University
of Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study program/specialization: Computer Science	Spring semester, 2018 Open / ~~Confidential~~
Author: Showmen Das Gupta	Showmen Dasgupta (signature of author)
Internal Supervisor: **Tomasz Wiktorski** External Supervisor(s): **Frank Rørtvedt (Siemens), Georg Schöler (Siemens)**	
Title of Master's Thesis: **Virtual Field Service Ecosystem (VSE) using AR (Augmented Reality) collaboration with SiemensAG**	
Credits: 30 ECTS	
Keywords: Augmented Reality, Virtual Environments, Virtual Reality, Machine Learning, AR applications, Data analysis	Number of pages: + supplemental material/other: - 123 (with source code in Appendix) Stavanger, June 15, 2018

Title page for Master's Thesis
Faculty of Science and Technology

University of Stavanger

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

Virtual Field Service Ecosystem (VSE) using AR (Augmented Reality) collaboration with SiemensAG

Master's Thesis in Computer Science
by

Showmen Das Gupta

Internal Supervisors

Tomasz Wiktorski (UiS)

External Supervisors

Frank Rørtvedt (Siemens)

Georg Schöler (Siemens)

June 15, 2018

"A dream doesn't become reality through magic; it takes sweat, determination and hard work"

Collin Powell

Abstract

With the huge advancement of technologies, our viewpoint to see, hear, observe and feel the surroundings around us is changing every single moment. Building a virtual ecosystem is an idea which needs much time and effort. The purpose of this project is building an ecosystem with AR applications combining machine learning features. In this way, users can gain access to information in a very interactive, contextualized ways which provide a deeper understanding of the physical problems around them and how to solve them easily. A smart machine learning algorithm is only possible if the data provided is concrete and huge to perform any thorough analysis. In this experiment, huge data containing different significant features of a single feed production machine from Siemens is provided where the quality of the product is depended on pressure. Analysis of that data is performed showing graphs, selecting features, validations, mathematical implementations or statistical analysis to propose a model. The significant part of the model building is predicting pressure value for advanced maintenance of the machine and accuracy of the model must be high. The predicted data, analysis of graphs and validation results is proposed to be stored on a cloud system. The AR application is supposed to show ML results. This includes showing every data that is stored in the cloud in the AR application. That way the AR and machine learning are combined in a single application which has the possibility to be extended later for bigger solutions.

Keywords: Augmented Reality, Virtual Environments, Virtual Reality, Machine Learning, AR applications, Data analysis

Acknowledgements

"First of all, I want to thank Siemens(Stavanger) and my external supervisor Frank Rørtvedt (Siemens) for giving me the opportunity to work with Siemens on this exciting project. They have helped me every possible way they can to make it a successful one. I also want to thank my internal supervisor Tomasz Wiktorski to help and guide me properly through the process of finishing my thesis. He has helped me to understand many important aspects that I have learned while working with him. Finally, I want to thank my family and friends because they have always been supportive and helpful to me."

Contents

Abstract **v**

Acknowledgements **vii**

Abbreviations **xi**

1 Introduction **1**

1.1 Motivation 1

1.2 Problem Description 2

1.2.1 Possible Solutions Approach: 3

1.3 Augmented Reality: 4

1.4 Machine Learning: 6

1.5 Evolution of Augmented Reality: 7

1.5.1 Different Types of AR Applications: 8

1.6 Evolution of Machine Learning: 10

1.6.1 Different Types of Machine Learning Algorithms: 11

1.6.2 Supervised Learning: 12

1.6.3 Unsupervised Learning 14

2 Literature Review **17**

2.1 Choosing the correct Regression Model: 18

2.2 Linear Regression 19

2.3 Support Vector Machine-Regression 21

2.4 K Nearest Neighbors-Regression 23

2.5 Random Forest Regression 25

3 Solution Approach **27**

3.1 Introduction 27

3.2 Data Visualization: 27

3.2.1 Merging or Deleting Variables 31

3.3 Feature Selection 32

3.3.1 Building Features . . . 33
3.3.2 Feature Modification . . . 33
3.3.3 StatsModel Evaluation . . . 34
3.3.4 Processing phase: . . . 35
3.4 Analysis to Select Model Algorithm . . . 37
3.5 Cross Validation split (KFold) . . . 40
3.6 Analysis . . . 43
3.7 Proposed Solution . . . 44

4 Experimental Evaluation **47**
4.1 Experimental Setup and Evaluation . . . 47
4.1.1 Residual Plots and Mean squared error . . . 47
4.1.2 True Value vs. Predicted Value and Accuracy . . . 52
4.1.3 RMSE,MAE,MedAE Calculations: . . . 55
4.2 Experimental Results . . . 57
4.2.1 Confusion Matrix and Classification Report . . . 57
4.2.2 Keras Model Evaluation . . . 59
4.2.3 Important Feature . . . 60
4.2.4 Comparison Graphs . . . 61
4.2.5 Forecasting and Plotting Regression . . . 66
4.2.6 Final Summary . . . 67

5 Discussion, Summary and Future Directions **69**
5.1 Problems Observed: . . . 70
5.2 Future Directions: . . . 70
5.3 Summary: . . . 71

List of Figures **72**

List of Tables **75**

A Source Code **77**

Bibliography **107**

Abbreviations

Acronym	**W**hat (it) **S**tands **F**or
AR	**A**ugmented **R**eality
VR	**V**irtual **R**eality
SVM	**S**ervice **V**ector **M**achine
SVR	**S**ervice **V**ector **R**egressor
KNN	**K** **N**earest **N**eighbors
UM	**U**ser **M**odeling
ML	**M**achine **L**earning

Chapter 1

Introduction

The idea of Augmented Reality came to light as a research idea or concept during early 1990's. It conjugates real and virtual objects basically computer-generated content in a real environment. This approach allows the user to make a connection with the real world by using computer generated interfaces. The idea, in past few years has been developed so much with innovation by the researchers related to it. With the increasing demand for data analysis and Machine Learning (ML), combining both to build up an ecosystem may provide more mobility and improvement. This can save time and money. Many big companies are stepping forward to make Machine Learning (ML) a media to develop systems. This may optimize every process, learn them, increase performance [1]. Analysis of a huge source of data is done by many data scientist, which later leads to building up a sophisticated model. Combining ML with Augmented reality strengthens this process more. Due to features of Machine Learning(ML), many significant problems are being fixed. For example, complexity, high-dimensionality frequent variability etc. It is not easy to instantly unleash knowledge and necessary information from real, unstructured and difficult large data-sets. Therefore, there is an urgent need for performing Machine Learning(ML) to big data [2].

1.1 Motivation

Siemens is one of the biggest companies in the world where they have so many industrial sectors that are producing many quality products all around the world. Every industrial sector has many sophisticated machines which are controlled by many well trained and educated engineers. Their goal is always to produce the best quality of product to maintain the priorities of Siemens. To continue this, the performances of machines must be perfect and maintained properly. Maintenance of those machines can be a big aspect there. The motivation here is predicting the maintenance early to save time and money. All those machines' production processes are stored as data to keep track. The purpose of this experiment is using those data for predicting maintenance early and send feedback via an AR application using Hololens. Machine Learning (ML) is an amazing tool to detect future possibilities. In this experiment, data from Siemens's, feed industry have been used where the pressure of a certain machine have been predicted as it influences the quality of production. There is a certain level where the pressure needs to stay to maintain the quality. If there is a significant change in the pressure values that means, there is something wrong with the machine then it requires maintenance.

1.2 Problem Description

Perfect problem visualization means to start with a small approach and make it larger successively. Building a virtual ecosystem is a complex task. So, a problem of a smaller scale has been discussed here and later it can be expanded. If a real-life scenario is discussed in the industries within Siemens, any specific machine or equipment can have technical problems or quality issues with the finished product. It may lead to reduce or even stop production. The challenge is how we can predict the problems in advance and provide effective solutions. We have a cattle feed machine which has several ingredients that controls the quality of the finished products. While on production time, any of the problems with the ingredients or any internal issues can occur. Alarms help to detect the problem. What if we can predict the problems in advance and solve it virtually? Thinking about it as an idea is very complicated but it is possible to materialize. To start, system architecture has been proposed to solve this problem in a smaller scale which has been expanded later. Predicting any problem is a big challenge. This is the part where Machine Learning (ML) plays a big role. Values that influence the quality of any finished goods can be analyzed or finished product parameters that ensure the quality can be useful as well. This data have been used to predict in potential future issues that can appear. With the predictions the solutions may be provided in advance. Cloud is a very strong option now a day when we consider storing our data or using the data to make proper solutions. Predictions and solutions have been proposed to upload instantly. AR application devices have been used to show the prediction and solution in the ecosystem. The application have been proposed to help us deciding about the solution and sending feedback to resolve it in advance. So as soon as the application is running in the devices, it starts analyzing and provides predictions of any future problems along with graphs.

1.2.1 Possible Solutions Approach:

To discuss the approach some specific steps to build up the system will be discussed below: -

1. First, Analyze the data that controls the quality of the system to figure out features.
2. Making statistical analysis and figuring out what we can predict from the data.
3. After analyzing a Machine Learning prediction model has been introduced to predict if there can be any possible problem that can occur in future.
4. Several graphs of validation and statistical analysis has been made from the data.
5. These validation analyses of data with graphs have been uploaded into the cloud.
6. The prediction results and solution for the problems have been stored in the cloud database as well.
7. AR Application has a user-friendly interface to alert and describe the predictions to the users by presenting graphs.
8. After predictions, the solutions and feedbacks have been sent to respective people with the application in Hololens.

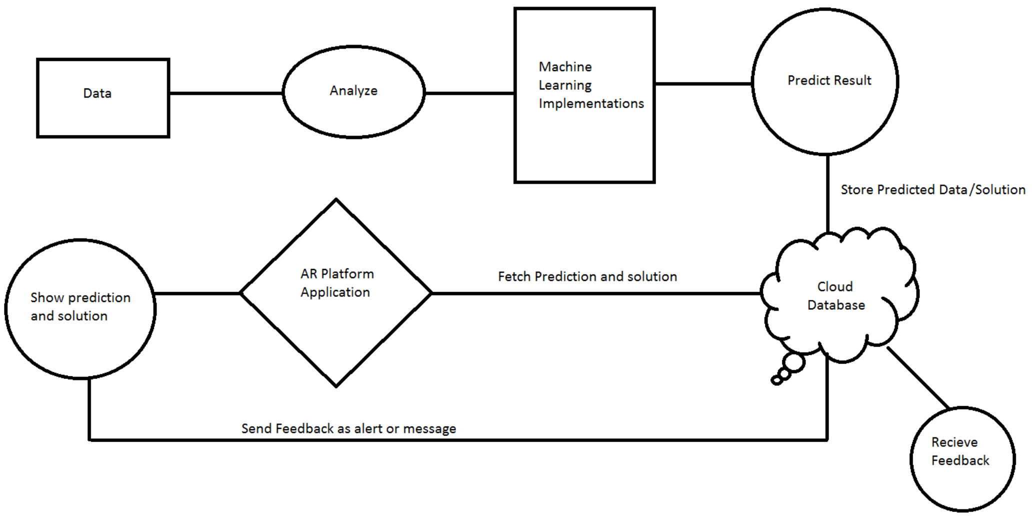

Data
Analyze
Machine Learning Implementations
Predict Result
Store Predicted Data/Solution
Show prediction and solution
AR Platform Application
Fetch Prediction and solution
Cloud Database
Send Feedback as alert or message
Recieve Feedback

Figure 1.1: Proposed System Architecture

The above figure 1.1 is a simple representation of the proposed solution in a graphical way. The steps that have been described before this figure sums up all of them in a graphical presentation. This proposed system architecture also represents each necessary steps to solve this problem.

1.3 Augmented Reality:

Augmented reality is a technology which is used to overlay and interact with digital information (images, 3D models, videos) onto a real-time environment. It can be imagined as the convergence of physical and virtual worlds into a new mixed reality. Simulation of structures in existing hardware and presence or absence of problems can be visualized very easily via AR solutions. It can also replace telephonic guided-assistance by AR navigation to the problem site, visual representation of task or problem at hand, augmented problem solving, inspection or assistance from experts at remote locations, installation and repair of faulty hardware (control electronics, sensors, actuators, motors), interactive documentation for service technicians on the field, visual representation of real-time values(sensor data, status, trends etc.) from physical assets for better understanding of process during service or optimization, planning of new equipment and machinery. The technology of augmented reality ensures integrating pictures of virtual objects into images of the real world. The images can be taken using a camera or a see-through head-mounted display that helps the user to have a direct view of the augmented world. Many large companies are trying to utilize the technology and improve their design and construction process using computers which will replace physical prototypes with virtual prototypes for packaging, assembly, and security evaluation. The technology of augmented information forms a component for user's visual context and the interaction is measured by fine-grained interfaces where relevance can be made as well as the search is refined. Retrieval process for such a system could be demonstrated as retrieval based on zooming through augmented reality or text entry using by zooming through forecasted alternative textual extensions. The respective scenario can be several elements. But first objects and people are recognized as potential cues with pattern recognition methods [3]. The best advantage of the technology is, it overlays computer graphics on the real world. These advantages of the technology work based on a defined field or describing many problems or summarizing development up to a certain point [4].

Within AR more general context is called mixed reality which represents a multi-axis spectrum of areas that covers Virtual Reality, AR, telepresence and other significant technologies. Virtual reality is a word used for computer-generated 3D environments that permit the user to get in and interact with a synthetic environment [4]. The users are capable to "merge" themselves to varying degrees in the computers artificial world which may either be a simulation of some form of reality or the simulation of a complex phenomenon [4]. AR is more likely treated a technology between VR and telepresence. In VR the environment is totally synthetic also telepresence is totally real but in AR the user sees the real world augmented with virtual objects. When structuring AR system three important aspects should be kept in mind: (1) Consolidation of real and virtual world (2) Interactivity in real time (3) Registration in 3D [4]. Wearable devices means the head-mounted displays (for example: Hololens) could be used to show the augmented scene though there are other different technologies. On the other hand, in some AR applications user is not allowed to move around much because of device limitations but in some, the application's user needs to move around a large environment. Here portability is the biggest issue [4].

1.4 Machine Learning:

Machine learning is a complex field of computer science which provides computers with the ability to learn about any big data and provide excellent solutions without being explicitly programmed [5]. ML is very much related to computational statistics which basically focuses on prediction making using computers. It provides strong mathematical optimization which delivers methods, theory and application domains to the field. ML has a very close relation with data mining. Any data can be complex and the more complex it is, the depth increases more. With the increase of depth training very deep networks becomes a big problem. Visualizing a virtual ecosystem is not an easy task, a lot of complexity and data analysis comes into the spotlight. The data can have hundreds of layers which need to be analyzed properly and later it will be very useful to propose a solution which can be extended to a bigger aspect.

