HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

MASTER THESIS

Lateral Flow Image Analysis with Python

Author

MD. ABDULLAH-AL-KAMRAN RIPON

255630

Supervisors

Prof. Dr. Matthias Kohl

Prof. Dr. Hans-Peter Deigner

A Thesis in the Field of Medical Engineering

for the Degree of Master in Bio-medical Engineering

Hochschule Furtwangen University

May 2020

Abstract

For a fast and point of care (POC) diagnosis, a machine learning (ML) approach could be used to measure the concentration of digoxin and its hapten, digoxigenin, in human serum from an image readout system of gold nanoparticle lateral flow (LF) assay [1]. In this work an image processing method is developed that can be applied to separate the region of interest (ROI) from the raw data, pre-process the data, and perform ML to predict the concentration of digoxigenin solution from the image intensity.

To accomplish the whole process a three-step algorithm was developed. At first, a two-step cropping method was developed to perform the cropping and separation of the ROI from raw images. In the next step, pre-processing was done, which includes building a data frame from cropped images and apply some feature engineering to improve the performance of the ML model. Lastly, ML models were developed to predict the concentration of the digoxigenin solution. ML model then evaluated by different evaluation methods and after that validated by K-fold cross-validation. The $adjusted - R^2$ value, accuracy of the model, for the final ML model is 81% and K-Fold cross-validation is 70.6%.

Keywords Python, Machine Learning, scikit-learn, scikit-image, Template matching, Lateral Flow Assay.

Zusammenfassung

Für eine schnelle und Point-of-Care-Diagnose (POC) könnte ein maschinelles Lernverfahren (ML) verwendet werden, um die Konzentration von Digoxin und seinem Hapten, Digoxigenin, in Humanserum aus einem Bildauslesesystem des Gold-Nanopartikel-Lateral-Flow (LF)-Tests [1] zu messen. In dieser Arbeit wird eine Bildverarbeitungsmethode entwickelt, die angewendet werden kann, um den interessierenden Bereich (ROI) von den Rohdaten zu trennen, die Daten vorzubereiten und eine ML durchzuführen, um die Konzentration der Digoxigeninlösung aus der Bildintensität vorherzusagen.

Um den gesamten Prozess durchzuführen, wurde ein dreistufiger Algorithmus entwickelt. Zunächst wurde ein zweistufiges Zuschneideverfahren entwickelt, um das Zuschneiden und Trennen des interessierenden Bereichs von den Rohbildern durchzuführen. Im nächsten Schritt wurde eine Vorverarbeitung durchgeführt, die das Erstellen eines Datenrahmens aus zugeschnittenen Bildern und das Anwenden von Feature-Engineering umfasst, um die Leistung des ML-Modells (Machine Learning) zu verbessern. Zuletzt wurden ML-Modelle entwickelt, um die Konzentration der Digoxigeninlösung vorherzusagen. Das ML-Modell wurde dann durch verschiedene Bewertungsmethoden bewertet und anschließend durch K-fache Kreuzvalidierung validiert. Der $adjusted - R^2$ Wert, Genauigkeit des Modells, für das endgültige ML-Modell beträgt 81% und die K-Fold-Kreuzvalidierung beträgt 70,6%.

Schlüsselwörter Python, Machine Learning, scikit-learn, scikit-image, Template matching, Lateral Flow Assay.

Statutory Declaration

I, the undersigned, Md. Abdullah-al-Kamran Ripon, hereby declare that I have written and developed this thesis entitled “Lateral Flow Image Analysis with Python” only with the listed auxiliary means and without any improper assistance.

The used literature sources are fully cited in the references.

Villingen- Schwenningen, 5.05.2020.

Adresse: Jakob-Kienzle-Strasse 17, 78054 Villingen-Schwenningen.

.....

Md. Abdullah-al-Kamran Ripon
ID: 255630

Acknowledgments

I would like to thank Prof. Dr. Matthias Kohl for his great support, guidance, and advice during my thesis work. During our several meetings, his motivation and encouragement help me to complete my thesis work. Prof. Kohl patiently read my thesis and gave me the advice to improve it. Without his support and guidance, it would be impossible for me to finish this work.

I would also like to thank my second supervisor, Prof. Dr. Hans-Peter Deigner, for his help and support.

Lastly, I would like to thank all my professors from HFU, BME program coordinator Mr. Andreas Dietz, all the staff members of HFU for their help and great support during my study at HFU.

Abbreviations and Acronyms

POC	Point of Care
ML	Machine Learning
LF	Lateral Flow
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
MAE	Median Absolute Error
COD	Coefficient of Determination
SSE	Sum of Squared Error
SST	Sum of Squared Total
ROI	Region of Interest
SD	Standard Deviation
DF	Data Frame
CI	Confidence Interval
RL	Regression Line
RA	Regression Analysis
HOR	Histogram of Residuals
Q-Q	Quantile-Quantile
FE	Feature Engineering
LR	Linear Regression

Table of Contents

List of Tables

List of Figures

Chapter I.

Introduction

1.1 Introduction

Fast and easy to use point of care (POC) devices, which could be operated by patients him or herself for the determination of drugs, or the measurement of the concentration of metabolic, is an essential process in the area of Precision Medicine. This easy to use system must be cost-effective, easy to operate, and should not require any additional equipment to measure or process the information [1].

To solve this issue a machine learning approach could be used to predict the concentration of digoxin and it's hapten, digoxigenin, from the image intensity. Digoxin and it's hapten, digoxigenin are used for the treatment of tachycardia with small concentration, 0.5 – 2.0 ng. mL^{-1}, in human serum solution. When the concentration of digoxin in serum solution is above 2.5 ng. mL^{-1}, it becomes very toxic, which should be avoided [1].

Main objectives of this thesis work is to

i. Design and development of an algorithm to prepare image data for analysis.
ii. Development of a machine learning model to predict the concentration of digoxin and it's hapten, digoxigenin, from image intensity.

1.2 Lateral Flow Assay

Lateral flow (LF) immunoassay is a very effective technique compared to the conventional detection method because of its simple nature. LF assays are very useful for a non-laboratory environment as a point of care (POC) device. LF assay can be used in many areas including clinical analysis, foodborne diseases, toxic pollutants, and heavy metals detection [12, 13].

LF assay is cellulose-based devices which use to detect the presence or absence of a target analyte in a liquid solution. The Detection process with LF assay is very fast, it reduces costly and sensitive equipment and could be used in the non-laboratory environments as a POC device [27].

Typically, a LF device contains one control line and one or more test lines. The main task of the control line is to check whether the test is working properly or not. As mentioned, each LF device contains only one control line but it could have more than one test line [28,75].

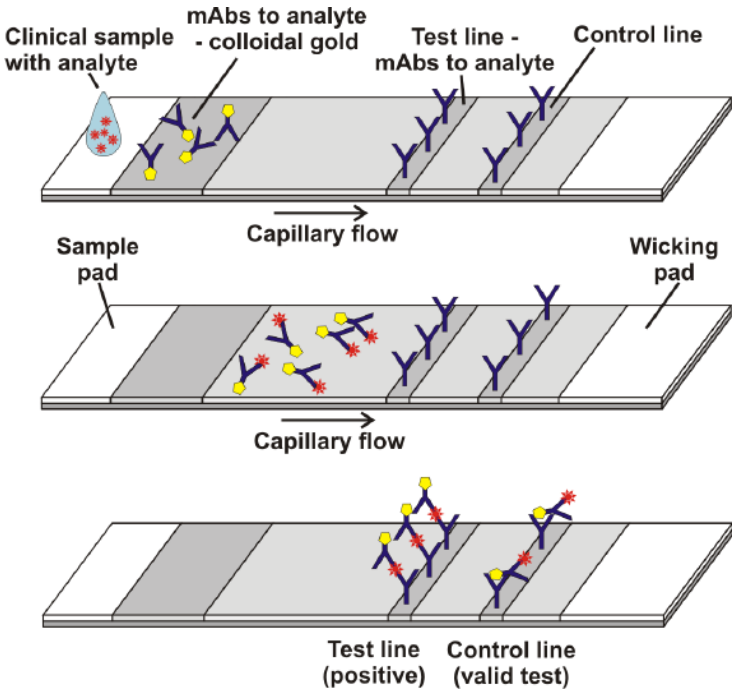


Figure 1: Schematic diagram of the LF assay with colloidal gold as a label [32, (pp:361)]

A series of capillary beds of porous paper, microstructure polymer [30,31] are sitting on one another to form a LF test strip. Each of these pads has the ability to transport the fluid such as blood, urine, or saliva impulsively. When the liquid solution is given to the LF sensor it quickly soaked the fluid and fluid reach to the second conjugate pad where freeze-dried bio-active particles, known as conjugates, are stored. These conjugates remain as salt-sugar matrix and contain all the reagents for the optimal chemical reaction between target molecule and antibody which is immobilized on the surface. As the fluid continues to flow and eventually it reaches to test line where it makes reaction with antibody and

produces an outcome like a color signal, which indicates the presence of target analytes e.g. a pregnancy test LF strips [29].

Lateral flow assay has many benefits. It is easy to use, cost-effective, reproducible, provides fast results, easy to store, does not require skill personals to handle it, and on-site detection. Some widely used LF strips are pregnancy test strip, glucose testing strip [27,28, 29].

1.3 Machine Learning

Machine Learning (ML) provides the system with the ability to learn automatically and improve itself from experience [57]. The main goal of ML is to develop a computer program or mathematical model, which has the ability to access the data and by using them, learn and improve themselves. This mathematical model is used to make predictions or decisions for a particular task. Machine learning has many applications some of them are, email filtering, image and speech recognition, medical diagnosis, predictions, fraud detection, etc [33,34,55].

Machine learning can be classified into three broad categories [37,38,39,40],

- **Supervised ML algorithms**: In this type of ML, the system is introduced with an example input and corresponding output data. The Objective of this process is to understand general rules and relationship which maps the input to output. Data used for learning purposes called training data. Supervised learning has many applications including risk evaluation, fraud detection, prediction of financial results, recognition of image objects, etc [37,38,55,59].
- **Unsupervised ML algorithms**: In this ML algorithm data has no labels or classifications, so ML algorithm must find patterns or structure by exploring the unlabelled data. Rather than finding the right output, the system describes the hidden structures. Item categorization, clustering customers, identify management, similar item recommendation are the common area where unsupervised ML used [38,39,55].

- **Reinforcement ML algorithms**: Reinforcement learning is a computer program that interacts with a dynamic situation where it must do a specific task. The system learns by a trial-and-error process which will lead to desired results. The main application of reinforcement learning in robotics, playing games, self-driving cars, etc [38, 40,55].

Chapter II.

Literature

2.1 Template Matching

Template matching is a method used to find similar features or a small part of an image by using a template image. Template matching is a process where numerical indexes are calculated by moving the template image within a large image in all possible directions. This calculation is an evaluation of how well the template image matches a source image. Template matching can be used in the industry to perform quality control or detecting edges of images [2,3,4,5,6].

In this work match_template function from the scikit-image library is used. This function uses fast, normalized cross-correlation to find instances of the template within the source image [7,58].

By normalizing the image and template vectors to unit length normalized cross-correlation matches the template. The basic definition of the normalized cross-correlation coefficient is shown in equation 1 [7].

$$\gamma(u,y) = \frac{\sum_{x,y}[f(x,y) - \bar{f}_{u,v}][t(x-u,y-v) - \bar{t}\,]}{\{\sum_{x,y}[f(x,y) - \bar{f}_{u,v}]^2[t(x-u,y-v) - \bar{t}]^2\}^{0.5}} \qquad \text{eq.(1)}$$

where, $f(x,y)$ is the original image, t is the template, $\bar{t}$ is the mean of the template and $\bar{f}_{u,v}$ is the mean of $f(x,y)$ in the region under the template. (x,y) and (u,v) represent pixel coordinates [7,76]. $\bar{f}_{u,v}$ can be express as [14]

$$\bar{f}_{u,v} = \frac{1}{N_x N_y} \sum_{x=u}^{u+N_x-1} \sum_{y=v}^{v+N_y-1} f(x,y) \qquad \text{eq.(2)}$$

Though template matching is a very effective technique, it also has some limitations. Below listed some limitation of template matching [16]:

- A small change in size, shape, or orientation can create problems in the final results.

- Template matching process could be very expensive especially if more than one template uses to search the entire image.
- A strong computational power requires for detection of patterns of the image to reduce the time.
- Template images which used to find a similar area of a large image could be different size and this could bring poor results.
- Template matching process could be easily parallelized.

2.2 Regression Analysis

Regression analysis (RA) is a statistical process, which finds the relationship between a dependent variable and one or more independent variables. Linear regression (LR) is the most popular and widely used regression analysis. LR finds the most closely fit line to a data point [8]. Regression analysis generally express as [15]

$$y = \beta_0 + \beta_1 x_1 + \epsilon \quad \text{eq.(3)}$$

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \epsilon \quad \text{eq.(4)}$$

Equation 3 is a simple linear regression and equation 4 multiple linear regression. where y is the target or dependent variable, x_1 and x_p independent variable, β_0, is y-intercept and β_1, β_p are coefficient of x_1 and x_p respectively, also known as the slope of the regression line, $\in$ is the error or residual [15].

Mainly regression analysis (RA) is used for two distinct purposes. First, RA is used for prediction and forecasting, which clearly overlay with the machine learning field. Second, it used to find causal relationships between the target variable and dependent variables. To use RA either of this case, relationships of dependent and independent variables must justify in the sense that why they have predictive power and why the relationship of these two variables has causal interpretation [8,9,10].

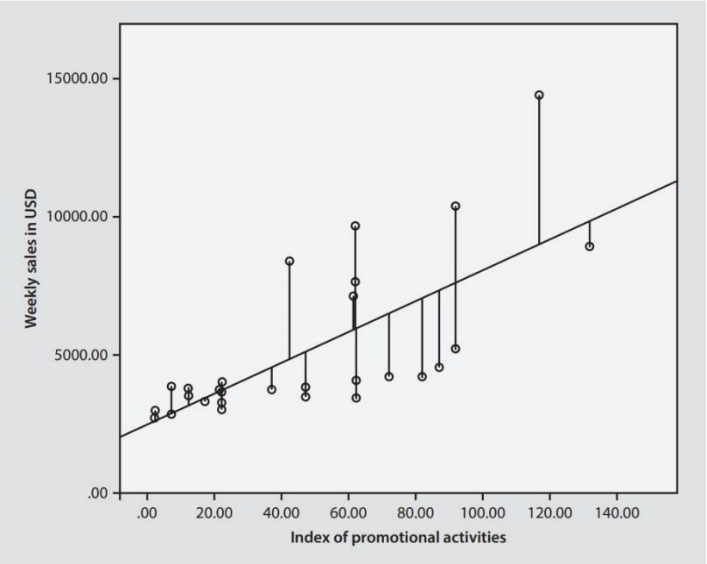


Figure 2: Representation of regression analysis [15, (pp:212)]

Figure 2 shows the regression analysis results, where dependent variable y is, the weekly sales in USD, plotted against the independent variable x, index of promotional activities. The Perpendicular line to fitted lines is the errors [15].

2.2.1 Types of regression analysis

There are various types of regression analyses available for various purposes. Below some of them will be discussed briefly.

2.2.1.1 Linear Regression

Linear regression modelled the relation between a dependent variable and one or more independent variables by fitting a linear equation. In this regression analysis, a relationship between one dependent variable and one or more independent variables establishes using a regression line [41,42].

When only one independent variable used to find a relationship with one dependent variable, it is then called simple linear regression and when more than one independent variable is used it is called multiple linear regression. Equation 3 & 4 represent simple and multiple linear regression respectively [42].

2.2.1.2 Logistic Regression

Probability of event to success or event to failure is investigated with logistic regression. Logistic regression is used if the nature of the dependent variable is binary or Boolean or should answer with yes or no. The predicted outcome of the dependent variables ranges from 0 to 1 [43].

To map probabilities from predicted values a sigmoid function is used. Sigmoid function maps any real value into a value between 0 to 1. Mathematically sigmoid function is defined as [44]

$$S(z) = \frac{1}{1 + e^{-z}} \qquad eq.(5)$$

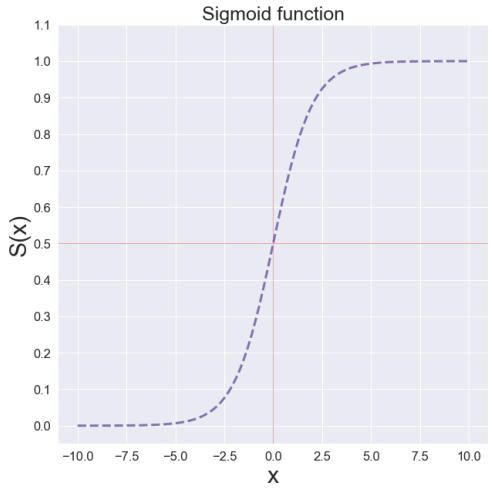


Figure 3: Sigmoid function, generated using equation 5

Figure 3 shows a general sigmoid function. When the value of "x" is less than zero value of S(x) below 0.5 and when the value of x greater than 0 then S(x) is greater than 0.5.

2.2.1.3 Polynomial Regression

Polynomial regression is a type of regression where the relation between dependent and independent variables modelled as n^{th} order polynomial. Polynomial regression fits a non-linear relationship between dependent and independent variables. Below equation represent a polynomial regression [26,79].

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon \qquad eq.(6)$$

where y is the target or dependent variable, x independent variable, $\beta_{0,}$ is y-intercept and β_1, β_p are coefficient of x and x^2 respectively, $\in$ is error or residual.

2.3 Evaluation

Evaluation of a machine learning (ML) model is the process of evaluating how well the ML model fits the data point. It is an estimation of the ML model accuracy. Evaluation helps us to identify how the ML model will perform on future data.

There are several methods available and widely used to calculate the difference between the actual value and the predicted value. Some of them are Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Median Absolute Error (MAE), Coefficient of determination (COD), also known as R^2 or $R\ squared$, adjusted R^2, residual plot, etc [17].

2.3.1 Mean Squared Error

Average squared difference between the actual value and the predicted value is the MSE. To calculate MSE first we take the square difference between the actual value and predicted value, then summed them together. After that, we divided this with the number of observations. Equation 7 define MSE mathematically [17, 45, 46,77],

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad \text{eq. (7)}$$

where

- y_i is true value
- $\hat{y}_i$ is predicted value.
- n is no. of observation

MSE is an estimation of how close the fitted line to the data point. Smaller the value of MSE indicates the better fit ML model to the data point [46,47].

