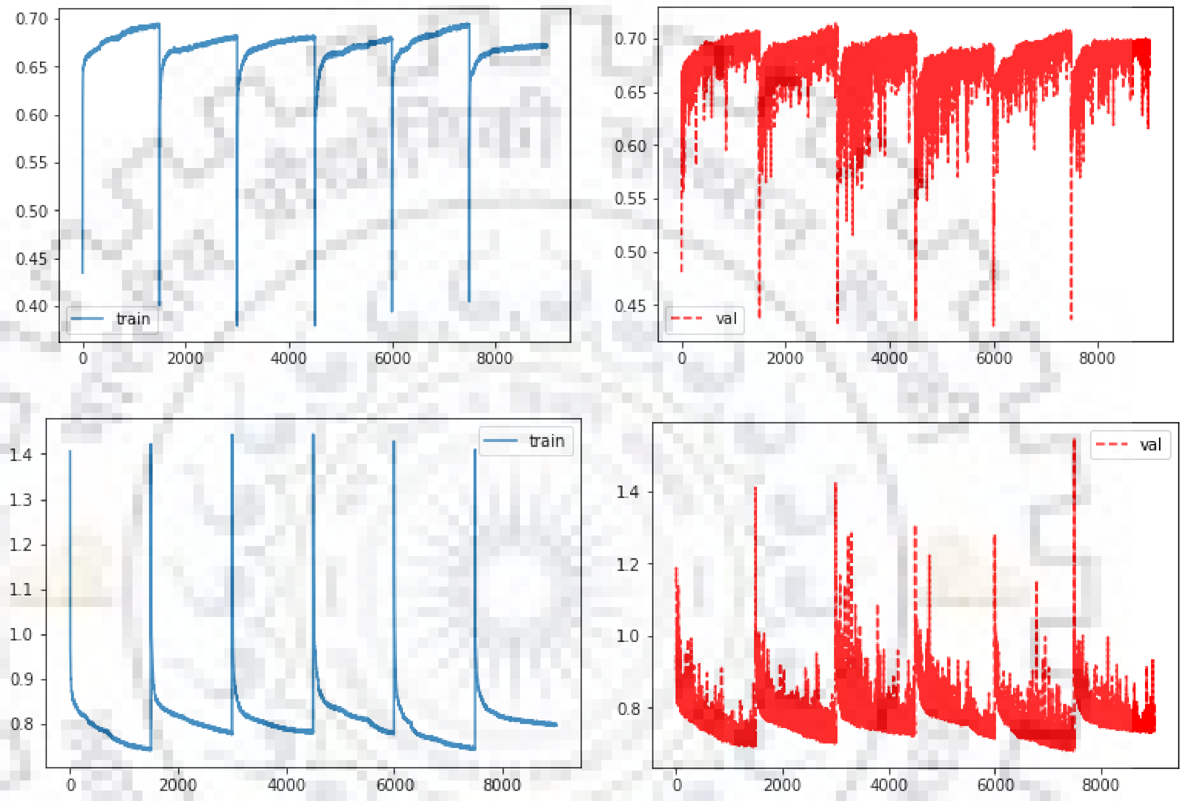


Figure 6.4: Training and Validation 1: Accuracy (top), Loss (bottom)

- 6-Fold cross-validation was used with each fold having an epoch of 1500 and dropout rate of 0.3.
- By using 6-Fold cross-validation, 6 neural networks were trained, ending up with 6 sets of predictions.
- The model shows an average F1 score of 0.697 on training dataset.
- Using Soft majority voting technique on 6 predictions, model achieves classification accuracy of 0.70 on the validation dataset.

Case 2: Drop out: 0.4, Fold: 6, Training_acc: 0.713, Validation_acc: 0.70

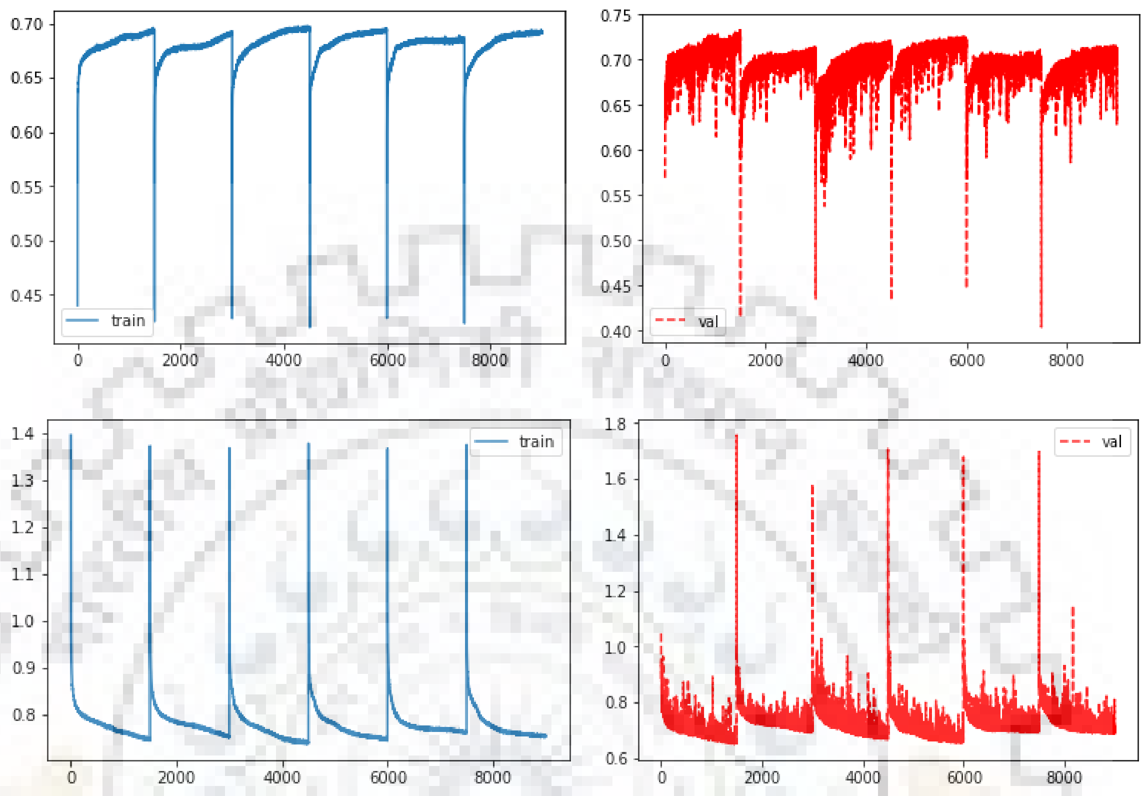


Figure 6.5: Training and Validation 2: Accuracy (top), Loss (bottom)

- 6-Fold cross-validation was used with each fold having an epoch of 1500 and dropout rate of 0.4.
- By using 6-Fold cross-validation, 6 neural networks were trained, ending up with 6 sets of predictions.
- The model shows an average F1 score of 0.713 on training dataset.
- Using Soft majority voting technique on 6 predictions, model achieves classification accuracy of 0.70 on the validation dataset.