Over the past two decades Machine Learning has become one of the mainstays of information technology and with that a rather essential and albeit mostly unseen part of our life. With the ever-expanding amounts of data becoming accessible there is a valid reason to believe that smart data analysis will become even more inescapable as a necessary ingredient for technological progress.

Machine learning can emerge in many aspects. To solve problems some basic tools from statistics and probability theory are used. They simplify the language in which many machine learning problems must be remarked to become convenient to fix. Finally bounding a set of basic, yet effective algorithms to solve an important problem is necessary. It is also important to recognize learning problems according to the type of data or amount of data is used. This helps when confronting new challenges since quite often problems on similar data types can be solved with very similar approaches [2].

Machine learning is closely related to calculating statistics which often targets on prediction making through the help of computers. It has a powerful affiliation with mathematical escalation which provides methods, theory and application domains to the field. It is always converged with data mining where the concluding sub-field spotlights more on exploratory data analysis and is recognized as unsupervised learning. Machine learning can also be unsupervised and be adapted to determine and build profiles for various items and then optimized to discover significant anomalies. Within the territory of data analytics, machine learning is an approach used to construct complex models and algorithms that allow themselves to prediction, in commercial use which is known as predictive analytics. These analytical models grant researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trend in data [6].

1.5 Evolution of Augmented Reality:

The early arrival of (AR) started in the 1950s when Morton Heilig, a cinematographer, speculated of cinema is an action that would have the capability to pull the spectator into the onscreen action by catching all the senses in a compelling manner. In 1962, Heilig framed a mock-up of his perception, which foreshadows digital computing. Next, Ivan Sutherland created the head mounted display in 1966. In 1968, Sutherland was the first pioneer to design and build an augmented reality system adopting an optical see-through head-mounted display [7].

In 1975, Mayron Krueger built the Video place, a room that grants users to communicate with virtual objects for the first time. Later, Tom Caudell and David Mizell from Boeing established the idiom Augmented Reality while assisting works to congregate wires and cable for an aircraft. They also initiate argument on advantages of Augmented Reality versus Virtual Reality (VR), such as needing less power since lesser pixels are needed. Same year L.B Rosenberg built one of the first functioning AR systems, called virtual fixtures and elaborated its benefit on human performance while Steven Feiner, Blair MacIntyre, and Doree Seligmann explained the first major paper on an AR system prototype name KARMA [7].

The reality virtual continuity seen is not defined until 1994 by Paul Milgram and Fumio Kishino as a continuity that stretched from the real environment to the virtual environment. In 1997, Ronal Azuma addresses the first survey in AR backing a widely acknowledged definition of AR by discovering it as a mixture of the real and virtual environment while being recorded in 3D and reciprocal in real time. The first outdoor mobile AR game, ARQuake, is built by Bruce Thomas in 2000 and explained during the International Symposium on wearable computers. In 2005, the horizon report predicts that AR technologies will take over fully within the 4-5years [7].

1.5.1 Different Types of AR Applications:

Several types of augmented reality technology exist, each with exceptional differences in their goals and applicational use cases. Different type processes of AR are given below: [8] 1. Marker Based AR. 2. Markless Augmented reality 3. Superimposition based Augmented reality

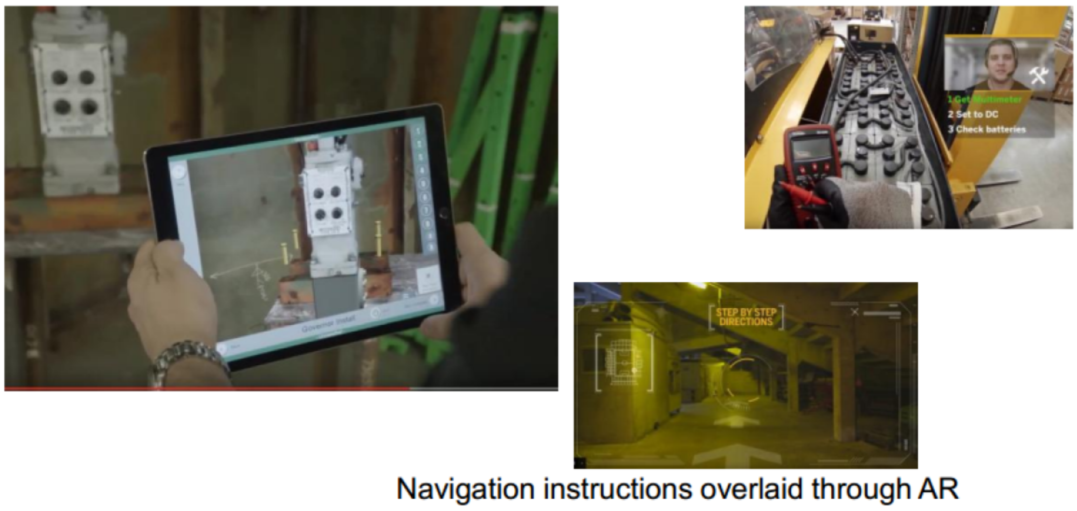


Figure 1.2: AR Application Example

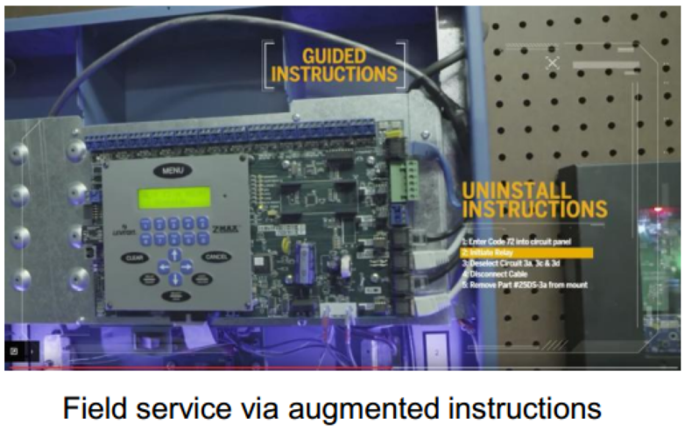

GUIDED INSTRUCTIONS
UNINSTALL INSTRUCTIONS
1. Enter Code 72 into circuit panel
2. Initiate Relay
3. Deselect Circuit 3a, 3c & 3d
4. Disconnect Cable
5. Remove Part #250S-3a from mount
Field service via augmented instructions

Figure 1.3: AR Application Example in Field Service

In Siemens, several kinds of AR technologies have already been applied to the field. In figure 1.2 and 1.3 explains that several kinds of research are going on to make a better experience for the customers and engineers. In figure 1.2 there are some examples of how navigation instructions are operated through AR and 1.3 represents some field service is being handled by AR an application.

Apart from these use cases, there are different kinds of AR applications which are applicable in the real scenarios. Applicable example scenarios are given below:

1. Virtual collaboration and remote-expert driven assistance for field operatives (ex: control room or command tower experts)
2. Visualization and interpretation made easier with 3D models projected onto the AR space to have a holistic overview of the incident in context.
3. Security staff and associated personnel can perform virtual "check-ins" at designated checkpoints to validate presence or localize suspicious activity.
4. Camera feeds of field operatives wearing AR devices can be used additionally alongside fixed surveillance cameras.
5. AR overlays of camera feed (see through walls), blind spots and public or restricted areas.

Roles applicable: Security staff, emergency personnel (fire brigades, evacuation team, etc.), Subject experts in incident handling and response.

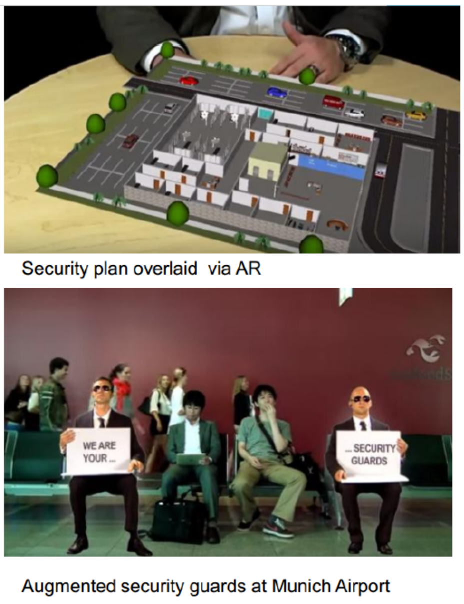

Security plan overlaid via AR
WE ARE YOUR ...
... SECURITY GUARDS
Augmented security guards at Munich Airport

Figure 1.4: AR Applications different roles applicable

Figure 1.4 shows some examples where AR applications are already being used for different roles where it is a very useful and handy tool.

1.6 Evolution of Machine Learning:

As rapidly the electronic computers appeared into adoption in the fifties and sixties, there evolved algorithms would allow modeling and analyzing large sets of data. From the very start, the three considerable branches of machine learning materialized. Classical work in figurative learning is described by Hunt et al. (1966), in statistical methods by Nilsson (1965) and in neural networks by Rosenblatt (1962). In last few years all three branches introduced advanced procedures (Michie et al.,1994): statistical or pattern recognition mechanisms, such as the k-nearest neighbors, discriminant analysis, and Bayesian classifiers, inductive learning of symbolic rules, such as top-down induction of decision trees, decision rules and induction of logic programs and artificial neural networks, such as multilayered feedforward neural network with back propagation learning, the Kohonen's self organizing network and the Hopfield's associative memory [9].

Research into the proper utilization of machine learning in past few decades has observed a significant change. It has achieved the capability to help user modeling pass through a duration of a downturn and then a revival, with the research area at the near twentieth century more alive and dynamic than at any earlier time. It is alluring to recognize the start of the ML for UM (user modeling) as being apparent by the publication of Self's (1988) paper in which he affirmed that a search difficulty that came into sight to inhibit an explicit machine learning approach to speculate possible cognitive process models for a relatively plain modeling task was 'clearly intractable'. While the paper did not disagree that student modeling was intractable per se, the phrase 'the intractable problem of student modeling', captured from the title of that paper, has been often duplicated, perhaps with less concentration to the excellent elaborate argument within the paper that might be expected. Without needing to impute causes to the ML for any implementations, it is eminent that it was anticipated by a decade of much work and effort. Notable examples from this era include the work of Brown and Burton (1978), Brown and VanLehn (1980), Gilmore and Self (1988), Langley and Ohlsson (1984), Mizoguchi et al. (1987), Reiser et al. (1985), Sleeman (1984), ValLehn(1982) and Younf and O'Shea(1981), much of it in the area of student modeling. In contrast, the period 1988-1994 saw relatively less activity in the sector [10].

With research described above, it is clear that real machine learning research started late, and its development process can be divided into 3 periods:

1. The early stage is from the middle of the 1950s to the middle of 1960s, which labeled as the warm period. 2. The second stage is from the middle of the 1960s to the middle of 1970s, which labeled as the calm period in machine learning. 3. The final stage is from the middle of the 1970s to the middle of 1980s, established as the rebirth period in machine learning.

The earliest stage starts in 1986. At that period, machine learning embraced the inclusive applications of psychology, neurophysiology and biology, and mathematics, automation and computer science which paved the way of establishing theoretical grounding on machine learning. Then through merging several learning approaches, they established a sophisticated learning system. Furthermore, the consensus of views of various normal problems of machine learning and artificial intelligence were developed, and the application field in several learning approaches continued to be broader. In the meantime, commercial machine learning products came to light, but also admissible academic tasks of machine learning were also passionately carried out. In 1989, J. G. Carbonell indicated four auspicious areas about machine learning: connection machine learning, symbol-based induced machine learning, genetic machine learning and analyzing machine learning. In 1997, T. G. Dietterich again conveyed another four new research directions: ensembles of classifiers, methods for scaling up supervised learning algorithm, reinforcement learning and learning complex stochastic models. In the rising history of machine learning, it is very significant to mention the father of the artificial brain, Professor Hugo de Garis. He built the CBM brain-machine which was able to operate the evolution of a neural network within few seconds and could handle approximately 0.1 billion artificial neurons. Its computing power was like 10000 personal computers [2].

Several years ago, Google, Facebook, Twitter, Microsoft, Netflix, Amazon and other international IT giants have observed the importance of machine learning and advanced its related research. 2014 was also an exceptional year because the amazing image processing and classifying techniques have been tested even in excellent paintings and several anonymous influences between famous artists were exposed [2].

1.6.1 Different Types of Machine Learning Algorithms:

Machine learning is practiced educating machines how to grasp the data more accurately. Occasionally after examining the data, we cannot make sense of the design or extract information from the data. In that case, we apply machine learning. With the affluence of data sets available, the interest in machine learning is in acceleration. Many industries from medicine to military use machine learning to extract relevant information. The goal of machine learning is to learn by themselves [11]. With the advancement of machine learning techniques, there are certain algorithms accessible we can use. The taxonomy of machine learning algorithms examine the training data during the model preparation process for getting the best result [2].

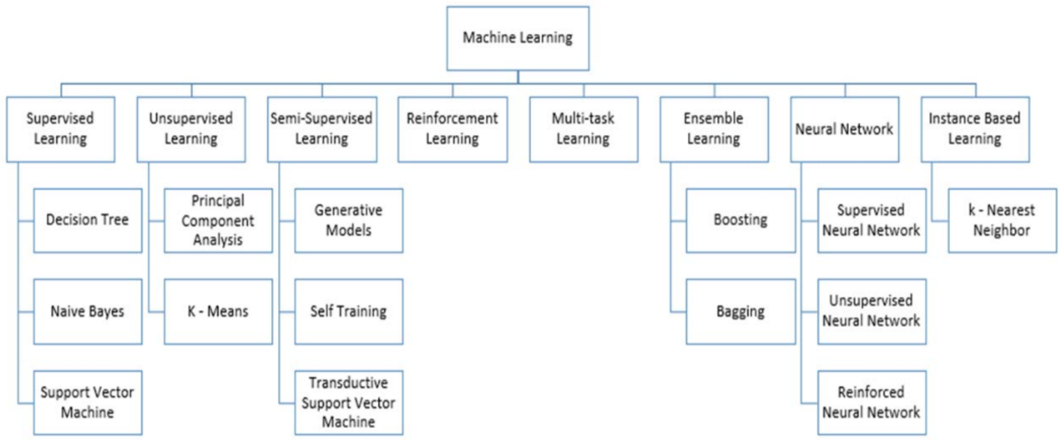


Figure 1.5: Machine Learning Algorithms Flow Chart [2]

Figure 1.5 is a flowchart of different machine learning approaches that are most likely to be used. All these algorithms are applied for different purposes and the different pattern of data. But here we are going to discuss only supervise and unsupervised learning.