2.3.2 Root Mean Squared Error

The square root of MSE is the RMSE. It can be defined as, the square root of the average squared difference between the actual value and the predicted value. Equation 8 define RMSE mathematically [17, 48, 60,77]

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \qquad eq.(8)$$

where

- y_i is true value
- n is no. of observation
- $\hat{y}_i$ is predicted value.

RMSE indicates how well the ML model will perform. Small value of RMSE indicates better performance of the regression model and large value indicates poor performance [78].

Though both MSE & RMSE are very popular and widely use to evaluate regression model, it has one drawback. Both evaluation methods are very sensitive to outliers. So, to use MSE & RMSE effectively outliers should be removed [46, 48].

2.3.3 Mean Absolute Error

MAE is the average absolute difference between the actual value and the predicted value. MAE is similar to MSE in the sense that, in MAE we take absolute value rather than a squared value. On the other hand, MSE takes the squared value. In both cases goal is to take non-negative values. Equation 9 shows the mathematical definition of MAE [17, 77]

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \qquad \text{eq.}(9)$$

Compare with MSE / RMSE, MAE can handle outliers better. So, a large error does not have much influence on small errors and provides much better results than MSE & RMSE [17, 78]

2.3.4 Coefficient of Determination (R^2)

Coefficient of determination or R-Square is the percentage of variation in the dependent variable, which will be predicted by using independent variables [49,83]. It provides a measure of how accurately observed values could be predicted using a regression model [50, 51, 52]. Value of R^2 gives the information on, strength of the relationship between the independent variable and dependent variables of the ML model [49,71].

The Coefficient of Determination (COD) normally ranges from 0 to 1. Where 1 represent perfectly fitted regression line and 0 represent no fitting at all [49,53,61,83]. Mathematically COD defines as [15, 71,83]

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{m}(y_i - \bar{y})^2} \qquad eq.(10)$$

where

- y_i is true value
- $\bar{y}$ is the mean of true value
- $\hat{y}_i$ is predicted value.
- m is no of observation

Numerator of equation 10 called Sum of Squared Error (SSE) and the denominator is called Sum of Squared Total (SST) [15, 49,83].

SSE is the sum of the squared difference between the actual value and predicted value [15]. SSE gives an indication of the model's prediction. A smaller value of SSE indicates good prediction [15, 54].

SST is the sum of the squared difference between the actual value and the mean of that actual or observed value [15]. It measures how the data varies around a certain number, what is the variation of observed data to its mean [15, 54].

2.3.5 $Adjusted\ -\ R^2$

It is not recommended to use R^2 to evaluate the ML model. When additional features added to the ML model, the value of R^2 will increase, even though those independent variables are not useful to the ML model [73, 74,83]. So, the bigger value of R^2 does not mean a good ML model. A good ML model can have a smaller R^2 value [18].

To interpret the goodness of the regression model, adjusted-R^2 is a much better indicator compares to R^2 [74]. Like R^2, $adjusted - R^2$ also shows how well a ML model can predict target variables, but it adjusts for the number of independent variables in the ML models [73,74,83]. Mathematically $adjusted - R^2$ can be defined as [15,49,83]

$$R^2_{adj} = 1 - \left[\frac{(1-R^2)(n-1)}{n-k-1}\right] \qquad eq.(11)$$

where

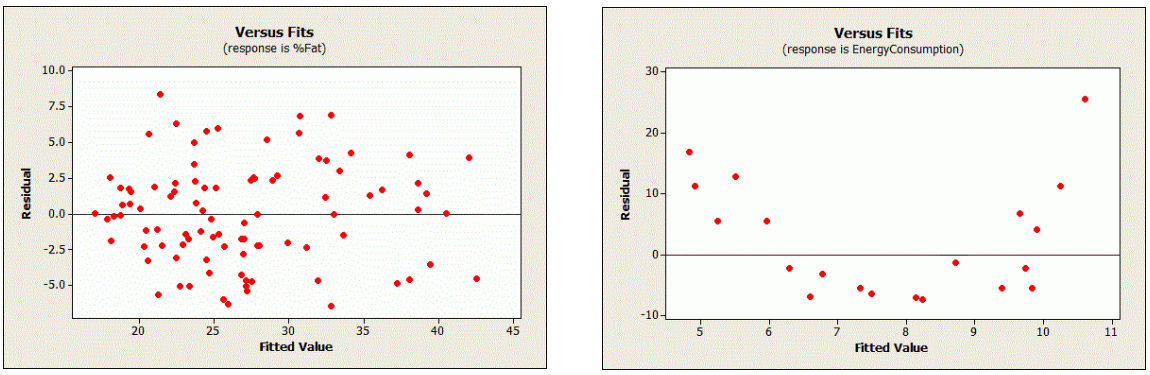
- n is no. of observation
- k is no. of independent variables

If the number of independent variables, denoted by k, increased, $adjusted - R^2$ will be decreased. $Adjusted - R^2$ only increases after adding one more independent variable unless those variables are truly useful to the ML models and eventually, they increase R^2 and which results in an increase in $adjusted - R^2$ [18,49].

2.3.6 Residual Plot

A residual plot is a type of visualization that shows dependent variables in the horizontal axis and residuals, which is the difference between the actual value and predicted value, in the vertical axis. It's a representation of how far the predicted value from the actual value in the vertical direction. If the residual value is above zero lines in the vertical axis, then the predicted value is bigger than the actual value and if the residual value is below zero than prediction is lower than the actual value [19,79].

If the distribution of data points on the residual plot shows a random nature or if they don't have any pattern around the horizontal axis, then the regression model is fitted properly to the data points [19, 20,79].



a b

Figure 4: Two residual plot, which plotted against the predicted value. a. Randomly distributed residual point around the horizontal axis which tells goodness of fit to the data point. b. Distribution of data point not random and shows a certain pattern that describes the regression model does not fit correctly [20].

2.4 Validation

Evaluation of the ML model could be tricky and sometime may be misleading. Specially we split the data to train the ML model and the same data used to test. But the question is how well the ML model will perform if an independent data set given to the ML model to predict. Different evaluation methods used to evaluate the accuracy of the ML model. But these evaluation methods however not very reliable. K-fold cross-validation could be used to overcome this problem and also an essential part of model selection [79].

Machine learning (ML) model validation is a step in which a trained model is being evaluated on the test data set. Validation process answer question how well the ML model will perform with an independent data set [55,56].

2.4.1 K-fold Cross Validation

Skills of the ML model could be estimated by cross-validation (CV). CV is a statistical method and widely used in the field of applied machine learning. It is a resampling method, used to select a ML model on a limited data sample for a predictive modelling problem. Cross-validation gives us a generalization view of how well the ML model will perform with an independent data set. CV is easy to understand, easy to

implement and results which produced by CV have very low bias compare to other methods [36, 60].

In k-fold cross-validation, iteration is done on a data set k^{th} number of times. In each round, data set is split into k parts. One part is for validation and remains k-1 part are for training purpose. Figure 5 shows a 5-fold cross-validation [35].

If k=5, then it will result in five different fitted models. These models are then fitted with the overlapping training data sets and then evaluate this model with a distinct validation set. Performance of cross-validation is the mean of k performance form validation sets. For training and test purpose k-fold cross-validation uses all the dataset [35].

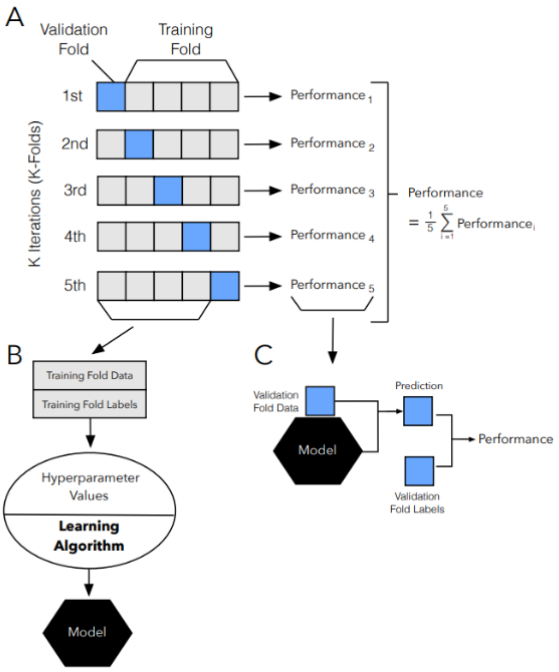


Figure 5: Illustration of the k-fold cross-validation, where k=5. [35, (pp:24)]

Below present the basic steps in K-fold cross-validation [22]

1. First, it randomly split the entire dataset into k equal subset. Each subset is known as a fold.
2. For each K-fold, a model of K-1 folds of a dataset will be built. After that model will be tested for the effectiveness of K^{th} fold.
3. For each prediction error will be recorded.

4. Repeat this process until k-fold served as a test set.
5. The average of all errors called cross-validation error and serve as performance metrics.

Chapter III.

Materials & Methods

3.1 Data

Mainly two types of solutions were prepared to do the experiments with gold nanoparticle lateral flow assay. For calibration, different concentrations, which are 1, 20, 40, 60, and 100 nmol L^{-1}, of digoxigenin solutions were prepared with phosphate-buffered saline. Second digoxigenin solutions were prepared with human serum with 0, 1, 5, 10, 15, 20, 25, 30, 40, 60, 80 and 100 nmol L^{-1} concentration [1].

Two different image acquisition systems were used to capture the images. One type of image was captured with Bio-Imager, which is ChemStudioPlus,16MP CCD-camera, and another type captured with iPhone 5S which is a CMOS smartphone camera [1].

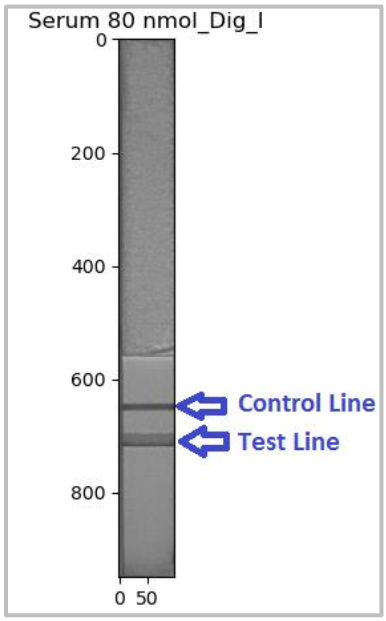


Figure 6: Resulted cropped Lateral flow strip from the first step of two steps of the cropping process

In the image mainly there are two areas of interest. One is called the test line, and another is control line, Figure 6. Test line is composed of digoxigenin-hapten and control line is composed of the secondary antibody for mouse Fc-fragments [1]. If digoxigenin

content increase in the samples, it reflects a strong coloured control line and light-coloured test line, Figure 7. Distance between the test line and the control line is 5 mm [1].

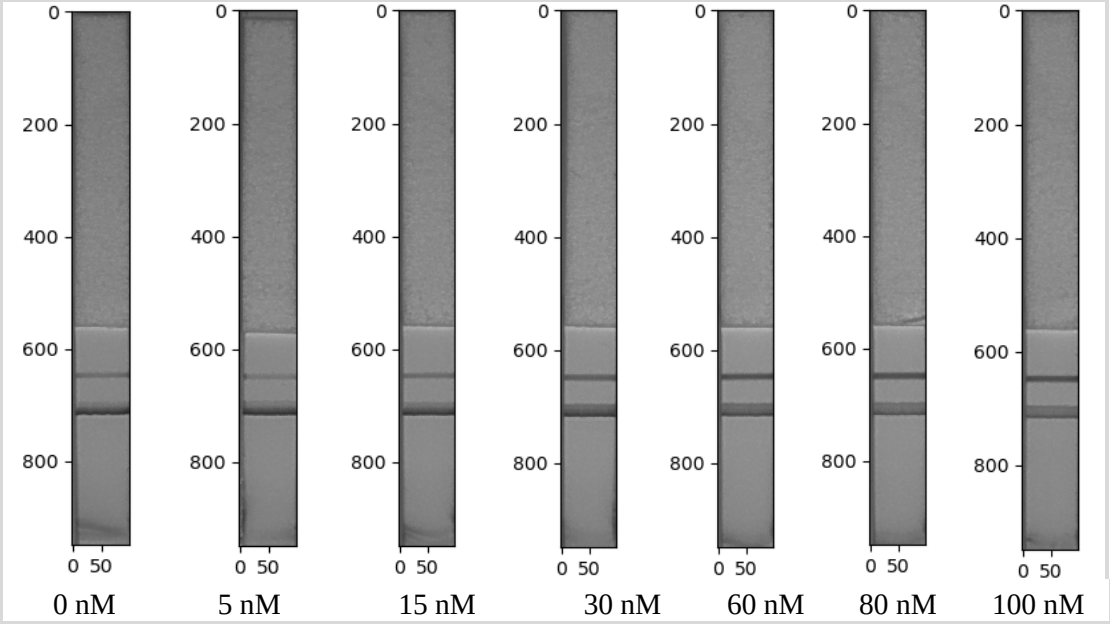


Figure 7: Lateral flow tests-strip for different concentration with different intensity

3.2 Python Programming

Python programming language, version 3.7, was used to analyse image data. Python was created by Dutch programmer Guido van Rossum and first released in 1991. Python is a very popular programming language, especially in the field of data science. Python is an interpreted, high-level programming language [62, 63].

Several Python libraries were used in this work. Including scikit-image [65], which is an image analysis tools, scikit-learn used for machine learning [64], SciPy is for scientific computation [66], StatsModels for statistical modelling and statistical tests [67], for data visualization matplotlib [71] & seaborn [70] was used, NumPy for numerical analysis [68] and Pandas for data frame manipulations [69]. All these libraries are open source and are widely used.

To analyse data a three-step algorithm was developed. In the first step, a two-step automated cropping technique is developed to crop the regions of interest (ROI). Next, this cropped image converted into a data frame and lastly a machine learning algorithm developed to predict the concentration of the sample from image intensity.

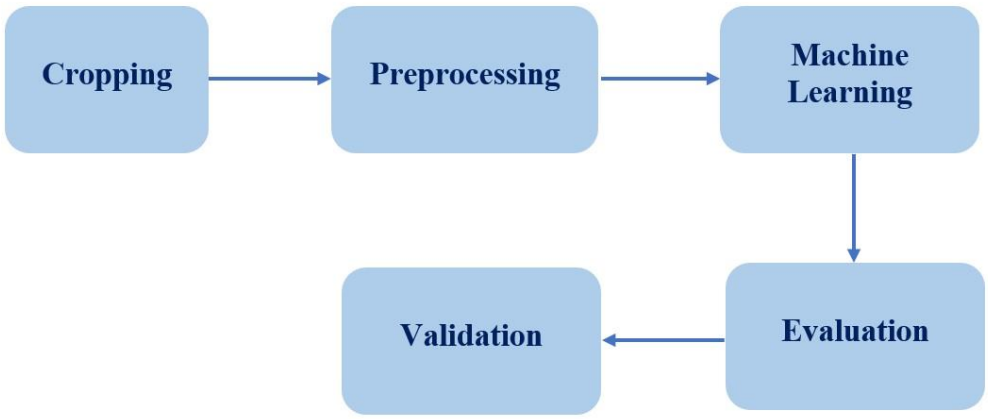


Figure 8: Image analysis algorithm flow diagram

Figure 8 shows different stages which developed to achieve results.

3.3 Cropping Region of Interest

Developing a proper cropping algorithm to crop-out region of interest (ROI) was an essential part of this work. Which should be done automatically. A two-step algorithm was developed to crop the ROI, test line and control line.

In the first step, whole lateral flow strip was cropped by using a template matching function called match_template [58] from scikit-image library [65], figure 6 & 7. Two template images were created manually, one is for imager images, figure 9 (a), and another for iPhone images, figure 9 (b). Using these two templates similar area searched all possible directions within the source image. When searched result found a similar feature in the large image it extracts that particular area and returns the result. Here is the sample code:

$$\boldsymbol{result = skimage.features.match\_template(image, template)}$$

In the scikit-image library, skimage, under features module, match_template function was called. This function takes two parameters. First, image which is the input image and second template, is the ready template image which will match a small area within the input image.

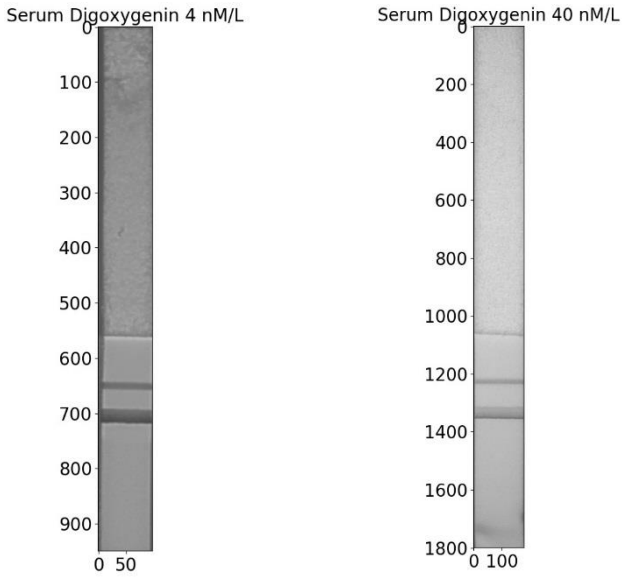


Figure 9: Template images, a. template for imager image, b. template for iPhone image

Figure 9 shows two template images, figure 9(a) is used to crop Imager images and figure 9(b) is used to crop iPhone images.

From the return result then we calculate the indexes of row and column. These two indices identify where in the source image matched area with temple image begins. Using these two calculated indices, we then able to select or slice a specific area with the source image and save it to a variable. Below code do the process:

$$\boldsymbol{crop\_img} = \boldsymbol{image}[\boldsymbol{y}:\boldsymbol{y} + \boldsymbol{imager\_temp.shape}[\mathbf{0}], \boldsymbol{x}:\boldsymbol{x} + \boldsymbol{imager\_temp.shape}[\mathbf{1}]]$$

where, image: source image; x: starting index of column; y: starting index of row; imager_temp: template image of Imager image; imager_temp.shape: return the shape of the template image.