Case 3: Drop out: 0.6, Fold: 6, Training_acc: 0.701, Validation_acc: 0.69

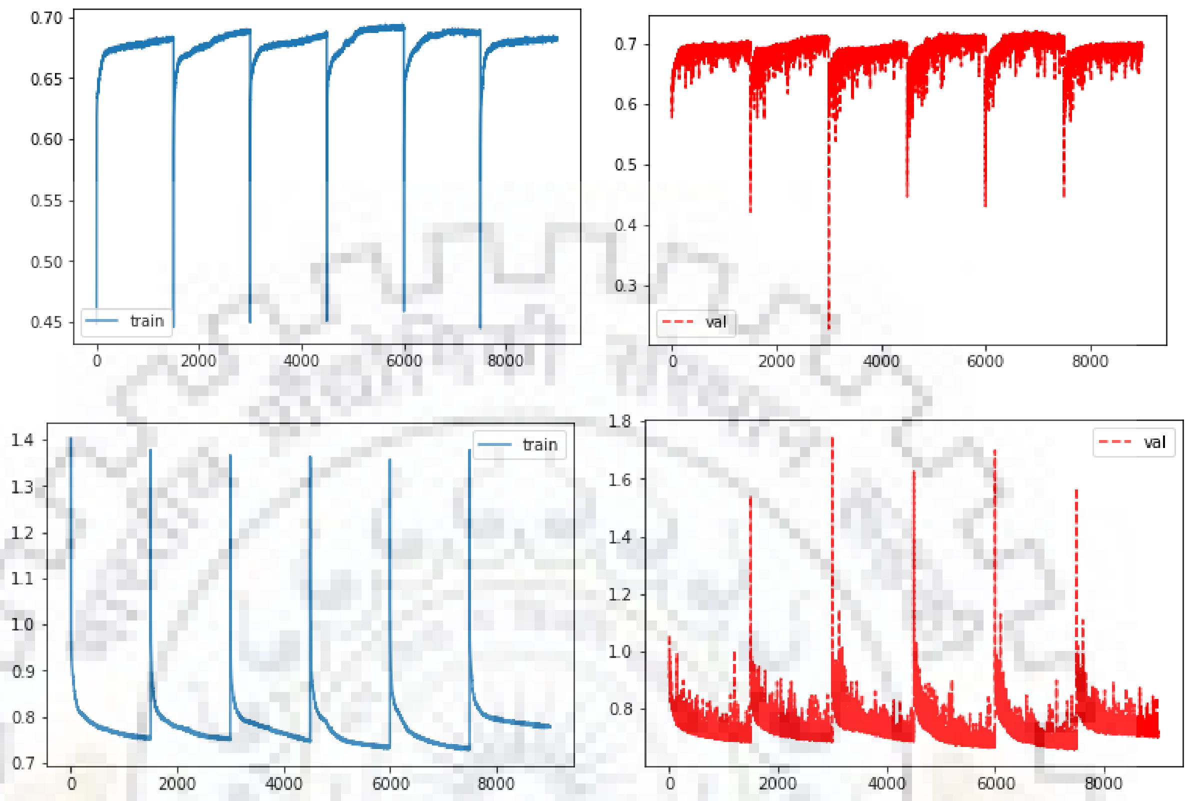


Figure 6.6: Training and Validation 3: Accuracy (top), Loss (bottom)

- 6-Fold cross-validation was used with each fold having an epoch of 1500 and dropout rate of 0.6.
- By using 6-Fold cross-validation, 6 neural networks were trained, ending up with 6 sets of predictions.
- The model shows an average F1 score of 0.701 on training dataset.
- Using Soft majority voting technique on 6 predictions, model achieves classification accuracy of 0.69 on the validation dataset.

Case 4: Drop out: 0.7, Fold: 3, Training_acc: 0.701, Validation_acc: 0.69

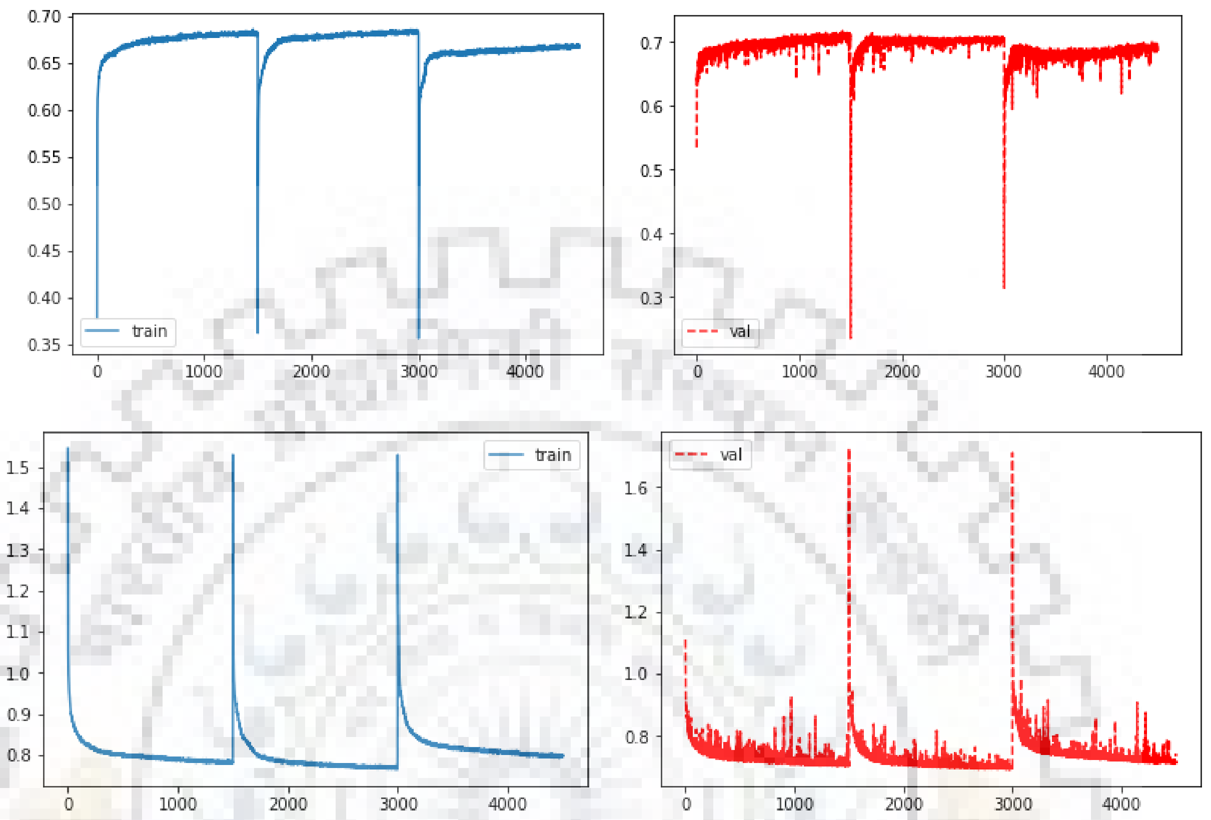


Figure 6.7: Training and Validation 4: Accuracy (top), Loss (bottom)

- 3-Fold cross-validation was used with each fold having an epoch of 1500 and dropout rate of 0.7.
- By using 3-Fold cross-validation, 3 neural networks were trained, ending up with 3 sets of predictions.
- The model shows an average F1 score of 0.701 on training dataset.
- Using Soft majority voting technique on 3 predictions, model achieves classification accuracy of 0.69 on the validation dataset.