1.6.2 Supervised Learning:

Supervised machine learning algorithms are such kind of algorithms which needs extraneous assistance. The input dataset is breaking down into train and test dataset. The training dataset contains output variable which needs to be predicted or classified. All algorithms educate themselves with patterns from the training dataset and bestow them to the test dataset for prediction or classification. The different processes of supervised machine learning algorithm are shown in the above picture [2]. Supervised learning is familiar in classification issues because the ambition is frequently to make the computer to understand a classification system that has been developed. Digit recognition is an acceptable example of classification learning. Most importantly, classification learning is applicable for any issue where figuring out a classification is advantageous, and the classification is simple and easy to conclude. Supervised learning frequently leaves the probability for inputs vague. This model is not required if the inputs are accessible, but if some of the input values are missing, it is not conceivable to conclude anything about the outputs [2].

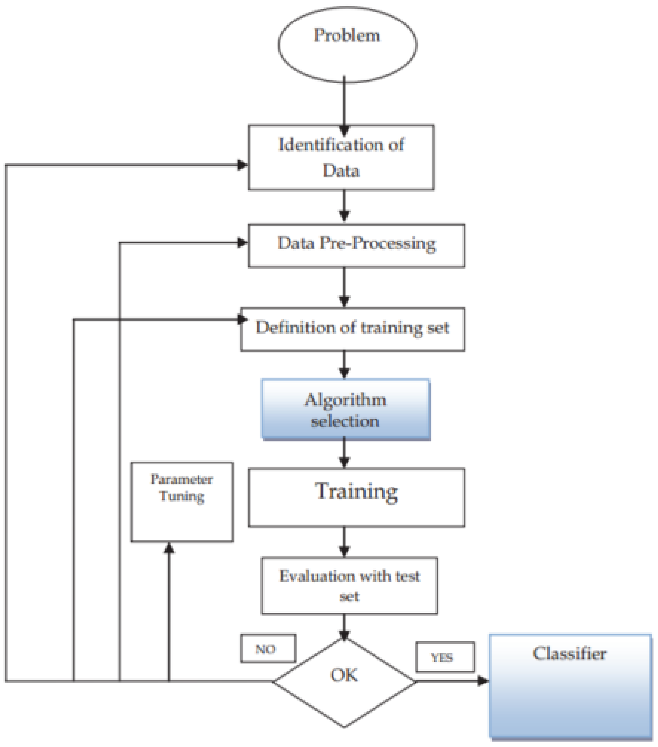


Figure 1.6: Machine Learning Supervised Process [12]

Figure 1.6 explains the logical flowchart of supervised learning. When a problem is processed through supervised process these important implementations are followed step by step. As we see the above figure problem-solving starts with the identification of the data and data goes through preprocessing, defining training sets, selecting the specific algorithm, training the model, evaluation with test set and so on as shown in the logical flowchart.

While dealing with supervised learning, an individual sample in the dataset is a combination of an input vector and an extrinsic output value or vector, that can be predicted. An implicit function is produced by evaluating the training set supporting supervised learning algorithm. The implicit function in the training model can be adapted to map or predict new samples. Both classification and regression are conventional supervised learning programs where there is an input vector X, and external output Y, and the task T is to learn the experience E from the input X to the output Y. Some Typical supervised learning algorithm types can be classified as follows: [2]

- Linear Regression

  Ordinary Linear Regression.

  Partial Least Square Regression.

  Penalized Regression.

- Nonlinear Regression:

  Multivariate Adaptive Regression Splines.

  Support Vector Machine.

  Artificial Neural Networks

  K-Nearest Neighbors

- Regression Trees:

  Bagging Tree.

  Random Forest.

  Boosted Tree.

1.6.3 Unsupervised Learning

Unsupervised learning appears much complex. The aim is to manufacture an application for the computer to learn how to do something on its own. There are two ways of unsupervised learning. The first way is to coach the application not by giving unambiguous classifications but by implementing some kind of remuneration system to display success. Particularly this kind of training will mainly fit into the decision problem framework because the aim is not to generate a classification but to make decisions that maximize remuneration. This way nicely concludes to the real world, where applications might be rewarded for doing certain activity and penalty for doing others. Always, a shape of support learning can be utilized for unsupervised learning, where the application conducts the implementations on the past remuneration and penalties without explicitly even learning any facts about the definite approaches that its activities influence the world. In the implementation process, all this knowledge is avoidable because by learning a remuneration function, the application simply understands how to work without any processing because it knows the precise remuneration it awaits to accomplish for each activity it could perform. This can be exceptionally constructive in the case where measuring every circumstance is very time absorbing. Also, it can be very time absorbing to learn by, essentially trail or error. But this kind of learning may become stronger because it concludes no pre-detected classification examples, in some cases, for example, the classification might not be best possible [12].

The second kind of unsupervised learning is known as clustering. In this kind of learning, the aim is not to inflate a utility function, but simply to discover likeness in the training data. The belief is generally that the cluster detected will counterpart good with a perceptive classification. For example, clustering distinctive based on demographics might end up in clustering of the rich in a single group and poor in another. Although the algorithm doesn't have brands to accredit new examples into one or other of the clusters, it can assemble them and then utilize those clusters to select new examples into one or the other of the clusters. This is a data-driven process that can perform well when there is enough data, for example, social information filtering algorithms [12]. So typical ways of the supervised learning are as follows:

- Clustering.
- Latent Variable Models:

  Expectation-Maximization algorithm.

  Methods of Moments.

  Artificial Neural Networks

  Blind Signal Separation techniques (e.g. Principal Component Analysis, Independent Components Analysis, Non-negative Matrix Factorization, Singular Value Decomposition)

Chapter 2

Literature Review

Regression analysis is considered as supervised machine learning algorithms for generating the regression model and assessing its accomplishment for a constant response depending on the bonding among different variables. It mainly involves linear regression, nonlinear regression, and regression trees. The theoretical ideas of these three kinds of regression are demonstrated and some of their classical algorithms will be reviewed here [2]. Every regression technique has some belief added to it which we must fulfill before starting an analysis. These algorithms may alter in terms of the type of dependent and independent variables and distribution [13]. It always comes to question what is Regression Analysis? Well in this section of the chapter where some of the Regression Algorithms will be described elaborately.

Regression analysis is a scheme of predictive modeling approach which explores the relationship between a dependent (target) and independent variable (s) (predictor). This approach is practiced for forecasting, time series modeling and discovering the causal effect relationship between the variables. Regression analysis is a decisive tool for modeling and analyzing data [14]. On average, analytic professionals know only 2-3 types of regression which are commonly practiced in the real world. But the fact is there are more than ten types of regression algorithms developed for different types of analysis. Each type has its own importance [15]. Every analyst must have knowledge which type of regression to process depending on nature of data and distribution. These various kinds of regression approaches are mostly driven by three metrics (number of independent variables, type of dependent variables and shape of the regression line) [14].

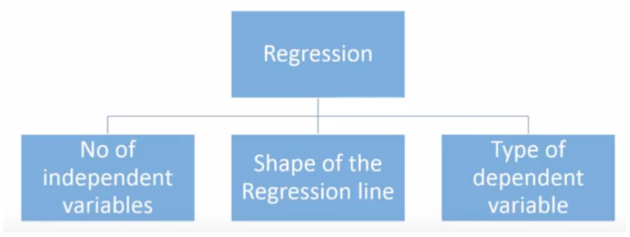

Regression
No of independent variables
Shape of the Regression line
Type of dependent variable

Figure 2.1: Machine Learning Supervised Process [14]

Figure 2.1 shows in supervised regression process the main factors that are followed throughout the process. Independent variables must be ignored, always consider the shape of the regression line and always consider the type of dependent variable.

The different types of Regressions are given below: [14] 1. Linear Regression. 2. Polynomial Regression. 3. Logistic Regression. 4. Quantile Regression. 5. Ridge Regression. 6. Lasso Regression. 7. Elastic-Net Regression. 8. Principal Component Regression. 9. Partial Least Square Regression. 10. Support Vector Regression. 11. Ordinal Regression. 12. Poisson Regression. 13. Negative Binomial Regression. 14. Quasi-Poisson Regression. 15. Cox Regression.

2.1 Choosing the correct Regression Model:

While building a model choosing correct Regression Model is very significant. Analysis of data helps to choose the correct model. So there are some certain terminologies that are used to decide which model is the best one for prediction.

1. Outliers:

   Assume there is a measurement in the dataset which is possessing a very high or very low value as measured to the other information's in the data, i.e. it does not belong to the population, such an observation is defined as an outlier. In plain words, it is intense value. An outlier is an issue because several times it impedes the results we get [15].

2. Multicollinearity:

   When the autonomous variables are eminently correlated with one another then the variables are presumed to be multicollinear. Different types of regression techniques expect multicollinearity should not exist in the dataset. The reason behind is it causes issues in ranking variables based on its importance. Or it causes issues while electing the most significant independent variable (factor) [15].

3. Heteroscedasticity:

   When reliant variable's variability is not balanced across values of an independent variable, it is called heteroscedasticity. Example - As one's income increases, the volatility of food consumption will expand. A poorer person will allocate a rather constant amount by always consuming inexpensive food; a rich person may seldom buy inexpensive food and most of the times eat expensive meals. Those with higher incomes display a greater variability of food consumption [15].

4. Under-fitting and Over-fitting:

   When we use useless explanatory variables, it might head to over-fitting. Over-fitting means that the algorithm performs well on the training set but is unable to perform better on the test sets. It is also called as a problem of high variance [15]. When the algorithm works so badly that it is incompatible to fit even training set well then it is said to under-fit the data. It is also recognized as a problem of high bias.

2.2 Linear Regression

Linear regression, also familiar as traditional least squares (OLS) and linear least squares, is the genuine powerhouse of the regression world. Linear regression is needed to comprehend the mean adjustment in a reliant variable providing a one-unit variation and a one unit adjustment in each independent variable. So, in theories of mathematics, linear regression is a statistical model to figure out the linear relationship between a reliant variable Y and one or more independent variables X. Suppose the conclusion of any action is designated by a random variable Y, called as dependent (or study) variable, depends on k independent (or explanatory) variables designated by [16] $X1, X2, ..., Xk$. Suppose the behaviour of Y can be explained by a relationship given by

$$f(X1, X2, ..., Xk, \beta 1, \beta 2, ..., \beta k) + \epsilon \tag{2.1}$$

where f is some well-illustrated function and $\beta 1, \beta 2, ..., \beta k$ are the parameters which describe the role and improvement of $X1, X2, ..., Xk$ and demonstrate that such relationship is not accurate in nature. When $\epsilon = 0$ then the relationship is called the mathematical model otherwise the statistical model. The term "model" is broadly practiced to serve any importance in a mathematical framework [16].

A model or relationship is defined as linear if it is linear in parameters and nonlinear if it is not linear in parameters. In other states, if all the partial derivatives of Y with respect to each of the parameters $\beta 1, \beta 2, ..., \beta k$ are independent of the parameters, then the model is known as a linear model. If any of the partial products of y considering with any of the $\beta 1, \beta 2, ..., \beta k$ is not independent of the parameters, the model is called nonlinear. It is also significant to perceive that the linearity and non-linearity of the model are not described by linearity or non-linearity of explanatory variables in the model. For example:

$$\beta 1 X1^2 + \beta 2 \sqrt{X2} + \beta 3 \log X3 + \epsilon \tag{2.2}$$

is a linear model because $\delta y/\delta \beta 1$, (i= 1,2,3) are independent of the parameters βi, (i=1,2,3). On the other hand,

$$y = \beta 1^2 X1 + \beta 2 X2 + \beta 3 \log X + \epsilon \tag{2.3}$$

is a nonlinear model because $\delta y/\delta \beta 1 = 2\beta 1 X1$ depends on $\beta 1$ although $\delta y/\delta \beta 3$ are independent of any of the $\beta 1$,$\beta 2$,$\beta 3$.

The typical ambition of the linear regression models is to detect assessments of the regression coefficient vector β in order to decrease mean squared error (MSE) considering the Variance-Bias trade-off. Primarily, the convenient benefit this model carries is that it acquires immense interpret-ability of the regression coefficients, can be distinctly clarified in this kind of model. The next part is that considering absolute expectations regarding model residuals distributions are fulfilled, we can precisely form use of the remaining statistical nature inside to get the standard errors of the regression parameters, and evaluate the performance of the predictive model [2].

However, as a result of immense interpret-ability, it is necessary that connection in the middle of each assessment of the parameter and the final feedback should take place along a flat hyper-lane. For instance, if there is only a single variable in the model, the connection between the variable and the feedback must be linear in a straight line. Thus, the nonlinear relationship among the regression coefficient and the predicted response cannot be clarified in this model [2].

2.3 Support Vector Machine-Regression

In the process of analyzing Machine learning algorithms, support vector machines (SVMs) along with support vector networks are recognized as supervised learning models which possess affiliated learning algorithms that evaluate data practiced for classification and regression analysis. If a set of training illustrations are provided, every single one of them is marked as belonging to one or the other of two categories, an SVM training algorithm produces a model which authorizes new instances to one category or the other, producing a non-probabilistic binary linear classifier. An SVM model is a portrayal of the illustrations as points in space, designed so that the illustrations of the different categories are distributed by a clear gap that is as wide as possible [17].

Support Vector Machines are regarded as the selected class of algorithms, classified by operation of kernels, deficiency of local minimal, thinness of the solution and adequacy control achieved by performing on the margin, or on the number of support vectors etc. They were created by Valdimir Vapnik and his co-workers, and first suggested at the Computational Learning Theory (COLT) 1992 conference with the paper. All these lucrative features nevertheless were already introduced in machine learning since 1960's. Anyway, it was not until 1993 that all of the important features were combined together to generate the maximal margin classifier, the basic Support Vector Machine, not until 1995 that the soft margin version was brought in.

Support Vector Machine can be utilized for classification issues and also in the problems of regression. Still, it possesses all the important features that identify maximum margin algorithm: a non-linear function rely on linear learning machine outlining into immense dimensional kernel analyzing feature space. The capacity of the system is maintained by parameters that do not rely on the amplitude of feature space [18].

Likewise, with classification implementations, there is a catalyst to explore and modify the abstraction bounds provided for regression. They are depended on explaining the loss function that avoids errors, which are established within the specified interval of the true value. This class of function is generally known as epsilon intensive or loss function.