In the next step two ROI, test line & control line, will be cropped by applying different conditions and masks on the resulted image from the previous step. At first, we

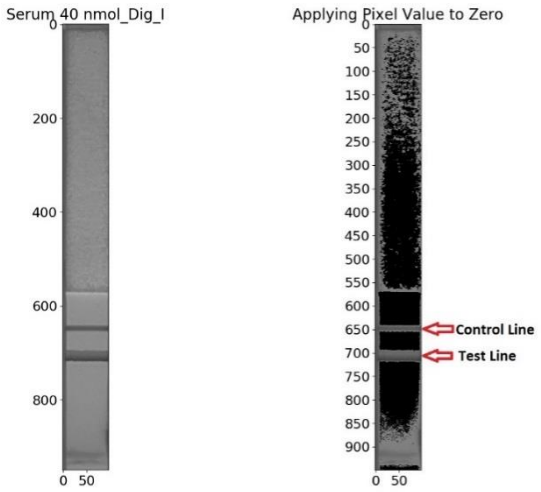
apply zero value to all the pixels whose value greater than the mean value of the source image by this command.

$$\boldsymbol{crop[crop > (crop.mean() + (crop.min())/15)] = 0}$$

where, crop is the resulted image from the previous step. crop.mean() & crop.min() return mean and minimum value of crop image, this command, $crop > (crop.mean() + (crop.min()) / 15)$, returns a Boolean value. When the value is true and it selects certain pixels within the image named crop and assign zero value to those pixels and save it to the original image.

Both the test line and control line much darker than the rest of the image area. So higher intensity value will fall on the above condition and their value will be replaced by zero. This process was very effective, and it works with all the image. Figure 10 shows the result.

Figure 10 shows, a. first cropped image using template image and b. shows after applying pixel value to zero. From the figure, we can see both the control line and test line are unaffected by the black pixel value because of their pixel value are much lower than the mean value of the entire image.

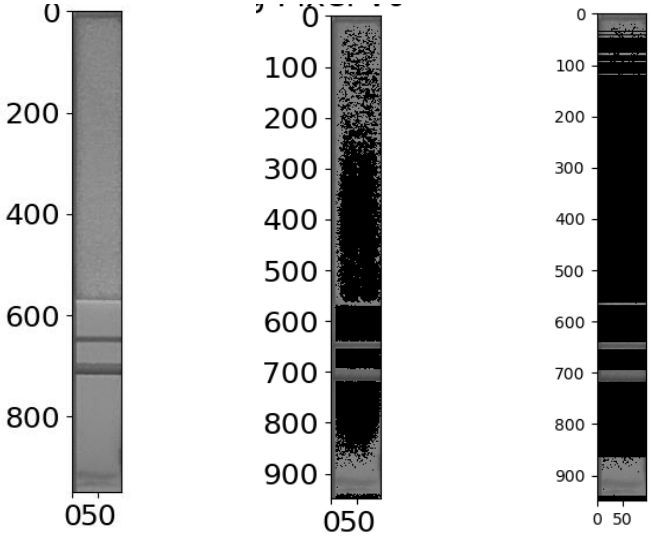


a b

Figure 10: Applying pixel value to zero, a. first cropped image b. after applying pixel value to zero.

In the next step, we will amplify the black pixel if it is fall in a certain condition. The idea is each row will scan and count the number of black pixels. If the number of black

pixels more than 20 we will apply zero value to all the pixels on that particular row. Figure 11(c) shows the result:



a b c

Figure 11: a. Result after cropping step 1, b. applying pixel value to zero where pixel value greater than the mean of the image c. Applying entire row pixel value to zero if the number of black pixel to that row more than 20.

Next, image was reshaped and sliced from half to all the way to end. Figure 12 shows the result.

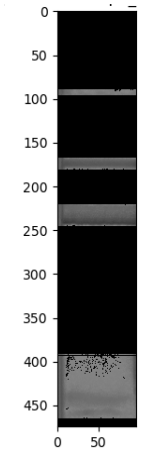


Figure 12: Reshaped Image

In the next step of the cropping process, we have to go through a couple of steps. From figure 12 we can see, couple of places not covered with the black mask. But from this image, we only crop two ROI, region in between index 150 to 250, see figure 12.

First, we will look for those indexes, whose corresponding total pixel value not equal to zero. If this condition full fill, we will gather those row indexes and use for the next step. Table 1 shows a Python list which contains all the indexes whose corresponding all pixel value summation not equal to zero.

Table 1 List of indexes which correspond to the sum of all pixel's value greater than zero

[91, 92, 93, 94, 95, 96, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 393, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465]

Next, within this Python list, we will look for indexes, which is much bigger than the previous index. For example, index value 168 is much bigger than the immediate previous index value 96. Row value 168 might be the beginning of one of the ROI.

We are looking for an index which has a different value than the next index. In our case, we said if preceding index value is greater than 49 compared to the previous index value, then we collect both of these indexes in this step. After this step, number of indexes reduced to maximum of 6 to 8. From the above python list, we able to separate six index value 168, 181, 222, 393, 393, 395.

Next, we apply another condition to separate only two indexes by saying if the difference between one index to immediate next index less than 45 but greater than 15 then we collect this two. After applying this idea, we able to separate two indexes which are 181 and 222.

Now we have two indexes. Using these two indices we are going to scan and search for ROI. First index will give us the control line and the second index will give us the test line. First, we will assign 181 to two variables. Using these two variables we will scan both upward, index value will be reduced to a certain value, and downward, index value will increase to a certain value, direction.

The idea is if we assign 181 to any variable n and if the sum of all pixel value at index n not equal to zero then we will reduce n by one, $n = n - 1$, and we will continue until the above condition is true. If the condition is false loop will stop and we will have a new value for n.

We will do the same but this time downward and we will increase the n by one, n = n+1. If this condition is true, we will have a new value for n downward direction. Using this new value of n, we can slice a certain portion of the image. The same Idea was implemented to crop test line from the image. Figure 13 shows two ROI, control line and test, separated from the input image.

These two steps cropping method able to successfully crop 99% of the images. But it should be noted that the algorithm developed to crop the image in this work highly dependent on the template image. Incorrect template image brings poor results.

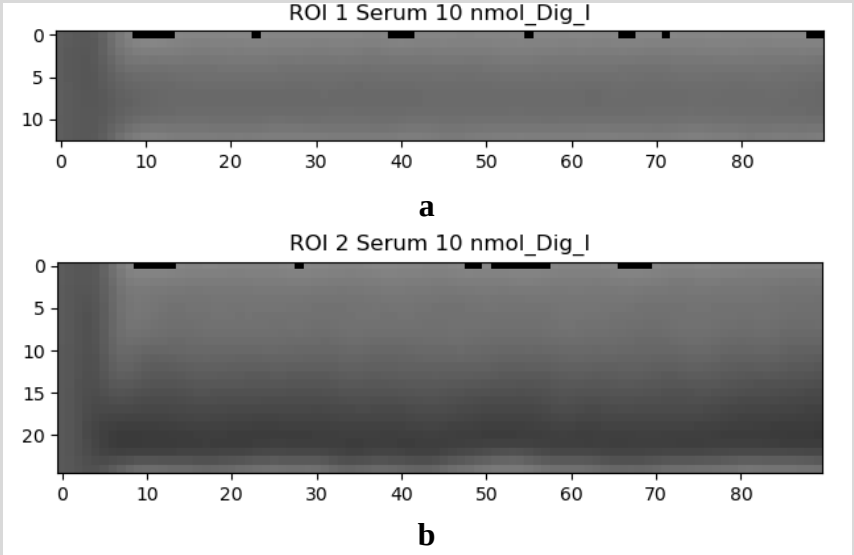


Figure 13: Cropped both ROI. a. Control line ROI, b. Test line ROI

3.4 Data Frame

In the data frame, eleven features are created. Mean of control & test line, median of control & test line and standard deviation (SD) of control & test line, ration of control line means to test line mean, ration of control line median to test line median, ration of control line SD to test line SD from each region of interest. Another feature is created by

taking the ratio of the width of the control line to width of the test line. This feature is important because just by looking at figure 7, simply it can be said that if the concentration of digoxigenin increases, the width and intensity, dark colour, of the control line clearly increases. Last feature is the target feature, which is concentration. Table 2 shows the head of the data frame with 6 rows and 12 columns.

Table 2: First few rows of Data Frame

ImageName	*ControlLine mean*	*ControlLine median*	*ControlLine std*	*TestLine mean*	*TestLine median*	*TestLine std*	*Ratio_of CL_to_TL*	*Ratio_of mean*	*Ratio_of median*	*Ratio_of std*	*Concentration*
0-I DS_P_Dig_I	13,35432	5,075174	3,96791	5,098702	5,010635	58,68377	0,819441	1,043404	1,032796	0,949209	0
0-II DS_P_Dig_II	13,35472	5,081404	3,993391	5,093553	5,010635	59,33419	0,952009	1,046125	1,036018	0,956095	0
0-III DS_P_Dig_III	12,92761	5,043425	4,040015	5,181346	5,129899	61,0914	0,025318	0,969177	0,957685	0,964468	0
1-I DS_P_Dig_I	12,38472	4,955827	4,422614	5,068827	4,962845	60,96991	0,942043	0,98221	0,996497	1,168963	1
1-II DS_P_Dig_II	13,25434	5,398163	4,515787	5,081956	4,983607	60,11641	1,047319	1,0443	1,230325	1,233372	1
1-III DS_P_Dig_III	13,43196	5,105945	3,998167	5,092768	4,990433	59,60233	0,97456	1,052589	1,059457	0,956223	1

3.5 Feature Engineering

Feature Engineering is a process of improving machine learning model performance by addition, groping, merging or transforming features of the data frame [80]. Couple of feature engineering is done in this work. Logarithm transformation, square root transformation of different variables whose data distribution has positive or negative skewness and lastly merging similar data on data frame.

3.5.1 Logarithm Transformation

It is widely used methods in feature engineering to remove or reduce the skewness of data and make the distribution close to normal [81]. Below figure 14 shows the distribution of data before and after logarithm and square root transformation.

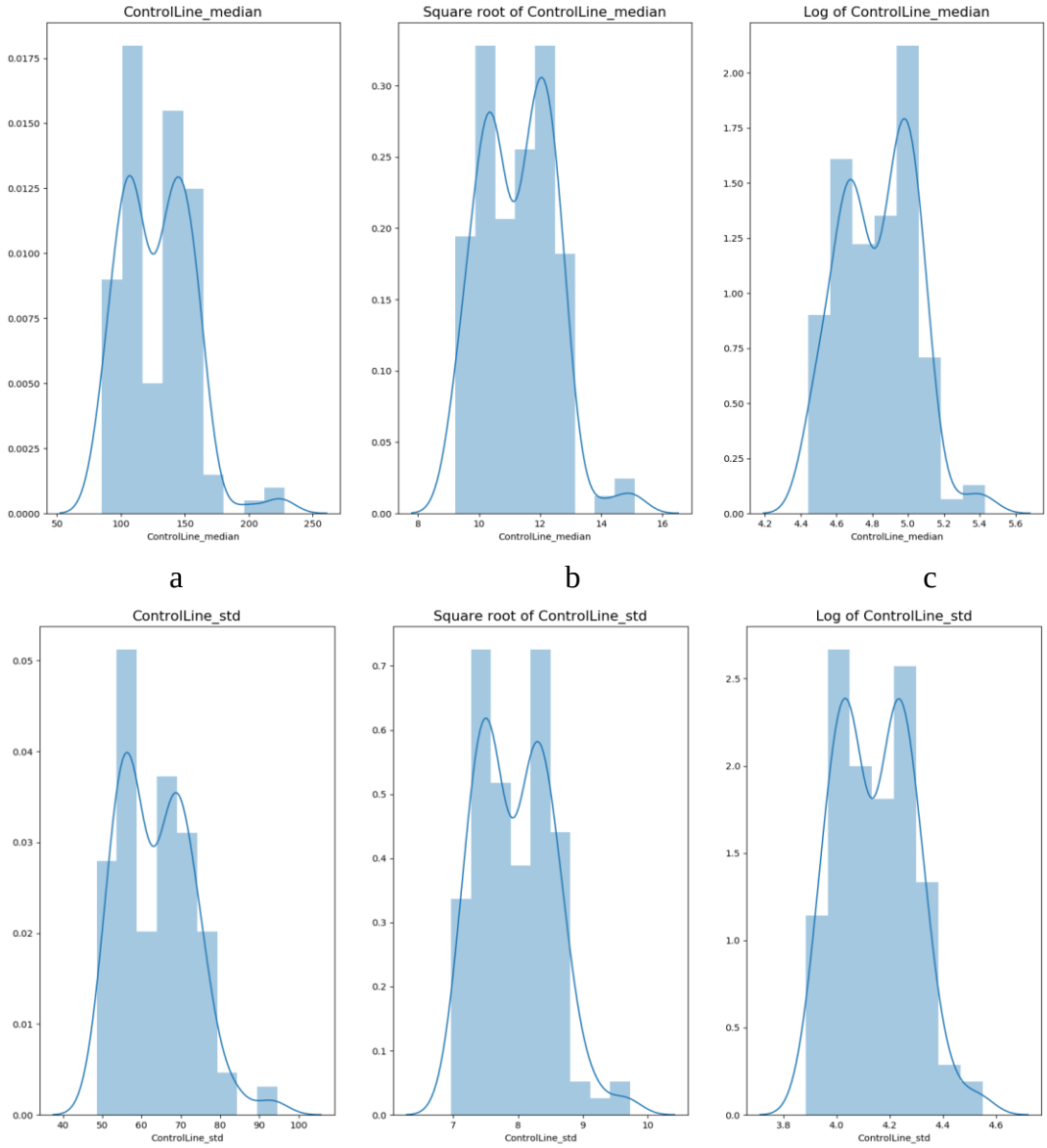


Figure 14: Data distribution of control line median and control line standard deviation. a. without transformation b. after square root transformation c. logarithm transformation.

Figure 14 shows, distribution and histogram of two features, control line median & control line standard deviation. From the control line median, it is clear that after logarithm transformation (c) distribution of data more symmetric compare to without any transformation (a) and square root transformation (b). Same result for control line standard deviation.

3.5.2 Merging Similar data

Images which show similar characteristics, values almost in the same range, they will be merge together and mean value will be collected. Below figure show data with image name show similar nature of data

	ImageName	ControlLine_mean	ControlLine_median	ControlLine_std	TestLine_mean	TestLine_median	TestLine_std	Ratio_of_CL_to_TL	Ratio_of_mean
0	0-I DS_P_Dig_I ROI_1	13.354320	5.075174	3.967910	5.098702	5.010635	58.683770	0.819441	1.088693
1	0-II DS_P_Dig_II ROI_1	13.354718	5.081404	3.993391	5.093553	5.010635	59.334190	0.952009	1.094378
2	0-III DS_P_Dig_III ROI_1	12.927609	5.043425	4.040015	5.181346	5.129899	61.091396	0.025318	0.939305

Figure 15: Part of the data frame shown three ROI from three images with different features

Figure 15 shows a small part of the data frame. Three images converted to different features. If we look closely, all the values within one feature shows almost similar range of value. Using all the image with similar characteristics for machine learning will generate poor results.

Before merging similar data, the data frame (DF) had 126 rows and 11 columns and after merging it become 37 rows and 11 columns. This merging also brings very good results. Using both DF, a machine learning model was developed, and we calculate R^2 and adjusted R^2. After merging similar rows both R^2 and adjusted R^2 value increase significantly. This comparison will discuss in the results section in details.

3.6 Machine Learning

Before selecting a final machine learning (ML) model, we built two ML model using all the features created during data frame development process. The main goal is to test the data and observe the performance of the ML model before and after doing feature engineering of data.

3.6.1 ML Model with Original Data frame:

In the data frame, we have 126 rows and 11 columns. Using all the features from the data frame we developed first machine learning (ML) model to test the performance. To do that first we select all the features except target variable from the DF and assign to

a variable X. The target variable is assigned to y. Then the DF was split into 70% for train and 30% for test using below command.

$$X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.30, random\_state = 0)$$

Where $train\_test\_split$ function was called from the $sklearn$ laibrary under $model\_selection$ module. This function takes a couple of variables. X is features; y is target variable and $test\_size$ is the percentage of splitting want to perform.

After that, a linear regression object is created using $LinearRegression()$ function and name it as lm. Next, we train our model using $lm.fit()$ command. This command takes two parameters, X_train and y_train. X_train is 70% of all features value and y_train 70% of target values. After train the model we predict the target variable using $predict()$ function from the scikit-learn library, which take only one parameter, X_test.

Now we have to evaluate our machine learning model. As we discuss in the literature section there are different methods for model evaluation. In this work, we calculate different errors, R^2 *and adjusted* R^2.

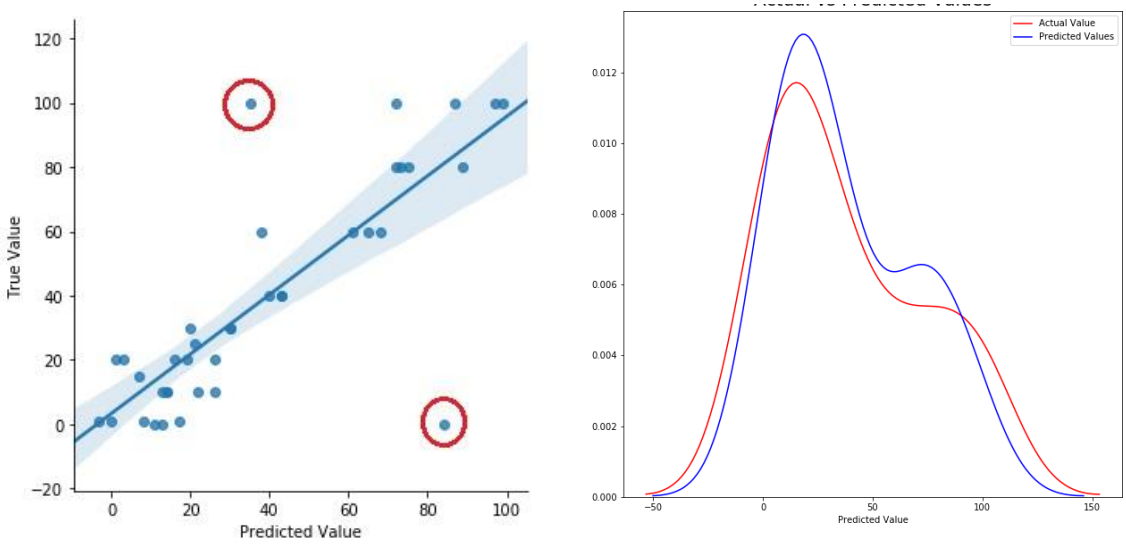
Table 3 Evaluation matrix of the first ML model

Mean Square Error	**= 393.16**
Root Mean Square Error	**= 19.83**
Mean Absolute Error	**= 11.22**
Median Absolute Error	**= 6.52**
R^2	**= 0.66**
Adjusted R^2	**= 0.56**

From the above result, we can say our model performance is quite good. Both R^2 *and adjusted* R^2 in the same range.