Case 5: Drop out: 0.6, Fold: 3, Training_acc: 0.701, Validation_acc: 0.67

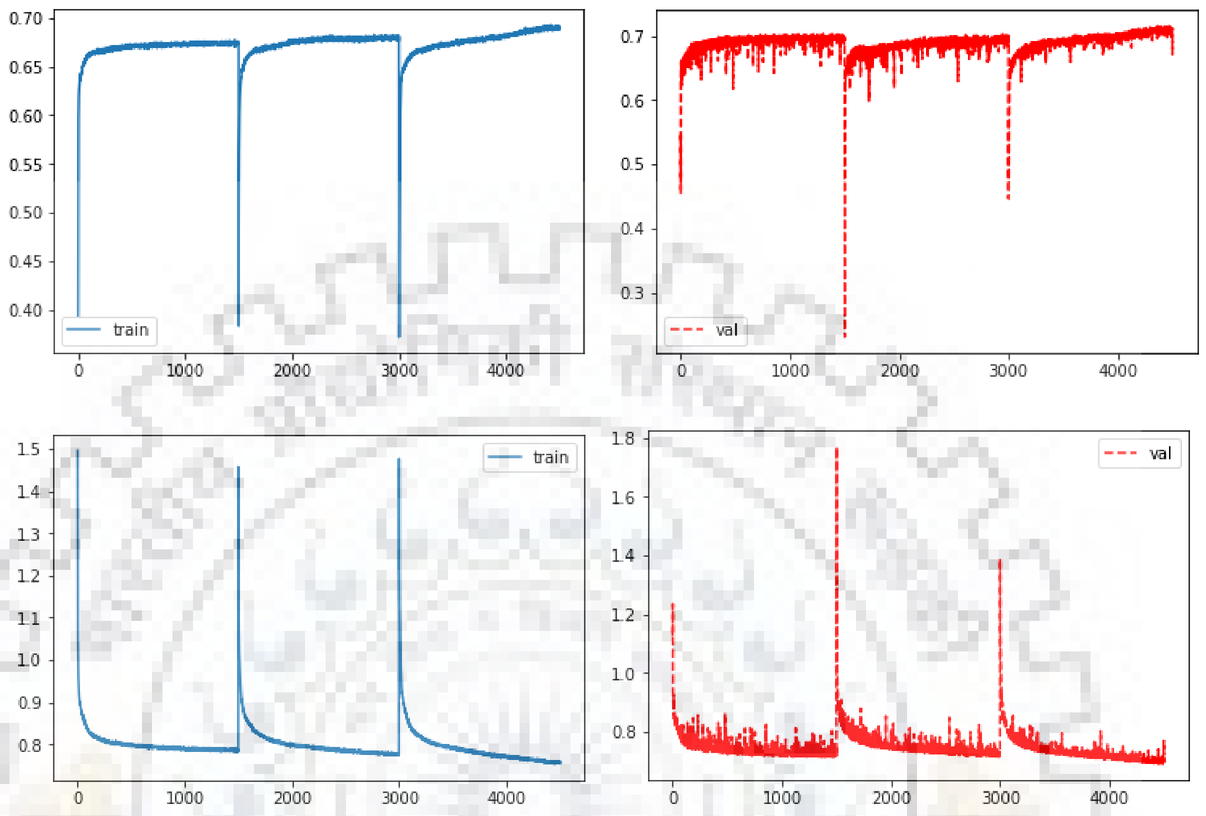


Figure 6.8: Training and Validation 5: Accuracy (top), Loss (bottom)

- 3-Fold cross-validation was used with each fold having an epoch of 1500 and dropout rate of 0.6.
- By using 3-Fold cross-validation, 3 neural networks were trained, ending up with 3 sets of predictions.
- The model shows an average F1 score of 0.701 on training dataset.
- Using Soft majority voting technique on 3 predictions, model achieves classification accuracy of 0.67 on the validation dataset.

Case 6: Drop out: 0.5, Fold: 3, Training_acc: 0.723, Validation_acc: 0.67

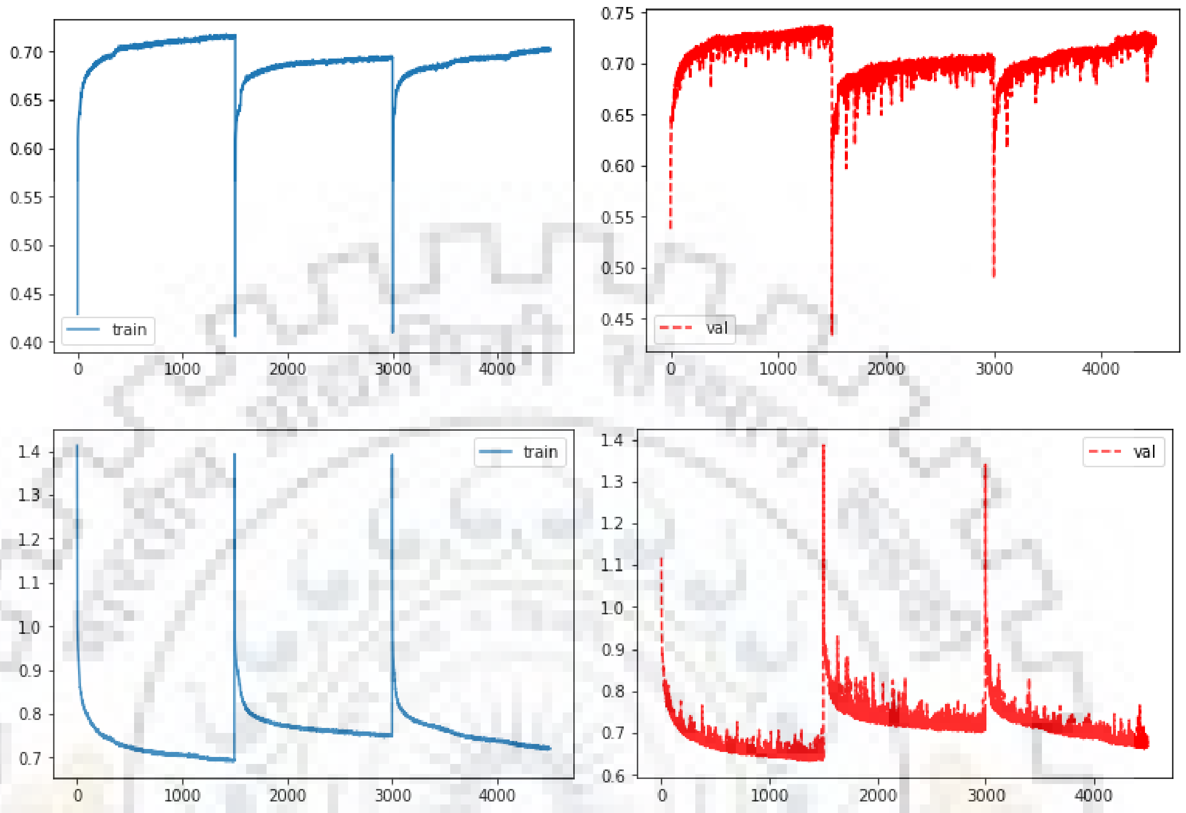


Figure 6.9: Training and Validation 6: Accuracy (top), Loss (bottom)

- 3-Fold cross-validation was used with each fold having an epoch of 1500 and dropout rate of 0.5.
- By using 3-Fold cross-validation, 3 neural networks were trained, ending up with 3 sets of predictions.
- The model shows an average F1 score of 0.723 on training dataset.
- Using Soft majority voting technique on 3 predictions, model achieves classification accuracy of 0.67 on the validation dataset.