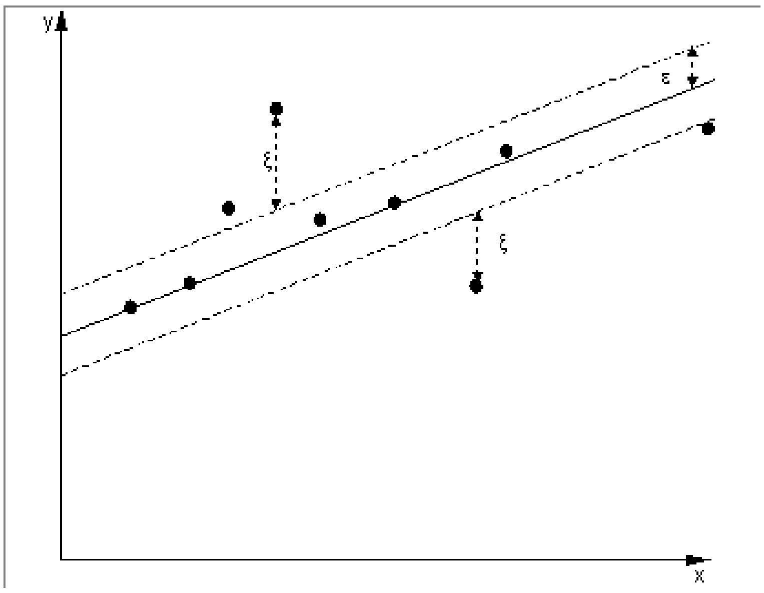


Figure 2.2: One-dimensional linear regression with epsilon intensive band. [18]

The figure 2.2 explains an example of one-dimensional linear regression function together with epsilon intensive band. The variables calculate the cost if there are errors in the training points. These are zero for all points that are inside the band.

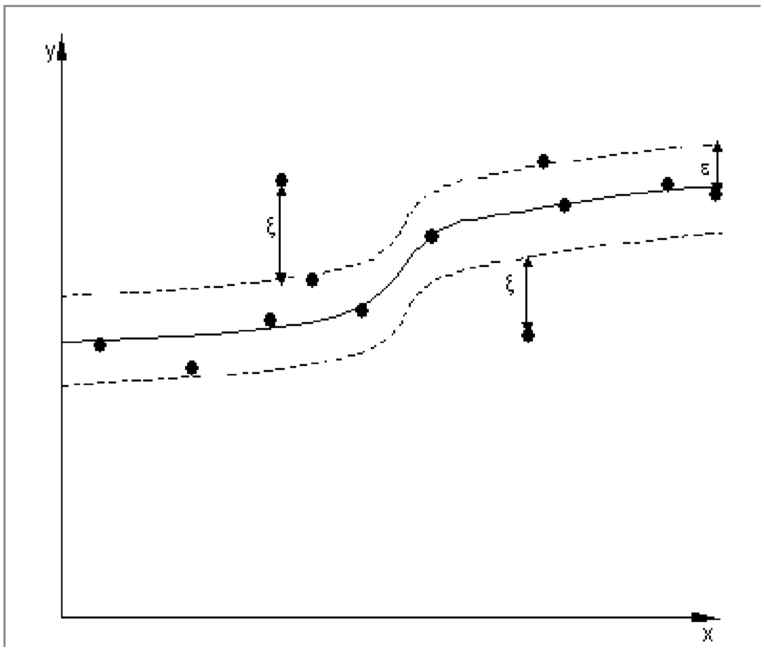


Figure 2.3: Non-linear regression function. [18]

The figure 2.3 explains similar example but for one-dimensional no-linear regression function together with epsilon intensive band.

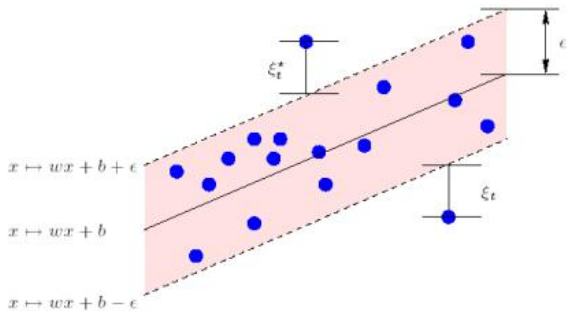


Figure 2.4: Detailed picture of epsilon band with slack variables and selected data points [18]

Figure 2.4 explains most significant concepts in Support Vector Classification and Regression cases is that introducing the explanation by the aid of limited subset of training points provides humongous computational advantages. Using the epsilon intensive loss function we make possible the continuation of the global minimum and simultaneously improving reliable generalization bound [18].

In SVM regression, the input x is first mapped onto a m-dimensional feature space using some fixed (nonlinear) mapping, and then a linear model is constructed in this feature space. Using mathematical notation, the linear model (int the feature space) $f(x, \omega)$ is given by [18]

$$f(x, \omega) = \sum_{j=1}^{m} \omega j g j(x) + b \tag{2.4}$$

where $gj(x), j = 1, ..., m$ denotes a set of nonlinear transformations, and b is the "bias" term. Often the data are assumed to be zero mean (this can be achieved by pre-processing), so the bias term is dropped.

2.4 K Nearest Neighbors-Regression

K-nearest neighbor regression is developed based on the UNN approach. The main theory behind this regression is to predict output values $y \in \Re^d$ to provided input values $x \in \Re^q$ based on sets of N input-output examples $(x_1, y_1), ..., (x_n, y_n)$. The idea here is to educate a function $f : x- > y$ which is recognized as regression function. If the data set having observed pairs $x_i, y_i \in X \times Y$ is given. KNN regression calculates the mean of the function values of its K-nearest neighbors [19].

$$\mathbf{f}_{knn}(\mathbf{x}') = \frac{1}{K} \sum_{i \in \mathcal{N}_K(\mathbf{x}')} \mathbf{y}_i$$

Figure 2.5: KNN Regression

With collection $N_k x$ possessing the indices of the K-nearest neighbors of x. The logic of KNN depends on the guessing of locality in data slot: In local neighborhoods of x patterns are expected to have similar output values y (or class labels) to f (x). Apparently, for an unknown x, the label must be identical to the labels of the nearest patterns, which is designed by the average of the result value of the K nearest samples. KNN has been proven well in various applications [19].

With KNN regression the main process is to compute the average of the numerical K nearest neighbors. Another way of calculation utilizes an inverse distance weighted average of the k nearest neighbors. KNN regression benefits the same distance function as KNN classification [20].

$$Euclidean = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2} \tag{2.5}$$

$$Manhattan = \sqrt{\sum_{i=1}^{k} |x_i - y_i|} \tag{2.6}$$

$$Minkowski = \sqrt{\sum_{i=1}^{k} (|x_i - y_i|)^q)^{1/q}} \tag{2.7}$$

These three equation given above are three distance calculations but only applicable if the variables are continuous. But if the problem is the criteria of categorical variables than Hamming distance is very important [20].

$$D_H = \sum_{i=1}^{k} |x_i - y_i| \tag{2.8}$$

where, $x = y => D = 0$ and $x! = y => D = 1$

2.5 Random Forest Regression

Usually, a random forest is a predictor combined accumulation of randomized base regression trees $r_n(X, \theta_m, D_n), m >= 1$, where θ_1, θ_2,.... are results of a randomizing variable θ. These random tress are added together to generate the aggregated regression estimate [21]

$$\overline{r_n}(X, D_n) = E_\theta[r_n(X, \theta, D_n)] \tag{2.9}$$

where E_θ demonstrates expectation considering the random parameter, conditionally on X and the data set D_n. In the following, to modify notation a little, the dependency of the estimates in the sample is removed, and for instance $\overline{r_n}(X)$ instead of $\overline{r_n}(X, D_n)$. It is important to mention that in reality, the above calculation is tested using Monte Carlo, that is, by producing M (usually large) random trees, and calculating the average of the separate outputs. The randomizing variable θ is utilized to figure out how the successive cuts are operated when constructing the individual trees, such as choosing the coordinate to split and position of the split [21].

In the model θ is regarded as free from X and the training sample D_n. This eliminates in generally any bootstrapping or re-sampling step in the training set. This also excludes any data reliant approach to generate the trees. For example exploring for excellent splits by modifying some principle on the existing observations [21].

Chapter 3

Solution Approach

3.1 Introduction

To start with the solution the focus was first to design a model to perform machine learning with any sort of industrial or plant data. The model is visualized on some plant data gathered from the customers of Siemens. Since different models demand different data preparation to perform a sophisticated machine learning, testing and analyzing those models with the data is a very important and necessary part which fulfills different requirements to the predictors in the process. Different data preparation can give rise to different predictive performance. After that visualization of feature selection is another significant part. Important feature selection removing unnecessary data can increase the performance. The cross-validated re-sampling technique can be often-used to evaluate the model in a generalized way, where a training set is used to fit a model and the testing set is used to estimate the efficiency. Several suitable machine learning algorithms are applied to build the model which includes validation, graphs, and statistical analysis to support the model.

3.2 Data Visualization:

The objective of this part is to visualize and analyze the data by plotting some graphs and performing some statistical calculations. When analyzing information from a quantitative study, there is always a possibility to deal with numbers and for that reason, it is important to begin with an understanding of the source of the numbers. The data that is provided to work is gathered from the process of a machine which is used for feed production where pressure plays an important role in maintaining the quality of the product. So, there are some different components which influence the pressure throughout the process. To determine how pressure reacts some pair plots are generated. There are some mathematical calculations done on the data to modify the features of the data.

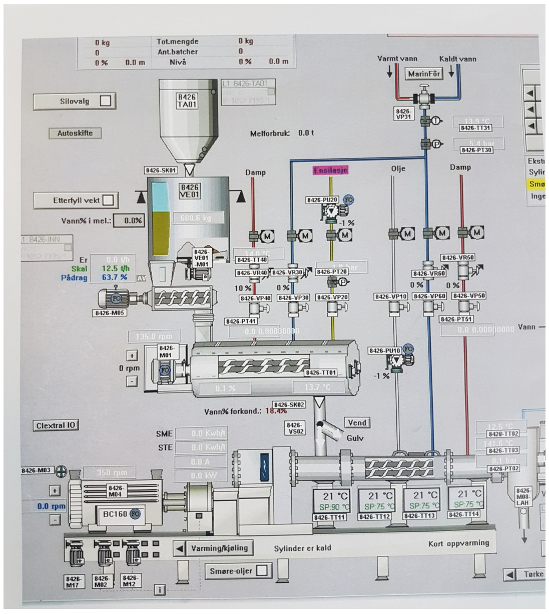


Figure 3.1: Feed Production Machine(1)(Siemens)

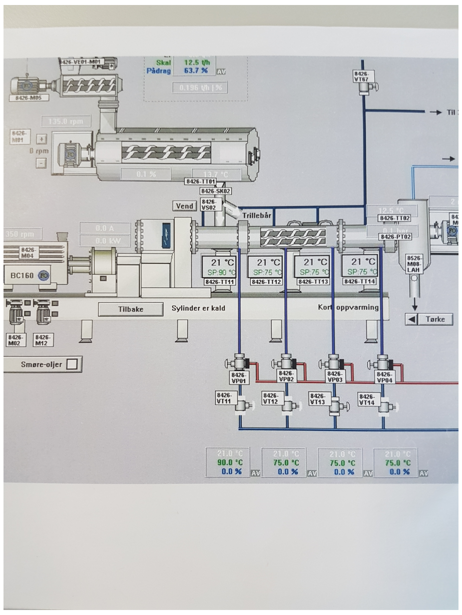


Figure 3.2: Feed Production Machine(2)(Siemens)

Figure 3.1 and 3.2 are some confidential pictures of Siemens feed production machines where each of the components are labeled. Each component represent different parameters which will be elaborated later.

The process of the system consists several components which represents several id's. The definition of each 'id' in the process is given below:

1. `E_8426_PT02` is the most important part which represents pressure.
2. `E_8426_TT02` and `E_8426_TT03` represents the end temperature.
3. `E_8426_M04_Freq_Speed` and `E_8426_M04_Current` represents engine electrical components.
4. `E_8426_TT1-TT14` represents temperature in each different zone that influences pressure.
5. `E_8426_LIW` represents machine floor materials.
6. `E_8426_VR50` and `E_8426_VR60` both represents damp on the process.
7. `E_8426_PU10` represents oil.

These are the components which influence the final output of the process. But the quality of the final product depends on the pattern of pressure value. If there is something wrong with pressure the quality might decrease. The other components of the process basically influence the pressure to be higher, lower or be in a level that is acceptable. These pair plots scatter graphs give us some good visualization of how each process data looks against the pressure. The graphs are given below:

```
sns.pairplot(df,x_vars=['E_8426_TT02','E_8426_TT03',
'E_8426_M04_Frq_Speed','E_8426_M04_Current,
y_vars='E_8426_PT02', size=7, aspect=0.7)
```

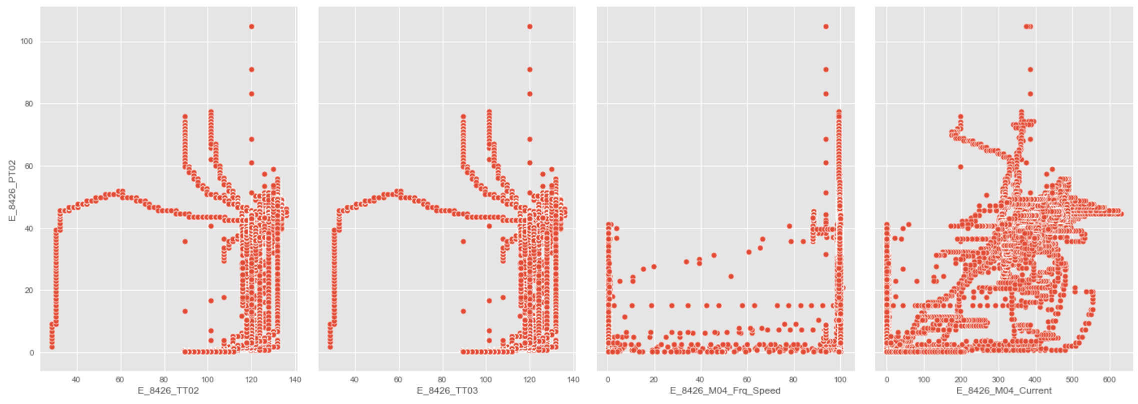


Figure 3.3: Pair Plots to show different data measure comparing pressure

In Figure 3.3 the end temperature data `E_8426_TT02` and `E_8426_TT03` looks very similar. but `E_8426_M04_Frq_Speed` and `E_8426_M04_Current` they are behaving differently than each other. `E_8426_M04_Current` has more influence on pressure as the graph looks denser.

```
sns.pairplot(df, x_vars=['E_8426_TT11',  'E_8426_TT12','E_8426_TT13',
'E_8426_TT14'],
y_vars='E_8426_PT02', size=7, aspect=0.7)
```