Figure 16 shows two visualizations based on the first ML model. First plotted actual value vs predicted value. Which shows a narrow confidence interval area around the fit line. From this plot, we also see two outliers, red circle, which affect the position of the regression line.

Second plot is the distribution of data. Which show two graphs, one is the actual value, and another is predicted value. This graph shows how far the predicted value from the actual value.



a b

Figure 16: First Machine Learning model evaluation, a. True value vs Predicted value using lmplot function from the seaborn library. b. Distribution plot which shows how far predicted value from the actual value.

3.6.2 ML with merged Data frame:

Next, we developed another machine learning model on the merged data frame. After performing merging data frame shrink in size. The shape of the new data frame is now 37 rows and 12 columns. We follow the same procedure as the last ML process. After performing machine learning on the new data frame, we did the same ML model evaluation. Preformation of the model shows significant compare to the last model. Below table shows the performance matrix of both ML modes.

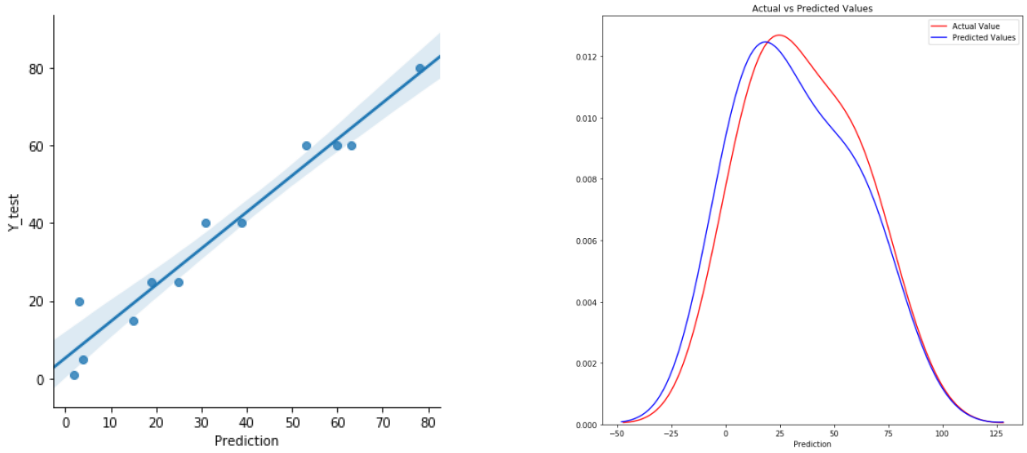
Table 4 Comparison of ML model performance with original DF and the data frame with featured engineered

Evaluation	Original DF	Merged DF
Mean Square Error	393.15	39.62
Root Mean Square Error	19.82	6.29
Mean Absolute Error	11.22	3.94
Median Absolute Error	6.52	1.33
R^2	0.66	0.93
Adjusted R^2	0.55	0.62

Table 4 shows significant improvement after performing ML on the merged data frame. All the errors reduced significantly and R^2 value also increases significantly. But adjusted R^2 increased only a small amount.

Figure 17 show also indicate very good performance of the ML model with the merged data frame. Figure 17 (a), which is plotted predicted value against actual value, shows most of the point very close to the regression line.

Second plot, Fig 17(b), show predicted value almost closely follow the actual value. Both the plot indicates good performance of this ML model.



a b

Figure 17: Machine Learning model evaluation, a. True value vs Predicted value using lmplot function from the seaborn library. b. density plot which shows how accurate or how far predicted value from actual value.

3.6.3 ML Model Selection

After building and evaluation of two ML model using all the features from the data frame, in the next step, we will select the final ML model. For that, we will select one and/or more feature/s from the data frame and build a ML model, and then we calculate both R^2 and adjusted R^2 of that ML model.

At first, we select each feature from the DF and calculate R^2 and adjusted R^2. Single Feature, which produces the biggest adjusted R^2, that feature will be selected for the ML model. Next, we will again iterate this process and this time we combine one more feature

with the previously selected feature from the last iteration. The combination which will give the highest value of R^2 and adjusted R^2 that will be chosen for the final model. This process will be continuing, and we will select maximum five features out of ten.

To execute this idea an iterative process, for loop, was created and run to a python list which contains the name, column name, of all features from the data frame. Below python list shows all the features name.

```
colums = ['ControlLine_mean', 'ControlLine_median', 'ControlLine_std',
          'TestLine_mean','TestLine_median', 'TestLine_std',
          'Ratio_of_CL_to_TL', 'Ratio_of_mean',
          'Ratio_of_median','Ratio_of_std', 'Concentration']
```

We will run a for loop on that list with this ***for col in colums***: command. Which will select each feature form the above python list and with this, $P = df2[[col]]$, command it will create a data frame with only one feature. After that we will split the data frame using this command, $X\_train2, X\_test2, y\_train2, y\_test2 = train\_test\_split(P, q, test\_size = 0.30, random\_state = 0)$. Where 70% of data will be used for training purpose and 30% for testing.

In the next step, we will train the model and after that, we will predict the target variable. Our goal is to find the feature which produces the biggest $adjusted\ R^2$. To do that each time we predict the target variable using only one feature we will calculate $adjusted\ R^2$, using this command $r2\_adj = r2 - (n\_parameters - 1)/(y\_train2.shape[0] - n\_parameters) * (1 - r2)$, and then we store it in an empty list. In the end, we will have ten calculated $adjusted\ R^2$.

Now we will plot these values to find out which feature produces highest $adjusted\ R^2$. Figure 18 shows the steps used to select final ML model.

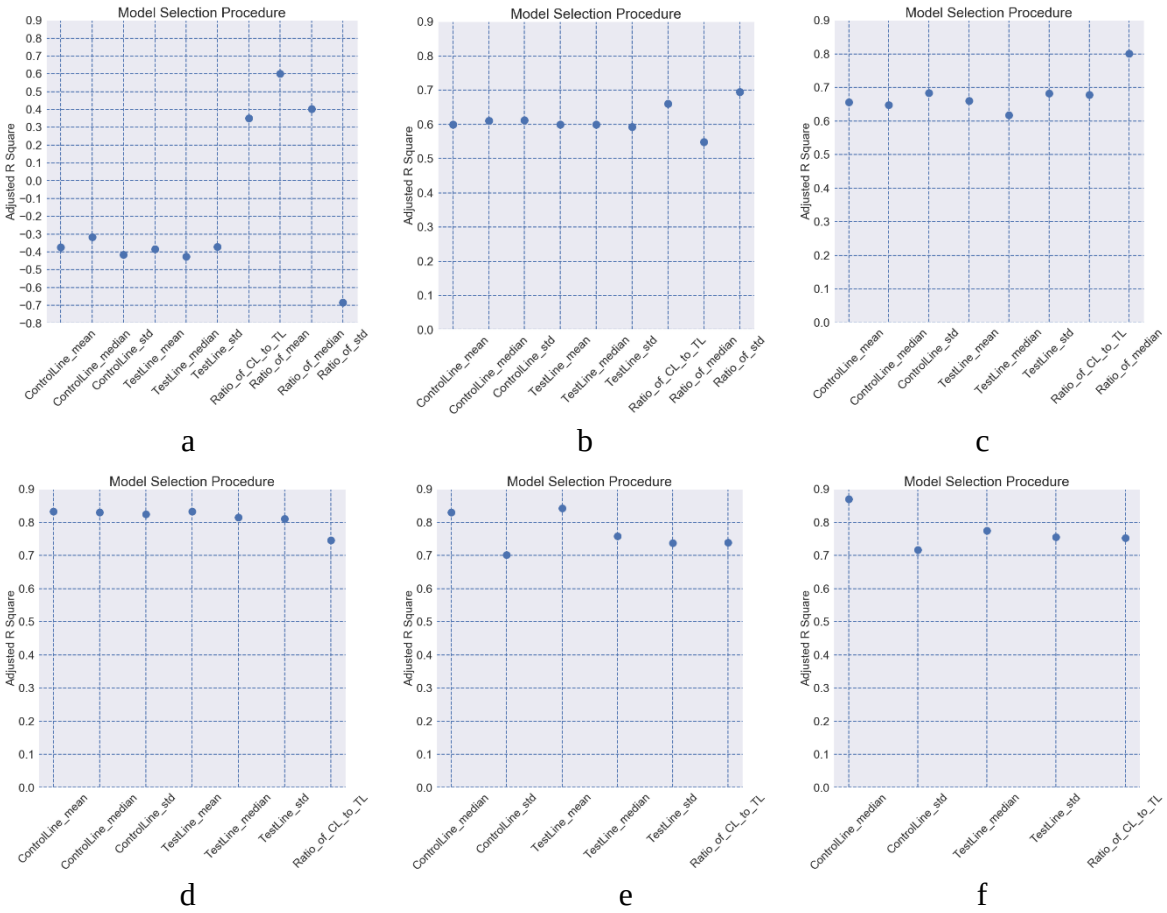


Figure 18: Model selection process from a to f. a. Using only one feature each time calculates adjusted R^2 and find out which feature return highest adjusted-R^2 b. Combine with "Ratio_of_mean", which return highest adjusted R^2in the last step, remaining each feature use to develop DF, predict target variable and calculate adjusted R^2. c. Combine with "Ratio_of_mean" & "Ratio_of_std",, d. Combine with "Ratio_of_mean", "Ratio_of_std" & "Ratio_of_median" e. combine with "Ratio_of_mean", "Ratio_of_std", "Ratio_of_median" &"ControlLine_mean", f. combine with "Ratio_of_mean", "Ratio_of_std", "Ratio_of_median", "ControlLine_mean" & "TestLine_mean"

Figure 18 shows six graphical representation of the $adjusted\ R^2$ value. Where, a. each time only one feature is used and calculate the $adjusted\ R^2$. b. Feature which returns the highest $adjusted\ R^2$ in the previous step, a, combine with one more feature out of the remaining nine features and again calculate $adjusted\ R^2$. c. Selected two features from the previous step, b, will combine with another feature and with each combination $adjusted\ R^2$ will be calculated. Three features will be calculated. d. A combination of four feature will be collected from this step. e. This step will generate five features and next step, f. will generate six features.

Our goal to select a model which produce higher $adjusted\ R^2$, produce the least errors and able to predict much closer to the true value. At the end features selected from this process are:

Ratio_of_mean
Ratio_of_std
Ratio_of_median
TestLine_mean
ControlLine_median

3.7 Model Evaluation

After developing the ML model, we evaluate each model's performance using different evaluation methods. Most common evaluation matrix used in this work are, Mean Squared Error, Root Mean squared Error, Mean Absolute Error, Median Absolute Error, Coefficient of Determination, $Adjusted - R^2$. Some additional evaluation methods such as Residuals plot, Histogram of residuals, quantile-quantile plot also used to evaluate and interpret ML model performance.

3.7.1 Mean Squared Error

To perform MSE, from scikit-learn library we call $metrics$ module and from $metrics$ module we use $mean\_squared\_error$ function. This function takes two variables, actual value and predicted value. Below code used to perform MSE

$$\boldsymbol{from\ sklearn\ import\ metrics}$$

$$\boldsymbol{metrics.mean\_squared\_error(y\_true, y\_predicted)}$$

where,

- y_true: true value
- y_predicted: Predicted value

3.7.2 Root Mean Squared Error

We follow the same procedure to perform RMSE. We called $metrics$ module from the scikit-learn library. There is a built-in function for RMSE but if we just take square root on MSE we get RMSE. Below command used to perform RMSE

$$\boldsymbol{import\ math}$$

$$\boldsymbol{math.sqrt(metrics.mean\_squared\_error(y\_true, y\_predicted))}$$

To perform square root, we call $math$ module and from that we use $sqrt$ function, which return square root of any given value.

3.7.3 Mean Absolute Error

From the $metrics$ module in $scikit-learn$ library $mean\_absolute\_error$ function is called. Which takes two value, true and predicted value. Below code used to perform MAE

$$\boldsymbol{metrics.mean\_absolute\_error(y\_true, y\_predicted)}$$

3.7.4 Median Absolute Error

To perform MAE Below code was used

$$\boldsymbol{metrics.median\_absolute\_error(y\_true, y\_predicted)}$$

3.7.5 Coefficient of Determination

Coefficient of determination also known as $R-Squared$ is very important evaluation metrics for the ML model. To perform this evaluation, $r2\_score$ function was called from the $metrics$ module. This function takes two variables, true and predicted values. Below code produces the results

$$\boldsymbol{metrics.r2\_score(y\_true, y\_predicted)}$$

where,

- y_true: true value
- y_predicted: Predicted value

3.7.6 Adjusted R-Squared

Another very important evaluation methods used to evaluate the ML model is $Adjusted\ R-Squared$. To calculate $Adjusted\ R-Squared$ we followed equation 11. Below code used to perform $Adjusted\ R-Squared$

$$r2\_adj = r2 - (n\_parameters - 1)/(y\_true.shape[0] - n\_parameters) * (1 - r2)$$

where,

- $r2$: Coefficient of Determination. Previous section shows how to calculate R^2.
- $n\_parameters$: No. of independent variables
- $y\_true.shape[0]$: this code returns no. of observation.

3.8 Model Validation

Last step of our predictive analysis is model validation. As discussed, only model evaluation is not enough to finalize the model selection process because model evaluation is tricky and could be misleading [79]. That is why model validation is an essential part of the ML model selection process. In this section, we will discuss how we developed k-fold cross validation steps in this work.

At first, we call all the necessary built-in function under the model selection module from the scikit-learn library. Below code used

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
from sklearn.ensemble import RandomForestRegressor
```

After that, we will estimate the parameter by using the grid search and cross-validation. To do that we used random forest regressor as a learning algorithm. First data will be stilted into train and test data using k-fold cross-validation.

First four parameters, max_depth, max_features, min_samples_split, min_samples_leaf, will be tuned for grid search. After that, an object will be created using $GridSearchCV$, which take two parameters, first one is estimator which is random forest and the second one is the parameter we created before. Next, we will train the model. Below code perform this task

```
model_RF_GS = GridSearchCV(RandomForestRegressor(), param_grid = params_RF, cv = 5)
model_RF_GS.fit(X_train, y_train)
```

After that, we will predict on test data. Below code perform the task