Case 7: Drop out: 0.4, Fold: 3, Training_acc: 0.712, Validation_acc: 0.70

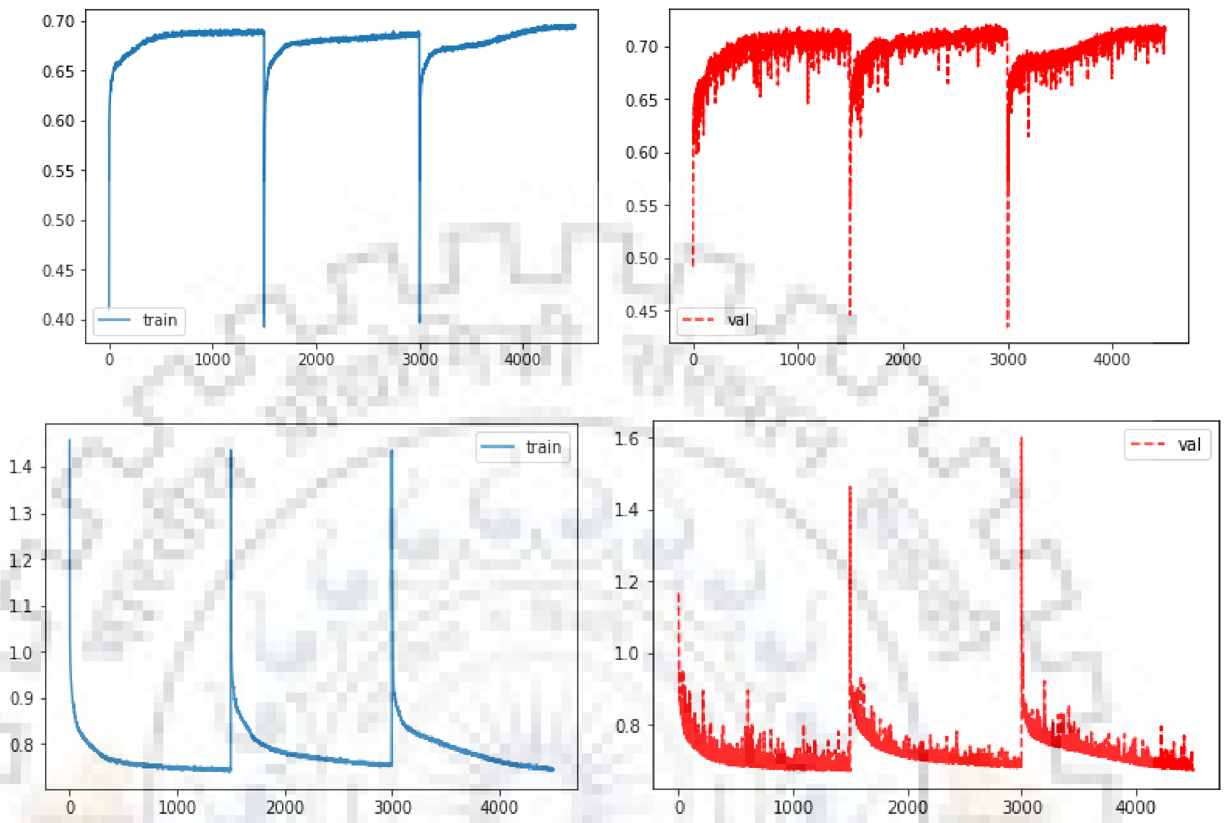


Figure 6.10: Training and Validation 7: Accuracy (top), Loss (bottom)

- 3-Fold cross-validation was used with each fold having an epoch of 1500 and dropout rate of 0.4.
- By using 3-Fold cross-validation, 3 neural networks were trained, ending up with 3 sets of predictions.
- The model shows an average F1 score of 0.712 on training dataset.
- Using Soft majority voting technique on 3 predictions, model achieves classification accuracy of 0.70 on the validation dataset.

Case 8: Drop out: 0.3, Fold: 3, Training_acc: 0.704, Validation_acc: 0.71

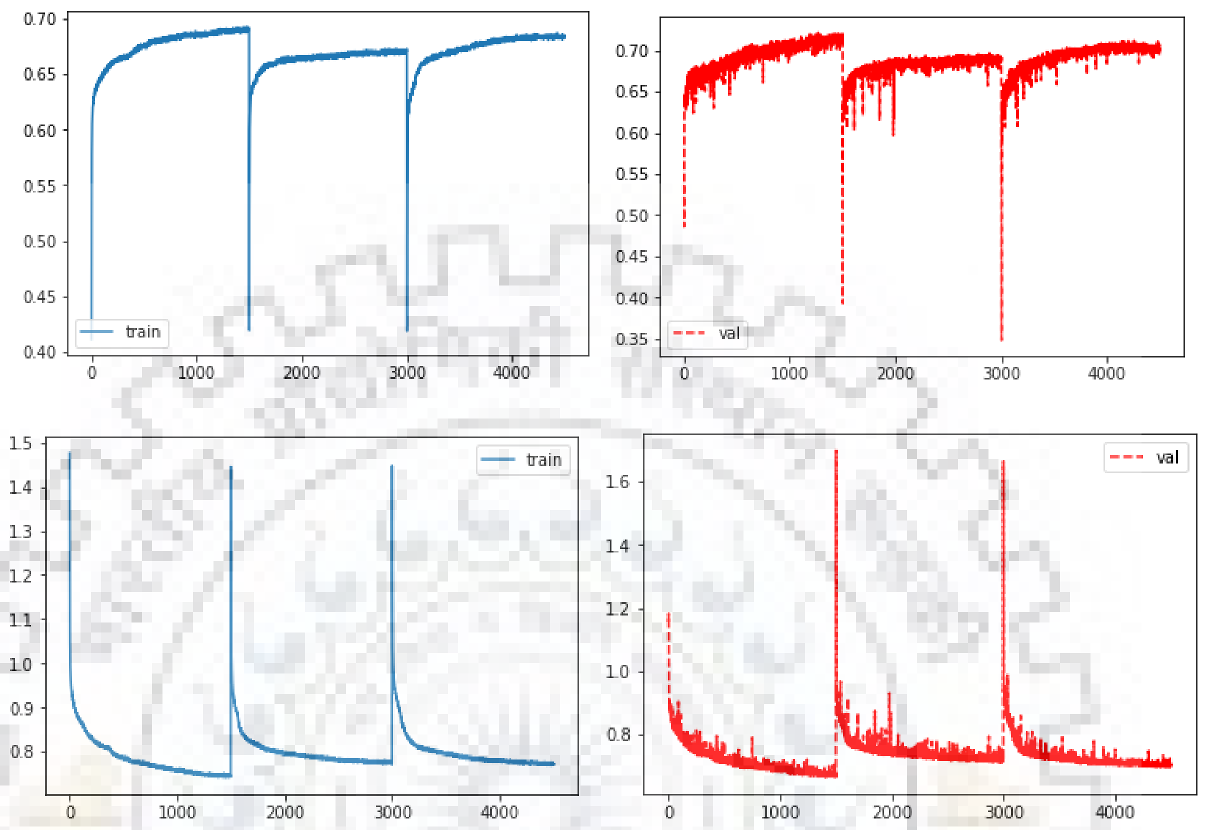


Figure 6.11: Training and Validation 8: Accuracy (top), Loss (bottom)