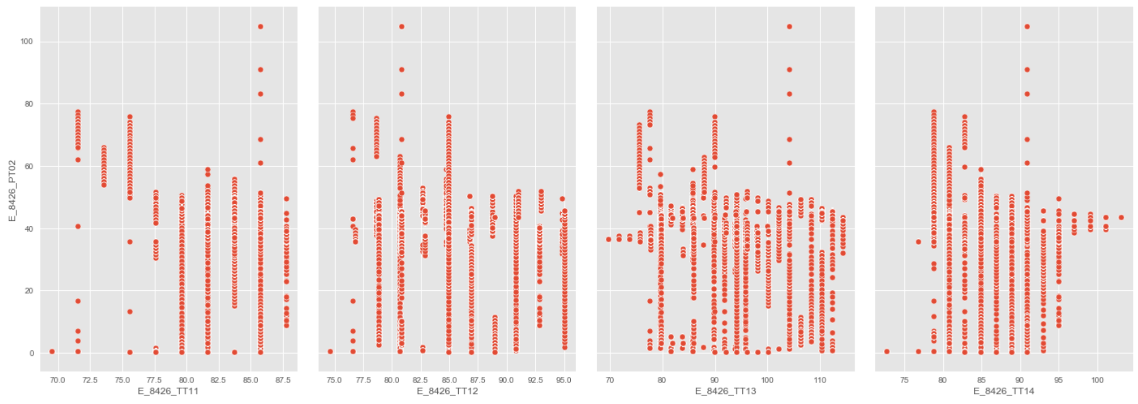


Figure 3.4: Pair Plots to show different data measure comparing pressure

Figure 3.4 represents temperature for different zones in the process are much closer to each other but `E_8426_TT13` and `E_8426_TT14` has more density compared to `E_8426_TT11` and `E_8426_TT12`. But the pattern of the graph looks more similar.

```
sns.pairplot(df, x_vars=['E_8426_LIW','E_8426_VR50','E_8426_VR60',
'E_8426_PU10'],
y_vars='E_8426_PT02', size=7, aspect=0.7)
```

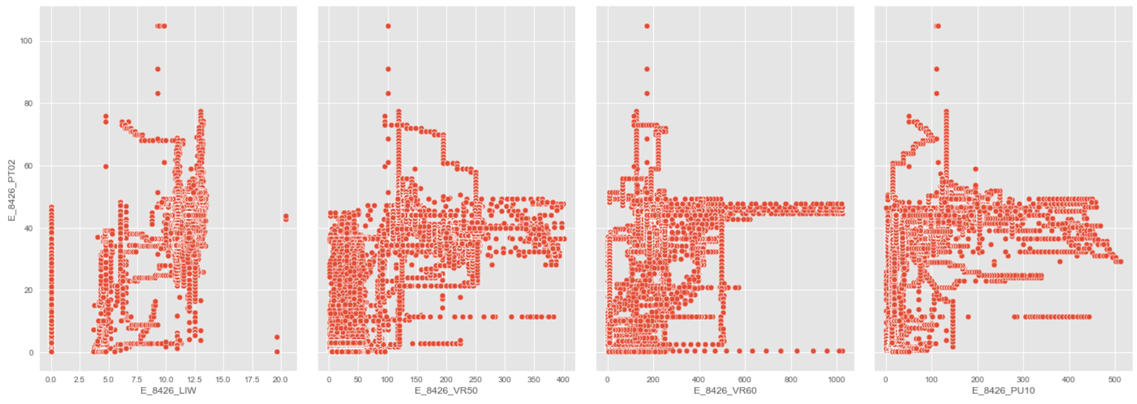


Figure 3.5: Pair Plots to show different data measure comparing pressure

In figure 3.5 the comparisons are for floor materials, damp, and oil which are quite different than each other for influencing pressure. So, they all can be an important feature for the prediction. Even `E_8426_VR50` and `E_8426_VR60` has some dissimilarities though they are damp data.

3.2.1 Merging or Deleting Variables

In the process of any implementation of phase-wise regression models, merging and deleting variables can be preserved until the designated stopping benchmark is met. In the backward phase-wise model, the model can be designed considering all the variables in the whole data-set, and after that eliminate them one by one until the performance of the model is improved. So, in the forward phase-wise model, the variables can be enumerated to the model one at a time, this processing is possible to halt when adding variables would not increase the fitness of the model at all [2].

There are various benefits to delete variables prior to modeling. First, abolishing unnecessary variables is one of the important tasks for facing with multicollinearity, which would make it impossible to figure out the individual coefficients and cause huge confidence interval for the parameters in the regression model. Second, deleting variables with deteriorated distributions helps to boost the resistance of the system enormously. Third, lesser variables mean lesser necessary resources, which influences storage space and computational time [2].

In this project, the data contained an unnecessary field for id. If the model is designed keeping the id as an index the performance of the model decreases enormously. The predictive value of an id field will differ considerably from data-set to data-set. So in various cases, it's probably fine to keep it but in others, it may cause trouble. In this project where it could have high predictive value (should be removed) as here the model is trying to predict pressure. It can also cause enormous over-fitting and some case under-fitting. So, after removing id DateTime has been kept as the index of the model.

```
pd.to_datetime(df['DateTime'])
df = df.set_index(df['DateTime'])
```

3.3 Feature Selection

In the process of building a machine learning model feature selection is a very significant and crucial part. It is also denoted as variable selection, attribute selection or variable subset selection which means it's a procedure to select a subset of relevant features (variables, predictors) to generate model construction. There are four ways of selection techniques [22]

1. simplification of models to make them easier to understand for researchers or users.
2. minimize training times.
3. to stay away from the curse of dimensionality.
4. increase generalization by reducing over-fitting.

There is a possibility while implementing feature selection techniques that the data may contain a lot of features which are unnecessary or irrelevant. This kind of features causes bigger issues while developing the model which can be eliminated without doing much loss for information. Unnecessary or irrelevant features are two specific notions, since one relevant feature may be redundant in the existence of another relevant feature. Feature selection techniques can be also named as feature extraction. Feature extraction generates new features from functions of the original features, whereas feature selection outputs a subset of the features [22]. There are different methods like filter methods, wrapper methods and embedded methods etc. to perform feature selection. In this implementation embedded methods are followed as much as possible.

3.3.1 Building Features

So, all the discussion above and analysis of data leads to select features for the model. As discussed above there are several components which influence pressure. As pressure control the quality of the end product so from the analysis it's obvious that prediction of pressure is the main goal to predict maintenance in future. All these implementations are performed in python using Pandas and Sckiit learn library for prediction, validation, and analysis.

At first, the id column has been removed and the date-time values in the data are set as an index of the data. All the columns in the data which contains values for different components of the machine are set in different data frames. To build these data frames pandas data frame library is used.

```
df = df.set_index(df['DateTime'])
df =df[['E_8426_PT02','E_8426_TT02',
  'E_8426_TT03','E_8426_M04_Frq_Speed',
  'E_8426_M04_Current','E_8426_TT11',
  'E_8426_TT12','E_8426_TT13','E_8426_TT14',
  'E_8426_LIW','E_8426_VR50',
  'E_8426_VR60','E_8426_PU10']]
```

3.3.2 Feature Modification

The next portion in feature selection is making some new features with the available labels. The objective of feature modification is to make similar data feature in a single acceptable feature as well as increasing the learning and prediction time of the model on this big (almost 68 thousand) data-set. The new features are designed by taking a deeper look at the current data.

1. First, `E_8426_TT02` and `E_8426_TT03` are close. So, an average data frame value is generated with both data and used as a new feature `Avg_TT0`.

```
        df['Avg_TT0'] = (df['E_8426_TT02']+ df['E_8426_TT03']) / 2
```

2. Similar way as the graphs comparing with pressure value looks similar for `E_8426_TT1`, `E_8426_TT2`, `E_8426_TT3`, `E_8426_TT4` they are averaged to one new feature for the temperatures called `Avg_TT_Series`

```
df['Avg_TT_Series'] = (df['E_8426_TT11']+ df['E_8426_TT12']
+df['E_8426_TT13']+df['E_8426_TT13'])/4
```

3. The values of `E_8426_M04_Current` and `E_8426_M04_Frq_Speed` shows a similar pattern but one represents the highest peak of engine electrical components and another represents the lowest peak. So, a new feature is introduced taking the percentile for both of the data and names `PCT_M04`

```
df['PCT_M04'] = (df['E_8426_M04_Current']
- df['E_8426_M04_Frq_Speed']) / df['E_8426_M04_Frq_Speed'] * 100.0
```

Other features of the data are kept same. New modified features are introduced using all the changes in a new data-frame including 8 features along with the pressure data `E_8426_PT02`

3.3.3 StatsModel Evaluation

StatsModel is an efficient Python module that contributes classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests and statistical data exploration. An extensive list of result statistics is available for each estimator [23].

The main goal here is before building the model estimation of the R squared value with StatsModel to visualize how the new features are reacting without splitting them into training and testing sets and how much accuracy it achieves. To get that first a fitted model is generated with all the features using multiple linear regression processes.

$$y = \beta 0 + \beta 1x1 + ... + \beta nxn \tag{3.1}$$

```
lm1 = smf.ols(formula='E_8426_PT02 ~ Avg_TT0 + Avg_TT_Series + PCT_M04
+ E_8426_LIW + E_8426_VR50 +
E_8426_VR60 + E_8426_PU10',
      data=df).fit()
```

As we can see the results without splitting data into training and testing sets the normal model is achieving 86% accuracy. Later it will be improved more.

```
print('R-squared value for the model:',lm1.rsquared)
Result: R-squared value for the model: 0.865464375161
```

Issues with StatsModel R-squared is, it will always increase as we add more features to the model, even if they are unrelated to the response. Selecting the model with the highest R-squared is not a reliable approach for choosing the best linear model. Solution adjusted R-squared penalizes model complexity (to control for overfitting), but it generally under-penalizes complexity [24].

Better Solution is Train/Test split or `model_selection` which are a more reliable estimate of reducing sample error and better for choosing which of your models will best generalize results out-of-sample data. There is extensive functionality for cross-validation in scikit-learn, including automated methods for searching different sets of parameters and different models Importantly, cross-validation can be applied to any model, whereas the methods described above only apply to linear models.

3.3.4 Processing phase:

In this phase of implementation the processing of the data and designing the model according to that has been initialized. Predictive modeling is an approach that utilizes mathematical and computational procedures to predict an occurrence or conclusion. A mathematical way utilizes an equation based model that demonstrates the circumstance under deliberation. The model is utilized to forecast an conclusion at some future state or time based upon adjustments to the model inputs. The model features benefits to demonstrate how model inputs control the outcome. For example time-series regression model for predicting airline traffic volume and predicting fuel efficiency based on a linear regression model of engine speed versus load [25].

The computational predictive modeling process contradicts from the mathematical way. The reason behind it is, it depends on models that are not simple to demonstrate in an equation pattern and it seldom requires simulation processes to generate a prediction. This type of process is generally recognized as "black box" predictive modeling due to the model architecture does not support observation to the factors that map model input to the conclusion [25].

Predictive modeling is generally developed processing curve and surface fitting, time series regression, or machine learning implementations. Regardless of the process performed, the implementations of generating a predictive model is the similar across the methods. The steps are:

1. Trim the data by eliminating exceptions and processing missing data.
2. Pinpointing a parametric on non-parametric predictive modeling process to use.
3. Pre-processing the data into a shape which is acceptable for the selected modeling algorithm.
4. Identify a subgroup of the data to be utilized for training the model.
5. Train, or estimate, model features from the training dataset.
6. Testing model performance or goodness-of-fit tests to evaluate model adequacy.
7. Validate predictive modeling accuracy on data not used for aligning the model.
8. Adopt the model for prediction if contented with its performance [25].

In this experiment, the processing starts with implementing several steps in the model. The steps are given below:

1. At first, a variable which symbolizes the forecast column which is defined as E_8426_PT02 the pressure of the process.

```
forcast_col = 'E_8426_PT02'
```

2. As trimming the unnecessary data is an important, the unused data has been dropped.

```
df.fillna(-99999, inplace=True)
```

3. In the next step a math ceiling implementation which returns a decimal point if the length of the data frame is a number. Here the length is 0.2 which means math ceiling will round that up to 1 making integer which will try to predict 20 percent of the data-frame number of days out.

```
    forcast_out = int(math.ceil(0.2*len(df)))
```

4. This phase includes pre-processing of data-frame. The variable 'x' contains all the other columns with values dropping the prediction column `E_8426_PT02` and variable 'y' contains the prediction column `E_8426_PT02`. One variable `'x_lately'` holds the values to forecast 20 percent data which will be used to predict future predictions.

```
    x = np.array(df.drop(['E_8426_PT02'],1))
    x_lately= x[-forcast_out:]
    y = np.array(df['E_8426_PT02'])
```

5. To pre-process 'x' values scikit-learn's library pre-processing is used. This package helps with several common useful functions and transformer classes to transform raw feature vectors into a representation that is more acceptable for downstream estimators. While learning algorithms it benefits to the standardization of the data set. So if any outliers are available in the data-set robust, scalers or transformers are more exact. As the amount of data is huge in 'x', the data is also sliced to make the learning faster while doing prediction or any validation.

```
        x = preprocessing.scale(x)
```

3.4 Analysis to Select Model Algorithm

In this phase, we test the data for various algorithms to select a perfect algorithm for the data and which provides us higher accuracy than normal Stats model accuracy. To do so some necessary validation parts have been run to find the perfect model. So, the x and y values are fitted through different algorithms such as LinearRegression, K neighborsRegressor, SVR, RandomForestRegressor to predict values before splitting the data in training and testing sets.