```
pred_RF_GS = model_RF_GS.predict(X_test3)
```

Chapter IV.

Results

In the previous section, three ML model was discussed, we defined and explained different development criteria, definition of different modules and functions, which uses to build models, stated different development stages, definition and explanation of different evaluation metrics and how to build them using the scikit-learn library and at the end, K-fold cross-validation was explained. In this section results from three ML model will be discussed and analyse.

4.1 Evolution Matrix

First evaluation and analysis were done on ML model performance with the evaluation matrix. Here we will analyse the results of the evaluation matrix for all three ML models.

4.1.1 Machine Learning Model with Unchanged Data Frame

As discussed, the first ML model has used ten independent variables from the initially developed data frame. The data frame was not modified in any means. After performing several steps in machine learning pipeline, we eventually predict the dependent variables and evaluate model performance. Here are the results of the evaluation matrix:

Table 5: Evaluation matrix results of the First ML model with the unchanged data frame

Mean Square Error	*= 393.16*
Root Mean Square Error	*= 19.83*
Mean Absolute Error	*= 11.22*
Median Absolute Error	*= 6.52*
R^2	*= 0.66*
Adjusted R^2	*= 0.55*

Table 5 shows the different evaluation's results. Most of the errors are very high and both R^2 and $adjusted - R^2$ are very low. From the R^2 value, ML model can be interpreted as only 66% of the actual or observed value can be predicted using this model.

As discussed, R^2 value goes up even though ineffective independent variables added to the ML model. An unbiased indicator to evaluate the ML model compare to R^2 is $adjusted - R^2$ [73,74]. Here we can see, R^2 and $adjusted - R^2$ are not in the same range. That means there are one or more useless features used in the ML model and should be removed.

4.1.2 Machine Learning Model with Merged Data Frame

For the second ML model, we merged the data where it shows similar characteristics. Merging data based on feature characteristics. Same as before here we use all the independent variables, in total ten, for machine learning model. Below table shows model performance:

Table 6 Evaluation matrix results of ML model with the modified data frame

Mean Square Error	= 39.62
Root Mean Square Error	= 6.30
Mean Absolute Error	= 3.94
Median Absolute Error	= 1.32
R^2	= 0.93
Adjusted R^2	= 0.62

Most of the error decreases significantly compared to the last ML model. Which is a good sign and we can say that merging similar data was a very effective measure.

R^2 value increases significantly but $adjusted - R^2$ doesn't increase that much. The significant difference between R^2 and $adjusted - R^2$ tells something wrong with the ML model. A good ML model should have a close range of R^2 and $adjusted - R^2$. We can make a conclusion here that, there might be one or more ineffective features used in the ML model and that feature(s) should be removed.

4.1.3 Final Machine Learning Model

In chapter three we discussed ML model selection procedure. Basically, we calculate $adjusted - R^2$ and find the best combination of independent variables which produces highest $adjusted - R^2$, also a similar range of R^2 and $adjusted - R^2$.

Table 7 Evaluation matrix for final ML model

Mean Square Error	= 68.72

Root Mean Square Error	*= 8.29*
Mean Absolute Error	*= 5.91*
Median Absolute Error	*= 3.67*
R^2	*= 0.88*
Adjusted R^2	*= 0.81*

Table 7 shows most of the error increases small amount compared to the last evaluation matrix, table 6. But both R^2 and $adjusted - R^2$ pretty high and in the same range, which tells performance of ML model very good. From R^2 value, it can be said that 88% value can be replicable with this model.

4.1.4 K-Fold Cross Validation

Table 8 present evaluation matrix of K-fold cross-validation. Table 8 shows most of the error increases small amount compared to the final ML model, table 7. But both R^2 and $adjusted - R^2$ are quite high and almost in the same range. So, we can say the ML model was selected is very effective and performing well.

Table 8 Evaluation matrix results of K-fold cross-validation

Mean Square Error	*= 106.82*
Root Mean Square Error	*= 10.34*
Mean Absolute Error	*= 9.0*
Median Absolute Error	*= 6.35*
R^2	*= 0.812*
Adjusted R^2	*= 0.706*

4.2 Predicted vs Actual Value

A scatter plot of predicted vs actual value with a regression line is a good way to visualize how well machine learning (ML) model able to predict [21]. If the model able to predict an actual value perfectly then the point will be on the regression line. Points stay under the regression line are lower than the actual value and points above the line higher than the actual value. Ideally, most of the points should remain close to the regression line.

4.2.1 Machine Learning Model with Unchanged Data Frame

Figure 19 shows a scatter plot between predicted and actual value. The regression line is the best-fitted line of the plotted data points.

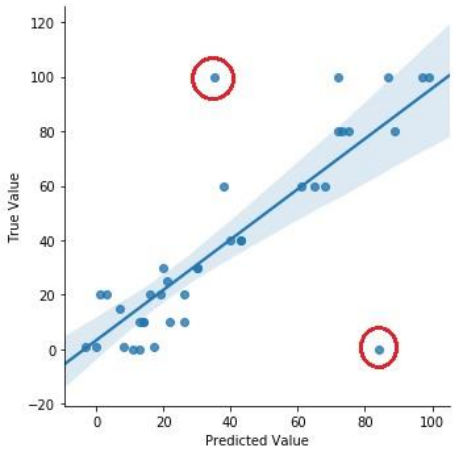


Figure 19: Predicted vs Actual value plot from First ML model

From figure 19 we can say, a few points are on the regression line (RL) and most of the points are very close to the RL. Points from approximately 9 to 35, some of the points are out of confidence interval (CV). But points greater than 40 well predicted and most of them very close to the regression line. From this plot, we can see two points are very far from the regression line, red circle. These two points are the outlier. Outlier influence the position of the regression line. So, these two points should be removed.

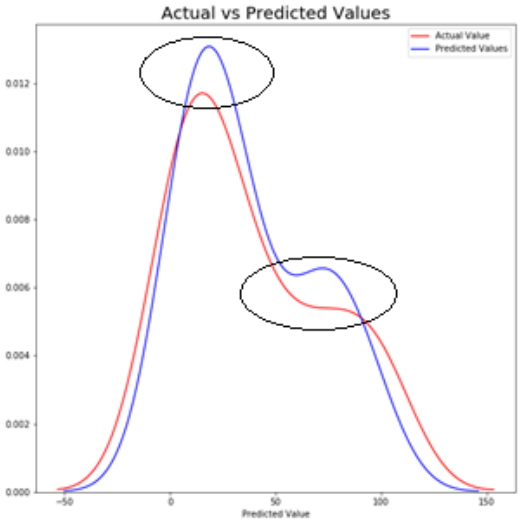


Figure 20: Distribution of data, red: actual and blue: predicted

Figure 20 shows data distribution of both actual value, red line, and predicted value, blue line. Both distribution curve shows positive skewness. From the figure, we can see that two areas where predicted value is much higher than the actual value, black circle. Those areas prediction is not accurate. But the rest of the place ML model performs quite good. Maybe because of the skewness of data, ML model performs incorrectly on those areas.

Though the scatter plot between predicted & actual value, shows the good distribution of data point around the regression line, but it won't be wise to conclude that, this model performing well. As we saw from the evaluation matrix, both R^2 and $adjusted - R^2$ are not big enough to choose this ML model.

4.2.2 Machine Learning Model with Merged Data Frame

Figure 21 shows, most of the points are very close to the regression line and there are no outliers on the plot. It's clear that after performing merging on similar data, the ML model performing much better than the previous ML model.

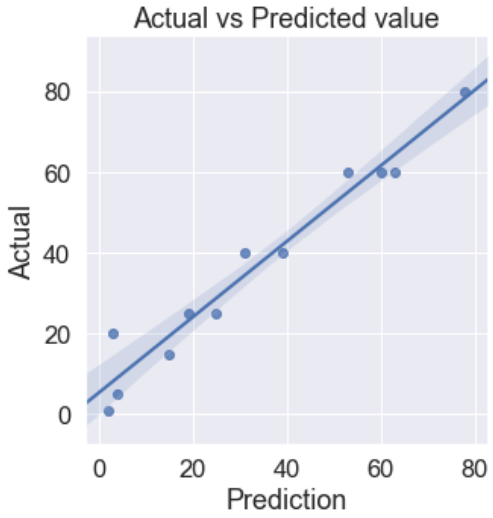


Figure 21: Predicted vs Actual value plot from ML model with the merged data frame

Figure 22 shows the distribution of data for the ML model with the merged data frame. Redline represents the actual value and blueline is true value. Both distribution curve shows a little bit of negative skewness. From the figure, most of the area of actual value is almost covered by predicted value, except top right corner where prediction little bit inaccurate. Otherwise the performance of this ML mode quite good.

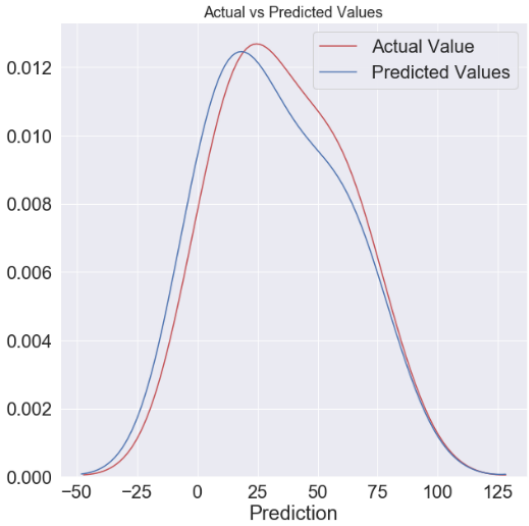


Figure 22: Distribution of data for the ML model with the merged data frame, red: actual and blue: predicted

4.2.3 Final Machine Learning Model

Figure 23 shows a scatter plot with a fitted line between actual and predicted value. From the plot we can say, almost all the points are very close to the regression line, which is a very good indication of the goodness of fit.

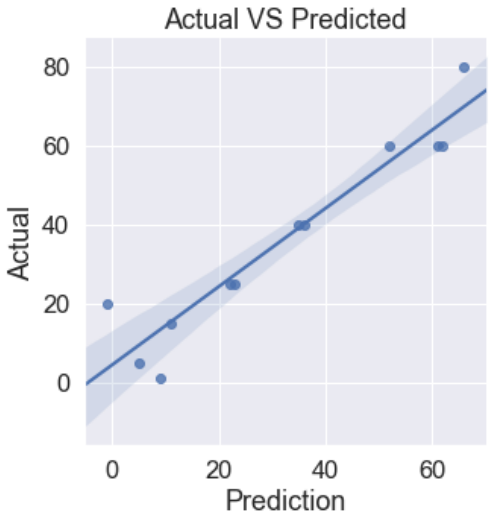


Figure 23: Predicted vs Actual value plot from Final ML model.

We can divide these points into three groups. Points between 0 to 20 tell, only 25% points are exactly on the fitted line, 50% points are within the confidence interval (CI) and 50% points are outside of the CI but still not far from the regression line (RL).

Points between 20 to 40, 75% points are on the line RL and 100% points are within the CI.

Points greater than 40, no points are on RL, but 75% points are within the CI. Points which outside of the CI are also close to RL

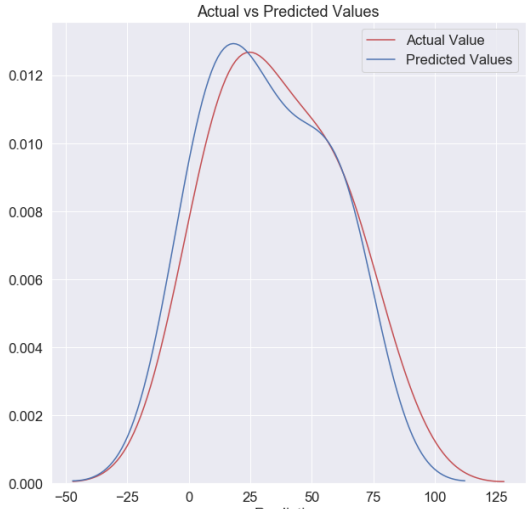


Figure 24: Distribution of data for final ML model, red: actual and blue: predicted

Figure 24 shows the distribution of data for the final ML model. Redline represents the actual value and blueline is true value. In this stage also both distribution curve shows a little bit of negative skewness. The blueline which is predicted value slightly higher than the actual value in the top area. On the top right corner again predicted value little bit incorrect otherwise most of the area it able to cover.

4.2.4 K-Fold Cross Validation

Figure 25 shows a scatter plot with a regression line between the actual and predicted value. From the graph we can say, most of the points are very close to the regression line and some of the points are outside of the CI, but still not far from the regression line.

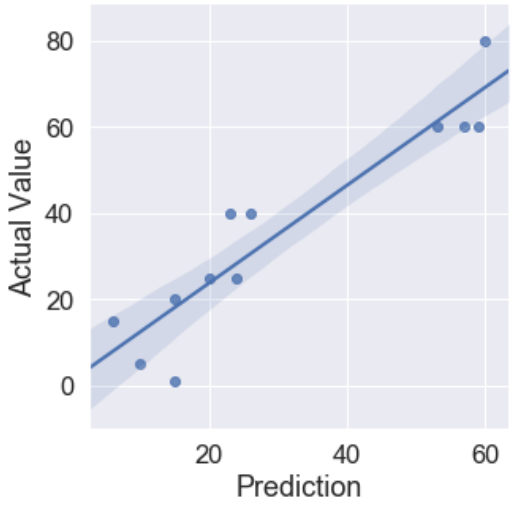


Figure 25: Predicted vs Actual value plot from K-fold cross-validation.

Figure 26 shows the distribution of data for K-fold cross-validation. On the top and right side of the area are much higher than the actual value. Also, in the top right corner predicted values are very lower than the actual value. Left and right side of the plot area where predicted value is able to follow the actual value.

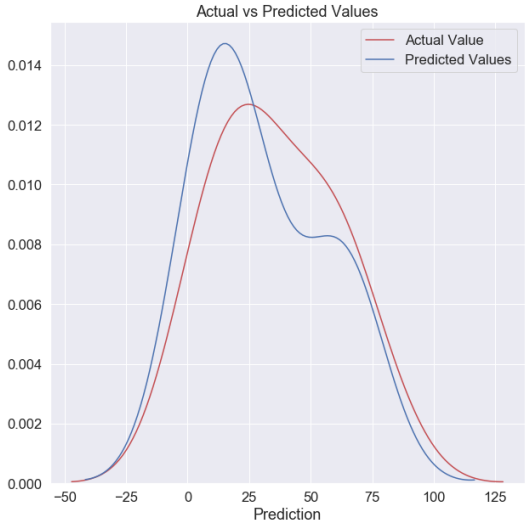


Figure 26: Distribution of data for final ML model, red: actual and blue: predicted

4.3 Residual Plot

Variance of data can be presented with the residual plot. Which is a very effective way to evaluate a ML model. As we know if the residual points are randomly distributed around the horizontal axis then the data points are properly fitted with the ML model [79].

4.3.1 Machine Learning Model with Unchanged Data Frame

Figure 27 shows the residuals plot for first ML model which used the unchanged data frame. From the plot, almost all the points are randomly distributed around the zero-line in the horizontal axis. On the plot, we can see two points are a little bit away from the zero-line, red circle. These two points are the outlier. Based on the plot we can say that ML model fit the data quite good.

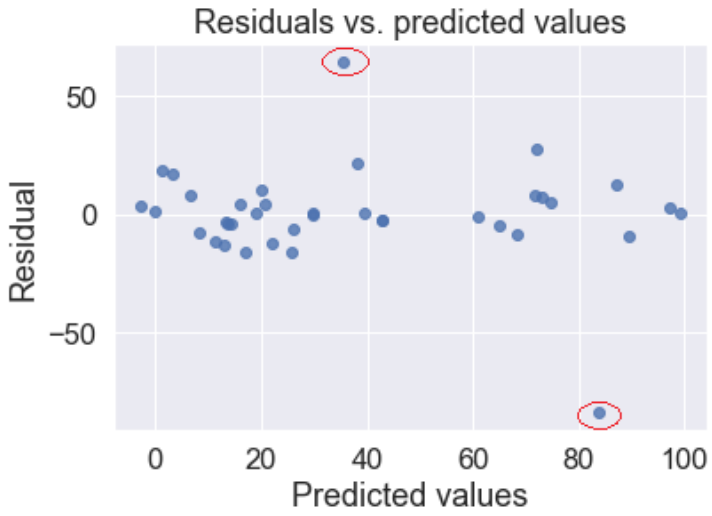


Figure 27: Residual plot for the ML model unchanged data frame

4.3.2 Machine Learning Model with Merged Data Frame

Figure 28 shows the residuals plot for the ML model for the merged data frame. From the plot we can say, almost 70% of points have remained very close to the zero-line in the horizontal axis and they are randomly distributed. Other points are also not too far from the zero line in the x-axis. But it should be noted here very few points remain under zero line, most of them either on the zero line or above the zero line in the x-axis.

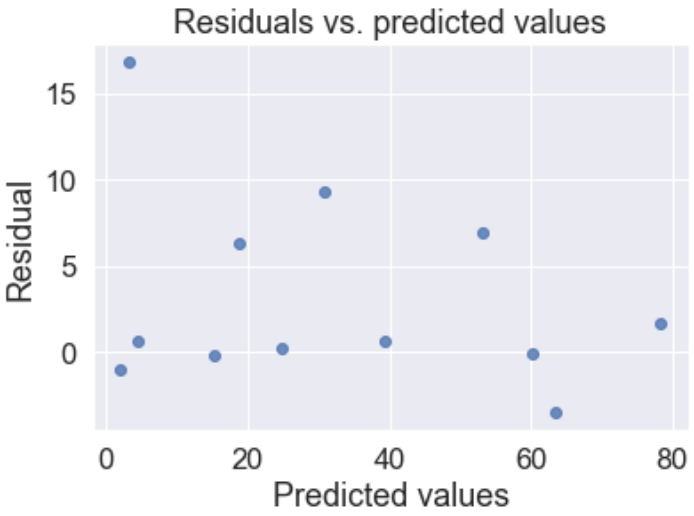


Figure 28: Residual plot for the ML model unchanged data frame

4.3.3 Final Machine Learning Model

Figure 29 shows the residual plot for the final ML model. All the points are randomly distributed around the horizontal axis. From the graph, most of the points are very close to the zero-line in the horizontal axis, which tells the goodness of fit of the ML model.

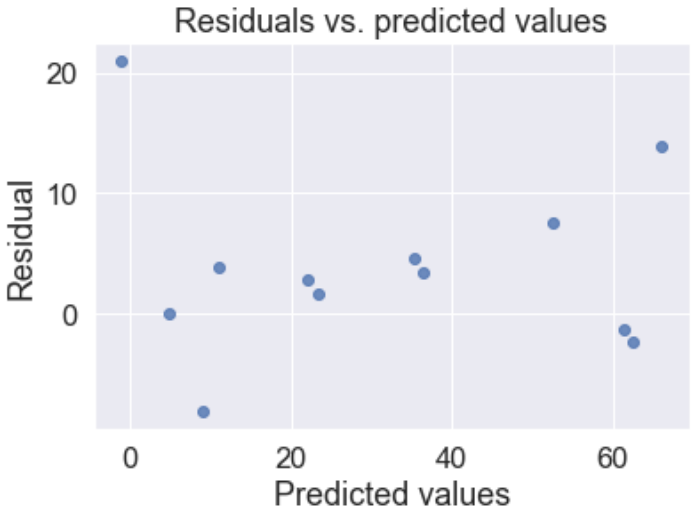


Figure 29: Residual plot for the final ML model

4..4 K-fold Cross Validation

Figure 30 shows the residual plot for K-fold cross-validation. Figure shows a random distribution of points around the horizontal axis. This display of random nature proves the goodness of fit. So we can say model we select for prediction will work properly even with an independent data set.

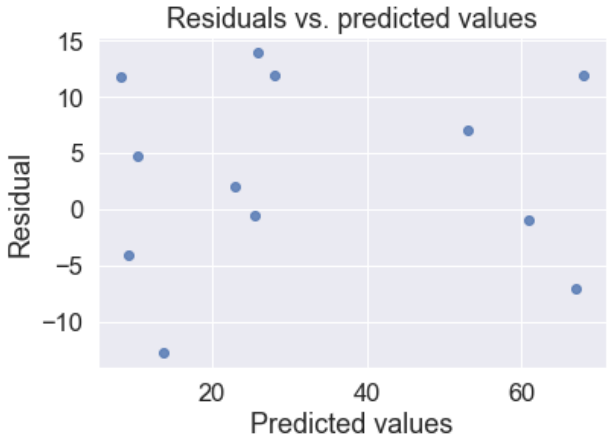
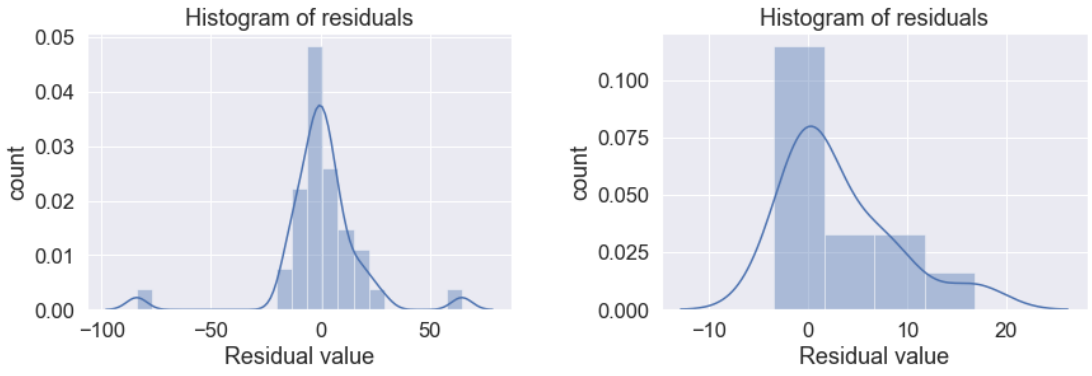


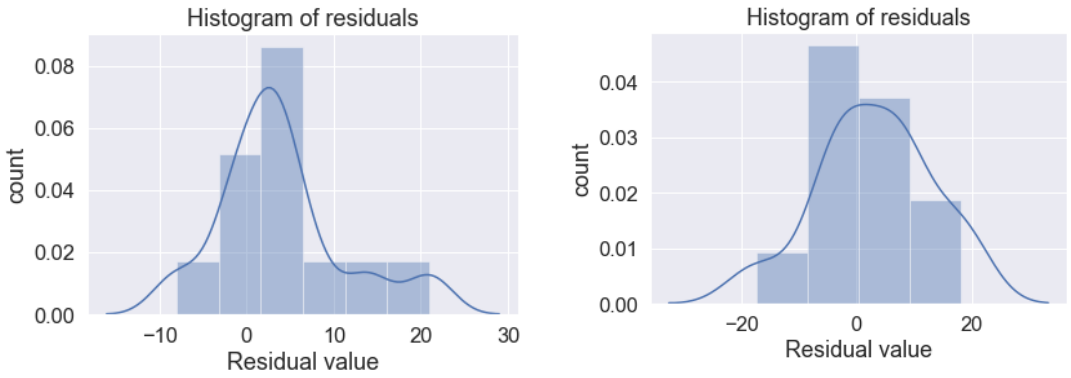
Figure 30: Residual plot for K-fold cross-validation

4.4 Histogram of Residuals

Histogram of residuals is also used to check whether the variance is normally distributed or not. A symmetric histogram shows the normality assumption is likely to be true. If the histogram of residual show errors is not normally distributed, then possibly model's underlying assumption may be violated. Figure 31 shows all four histograms of residual from three ML models and fig 31(d) from K-fold cross-validation [82].



a b



c d

*Figure 31: Histogram of residuals of **a.** ML model with unchanged data frame **b.** ML model with merged data frame **c.** final ML model **d.** K-fold cross-validation*

Figure 31-a shows the histogram of residuals (HOR) of the first machine learning model with unchanged data frame (DF). From figure 31-a, the residuals are normally distributed. But there are two extreme outliers.

Figure 31-b shows HOR for the ML model with the merged data frame. This HOR suggests that the residuals and the error terms are not normally distributed. The distribution of the residuals is highly positively skewed. But there is no outlier in this ML model.

Figure 31-c shows HOR for final ML model. This HOR suggests that the residuals and the error terms are not normally distributed. There are too many extreme positive and negative residuals. The distribution is strongly tailed.

Figure 31-d shows HOR for K-fold cross-validation. This HOR suggests that the residuals and the error terms are normally distributed.

4.5 Quantile-Quantile Plot

Quantile-Quantile (Q-Q) plot is a type of scatter plot which plotted one quantile of data sets against another quantile of data set. Quantiles also refer to as percentiles are points in the data, below that point a certain proportion of data falls. A quantile 0.3 means 30% data fall below that point and 70% above that point [24,25].

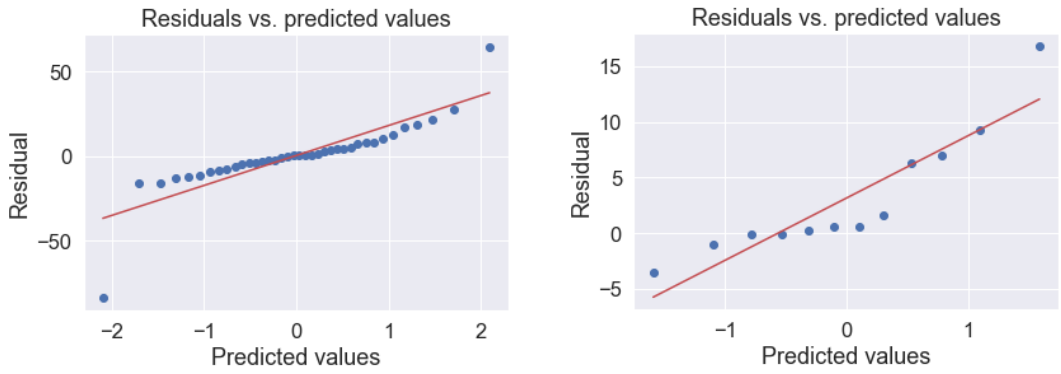
If two points have come from a similar distribution of data, then the point will fall on the 45-degree reference line, Figure 32: red line. More the points are away from the

reference line more it proves that two quantiles are from different distributions of data [23, 24].

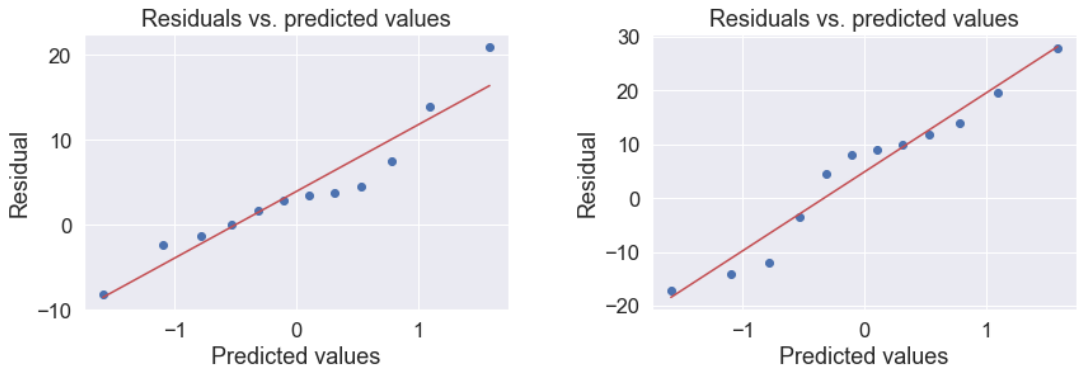
A Q-Q plot can be used to evaluate and analyse the data distributions structure, normally distributed or distribution with skewness. It provides a graphical representation of two distribution of data on location, scale or skewness and represents an analytical judgment on their similarities or differences. The graphical representation of the Q-Q plot can be used to assess the goodness of fit rather than using a numerical summary [23].

Points are plotted on Q-Q plot are always shows an increasing trend if we observed the plot form left to right. As mentioned, if both quantiles are from the same distribution or data distribution are identical then points in the Q-Q plot follow the 45° reference line. If trends of Q-Q plot are flatter compare to reference line, then the distribution of quantiles on the horizontal axis's is more dispersed than the distribution of quantiles on the vertical axis. But if trends of Q-Q plot are steeper than the reference line then the distribution of quantiles on vertical axis's is more dispersed than the distribution of quantiles on the horizontal axis. Sometime Q-Q plot shape like arced or "S" shaped, in this case, it indicates one distribution more skewed than other [23, 79].

Below figure shows four Q-Q plots from four machine learning (ML) model.



a b



c d

*Figure 32: Quantile-Quantile plot of **a.** ML model with unchanged data frame **b.** ML model with merged data frame **c.** final ML model **d.** K-fold cross-validation*

Figure 32-a shows Quantile-Quantile (Q-Q) plot of first machine learning model with unchanged data frame (DF). From figure 32-a, the Q-Q plot is flatter compare to the reference line, which tells quantile in the horizontal axis, in this case, predicted value, are more dispersed than the vertical axis. The Q-Q plot has a slightly arched shape which tells one distribution has little bit skewness than others. From this plot, it can also be observed there are two outliers, which also influence the skewness of the distribution.

Figure 32-b shows a Q-Q plot of the ML model with merged data frame (DF). The Q-Q plot shows an arched shape structure, and this could be interpreted as one distribution has more skewness than other.

Figure 32-c shows a Q-Q plot of the final ML model. The Q-Q plot shows an arched shape structure, and this could be interpreted as one distribution has more skewness than other.

Figure 32-d shows Q-Q plot of K-fold cross-validation. The Q-Q plot has almost no arched shaped, most of the point follow the reference line. This plot describes the distribution of data symmetric. The relation between two quantiles is linear.

Chapter V.

Discussion

In the area of precision medicine, it is essential to have a measurement system of drug and other metabolite concentration at a high speed and easy to use point of care (POC) technique. For the treatment of tachycardia digoxin and digoxigenin are used in serum solution with a small concentration, mostly in the range 0.5 – 2.0 ng. mL^{-1}. It is very important to monitor the concentration of digoxin in human serum solution. If the concentration of digoxin in serum is above 2.5 $ng.mL^{-1}$ then the toxic effect of digoxin is very much possible and dangerous for human health [1].

For monitoring the digoxin concentration in a fast and easy to use POC fashion, a machine learning approach could be used to measure the concentration of digoxin and its hapten, digoxigenin, in human serum [1].

For developing such a system, a gold nanoparticle lateral flow (LF) assay for digoxigenin is used for smartphones and other readout systems for use in a non-laboratory environment. After performing tests with LF and prepared solution with different concentration, images of those LF was captured using two image readout system [1].

To calculate the concentration of digoxin from image intensity a three steps algorithm in Python programming was developed. In the first step, the region of interest (ROI), control line and test line, from raw images was extracted. This step requires an additional two steps. At first, a built-in function, match template, from scikit-learn library is used to extract LF from raw images. Next, several conditions and masking ideas used to extract ROI.

After that, these images then converted into a data frame (DF). To improve machine learning (ML) model performance, a couple of feature engineering (FE) was performed. Transformation of data using logarithm & square root and merging similar characteristics data based on the feature. After performing these FE, ML model performance improve significantly.

In the last step, ML model was developed to predict the concentration of digoxin from image intensity. The Performance of ML model was then evaluated and validated.

In the data frame (DF) we created ten features from cropped images. Before selecting a final ML model, we used all the features form DF and created two ML models to test and analyse the performance of DF. Very first ML model which is used unmodified DF motivates to perform FE on the DF.

Several evaluation methods used to evaluate the performance of the ML model but the most important evaluation methods we used is $adjusted - R^2$. Which gives much better indicator compare to R^2.

From the first ML model with unchanged DF, where ten independent variables were used. The value of R^2 and $adjusted - R^2$ was 0.66 and 0.55 respectively. Even though $adjusted - R^2$ not far from R^2 but both are very small. From R^2, only 66% actual value could be predicted correctly with this model, which is not high enough. Also, some evaluation method shows some outliers on the predicted value.

To improve the performance of the ML model a couple of feature engineering was performed on the DF. One important FE was done by merging similar data based on feature characteristics. Which improves the performance of the ML model. From the second ML model, R^2 and $adjusted - R^2$ were 0.93 and 0.62. Value of R^2 is very high but $adjusted - R^2$ very low compare to R^2. As discussed, a good ML model should return a similar range of R^2 and $adjusted - R^2$. Because they have a high deviation, there might be one or more bad independent variables used in the ML model.

Next, we select our final ML model. For that, we calculate $adjusted - R^2$ every time we choose a feature for the ML model. Wherever we got a higher value of $adjusted - R^2$ we select that feature or combination of features. That is how we select our final ML mode. The value of R^2 and $adjusted - R^2$ of the final ML model were 0.88 and 0.81 respectively. Which is quite high and both of them are in a similar range. Which indicates the goodness of fit of the regression line.

Lastly, we validate the ML model using K-fold cross-validation. For this method parameter estimation was performed by using the grid search method and the learning

algorithm estimator was random forest regressor. The value of R^2 and $adjusted - R^2$ was 0.81 and 0.706 respectively. The value of R^2 and $adjusted - R^2$ is quite high and both of them in a similar range. Which indicates the goodness of fit of the regression line.

Chapter VI.

Conclusion

This work demonstrates a method of preparing data to analyse and develop a machine learning model to predict the concentration of digoxin from the intensity of the image. This idea could be used to further develop a smartphone application which enables a first and easy to use POC measurement technique in the field of precision medicine.

Though this work demonstrates a very effective way to prepare data and eventually develop a ML model to predict the concentration, it has several limitations. Cropping steps extremely dependent on the template image. A wrong or inaccurate template image could result in poor cropping result.

A machine learning approach combine with an automated cropping algorithm developed to calculate the concentration of digoxigenin from image intensity. This work demonstrates an efficient pre-processing step which is a crucial part for the final machine learning step.

This work could be used to further develop a mobile application that could be used in a non-laboratory environment to calculate the concentration of digoxigenin in human serum.

Appendix 1.

Python Code for Cropping

Library