- 3-Fold cross-validation was used with each fold having an epoch of 1500 and dropout rate of 0.3.
- By using 3-Fold cross-validation, 3 neural networks were trained, ending up with 3 sets of predictions.
- The model shows an average F1 score of 0.704 on training dataset.
- Using Soft majority voting technique on 3 predictions, model achieves classification accuracy of 0.71 on the validation dataset.

Case 9: Drop out :0.0, Fold: 3, Training_acc: 0.700, Validation_acc: 0.68

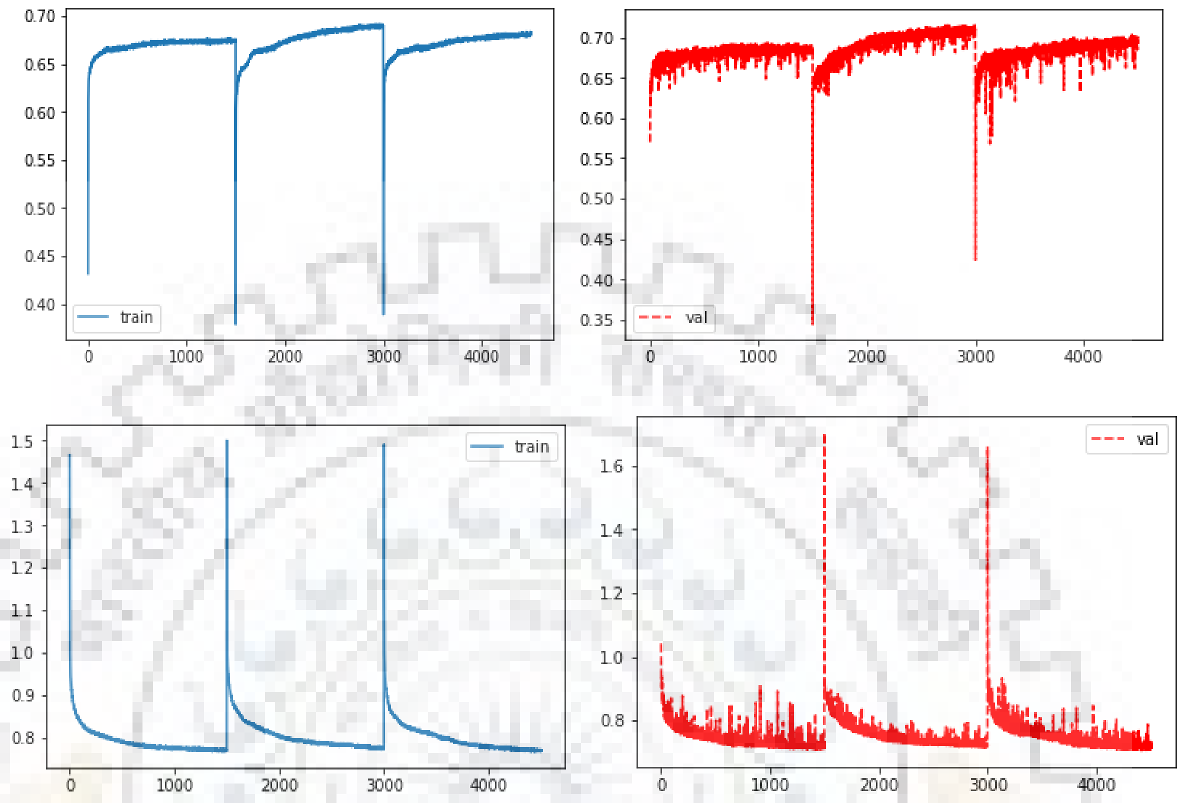


Figure 6.12: Training and Validation 9: Accuracy (top), Loss (bottom)

- 3-Fold cross-validation was used with each fold having an epoch of 1500 and dropout rate of 0.0.
- By using 3-Fold cross-validation, 3 neural networks were trained, ending up with 3 sets of predictions.
- The model shows an average F1 score of 0.700 on training dataset.
- Using Soft majority voting technique on 3 predictions, model achieves classification accuracy of 0.68 on the validation dataset.