Scatter graph to compare true pressures and the predicted pressures is implemented to analyze the data furthermore and see how each algorithm is reacting with the prediction before splitting into training and testing sets. For all the graphs Yi is pressure in data and $\hat{Y}$ is the predicted pressure for a respective algorithm. Different algorithms will definitely show different patterns. The Scatter graphs are given below with some code snippets:

```
lm = LinearRegression()
lm.fit(x,y)
plt.scatter(y, lm.predict(x))
```

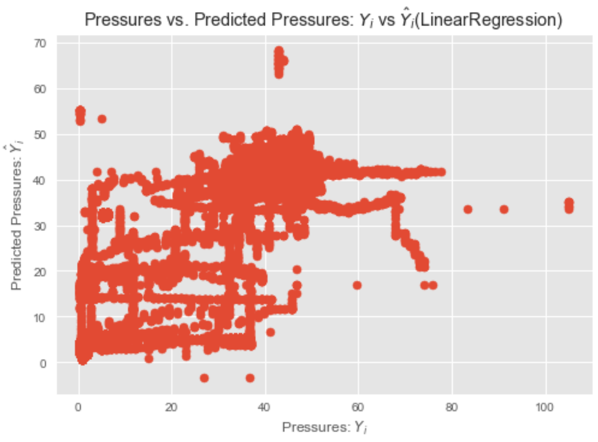


Figure 3.6: Pressures vs. Predicted Pressures(LinearRegression)

In Figure 3.6 Yi is pressure in data and $\hat{Y}$ is the predicted pressure for LinearRegression. The data looks quite scattered in this graph.

```
vm = svm.SVR()
vm.fit(x,y)
plt.scatter(y, vm.predict(x))
```

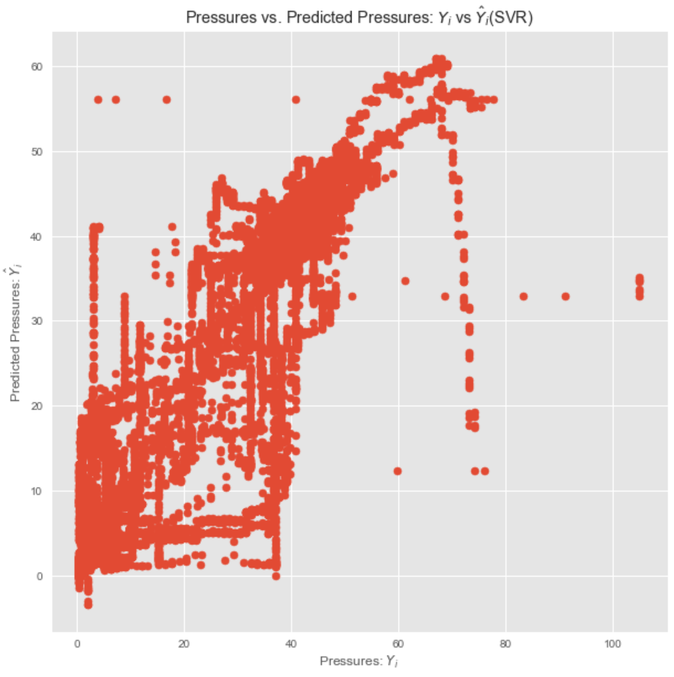


Figure 3.7: Pressures vs. Predicted Pressures(SVR)

In Figure 3.7 Yi is pressure in data and $\hat{Y}$ is the predicted pressure for SVR. The data looks scattered but better than LinearRegression.

```
kn = neighbors.KNeighborsRegressor()
kn.fit(x,y)
plt.scatter(y, kn.predict(x))
```

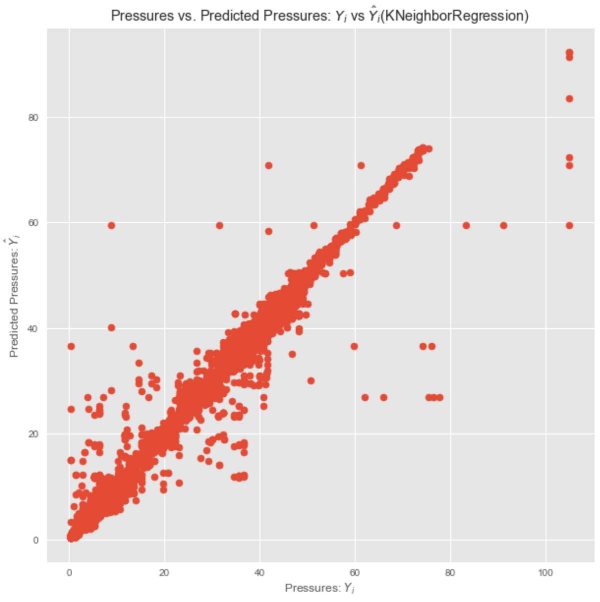


Figure 3.8: Pressures vs. Predicted Pressures(KNeighborRegression)

Figure 3.8 shows Yi is pressure in data and $\hat{Y}$ is the predicted pressure for KNeigborsRegressor. The result looks in good shape forming a sort of straight line from point zero.

```
        rfg = RandomForestRegressor()
        rfg.fit(x,y)
        plt.scatter(y, rfg.predict(x))
```

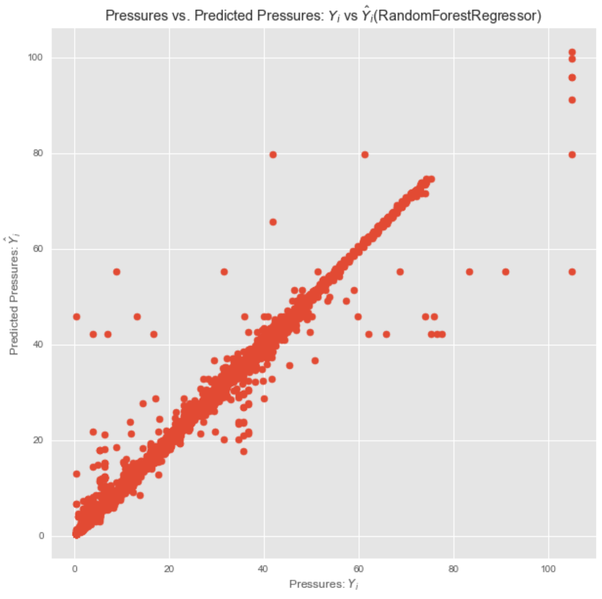


Figure 3.9: Pressures vs. Predicted Pressures(RandomForestRegressor)

Figure 3.9 shows Yi is pressure in data and $\hat{Y}$ is the predicted pressure for RandomForestRegressor. This result looks as it's also forming a sort of straight line from point zero.

This analysis shows that RandomForestRegression and KNeighborRegression shows better output than other two algorithm. Without splitting them or validating them it is quite impossible to proof that which algorithm actually fits the model. Some statistical calculations are also needed for further proofs.

3.5 Cross Validation split (KFold)

In this significant part of the implementation splitting data into train and test data is the mandatory task. There are several ways to do it but it is necessary to understand that inefficient splitting can lead to over-fitting or under-fitting in the model which is not acceptable. The most efficient process is to use cross-validation while splitting the data.

Cross-validation is recognized as a model evaluation process which performs better than the residuals. The issue while performing residual evaluations is that they never provide a hint that how precise the learner will perform when there is any implementation done to perform fresh predictions for data that has not been observed. There are several ways to fix this but one efficient way to fix this problem is to prohibit using the whole data set when the learner is being trained. Some portion of the data is already eliminated earlier when training commences. After training is finished, the portion of data that has been eliminated can be utilized to evaluate the performance or confidence of the learned model on 'new' data. This is the elemental concept for a whole class of model evaluation methods called cross-validation [26]. There are several methods of Cross-validation. Some of them are discussed here:

1. The holdout method:

   The holdout process of cross-validation is the straightforward type of cross-validation. The dataset is basically divided into two sets, called the training set and the testing set. Approximator a defined function in the process fits a function using the training set only. Later function approximator is used to predict the output results for the data in the testing set (it has never observed these output values before). The errors the process produces are acquired as before to provide the mean absolute test set error, which is needed to assess the model. The leverage of this process is that it is usually preferable to the residual method and takes no longer to compute. However, appraisal can produce a high variance. The appraisal only depend heavily on which data points end up in the training set and which end up in the test set, and thus the evaluation may be naturally distinctive relying upon how the division is done [26].

2. K-fold cross validation:

   K-fold cross-validation is an improved implementation comparing the holdout method. The data set is broken down into k subsets, and the holdout method is duplicated k times. Each time, one of the k subsets is operated as the test set and the other k-1 subsets are assembled to generate a training set. After that average error over all k trials is estimated. The benefit of this process is that it concerns less how the data gets split. Each data point obtains the ability to be in a test set exactly once and gets to be in a training set k-1 times. The variance of the final estimate is lessened. The prejudice of this process is that the training algorithm has to be repetition from scratch k times, which means it takes k times as much computation to make an evaluation. A variant of this process is to randomly separate the data into a test and training set k different times. The leverage of doing this is that in the process it can individually determine how large each test set is and how many trials should be performed [26].

3. Leave-one-out cross validation:

   Leave-one-out cross-validation is a part of K-fold cross-validation performed to its logical extreme, with K equal to N, the number of data points in the set. That demonstrates that N separate times, the defined function approximator is trained on whole data excluding one point and a prediction is performed only for that point. But before the average error is calculated or performed to calculate the model. The results given by leave-one-out cross-validation error (LOO-XVE) is fine, but at first phase, it looks it takes time to finish and costly to compute. Luckily, locally weighted learners can make LOO predictions just as simple as they perform regular predictions. That means measuring the LOO-XVE performs no more time than computing the residual error and it is a much efficient way to evaluate models [26].

Analyzing the dataset in this experiment KFold cross-validation has been used. In K-Folds Cross validation of the data is divided into k different subsets (or folds). There are k-1 subsets used to train our data and leave the last subset (or the last fold) as test data. Later in the implementation, the average of the model against each of the folds is computed and then the model is concluded. Later it's tested with the test set.

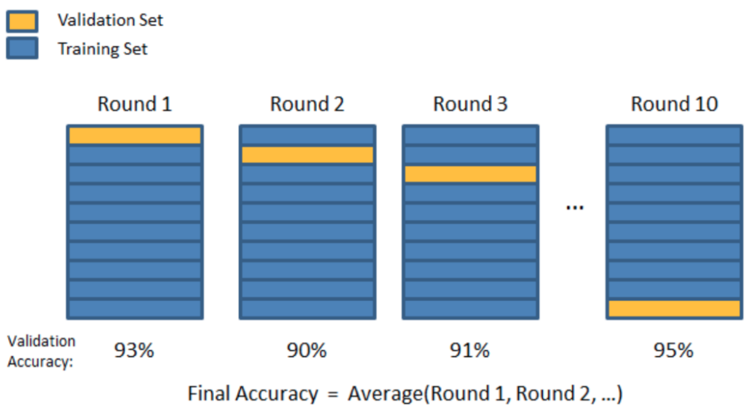


Figure 3.10: K-Fold Cross Validation [26]

Figure 3.10 is a graphical representation of how KFold cross-validation works in each round. In each round, the accuracy seems to be increasing but in our case, two folds were enough to produce expected results.

Before splitting in the experiment the 'x' is sliced without 20 percent of the data that has been used to do future prediction but the rest of the big chunk of data is used and later implemented k-fold split on that portion of data. The data has been split into 2 folds which returns the number of splitting iteration in the cross-validation

```
kf = KFold(n_splits=2) # Define the split - into 2 folds
folds = kf.get_n_splits(x)
for train_index, test_index in kf.split(x):
    print("TRAIN:", train_index, "TEST:", test_index)
    x_train, x_test = x[train_index], x[test_index]
    y_train, y_test = y[train_index], y[test_index]
```

3.6 Analysis

Different regression algorithms LinearRegression, K neighborsRegressor, SVR, and RandomForestRegressor is used to fit the data with training sets `'x_train'` and `'y_train'`. All the fitted data is pickled in various files for each algorithm which makes the process much faster. In the specific case of the scikit, it may be more interesting to use pickle which is more efficient on objects that carry large numpy arrays internally as it is often the case for fitted scikit-learn estimators [27].

The purpose of this section is to set up the model with important features and modify a different section of implementation. All these processes include data analysis and setting up everything according to it. Now on the next part of the implementation is validating the model which includes different statistical analysis, graphs, most importantly making predictions and verifying those predictions. In the next chapter, all of these important implementations will be discussed elaborately. This analysis with machine learning is the most significant part of the concept of this project.

3.7 Proposed Solution

After analysis and approaching with the solution is the primary goal. This part of the chapter includes only some figures which demonstrate how the solution of this experiment problem is proposed and will be conducted further.

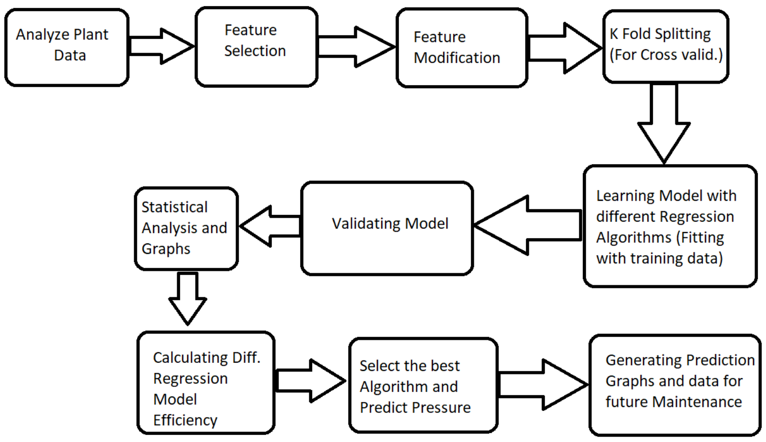

Analyze Plant Data
Feature Selection
Feature Modification
K Fold Splitting (For Cross valid.)
Learning Model with different Regression Algorithms (Fitting with training data)
Validating Model
Statistical Analysis and Graphs
Calculating Diff. Regression Model Efficiency
Select the best Algorithm and Predict Pressure
Generating Prediction Graphs and data for future Maintenance

Figure 3.11: Machine Learning Approach

Figure 3.11 already defines that first four processes are already discussed in this chapter. This is a flowchart of how the machine learning approach for this experiment is conducted. Next phase of the application will be really crucial as this model will be evaluated with proper analysis to get the best results.

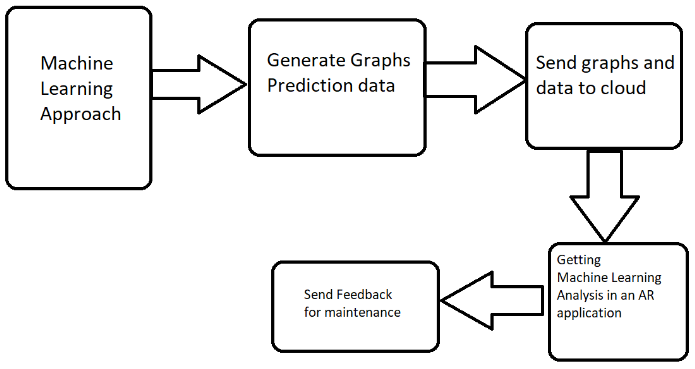

Machine Learning Approach
Generate Graphs Prediction data
Send graphs and data to cloud
Getting Machine Learning Analysis in an AR application
Send Feedback for maintenance

Figure 3.12: Machine Learning collaboration with AR

Figure 3.12 is a representation the combination of AR and Machine learning as a concept on the virtual ecosystem which is yet not implemented on a bigger scale. Proposing it leads to building collaboration which includes all the exceptional implementations that can be conducted in future works. Combining both of the concepts is the main objective here.