```
from skimage.io import imread, imshow
from skimage.color import rgb2gray
import matplotlib.pyplot as plt
import numpy as np
from skimage.feature import match_template
import os
```

Iphone Templet

```
iphone = imread('40-II.jpg')
iphone = rgb2gray(iphone)
iphone_temp = iphone[450:2250,1017:1195]
```

Imager Templet

```
immg = imread('Dig 40 nM.tif')
immg = rgb2gray(immg)
imager_temp = immg[600:1550,1542:1639]
```

Data Directory (read & save)

```
data_dir = ('../../Dataset Gold-NP-LFA/IMAGES GOLD-DIGOXIGENIN-LFA/Data Serum/14-11-2018 Dig Serum BioImager/DIG-Serum I/')

directory = data_dir.split('/')
directory = directory[4:]
save_dir = 'E:/02 Study/MasterThesis/Test/Masking/SavedFiles/'
save_dir = save_dir + '/'.join(directory)

try:
   os.makedirs(save_dir)
except:
   pass
```

Reading Image File

```
for img in os.listdir(data_dir):
```

Cropping Begin for the Image ends with JPG (Iphone image)

```
   if img.endswith('.JPG') or img.endswith('jpg'):
      try:
         image= imread(data_dir+img)
         image = rgb2gray(image)
```

Template Matching Part

```
         result = match_template(image, iphone_temp)
         ij = np.unravel_index(np.argmax(result), result.shape)
         x, y = ij[::-1]
         crop_img = image[y:y+iphone_temp.shape[0],x:x+iphone_temp.shape[1]]
         plt.figure()
         imshow(crop_img)
         plt.title(img.split('.'))
```

Applying Mask

```
crop_img_BM = crop_img.copy()
crop_img_BM[crop_img_BM>(crop_img_BM.mean()+(crop_img_BM.min())/15)] = 0
plt.figure()
imshow(crop_img_BM)
plt.title(img)
```

Amplifying Black Pixel

```
n=0
crop1 = crop_img_BM.copy()
for k in crop1:
    i = 0
    for y in k:
        if y == 0 or y == 0.0:
            i = i +1
    if i > 9:
        crop1[n, :] = 0
    n = n + 1

plt.figure()
imshow(crop1)
plt.title('Crop1 '+ img.split('.')[0])
plt.yticks(range(0,crop1.shape[0],50))
pts = np.argwhere(crop_img_BM > 0)
y1,x1 = pts.min(axis=0)
y2,x2 = pts.max(axis=0)

crop2 = crop1[y1:y2, x1:x2]
```

Reshaping Image

```
if crop2.shape[0] > 1500:
    crop2 = crop2[(round(crop2.shape[0]/1.5)):,:]
elif crop2.shape[0] > 600:
    crop2 = crop2[(round(crop2.shape[0]/2)):,:]

plt.figure()
imshow(crop2)
plt.title('Removing Zero pixel area crop2_1 '+ img.split('.')[0])
plt.yticks(range(0,crop2.shape[0],50))
```

Finding Row Indexes with Non-zero Pixels

```
zero_list = list()
for i, j in enumerate(crop2):
    if sum(j) != 0:
        zero_list.append(i)
```

Slicing Image

```
crop3 = crop2[zero_list[0]:zero_list[-1],:]
plt.figure()
imshow(crop3)
plt.title(img+' Zero List')
pts = np.argwhere(crop3 > 0)
y1,x1 = pts.min(axis=0)
y2,x2 = pts.max(axis=0)
```

```
crop4 = crop3[y1:y2, x1:x2]

crop_index=list()
for l,k in enumerate(crop4):
    if sum(k) != 0:
        crop_index.append(l)

crop5 = crop4[crop_index[0]:,:]
plt.figure()
imshow(crop5)
plt.title(img+' crop5')
plt.yticks(range(0,crop5.shape[0],15))

list_list = list()
for z,q in enumerate(crop5):
    if sum(q)==0:
        break
    else:
        list_list.append(z)
if list_list[-1] - list_list[0] < 10:
    crop5 = crop5[list_list[-1]+1:,:]
plt.figure()
imshow(crop5)
plt.title(img+' crop5_1')
plt.yticks(range(0,crop5.shape[0],15))

Npix_idx = list()
for idx, Npix in enumerate(crop5):
    if sum(Npix) != 0:
        Npix_idx.append(idx)
    else:
        pass
crop6 = crop5[Npix_idx[1]:,:]

plt.figure()
imshow(crop6)
plt.title(img+' crop6')
plt.yticks(range(0,crop6.shape[0],15))
roi1_indx=list()
for indx, roi1 in enumerate(crop6):
    if sum(roi1) != 0:
        roi1_indx.append(indx)
    else:
        pass
testtest = list()
for x in roi1_indx:
    try:
        if x+1 == roi1_indx[roi1_indx.index(x)+1]:
            testtest.append(x)
        else:
            break
    except:
        pass
```

Separating Control Line

```
roi1 = crop6[testtest[1]:testtest[-1],:]
```

```
try:
    roi1_idx=list()
    for i, row in enumerate(roi1):
        try:
            if sum(row) != 0:
                roi1_idx.append(i)
        except:
            pass
except:
    pass

roi1 = roi1[roi1_idx[0]:roi1_idx[-1],0:166]
plt.figure()
imshow(roi1)
crop7_index=list()
for index in roi1_indx:
    try:
        if index+1 != roi1_indx[roi1_indx.index(index)+1]:
            crop7_index.append(roi1_indx[roi1_indx.index(index)+1])
    except:
        pass

crop7 = crop6[crop7_index[0]:,:]
plt.figure()
imshow(crop7)
plt.title(img+' crop7')
plt.yticks(range(0,crop7.shape[0],35))
roi2_indx=list()
for indx2, ro2 in enumerate(crop7):
    if sum(ro2) != 0:
        roi2_indx.append(indx2)

        n = n - 1
    else:
        pass

crop8 = crop7[roi2_indx[7]:roi2_indx[48],:]
roi2_index = list()
for i, indx in enumerate(crop8):
    if sum(indx) != 0:
        roi2_index.append(i)

roi2_final_inx = list()
for x in roi2_index:
    try:
        if roi2_index[roi2_index.index(x)+1] - x < 55:
            roi2_final_inx.append(x)
        else:
            break
    except:
        pass
```

Separating Test Line

```
roi2 = crop8[roi2_final_inx[1]:roi2_final_inx[-7],0:166]

plt.figure()
```

```
        imshow(roi2)

    except:
        pass
```

Cropping Begin for the Image ends with TIF (Imager image)

```
For img in os.listdir(data_dir)
  if img.endswith('.TIF') or img.endswith('tif'):
    try:
      image= imread(data_dir+img)
      image = rgb2gray(image)
```

Template Matching Part