- Model with best accuracy: **Drop out: 0.3, Fold: 3**

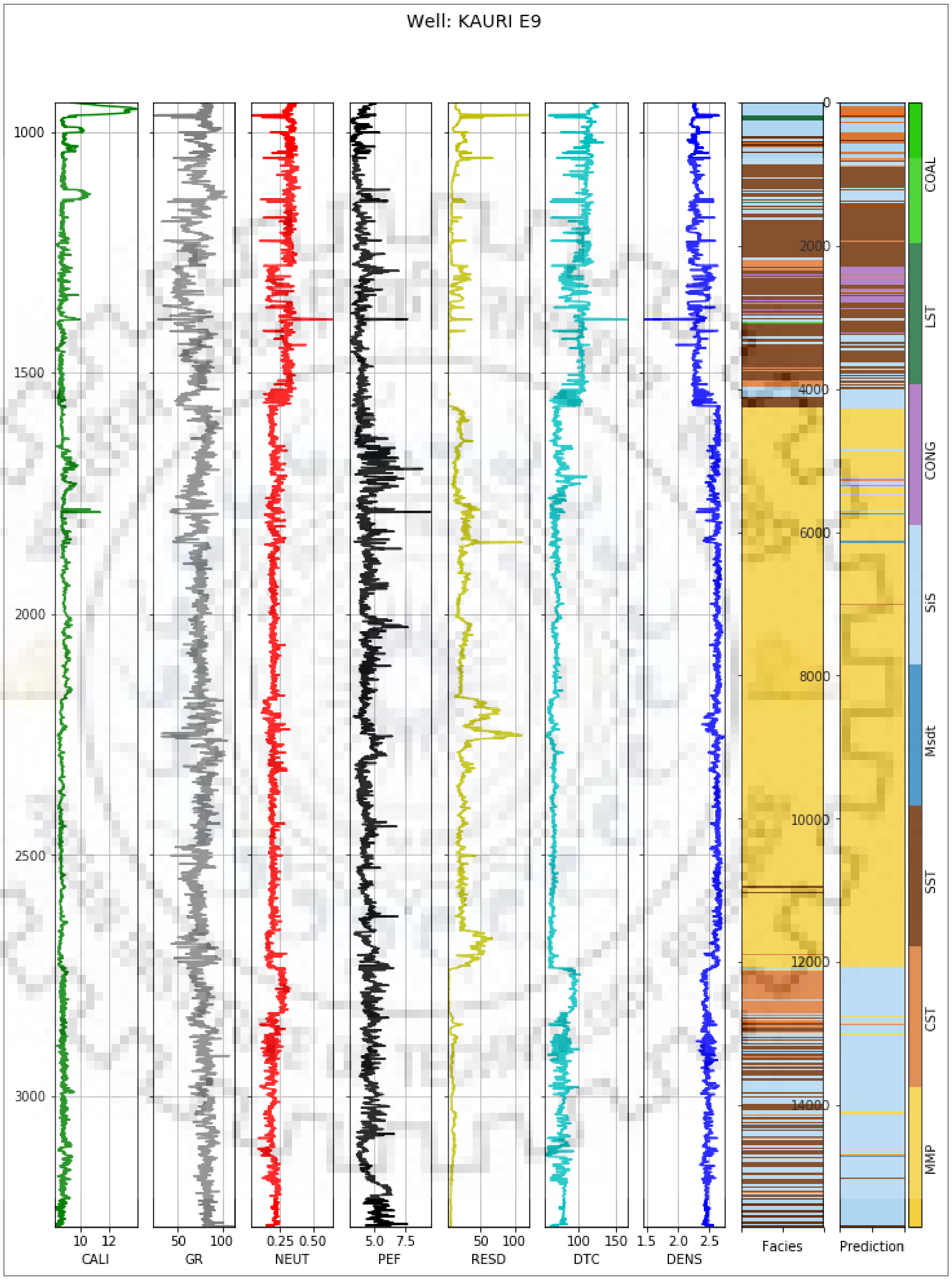
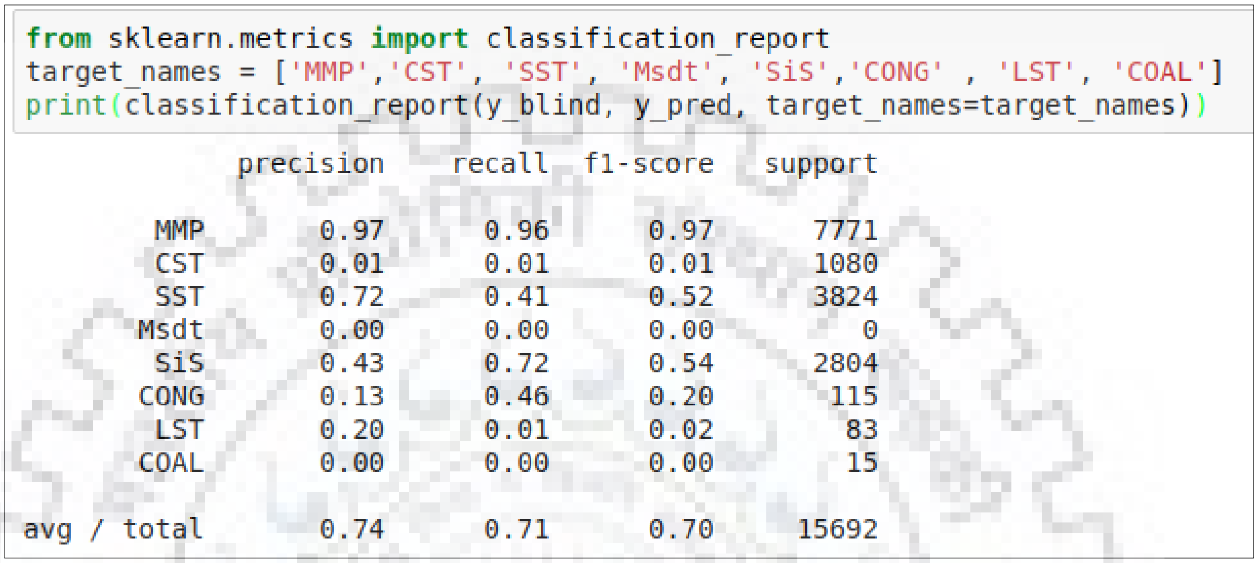


Figure 6.13: Facies prediction result from CNN

The best prediction accuracy of 0.71 was obtained at dropout rate: 0.3 and 3 fold cross-validation.

Classification report:

Figure 6.14 shows the high dependency of model on the number of training examples. A positive correlation is found between the f1-score and support of different facies.



```
from sklearn.metrics import classification_report
target_names = ['MMP','CST', 'SST', 'Msdt', 'SiS','CONG' , 'LST', 'COAL']
print(classification_report(y_blind, y_pred, target_names=target_names))
```

```
             precision    recall  f1-score   support

        MMP       0.97      0.96      0.97      7771
        CST       0.01      0.01      0.01      1080
        SST       0.72      0.41      0.52      3824
       Msdt       0.00      0.00      0.00         0
        SiS       0.43      0.72      0.54      2804
       CONG       0.13      0.46      0.20       115
        LST       0.20      0.01      0.02        83
       COAL       0.00      0.00      0.00        15

avg / total       0.74      0.71      0.70     15692
```