Chapter 4

Experimental Evaluation

4.1 Experimental Setup and Evaluation

In the process of building a model only getting the desired result is not enough. To validate the model and make it acceptable there are certain evaluations or experiments needed to be implemented which includes statistical graphs, analysis, mathematical calculations and comparison of different algorithms. In this experiment, different regression algorithms are being compared by performing some several validations. As discussed previously after splitting the data it is fitted with training sets for the regression algorithms which means the experimental setup on the data set is done. Now the model is ready to be evaluated.

4.1.1 Residual Plots and Mean squared error

1. **Residual Plots:**

   Residual plots are a good way to visualize the errors in the data. If the process is done in a good way then the data should be randomly scattered around line zero. If the structure doesn't seem like that in the data, that means the model is not capturing something. Maybe there is an interaction between two variables that have not been considered, or maybe the measurement is only time dependent. If there is some structure in the data, the model needs to be checked whether it's doing a good job with the current parameters. In this part of the experimental evaluation, residual plots are implemented on different regression algorithms. This is the first step to find the best algorithm for the model. This also paves a way which algorithm should be used to make the future prediction from the true values. Different residual graphs are given below:

```
plt.scatter(clf.predict(x_train),
clf.predict(x_train) - y_train, c='b', s=40, alpha=0.5)
plt.scatter(clf.predict(x_test),
clf.predict(x_test) - y_test, c='g', s=40)
```

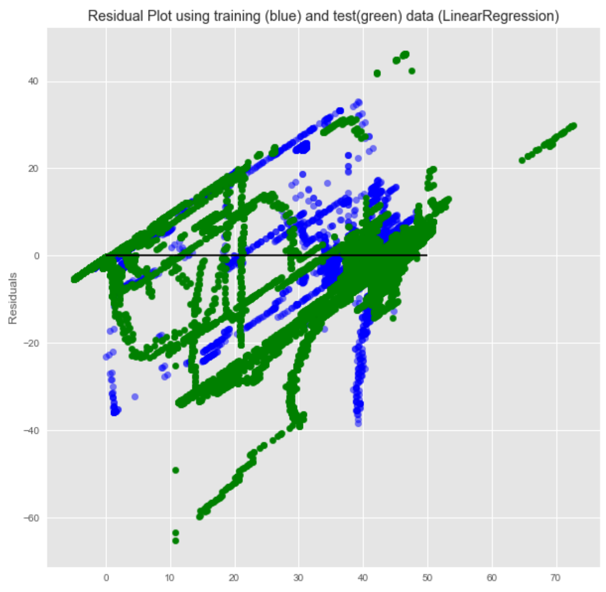


Figure 4.1: Residual Plot using training (blue) and test(green) data (LinearRegression)

Figure 4.1 is the residual plot between training and test data using LinearRegression after splitting. The blue part of the graph represents training set and green part shows testing set and scattered nicely around in pretty same proportion line zero.

```
plt.scatter(vm.predict(x_train),
vm.predict(x_train) - y_train, c='b', s=40, alpha=0.5)
plt.scatter(vm.predict(x_test),
vm.predict(x_test) - y_test, c='g', s=40)
```

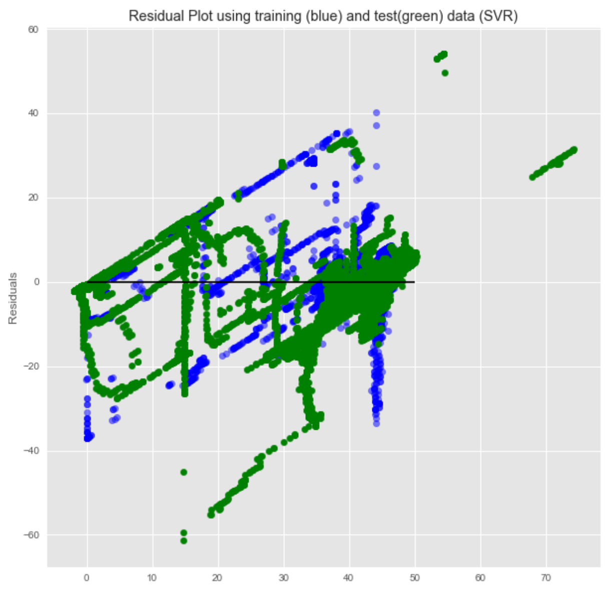


Figure 4.2: Residual Plot using training (blue) and test(green) data (SVR)

Figure 4.2 is the residual plot between training and test data selecting SVR as the model after splitting. The blue part of the graph represents training set and green part shows testing set which scattered densely enough in same proportion around the zero line.

```
plt.scatter(kng.predict(x_train),
kng.predict(x_train) - y_train, c='b', s=40, alpha=0.5)
plt.scatter(kng.predict(x_test),
kng.predict(x_test) - y_test, c='g', s=40)
```

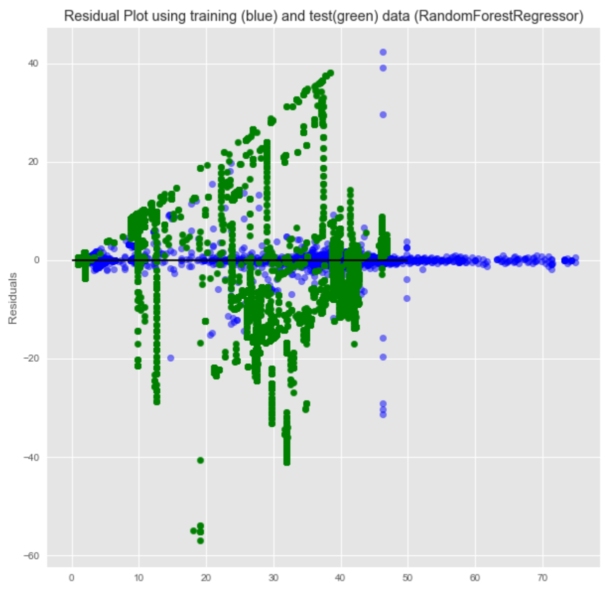


Figure 4.3: Residual Plot using training (blue) and test(green) data (KNeighborsRegressor)

Figure 4.3 is the residual plot between training and test data selecting KNeighborsRegressor as the model after splitting. The blue part of the graph represents training set and green part shows testing set which seems to be not scattered around line zero in the same proportion.

```
plt.scatter(rfg.predict(x_train),
rfg.predict(x_train) - y_train, c='b', s=40, alpha=0.5)
plt.scatter(rfg.predict(x_test),
rfg.predict(x_test) - y_test, c='g', s=40)
```

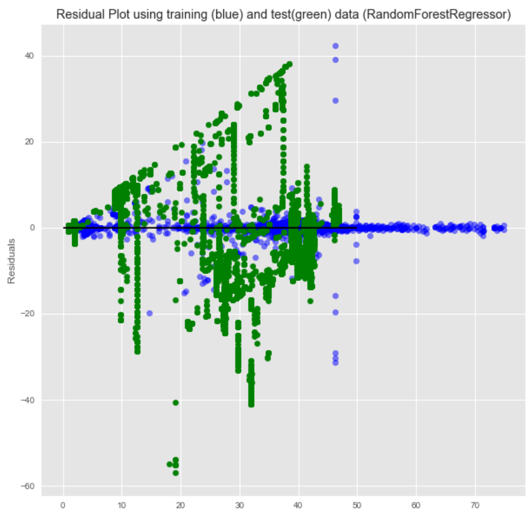


Figure 4.4: Residual Plot using training (blue) and test(green) data (RandomForestRegressor)

Figure 4.4 represents the residual plot between training and test data selecting RandomForestRegressor as the model after splitting. This shows pretty similar results like the previous graph.

In this residual plot analysis, different algorithms are showing different patterns. LinearRegression and SVR showing that more data are randomly scattered at zero in same proportion rather than K neighborsRegressor and RandomforestRegressor. Which can be a lead that any of those two algorithms will have a better result on the prediction score.

2. **Mean Squared Error(MSE):**

The mean squared error demonstrates how close a regression line is to a set of points. It implements it by taking the distances from the points to the regression line (these distances are the "errors") and squaring them. The squaring is very significant to remove any negative signs. It also provides more weight to larger differences. It's called the mean squared error and it works always finding the average of the set of errors [28]. The smaller the means squared error, the closer it is to find the line of best fit. Depending on the data, it may be impossible to get a very small value for the mean squared error [28].

if $\hat{Y}$ is a vector of n predictions, and Y is the vector of observed values of the variable being predicted, then the within-sample MSE of the predictor is calculated as,

$$MSE = 1/n \sum_{i=1}^{n} (Yi - \hat{Y}i)^2 \tag{4.1}$$

After residual plots,fitting a model with `x_train` MSE is calculated first with `y_train` and later with `x_test` and `y_test`. The list of MSE with calculations by fitting the model with `x_train` for each algorithm is given below:

Fit model with `x_train`				
	Linear	SVR	Kneighbors	Randomforest
MSE with `x_test`, `y_test`	1.72083640817	1.61179027145	5.91559859264	4.94582563098

Table 4.1: MSE with `x_test`, `y_test`

As we see in this Table 4.1 SVR is giving less MSE while calculating with `x_test` and `y_test` but other algorithms have higher MSE results.

4.1.2 True Value vs. Predicted Value and Accuracy

While validating the model many significant types of validation is required to establish it strongly along the process. As a part of validation in this section, the true values of the data will be compared with the predicted value for each regression algorithm. These graphs will show how each algorithm is reacting when it comes to the point of prediction. Next, the accuracy has been calculated for each algorithm which gives an idea of how the model is learning in the process. A good learner is the one which has good prediction accuracy, in other words, which has lesser prediction error. So the goal is here to find the best predictive model. Most of the predictive modeling implementations have several tuning parameters which empower the model to flex and discover the structure in the data. It helps to recognize settings for the model's parameters that provide the best and most practical predictive performance.

1. **True vs Predicted value graphs:**

   As these graphs are the part of validation, test data that has been generated after cross-validation is used to generate the graphs. Basically `y_test` prediction value with `x_test` value has been compared for different models.

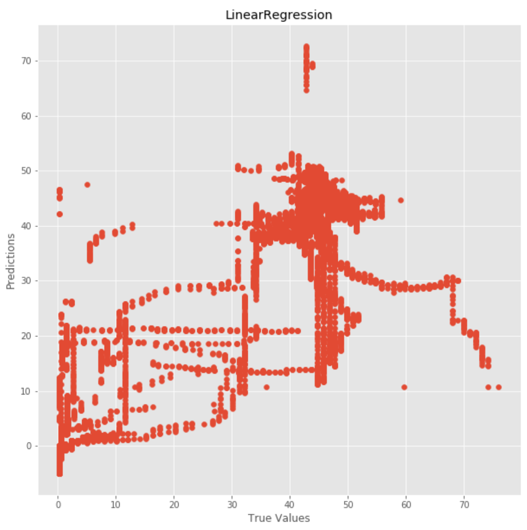


Figure 4.5: True vs. Predicted Value (LinearRegression)

Figure 4.5 is the validation graph with test values `x_test` and `y_test`. The model is being fitted with LinearRegression and the `y_test` values are predicted with the model. This graph represents true values vs. predicted values. The graph shows they are scattered but some values are very close or same.

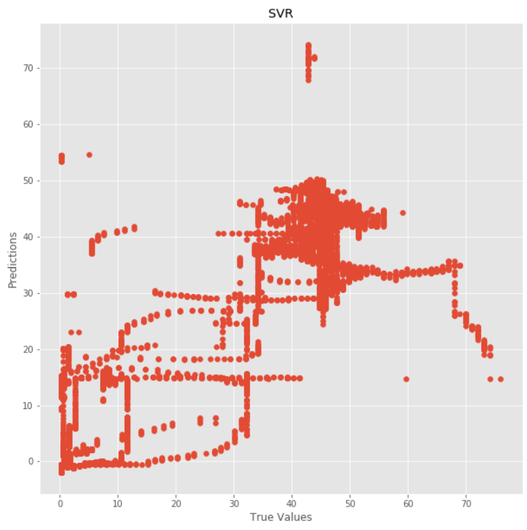


Figure 4.6: True vs. Predicted Value (SVR)

Figure 4.6 is the validation graph taking test values `x_test` and `y_test`. This time, the model is being fitted with SVR and the `y_test` values are predicted with the model. This graph represents similarly true values vs. predicted values with SVR. This graph is also scattered enough and some values are very close or same, actually better than the previous one.

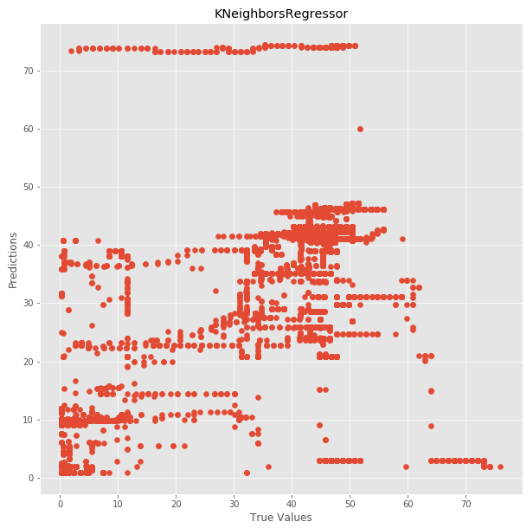


Figure 4.7: True vs. Predicted Value (KNeighborsRegressor)

Figure 4.7 is the validation graph taking test values `x_test` and `y_test`. This time, the model is being fitted with KNeighborsRegressor and the `y_test` values are predicted with the model. This graph represents similarly like other two true values vs. predicted values with KNeighborsRegressor. This graph looks much more scattered and few values are matching each other.

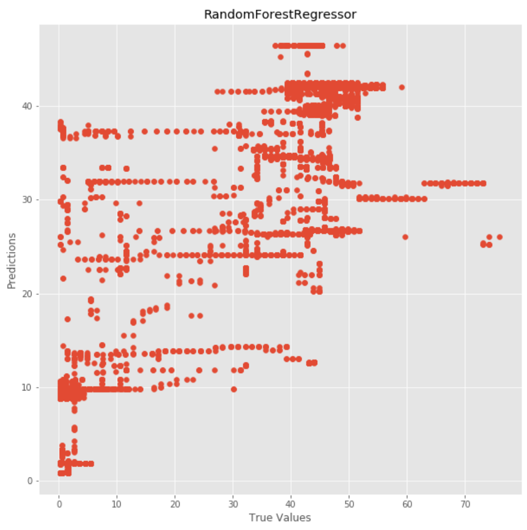


Figure 4.8: True vs. Predicted Value (RandomForestRegressor)

Figure 4.8 is the validation graph taking test values `x_test` and `y_test`. Finally, the model is being fitted with RandomForestRegressor and the `y_test` values are predicted with the model. This graph represents true values vs. predicted values for RandomForestRegressor. This graph also looks much more scattered as few values are matching but showing better results than the RandomForestRegressor graph.