```
      result = match_template(image, imager_temp)

      ij = np.unravel_index(np.argmax(result), result.shape)
      x, y = ij[::-1]
      crop_img = image[y:y+imager_temp.shape[0],x:x+imager_temp.shape[1]]
      plt.figure()
      imshow(crop_img)
      plt.yticks(fontsize=18)
      plt.xticks(fontsize=18)
      plt.title(img.split('.')[0], fontsize=20)
```

Applying Mask

```
      crop = crop_img.copy()
      crop[crop > (crop.mean()+(crop.min())/15)] = 0

      plt.figure()
      imshow(crop)
      plt.yticks(range(0,crop.shape[0],100), fontsize=18)
      plt.xticks(range(0,crop.shape[1],50), fontsize=18)
      plt.title('Applying Pixel Value to Zero',fontsize=20)
```

Amplifying Black Pixel

```
      n=0
      crop1 = crop.copy()
      for k in crop:
        i = 0
        for y in k:
          if y == 0 or y == 0.0:
            i = i +1
        if i > 20:
          crop1[n, :] = 0
        n = n + 1
      plt.figure()
      imshow(crop1)
      plt.title('Crop1 '+ img.split('.')[0])
      plt.yticks(range(0,crop1.shape[0],100))

      pts = np.argwhere(crop > 0)
      y1,x1 = pts.min(axis=0)
      y2,x2 = pts.max(axis=0)

      crop2 = crop1[y1:y2, x1:x2]
```

Reshaping Image

```
if crop2.shape[0] > 600:
   crop2 = crop2[(round(crop2.shape[0]/2)):,:]

plt.figure()
imshow(crop2)
plt.title('Removing Zero pixel area crop2_1 '+ img.split('.')[0])
plt.yticks(range(0,crop2.shape[0],5))
```

Finding Row Indexes with Non-zero Pixels

```
zero_list = list()
for i, j in enumerate(crop2):
   if sum(j) != 0:
      zero_list.append(i)
```

Separating Useful Indexes

```
index_list = list()
for num in zero_list:
   try:
      if num + 1 != zero_list[zero_list.index(num) + 1]:
         differ = zero_list[zero_list.index(num) + 1] - num
         if differ > 49:
            index_list.append(zero_list[zero_list.index(num) + 1])
         else:
            index_list.append(num)
            index_list.append(zero_list[zero_list.index(num) + 1])

   except:
      pass

new_row = list()
for ele in index_list:
   try:
      diff = index_list[index_list.index(ele)+1] - ele
      if diff < 45 and diff > 15:
         new_row.append(ele)
         new_row.append(index_list[index_list.index(ele)+1])

   except:
      None

new_row = list(dict.fromkeys(new_row))

final_index=list()
final_index.append(new_row[0])
for  index in new_row:
   try:
      index_diff = new_row[new_row.index(index)+1] - new_row[0]
      if index_diff < 100:
         final_index.append(new_row[new_row.index(index)+1])
   except:
      None
```

Separating ROI

```
new_row = final_index.copy()

n = new_row[0]
m = new_row[0]

try:
   while n > 0 :
      if sum(crop2[n-1,:]) != 0:
         n = n-1
      else:
         break
except:
   None

try:
   while m < new_row[1]:
      if sum(crop2[m+1,:]) != 0:
         m = m+1

      else:
         break
except:
   pass

try:
   if n != new_row[0]:
      ROI_1 = crop2[n:new_row[0],:]

      plt.figure()
      imshow(ROI_1)

   else:
      ROI_1 = crop2[new_row[0]:m,:]
      plt.figure()
      imshow(ROI_1)

except:
   pass

p = new_row[-1]
q = new_row[-1]

try:
   while p > 0 :
      if sum(crop2[p-1,:]) != 0:
         p = p-1
      else:
         break
except:
   None

try:
   while q < crop2.shape[0]:
      if sum(crop2[q+1,:]) != 0:
```

```
            q = q+1
        else:
            break
    except:
        pass

    try:
        if p != new_row[-1]:
            ROI_2 = crop2[p:new_row[-1],:]
            plt.figure()
            imshow(ROI_2)
            plt.title('ROI 2 ' + img.split('.')[0])
        else:
            ROI_2 = crop2[new_row[-1]:q,:]
            plt.figure()
            imshow(ROI_2)

    except:
        pass

except:
    pass
```

Appendix 2.

Code for Pre-processing

Library

```
import pandas as pd
from skimage.io import imread, imshow
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import os
import math
import re
```

Defining Data Directory

```
data_dir = 'E:/02 Study/MasterThesis_HFU/Test/Masking/ML_Data/'
```

Function to collect concentration value from image name

```
def getNumbers(str):
    array = re.findall(r'[0-9]+', str)
    return array
```

Empty Feature List

```
img_name = list()
roi1_mean = list()
roi1_median = list()
roi1_std = list()
roi2_mean = list()
roi2_median = list()
roi2_std = list()
img_ratio = list()
concentration = list()
Ratio_of_mean = list()
Ratio_of_median = list()
Ratio_of_std = list()
```

Reading Image Files

```
for i, roi in enumerate(os.listdir(data_dir)):
   indexlist = list()
   img = imread(data_dir+roi)
#    plt.figure()
#    imshow(img)
#    plt.title(roi)
```

Selecting Right Within the Image

```
for jj, ii in enumerate(img):
  if np.mean(ii) == 255:
    pass
```

```
else:
indexlist.append(jj)

listlist=list()
for x in indexlist:
try:
if x+1 == indexlist[indexlist.index(x) +1]:
listlist.append(x)
except:
pass
```

Slicing ROI from Saved Image

```
img = img[listlist[3]:listlist[-12],55:622]
```

Creating Feature Ratio of Control to Test Line

```
try:
  if roi.split('.')[0].split()[-1] == 'ROI_1':
    ReOI1 = img.shape[0]
    #print('ReOI1',ReOI1)
  else:
     ReOI2 = img.shape[0]
     #print('ReOI2',ReOI2)
     img_ratio.append(ReOI1/ReOI2)
except:
   pass
```

Creating Image Name Feature

```
images = os.listdir(data_dir)
imgname_1 = roi
try:
   imgname_2 = images[images.index(imgname_1)+1]
   if imgname_1.split()[:-1] == imgname_2.split()[:-1]:
   img_name.append(roi.split('.')[0])
except:
   pass
```

Creating Control Line Features

```
if roi.split()[-1].split('.')[0] == 'ROI_1':
   roi1_mean.append(np.mean(img))
   roi1_median.append(np.median(img))
   roi1_std.append(np.std(img))
   roi1_Mean = np.mean(img)
   roi1_Median = np.median(img)
   roi1_Std = np.std(img)
```

Creating Test Line Feature

```
if roi.split()[-1].split('.')[0] == 'ROI_2':
   roi2_mean.append(np.mean(img))
   roi2_median.append(np.median(img))
   roi2_std.append(np.std(img))
   roi2_Mean = np.mean(img)
   roi2_Median = np.median(img)
   roi2_Std = np.std(img)
   Ratio_of_mean.append(roi1_Mean/roi2_Mean)
   Ratio_of_median.append(roi1_Median/roi2_Median)
   Ratio_of_std.append(roi1_Std/roi2_Std)
```

Creating Concentration Features

```
name=roi
try:
name2 = images[images.index(name)+1]
if name.split()[:-1] == name2.split()[:-1]:
arr = getNumbers(name)
try:
concentration.append(np.int(arr[0]))
except:
pass

except:
pass
```

Converting List into DF

```
zippedList =  list(zip(img_name, roi1_mean, roi1_median, roi1_std, roi2_mean,
roi2_median,roi2_std, img_ratio,Ratio_of_mean, Ratio_of_median,
Ratio_of_std, concentration))
df = pd.DataFrame(zippedList, columns = ['img_name', 'roi1_mean', 'roi1_median','roi1_std',
'roi2_mean', 'roi2_median','roi2_std', 'img_ratio',
'Ratio_of_mean', 'Ratio_of_median', 'Ratio_of_std','concentration'])
df.columns = ['ImageName', 'ControlLine_mean', 'ControlLine_median','ControlLine_std',
'TestLine_mean','TestLine_median',
'TestLine_std','Ratio_of_CL_to_TL','Ratio_of_mean', 'Ratio_of_median', 'Ratio_of_std',
'Concentration']
```

Visualization of Logarithm and Square root transformation and original Data

```
for x in df.columns[1:-1]:
    f, [ax1,ax2,ax3] = plt.subplots(1,3)
    sns.distplot(df[x],ax=ax1)
    sns.distplot(np.sqrt(df[x]), ax=ax2)
    sns.distplot(np.log(df[x]), ax=ax3)
    ax1.set_title(x, fontsize = 15)
    ax2.set_title('Square root of '+ x, fontsize=15)
    ax3.set_title('Log of ' + x, fontsize=15)
```

Appendix 3.
Code for Merging data, Machine Learning, Evaluation, Validation

Library

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import linear_model
import sklearn.metrics as sklm
import sklearn.model_selection as ms
import math
from sklearn.model_selection import cross_val_score
import statsmodels.formula.api as smf
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
from sklearn.ensemble import RandomForestRegressor
```

Reading data frame

```
df = pd.read_csv("Concentration_Prediction_DataFrame.csv", index_col=0)
```

Developing first ML model

```
X = df[['ControlLine_mean', 'ControlLine_median', 'ControlLine_std',    'TestLine_mean',
        'TestLine_median', 'TestLine_std', 'Ratio_of_CL_to_TL', 'Ratio_of_mean',  'Ratio_of_median',
        'Ratio_of_std']]

y = df['Concentration']

X_train1, X_test1, y_train1, y_test1 = train_test_split(X, y, test_size=0.30, random_state=0)

lm = LinearRegression(fit_intercept=False)
lm.fit(X_train1,y_train1)
predictions = lm.predict(X_test1)
```

Evaluation of first ML model

Evaluation Matrix

```
def print_metrics(y_true, y_predicted, n_parameters):
    ## First compute R^2 and the adjusted R^2
    r2 = sklm.r2_score(y_true, y_predicted)
    r2_adj = r2 - (n_parameters - 1)/(y_true.shape[0] - n_parameters) * (1 - r2)
    r2_adj_2 = 1 - (((1-r2) * (y_true.shape[0] - 1))/(y_true.shape[0] - n_parameters-1))

    ## Print the usual metrics and the R^2 values
    print('Mean Square Error      = ' + str(sklm.mean_squared_error(y_true, y_predicted)))
    print('Root Mean Square Error = ' + str(math.sqrt(sklm.mean_squared_error(y_true, y_predicted))))
    print('Mean Absolute Error    = ' + str(sklm.mean_absolute_error(y_true, y_predicted)))
    print('Median Absolute Error  = ' + str(sklm.median_absolute_error(y_true, y_predicted)))
    print('R^2                    = ' + str(r2))
    print('Adjusted R^2           = ' + str(r2_adj))
```

```
    #print('Adjusted R^2          = ' + str(r2_adj_2))

print_metrics(y_test1, predictions, 10)
```

Residual Plot

```
def resid_plot(y_test, y_score):
    ## first compute vector of residuals.
    resids = np.subtract(y_test, y_score)
    ## now make the residual plots
    plt.figure()
    sns.regplot(y_score, resids, fit_reg=False)
    sns.set(font_scale=1.5)
    plt.title('Residuals vs. predicted values')
    plt.xlabel('Predicted values')
    plt.ylabel('Residual')

resid_plot(y_test1, predictions)
```

Historgam of Residual

```
def hist_resids(y_test, y_score):
    ## first compute vector of residuals.
    resids = np.subtract(y_test, y_score)
    ## now make the residual plots
    plt.figure()
    sns.distplot(resids)
    sns.set(font_scale=1.5)
    plt.title('Histogram of residuals')
    plt.xlabel('Residual value')
    plt.ylabel('count')

hist_resids(y_test1, predictions)
```

Quantile-Quantile Plot

```
def resid_qq(y_test, y_score):
    ## first compute vector of residuals.
    resids = np.subtract(y_test, y_score)
    ## now make the residual plots
    plt.figure()
    ss.probplot(resids, plot = plt, rvalue=False)
    sns.set(font_scale=1.5)
    plt.title('Residuals vs. predicted values')
    plt.xlabel('Predicted values')
    plt.ylabel('Residual')

resid_qq(y_test1, predictions)
```

Scatter Plot between predicted vs actual value with regression line

```
z = list(zip(predictions.round(),y_test1))
dff = pd.DataFrame(z, columns=['Predicted Value','True Value'])
sns.lmplot(x = 'Predicted Value',y='True Value',data=dff)
```

```
sns.set(font_scale=1.5)
plt.title("True Vs Predicted Value")
```

Distribution Plot

```
plt.figure(figsize=(10, 10))
ax1 = sns.distplot(dff['True Value'], hist=False, color="r", label="Actual Value")
sns.distplot(dff['Predicted Value'], hist=False, color="b", label="Predicted Values" , ax=ax1)
sns.set(font_scale=1.5)
plt.title('Actual vs Predicted Values')
```

Merging Similar Data

- **Creating Empty List**

```
img_name= list()
ControlLine_mean = list()
ControlLine_median = list()
ControlLine_std = list()
TestLine_mean = list()
TestLine_median = list()
TestLine_std = list()
Ratio_of_CL_to_TL = list()
Ratio_of_mean = list()
Ratio_of_median = list()
Ratio_of_std = list()
Concentration = list()

df['Concentration'].unique()
df_new = df[df['Concentration'] == 0]

df_new_1 = df_new.loc[88:90]

img_name.append(df_new_1.ImageName.loc[88])
#img_name.append('60nM DC_I_P_single_V ROI_1')
#img_name.append(df_new_1[0])

ControlLine_mean.append((df_new_1.mean(axis=0))[0])
ControlLine_median.append((df_new_1.mean(axis=0))[1])
ControlLine_std.append((df_new_1.mean(axis=0))[2])
TestLine_mean.append((df_new_1.mean(axis=0))[3])
TestLine_median.append((df_new_1.mean(axis=0))[4])
TestLine_std.append((df_new_1.mean(axis=0))[5])
Ratio_of_CL_to_TL.append((df_new_1.mean(axis=0))[6])
Ratio_of_mean.append((df_new_1.mean(axis=0))[7])
Ratio_of_median.append((df_new_1.mean(axis=0))[8])
Ratio_of_std.append((df_new_1.mean(axis=0))[9])
Concentration.append((df_new_1.mean(axis=0))[10])
```

List Converted into DF

```
zippedList = list(zip(img_name, ControlLine_mean, ControlLine_median,
                      ControlLine_std, TestLine_mean, TestLine_median,
                      TestLine_std,Ratio_of_CL_to_TL, Ratio_of_mean, Ratio_of_median,
                      Ratio_of_std, Concentration))

df2 = pd.DataFrame(zippedList, columns = ['img_name', 'ControlLine_mean',
```

```
'ControlLine_median', 'ControlLine_std',
'TestLine_mean', 'TestLine_median', 'TestLine_std',
'Ratio_of_CL_to_TL', 'Ratio_of_mean',
'Ratio_of_median', 'Ratio_of_std', 'Concentration'])
```

ML Model with Merged Data Frame

- **Reading DF**

```
df2 = pd.read_csv('Concentration_Prediction_DataFrame_Merged.csv', index_col=0)
```

- **Developing ML model**

```
P = df2[['ControlLine_mean', 'ControlLine_median', 'ControlLine_std', 'TestLine_mean',
        'TestLine_median', 'TestLine_std', 'Ratio_of_CL_to_TL', 'Ratio_of_mean',
        'Ratio_of_median', 'Ratio_of_std']]
q = df2['Concentration']

X_train2, X_test2, y_train2, y_test2 = train_test_split(P, q, test_size=0.30, random_state=0)
lm = LinearRegression(fit_intercept=False)
lm.fit(X_train2,y_train2)
lm.score(X_test2, y_test2)

predictions2 = lm.predict(X_test2)
```

Evaluation of ML model with merged DF

- **Evaluation Matrix**

```
print_metrics(y_test2, predictions2, 10)
```

- **Residual Plot**

```
resid_plot(y_test2, predictions2)
```

- **Historgam of Residual**

```
hist_resids(y_test2, predictions2)
```

- **Quantile-Quantile Plot**

```
resid_qq(y_test2, predictions2)
```

- **Scatter Plot between predicted vs actual value with regression line**

```
z = list(zip(predictions2.round(),y_test2))
dff2 = pd.DataFrame(z, columns=['Prediction','Actual'])
sns.set(font_scale=1.5)
sns.lmplot(x = 'Prediction',y='Actual',data=dff2)
plt.title("Actual vs Predicted value")
```

- **Distribution Plot**

```
plt.figure(figsize=(10, 10))

sns.set(font_scale=1.5)
ax1 = sns.distplot(dff2['Actual'], hist=False, color="r", label="Actual Value")
sns.distplot(dff2['Prediction'], hist=False, color="b", label="Predicted Values" , ax=ax1)
plt.title('Actual vs Predicted Values')
```

Final ML model selection procedure

```
colums = ['ControlLine_mean', 'ControlLine_median', 'ControlLine_std', 'TestLine_mean',
          'TestLine_median', 'TestLine_std', 'Ratio_of_CL_to_TL', 'Ratio_of_mean',
         'Ratio_of_median', 'Ratio_of_std']

adj_r2 = list()
r_square = list()
n_parameters =1
q = df2['Concentration']
cols = [ col]

for col in colums:
    P = df2[[col]]
    X_train2, X_test2, y_train2, y_test2 = train_test_split(P, q, test_size=0.30, random_state=0)
    lm = LinearRegression()
    lm.fit(X_train2,y_train2)
    predictions = lm.predict(X_test2)
    r2 = sklm.r2_score(y_test2, predictions)
    r2_adj = r2 - (n_parameters - 1)/(y_train2.shape[0] - n_parameters) * (1 - r2)
    adj_r2.append(r2_adj)
    r_square.append(r2)

print("Adjusted R^2")
print(adj_r2)
print("R^2")
print(r_square)
plt.figure(figsize=(10,10))
plt.scatter(range(len(adj_r2)), adj_r2)
plt.grid()
plt.xticks(range(len(adj_r2)), cols, size='small',fontsize=12)
plt.xticks(rotation=45)
plt.title('Model Selection With Three Independent Variable',fontsize=12)
plt.ylabel('Adjusted R Square',fontsize=10)

P = df2[['Ratio_of_mean','Ratio_of_std','Ratio_of_median','ControlLine_mean','TestLine_mean']]
q = df2['Concentration']
X_train2, X_test2, y_train2, y_test2 = train_test_split(P, q, test_size=0.30, random_state=0)
lm = LinearRegression(fit_intercept=False)
lm.fit(X_train2, y_train2)
predictions3 = lm.predict(X_test2)
```