Figure 6.14: Classification report from ConvNet

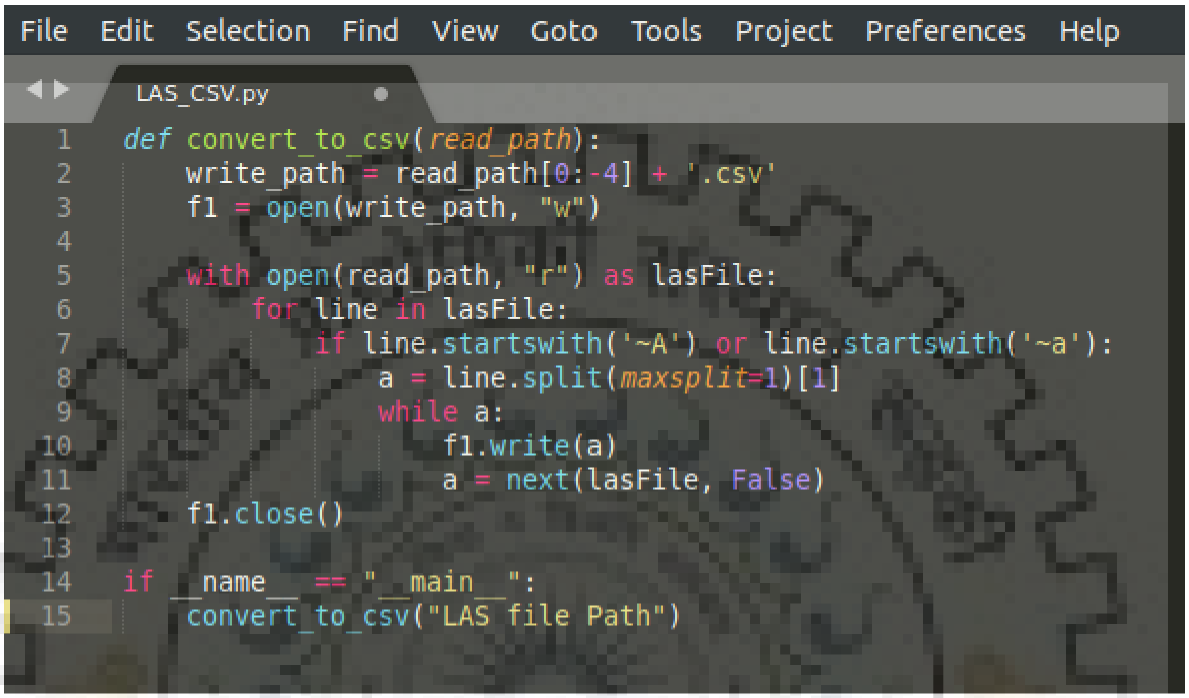
Chapter 7: Conclusion

- From the work carried out in the thesis, it can be concluded that machine learning and deep learning techniques can be applied to predict the lithofacies of the wells in the regions where we have only few mud logs available with us.
- Results obtained on a set of seven wells validate the proposed approach, which highlights the positive impact of the developed feature augmentation strategy.
- The results obtained while validating our model on blind well also give a confirmation to a good capacity of this model to generalize to new data. Using deep learning strategies for feature learning and classification (e.g., ConvNets), improved results were obtained.
- Both methods have limitation in terms of skewed dataset therefore resulted in poor precision and recall score for those particular facies.
- Both these methods have very high dependency to the support of different lithofacies in the training samples. Higher the support of a particular facies in the training dataset, higher is the probability of correctly classifying that particular facies to its correct class.
- Considering the achieved promising results, future work will be devoted on validating the possibility of adding geological constraints to drive classification with the help of apriori information about rock formation.

Reference

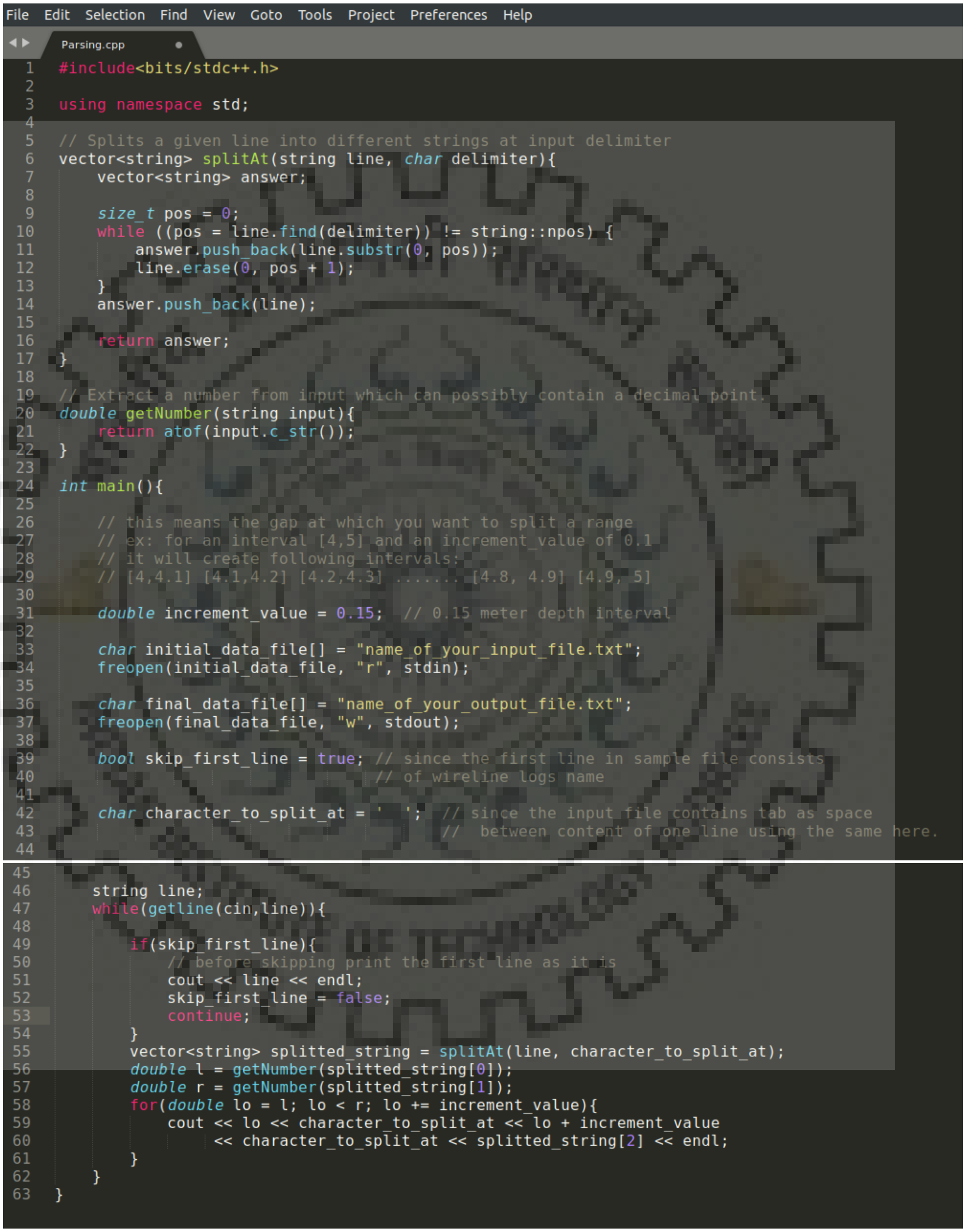
- Bohling, G., and M. Dubois, 2003, An integrated application of neural network and markov chain techniques to the pre-diction of lithofacies from well logs.
- Busch, J., W. Fortney, and L. Berry, 1987, Determination of lithology from well logs by statistical analysis: SPE (Society of Petroleum Engineers) Format. Eval.; (United States),2:4
- del Monte, A. A., 2015, Seismic petro physics: Part 1: The Leading Edge,34, 440–442.
- Dubois, M. K., G. C. Bohling, and S. Chakrabarti, 2007, Comparison of four approaches to a rock facies classification problem: Comput. Geosci.,33, 599–617.
- Hall, B., 2016, Facies classification using machine learning: The Leading Edge,35, 906–909.
- LeCun, Y., Y. Bengio, and G. Hinton, 2015, Deep learning: Nature,521, 436–444.
- Lecun, Y., L. Bottou, Y. Bengio, and P. Haffner, 1998, Gradient-based learning applied to document recognition: Proceedings of the IEEE,86, 2278–2324.
- Pires de Lima, Rafael & Surianam, Fnu & J. Marfurt, Kurt & J. Pranter, Matthew. (2019). CONVOLUTIONAL NEURAL NETWORKS AS AID IN CORE LITHOFACIES CLASSIFICATION. Interpretation. 1-50. 10.1190/int-2018-0245.1.
- Rabaute, A., 1998, Inferring a continuous lithology and mineralogy from multivariate statistical analysis of well-logging data : Examples from some sedimentary structures associated with tectonic plates convergence zones (ocean drilling program leg 134, 156 and 160): Institut des sciences dela terre, de l'eau et de l'espace de Montpellier, Université Montpellier II. Mémoires géosciences-Montpellier.
- Srivastava, N., G. E. Hinton, A. Krizhevsky, I. Sutskever, andR. Salakhutdinov, 2014, Dropout: a simple way to pre-vent neural networks from overfitting.: Journal of Machine Learning Research,15, 1929–1958.
- Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov,D. Erhan, V. Vanhoucke, and A. Rabinovich, 2015, Going deeper with convolutions: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1–9.
- Tschannen, Valentin, Matthias Delescluse, Mathieu Rodriguez and Janis Keuper. "Facies classification from well logs using an inception convolutional network." CoRR abs/1706.00613 (2017): n. pag.
- Wang, G., and T. R. Carr, 2012, Methodology of organic-rich shale lithofacies identification and prediction: A case study from marcellus shale in the appalachian basin: Computers&Geosciences,49, 151–163.