Evaluation of Final ML model

- **Evaluation Matrix**

```
print_metrics(y_test2, predictions3, 5)
```

- **Residual Plot**

```
resid_plot(y_test2, predictions3)
```

- **Historgam of Residual**

```
hist_resids(y_test2, predictions3)
```

- **Quantile-Quantile Plot**

```
resid_qq(y_test2, predictions3)
```

- **Scatter Plot between predicted vs actual value with regression line**

```
z = list(zip(predictions3.round(),y_test2))
dff = pd.DataFrame(z, columns=['Prediction','Actual'])
plt.figure(figsize=(10,10))
sns.lmplot(x = 'Prediction',y='Actual',data=dff)
plt.title("Actual VS Predicted")
sns.set(font_scale=1.5)
```

- **Distribution Plot**

```
plt.figure(figsize=(10, 10))

ax1 = sns.distplot(dff['Actual'], hist=False, color="r", label="Actual Value")
sns.distplot(dff['Prediction'], hist=False, color="b", label="Predicted Values" , ax=ax1)
sns.set(font_scale=1.5)
plt.title('Actual vs Predicted Values')
```

K-fold Cross Validation

```
df2 = pd.read_csv('Concentration_Prediction_DataFrame_Merged.csv', index_col=0)

R = df22[['ControlLine_mean', 'ControlLine_median', 'ControlLine_std', 'TestLine_mean',
          'TestLine_median', 'TestLine_std', 'Ratio_of_CL_to_TL', 'Ratio_of_mean',
          'Ratio_of_median', 'Ratio_of_std']]
K = df22['Concentration']

R = df2[['Ratio_of_mean','Ratio_of_std','Ratio_of_median','ControlLine_mean','TestLine_mean']]
K = df2['Concentration']

X_train3, X_test3, y_train3, y_test3 = train_test_split(R, K, test_size=0.30, random_state=0)

params_RF = {"max_depth": [3,5,6,7,8,9],
          "max_features":['auto', 'sqrt', 'log2'],
          "min_samples_split": [2, 3,5,7],
          "min_samples_leaf": [1, 3,5,6]}

model_RF_GS = GridSearchCV(RandomForestRegressor(), param_grid=params_RF,cv=4)
model_RF_GS.fit(X_train3,y_train3)

pred_RF_GS = model_RF_GS.predict(X_test3)
sklm.r2_score(y_test3,pred_RF_GS)
```

Evaluation of Final ML model

- **Evaluation Matrix**

```
print_metrics(y_test2, predictions3, 5)
```

- **Residual Plot**

```
resid_plot(y_test3, pred_RF_GS)
```

- **Historgam of Residual**

```
hist_resids(y_test3, pred_RF_GS)
```

- **Quantile-Quantile Plot**

```
resid_qq(y_test3,pred_RF_GS)
```

- **Scatter Plot between predicted vs actual value with regression line**

```
z = list(zip(pred_RF_GS.round(),y_test3))
dff = pd.DataFrame(z, columns=['Prediction','Actual Value'])
sns.lmplot(x = 'Prediction',y='Actual Value',data=dff)
sns.set(font_scale=1.5)
```

- **Distribution Plot**

```
plt.figure(figsize=(10, 10))
ax1 = sns.distplot(dff['Actual Value'], hist=False, color="r", label="Actual Value")
sns.distplot(dff['Prediction'], hist=False, color="b", label="Predicted Values" , ax=ax1)
sns.set(font_scale=1.5)
plt.title('Actual vs Predicted Values')
```

Code for Sigmoid Function

- ***Library***

```
import numpy as np
import matplotlib.pyplot as plt
```

- ***Generating Linearly Spaced Number***

```
x = np.linspace(-10,10,100)
```

- ***Creating Sigmoid Function***

```
def sig(z):
  s = 1/(1+np.exp(-z))
  return s
```

- ***Plotting***

```
plt.figure(figsize=(10,10))
plt.plot(x,sig(x),'k--',c='m',linewidth=4)
plt.yticks(np.arange(0, 1.2, step=0.1))
plt.axhline(0.5,0,10,linewidth=.5, color='r')
plt.axvline(0,0,10,linewidth=.5, color='r')
plt.title('Sigmoid function', fontsize=25)
plt.xlabel('x', fontsize=30)
plt.ylabel('S(x)', fontsize=30)
```

References

[1] Ruppert, C., Phogat, N., Laufer, S., Kohl, M. and Deigner, H.P., 2019. A smartphone readout system for gold nanoparticle-based lateral flow assays: application to monitoring of digoxigenin. Microchimica Acta, 186(2), p.119. https://doi.org/10.1007/s00604-018-3195-6

[2] Brunelli, R., 2009. Template matching techniques in computer vision: theory and practice. John Wiley & Sons.

[3] Aksoy, M.S., Torkul, O. and Cedimoglu, I.H., 2004. An industrial visual inspection system that uses inductive learning. Journal of Intelligent Manufacturing, 15(4), pp.569-574.

[4] Kyriacou, T., Bugmann, G. and Lauria, S., 2005. Vision-based urban navigation procedures for verbally instructed robots. Robotics and Autonomous Systems, 51(1), pp.69-80.

[5] Yang, W.C., 1985. Edge Detection Using Template Matching (Image Processing, Threshold Logic, Analysis, Filters). Duke University, 288.

[6] Talmi, I., Mechrez, R. and Zelnik-Manor, L., 2017. Template matching with deformable diversity similarity. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 175-183).

[7] Lewis, J.P., 2005. Fast normalized cross-correlation, Industrial Light and Magic.

[8] „ Regression analysis “, Wikipedia, last modified December 16, 2019, https://en.wikipedia.org/w/index.php?title=Regression_analysis&action=history

[9] Freedman, D.A., 2009. Statistical models: theory and practice. cambridge university press.

[10] Cook, R.D. and Weisberg, S., 1982. Criticism and influence analysis in regression. Sociological methodology, 13, pp.313-361.

[11] Rencher, A.C. and Christensen, W.F., 2012. Chapter 10, Multivariate regression– Section 10.1, Introduction. Methods of Multivariate Analysis, Wiley Series in Probability and Statistics, 709, p.19.

[12] Rong-Hwa, S., Shiao-Shek, T., Der-Jiang, C. and Yao-Wen, H., 2010. Gold nanoparticle-based lateral flow assay for detection of staphylococcal enterotoxin B. Food Chemistry, 118(2), pp.462-466.

[13] Chen, A. and Yang, S., 2015. Replacing antibodies with aptamers in lateral flow immunoassay. Biosensors and bioelectronics, 71, pp.230-242.

[14] Briechle, K. and Hanebeck, U.D., 2001, March. Template matching using fast normalized cross correlation. In Optical Pattern Recognition XII (Vol. 4387, pp. 95-102). International Society for Optics and Photonics.

[15] Mooi, E., Sarstedt, M. and Mooi-Reci, I., 2018. Regression analysis. In Market Research (pp. 210-263). Springer, Singapore.

[16] Swaroop, P. and Sharma, N., 2016. An overview of various template matching methodologies in image processing. International Journal of Computer Applications, 153(10), pp.8-14.

[17] "MODEL EVALUATION – REGRESSION MODELS", "datavedas.com", visited on April 2, 2020, Available on https://www.datavedas.com/model-evaluation-regression-models/

[18] "How to Interpret Adjusted R-Squared and Predicted R-Squared in Regression Analysis", "https://statisticsbyjim.com/", visited on April 3, 2020, https://statisticsbyjim.com/regression/interpret-adjusted-r-squared-predicted-r-squared-regression/

[19] "Statistics Dictionary", "https://stattrek.com/", visited on April 4, 2020, "https://stattrek.com/statistics/dictionary.aspx?definition=residual%20plot"

[20] "Check Your Residual Plots to Ensure Trustworthy Regression Results!", "Statistics By Jim", visited on April 4, 2020, "https://statisticsbyjim.com/regression/check-residual-plots-regression-analysis/"

[21] Piñeiro, G., Perelman, S., Guerschman, J.P. and Paruelo, J.M., 2008. How to evaluate models: observed vs. predicted or predicted vs. observed?. Ecological Modelling, 216(3-4), pp.316-322.

[22] "Improve Your Model Performance using Cross Validation (in Python and R)", "Analytics Vidhya", visited on April 12, 2020, "https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r/"

[23] "Q-Q plot", Wikipedia, last modified: December 30, 2019; Visited on April 12, 2020; "https://en.wikipedia.org/wiki/Q%E2%80%93Q_plot"

[24] "Quantile-Quantile Plot", "Engineering Statistics", Visited on April 12, 2020; "https://www.itl.nist.gov/div898/handbook/eda/section3/qqplot.htm"

[25] "Understanding Q-Q Plots", Visited on April 12, 2020; "https://data.library.virginia.edu/understanding-q-q-plots/"

[26] "Polynomial Regression", last modified December 1, 2019; visited on April 15, 2020; "https://en.wikipedia.org/wiki/Polynomial_regression"

[27] Yetisen, A.K., Akram, M.S. and Lowe, C.R., 2013. based microfluidic point-of-care diagnostic devices. Lab on a Chip, 13(12), pp.2210-2251.

[28] "ABINGDON HEALTH", visited on April 16, 2020; "https://www.abingdonhealth.com/contract-services/what-is-a-lateral-flow-immunoassay/"

[29] "Lateral flow test", (2020) Wikipedia, Available at: https://en.wikipedia.org/wiki/Lateral_flow_test , (Accessed: 16 April 2020)

[30] Hansson, J., Yasuga, H., Haraldsson, T. and Van der Wijngaart, W., 2016. Synthetic microfluidic paper: high surface area and high porosity polymer micropillar arrays. Lab on a Chip, 16(2), pp.298-304.

[31] Guo, W., Hansson, J. and van der Wijngaart, W., 2016. Viscosity Independent Paper Microfluidic Imbibition. In The 20th International Conference on Miniaturized Systems for

Chemistry and Life Sciences, MicroTAS 2016, 9-13 October 2016, Dublin, Ireland (pp. 13-14). MicroTAS.

[32] Lee, L.G., Nordman, E.S., Johnson, M.D. and Oldham, M.F., 2013. A low-cost, high-performance system for fluorescence lateral flow assays. *Biosensors*, *3*(4), pp.360-373.

[33] "What is Machine Learning? A definition", (2017), Expert System, Available at: https://expertsystem.com/machine-learning-definition/ , (Accessed:16 April 2020)

[34] "Machine Learning", (2020), Wikipedia, Available at: https://en.wikipedia.org/wiki/Machine_learning , (Accessed: 16 April 2020)

[35] Raschka, S., 2018. Model evaluation, model selection, and algorithm selection in machine learning. *arXiv preprint arXiv:1811.12808*.

[36] A Gentle Introduction to k-fold Cross-Validation (2018), Machine Learning Mastery, Available at: https://machinelearningmastery.com/k-fold-cross-validation/ , (Accessed: 17 April 2020)

[37] "Supervised learning", (2020) Wikipedia, Available at: https://en.wikipedia.org/wiki/Supervised_learning , (Accessed: 16 April 2020)

[38] "Machine Learning – Existing Applications", (2017) Available at: https://www.apriorit.com/dev-blog/472-machine-learning-applications , (Accessed: 17 April 2020)

[39] "Unsupervised learning", (2020) Wikipedia, Available at: https://en.wikipedia.org/wiki/Unsupervised_learning , (Accessed: 16 April 2020)

[40] "Three Things to Know About Reinforcement Learning" (2019) Available at: https://www.kdnuggets.com/2019/10/mathworks-reinforcement-learning.html , (Accessed: 17 April 2020)

[41] Freedman, D.A., 2009. Statistical models: theory and practice. cambridge university press.

[42] "Linear regression", (2020) Wikipedia, Available at: https://en.wikipedia.org/wiki/Linear_regression , (Accessed: 16 April 2020)

[43] "7 Regression Techniques you should know!" (2015) Analytics Vidhya, Available at: https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/ , (Accessed: 24 April 2020)

[44] "Logistic regression", (2020) Wikipedia, Available at: https://en.wikipedia.org/wiki/Logistic_regression , (Accessed: 24 April 2020)

[45] "Mean squared error", (2020), Wikipedia, Available at: https://en.wikipedia.org/wiki/Mean_squared_error , (Accessed: 24 April 2020)

[46] "What are Mean Squared Error and Root Mean Squared Error?", (2018), vernier.com, Available at: https://www.vernier.com/til/1014 , (Accessed: 24 April 2020)

[47] "Mean Squared Error: Definition and Example", (2020), Statistics How To (.com), Available at: https://www.statisticshowto.com/mean-squared-error/ , (Accessed: 24 April 2020)

[48] "MAE and RMSE — Which Metric is Better?", (2016), medium.com, Available at: https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d , (Accessed: 24 April 2020)

[49] "Coefficient of determination", (2020), Wikipedia, Available at: https://en.wikipedia.org/wiki/Coefficient_of_determination , (Accessed: 24 April 2020)

[50] Carpenter, R.G., 1960. Principles and procedures of statistics, with special reference to the biological sciences. *The Eugenics Review*, *52*(3), p.172.

[51] Glantz, S.A., Slinker, B.K. and Neilands, T.B., 1990. *Primer of applied regression and analysis of variance* (Vol. 309). New York: McGraw-Hill.

[52] Draper, N.R. and Smith, H., 1998. *Applied regression analysis* (Vol. 326). John Wiley & Sons.

[53] "Coefficient of Determination", (2020), Investopedia.com, Available at: https://www.investopedia.com/terms/c/coefficient-of-determination.asp , (Accessed: 24 April 2020)

[54] "Sum of Squares Total, Sum of Squares Regression and Sum of Squares Error", (2020), 365datascience.com, Available at: https://365datascience.com/sum-squares/ , (Accessed: 24 April 2020)

[55] Alpaydin, E., 2020. *Introduction to machine learning*. MIT press.

[56] Wang H., Zheng H. (2013) Model Validation, Machine Learning. In: Dubitzky W., Wolkenhauer O., Cho KH., Yokota H. (eds) Encyclopedia of Systems Biology. Springer, New York, NY

[57] Mitchell, T.M., 1997. Machine learning.

[58] "Template Matching", scikit-image (documentation), Available at: https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_template.html , (Accessed: 24 January 2020)

[59] Tataru, A., 2017. Metrics for Evaluating Machine Learning Cloud Services.

[60] Zheng, A., 2015. Evaluating machine learning models: a beginner's guide to key concepts and pitfalls.

[61] Legates, D.R. and McCabe Jr, G.J., 1999. Evaluating the use of "goodness-of-fit" measures in hydrologic and hydroclimatic model validation. *Water resources research*, *35*(1), pp.233-241.

[62] Oliphant, T.E., 2007. Python for scientific computing. *Computing in Science & Engineering*, *9*(3), pp.10-20.

[63] Van Rossum, G., 2007, June. Python Programming Language. In *USENIX annual technical conference* (Vol. 41, p. 36).

[64] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J., 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, *12*, pp.2825-2830.

[65] Van der Walt, S., Schönberger, J.L., Nunez-Iglesias, J., Boulogne, F., Warner, J.D., Yager, N., Gouillart, E. and Yu, T., 2014. scikit-image: image processing in Python, PeerJ, 2, e453.

[66] Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J. and van der Walt, S.J., 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods*, *17*(3), pp.261-272.

[67] Seabold, S. and Perktold, J., 2010, June. Statsmodels: Econometric and statistical modeling with python. In *Proceedings of the 9th Python in Science Conference* (Vol. 57, p. 61). Scipy.

[68] Developers, N., 2013. NumPy. *NumPy Numpy. Scipy Developers*, p.31.

[69] McKinney, W., 2015. Pandas, Python Data Analysis Library. *see http://pandas. pydata. org*.

[70] Waskom, M., Botvinnik, O., Hobson, P., Warmenhoven, J., Cole, J.B., Halchenko, Y., Vanderplas, J., Hoyer, S., Villalba, S., Quintero, E. and Miles, A., 2014. Seaborn: statistical data visualization. *URL: https://seaborn. pydata. org/(visited on 2017-05-15)*.

[71] Hunter, J.D., 2007. Matplotlib: A 2D graphics environment. *Computing in science & engineering*, *9*(3), pp.90-95.

[72] Barrett, J.P., 1974. The coefficient of determination—some limitations. *The American Statistician*, *28*(1), pp.19-20.

[73] Akossou, A.Y.J. and Palm, R., 2013. Impact of data structure on the estimators r-square and adjusted r-square in linear regression. *Int. J. Math. Comput*, *20*, pp.84-93.

[74] Ricci, L., 2010. Adjusted R-squared type measure for exponential dispersion models. *Statistics & probability letters*, *80*(17-18), pp.1365-1368.

[75] Posthuma-Trumpie, G.A., Korf, J. and van Amerongen, A., 2009. Lateral flow (immuno) assay: its strengths, weaknesses, opportunities and threats. A literature survey. *Analytical and bioanalytical chemistry*, *393*(2), pp.569-582.

[76] Sarraf, S., Saverino, C. and Golestani, A.M., 2016, February. A robust and adaptive decision-making algorithm for detecting brain networks using functional mri within the spatial and frequency domain. In *2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)* (pp. 53-56). IEEE.

[77] Willmott, C.J. and Matsuura, K., 2005. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate research*, *30*(1), pp.79-82.

[78] Chai, T. and Draxler, R.R., 2014. Root mean square error (RMSE) or mean absolute error (MAE)?. *GMDD*, *7*(1), pp.1525-1534.

[79] Rawlings, J.O., Pantula, S.G. and Dickey, D.A., 2001. *Applied regression analysis: a research tool*. Springer Science & Business Media, (pp 228 - 389).

[80] "Feature engineering, Explained", (2018), Available at: https://www.kdnuggets.com/2018/12/feature-engineering-explained.html , (Accessed: 28 January 2020)

[81] Changyong, F.E.N.G., Hongyue, W.A.N.G., Naiji, L.U., Tian, C.H.E.N., Hua, H.E. and Ying, L.U., 2014. Log-transformation and its implications for data analysis. *Shanghai archives of psychiatry*, *26*(2), p.105.

[82] "Residual Plot Analysis", Available at: https://www.originlab.com/doc/Origin-Help/Residual-Plot-Analysis , (Accessed: 28 January 2020)

[83] Harel, O., 2009. The estimation of R 2 and adjusted R 2 in incomplete data sets using multiple imputation. *Journal of Applied Statistics*, *36*(10), pp.1109-1118.