Appendix A


File Edit Selection Find View Goto Tools Project Preferences Help
LAS_CSV.py

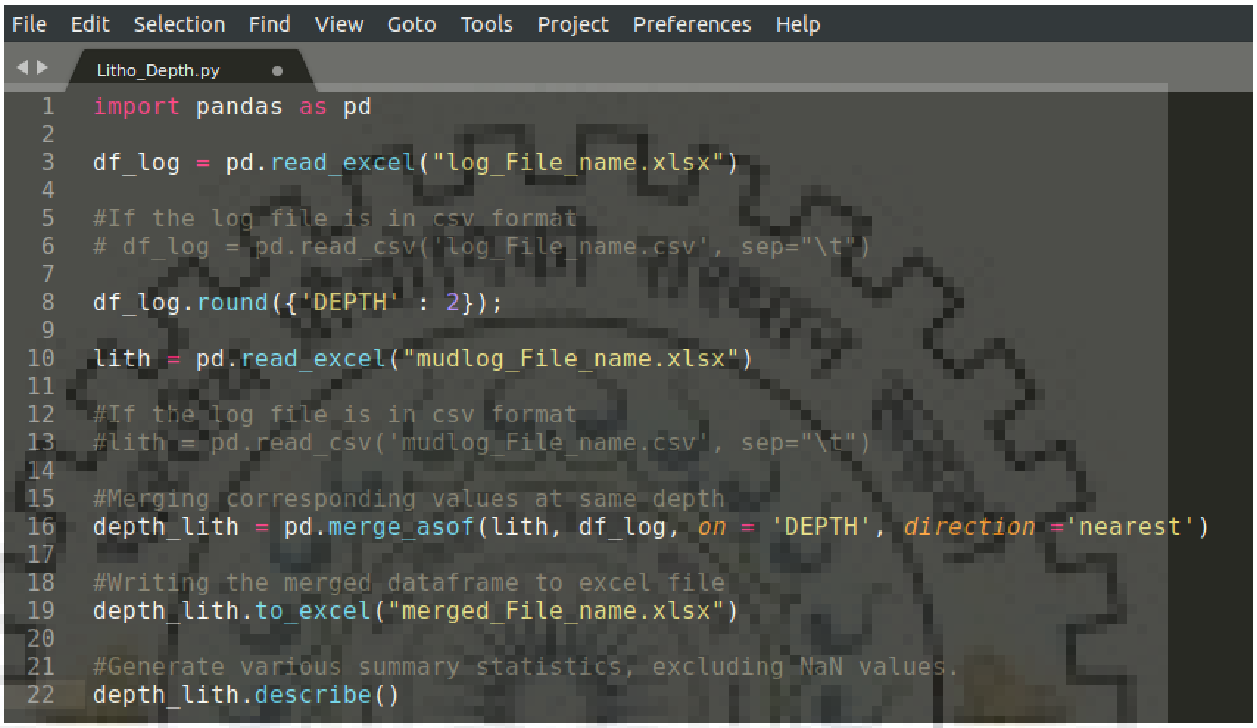
```python
def convert_to_csv(read_path):
    write_path = read_path[0:-4] + '.csv'
    f1 = open(write_path, "w")

    with open(read_path, "r") as lasFile:
        for line in lasFile:
            if line.startswith('~A') or line.startswith('~a'):
                a = line.split(maxsplit=1)[1]
                while a:
                    f1.write(a)
                    a = next(lasFile, False)
    f1.close()

if __name__ == "__main__":
    convert_to_csv("LAS file Path")
```

Appendix B


File Edit Selection Find View Goto Tools Project Preferences Help
Parsing.cpp

```cpp
#include<bits/stdc++.h>

using namespace std;

// Splits a given line into different strings at input delimiter
vector<string> splitAt(string line, char delimiter){
    vector<string> answer;

    size_t pos = 0;
    while ((pos = line.find(delimiter)) != string::npos) {
        answer.push_back(line.substr(0, pos));
        line.erase(0, pos + 1);
    }
    answer.push_back(line);

    return answer;
}

// Extract a number from input which can possibly contain a decimal point.
double getNumber(string input){
    return atof(input.c_str());
}

int main(){

    // this means the gap at which you want to split a range
    // ex: for an interval [4,5] and an increment_value of 0.1
    // it will create following intervals:
    // [4,4.1] [4.1,4.2] [4.2,4.3] ....... [4.8, 4.9] [4.9, 5]

    double increment_value = 0.15;  // 0.15 meter depth interval

    char initial_data_file[] = "name_of_your_input_file.txt";
    freopen(initial_data_file, "r", stdin);

    char final_data_file[] = "name_of_your_output_file.txt";
    freopen(final_data_file, "w", stdout);

    bool skip_first_line = true; // since the first line in sample file consists
                                 // of wireline logs name

    char character_to_split_at = '	';  // since the input file contains tab as space
                                        //  between content of one line using the same here.


    string line;
    while(getline(cin,line)){

        if(skip_first_line){
            // before skipping print the first line as it is
            cout << line << endl;
            skip_first_line = false;
            continue;
        }
        vector<string> splitted_string = splitAt(line, character_to_split_at);
        double l = getNumber(splitted_string[0]);
        double r = getNumber(splitted_string[1]);
        for(double lo = l; lo < r; lo += increment_value){
            cout << lo << character_to_split_at << lo + increment_value
                 << character_to_split_at << splitted_string[2] << endl;
        }
    }
}
```

Appendix C


File Edit Selection Find View Goto Tools Project Preferences Help
Litho_Depth.py

```python
import pandas as pd

df_log = pd.read_excel("log_File_name.xlsx")

#If the log file is in csv format
# df_log = pd.read_csv('log_File_name.csv', sep="\t")

df_log.round({'DEPTH' : 2});

lith = pd.read_excel("mudlog_File_name.xlsx")

#If the log file is in csv format
#lith = pd.read_csv('mudlog_File_name.csv', sep="\t")

#Merging corresponding values at same depth
depth_lith = pd.merge_asof(lith, df_log, on = 'DEPTH', direction ='nearest')

#Writing the merged dataframe to excel file
depth_lith.to_excel("merged_File_name.xlsx")

#Generate various summary statistics, excluding NaN values.
depth_lith.describe()
```