The graphs above explains that with LinearRegression and SVR the results of True values vs. Prediction values looks denser and the reason will be clarified when accuracy calculation is done. But so far the results for these two algorithms looks much better than the KneighborsRegressor and RandomForestRegressor. If a straight line is drawn from the center of the graph the results from LinearRegression and SVR looks more densely populated around it than other two.

2. **Accuracy Score:**

   Accuracy calculation for all the algorithm will lead which regression algorithm will be finally used to predict pressure data for the model. The table below explains the accuracy of each model:

```
accuracy = clf.score(x_test,y_test) //accuracy for linearregression
accuracy = vm.score(x_test,y_test) //accuracy for SVR
accuracy = kng.score(x_test,y_test) //accuracy for KNeighborsRegression
accuracy = rfg.score(x_test,y_test)//accuracy for RandomForestRegression
```

Accuracy Score			
Linear	SVR	Kneighbors	Randomforest
0.836039535805	0.8746783443	0.645839093481	0.800279614849

Table 4.2: Accuracy Score For Different Models

In Table 4.2 the accuracy calculations are performed using test values (`x_test`,`y_test`) values. So far SVR shows better accuracy than other algorithms. This accuracy has been reached doing some tuning with the regression. For SVR the 'kernel' value is set to linear which lead to higher accuracy than other algorithms. Other algorithms are not tuned because even without tuning SVR showed higher accuracy than other algorithms.

4.1.3 RMSE,MAE,MedAE Calculations:

1. **Root Mean Square (RMSE):**

   Root Mean Square Error (RMSE) is the standard deviation of the residuals(prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how to spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Root mean square error is commonly used in climatology, forecasting and regression analysis to verify experimental results [29].

   Formula:

$$RMSE = \sqrt{(f-o)^2} \tag{4.2}$$

Where: f= forecasts(expected values or unknown results), o=observed values(known results) [29]

The bar above the squared differences is the mean (similar to x). The same formula can be written with the following slightly different notation: [29]

$$RMSE_{f}o = [\sum_{i=1}^{N}(Z_{f}i - Z_{o}i)^2/N]^{1/2} \tag{4.3}$$

2. **Mean Absolute Error(MAE):**

   Mean absolute error (MAE) is a measure of the difference between two continuous variables. Assume X and Y are variables of paired observations that express the same phenomenon. Examples of Y versus X include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement. Consider a scatter plot of n points, where point i has coordinates (xi, yi)... Mean Absolute Error (MAE) is the average vertical distance between each point and the Y=X line, which is also known as the One-to-One line. MAE is also the average horizontal distance between each point and the Y=X line [30].

$$MAE = \sum_{i=1}^{n}(|y_i - x_i|)/n = \sum_{i=1}^{n}|e_i|/n \tag{4.4}$$

3. **Median Absolute Error(MedAE):**

   The MedAE is similar to the MSE, but we start with the absolute values of the residuals and we use median instead of mean as the measure for centrality. The MedAE is also analogous to variance and is ideally zero or very small. Taking the absolute value instead of squaring potentially avoids numerical instability and speed issues and median is more robust for outliers than the mean. Also taking the square rends to emphasize large errors [31].

$$MedAE(y, \hat{y}) = median(|y_1 - \hat{y_1}|, ..., |y_n - \hat{y_n}|) \tag{4.5}$$

All these calculations have been done to see which model has a lesser error and the accuracy that has been observed is feasible or not. All these error calculations are done using Sckit learn's Metrics library. These are some statistical calculation which helps to find errors in the data that has been used in the process. Results for these parameters for different algorithms are given below:

RMSE, MAE, MedAE Calculation			
Type	RMSE	MAE	MedAE
Linear Regression	7.4798213597	4.82	3.73
SVR	6.53935204454	4.27	3.37
KNeighbors Regressor	10.9931452553	6.62	3.36
RandomForest Regressor	8.25529987317	5.68	4.08

Table 4.3: RMSE, MAE, MedAE Calculation

In Table 4.3, all these statistical results on different algorithms show that SVR has a lesser error than any of the other algorithms in the model when it comes to predicting data. All these proofs lead to select SVR as the primary prediction model for the experiment

4.2 Experimental Results

After all the statistical error analysis it is obvious that Support Vector Regressor (SVR) will be used for further prediction on the data-set but before that SVR is tested with some important analysis like confusion matrix, precision, recall,f1-score, Keras evaluation. Only important feature analysis is done by RandomforestRegressor. In the next phase of SVR modeling, the whole data will be predicted and then later compared with real data. Some time interval graphs are implemented to see how the prediction is behaving with the real data and these results give proofs of accuracy that has been reached by SVR. Later Future prediction graph is implemented to fulfill experimental results.

4.2.1 Confusion Matrix and Classification Report

To support the model confusion matrix and classification reports which includes some important statistical parameters precision, recall, f1score has been implemented. These calculations are performed based on y(test) and predicted(test) values. But before that, the data has been reshaped to Integer values to make it acceptable in the function parameters of the Scikit- learn metrics library.

1. **Confusion Matrix**:

   A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing [32].

2. **Precision and Recall**:

   In pattern recognition, information retrieval and binary classification, precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances, while recall (also known as sensitivity) is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. Both precision and recall are therefore based on an understanding and measure of relevance [33].

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

Figure 4.9: Precision

   Figure 4.9 represents the mathematical way to calculate precision in a model to find errors.

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

Figure 4.10: Recall

   Figure 4.10 represents the mathematical way to calculate recall in a model which works same as precision by understanding and measuring relevance.

3. **F1 Score**:

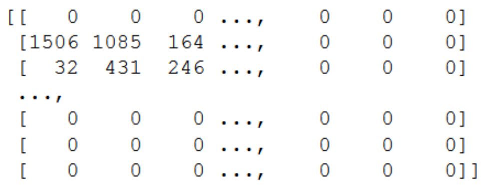
   In a statistical analysis of binary classification, the F1 score (also F-score or F-measure) is a measure of a test's accuracy [34].

$$F_1 = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

Figure 4.11: F1-Score

   Figure 4.11 represents the mathematical way to calculate F1-Score in a model is basically depended on precision and recall results.

```
y_regression = y_test.astype(np.int).reshape((-1, 1))
pred_test_value = pred_test_vm.astype(np.int).reshape((-1, 1))
matrix = metrics.confusion_matrix(y_regression, pred_test_value)
report = metrics.classification_report(y_regression, pred_test_value)
binary = np.array(matrix)
```



```
[[   0    0    0 ...,    0    0    0]
 [1506 1085  164 ...,    0    0    0]
 [  32  431  246 ...,    0    0    0]
 ...,
 [   0    0    0 ...,    0    0    0]
 [   0    0    0 ...,    0    0    0]
 [   0    0    0 ...,    0    0    0]]
```

Figure 4.12: Confusion Matrix Result

In Figure 4.12, the confusion matrix values are coming diagonally which means the classification of the model is satisfactory.

Precision, Recall, F1-Score Calculation		
Precision	Recall	F1-Score
0.17	0.11	0.13

Table 4.4: Precision, Recall, F1-Score Calculation

The results for precision in Table 4.4 shows that recall, and f1-score are pretty close to each other. These are some satisfactory measurements and proofs that the SVR is the prediction model for this experiment.

4.2.2 Keras Model Evaluation

In this section, experimental evaluation is performed on the developed model. To implement this evaluation a python library Keras evaluation has been used which runs TensorFlow in its background.

Keras is a high-level neural networks API, written in Python which is capable of running on top of TensorFlow and CNTK or Theano. It was developed with a vision for establishing fast experiment and be able to switch fast from idea to result with the least possible delay which is a key to good research [35].

So to perform this evaluation a test function is implemented which takes as input test and prediction values. But before passing the values to the function the data is shaped. The functions first calculate MSE manually and later it uses Keras model evaluation which calculates MSE too. If they are same that means the model that has been developed here has lesser error values [36].

Results:

```
preds: float32, (13491, 1)
y_regression: float32, (13491, 1)
manual result: mse=541.165100
evaluate result:
mse=541.1650862832759, mae=16.290467167155118,
mape=2003.6773339970948
```

The results have been calculated for test and prediction values. The results of the MSE for manual and evaluation result is same. The same implementation is done for test values and forecasted prediction values which provide the same result for both. That means the model is providing lesser error while doing any prediction.

4.2.3 Important Feature

Every model of machine learning approach requires certain features that shape up the implementation. Among them, some feature really influence the model on prediction or reaching the accuracy it deserves. In this case, a bar plot is generated to determine which feature played the most significant part in the model. RandomForestRegressor has its own component `feature_importances_` which provides a list of each feature importance results. In the implementation, the columns are stored in one specific array and later the importance of each column is sorted from higher to lower.

```
                importance
feature
E_8426_LIW          0.793
E_8426_VR50         0.097
Avg_TT0             0.060
PCT_M04             0.023
E_8426_VR60         0.010
E_8426_PT02         0.009
Avg_TT_Series       0.008
```

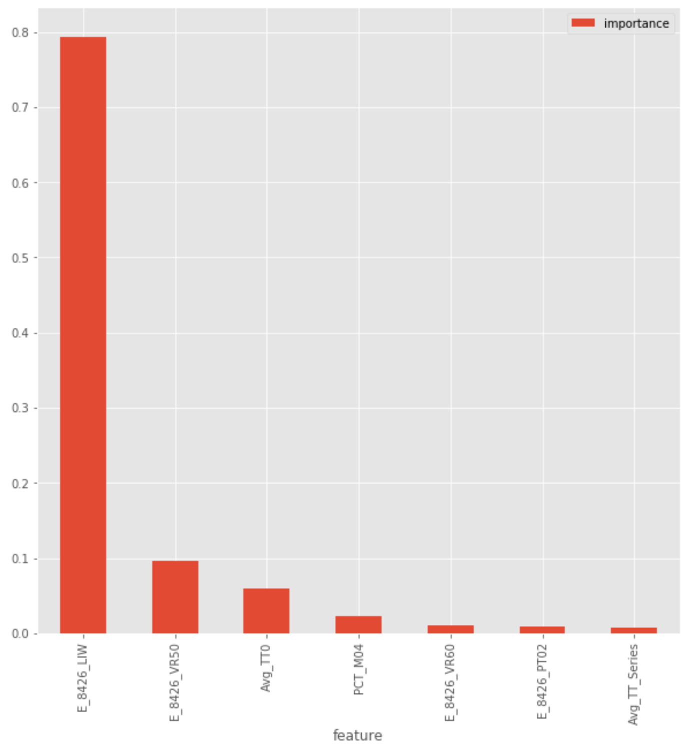


Figure 4.13: Important Feature Bar Plot

In Figure 4.13, it's visible that `E_8426_LIW` is the most important feature in the model as the bar plot shows high frequency than other features. This feature actually represents machine floor materials on the process. Other features are sorted as from most important to less important with values in the list.

4.2.4 Comparison Graphs

Among all the evaluations and validations it may come into question how the prediction and model work against the real data. In this phase, the real data is compared with prediction data for the same timeline that the data has been provided by Siemens. It means the prediction is done for the data on the same time frame and observed how close they are in the graph. The comparison is basically done on the whole data. As the data set is huge to make the prediction faster the data has been sliced down, predicted and the prediction is merged together. It helped to make the learning process faster. Different time interval graphs are implemented to see the how the real data and predicted data reacted on a different interval of time and if the difference is pretty close or not. As SVR is our main prediction model, these graphs also show that this model with the current features and implementations is capable of good prediction for a virtual ecosystem.

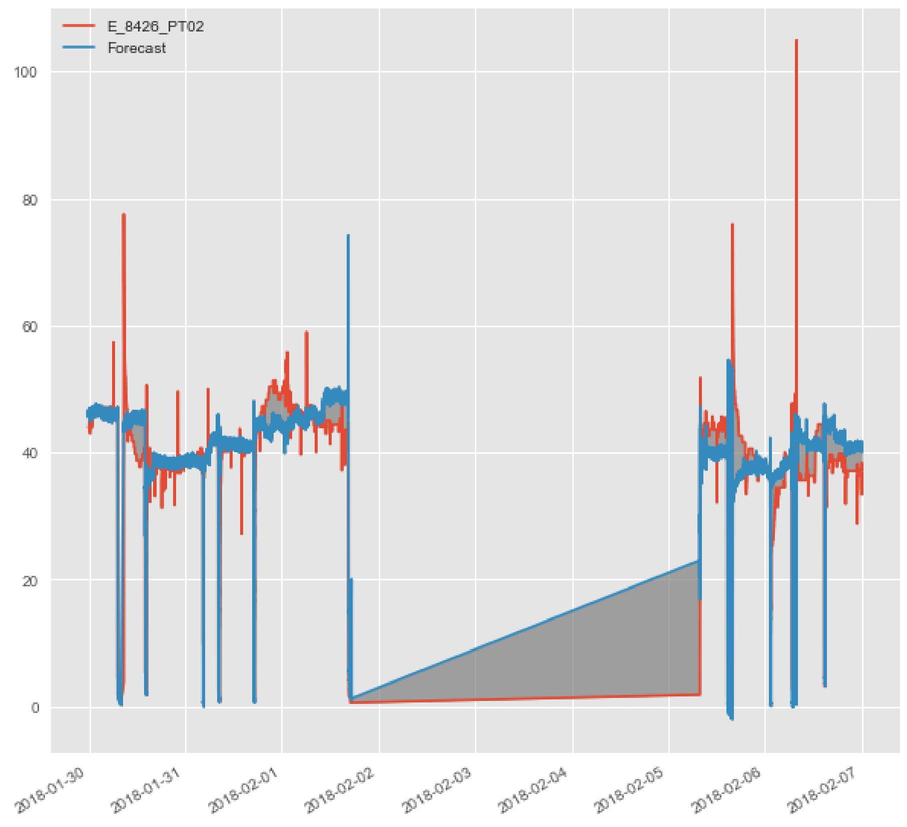


Figure 4.14: Comparing real data vs. forecast on the same data set

Figure 4.14 compares the real data and forecast data on the same data set. The graph shows that the model is predicting most of the up and downfall of the real data. There are some places where it is not able to predict properly. On two or three occasions the model was unable to predict the rise or downfall of pressure because there are some errors as we observed in the statistical analyses. The differences between both results are shown with grey color fill between them. Nevertheless, the model is predicting most of the data pretty close and ensures the accuracy has been reached is quite fine.

Next, different interval graphs for 10.20,30,40,50,60 minutes is produced. All these interval graphs are generated for both prediction and real values with the whole data.

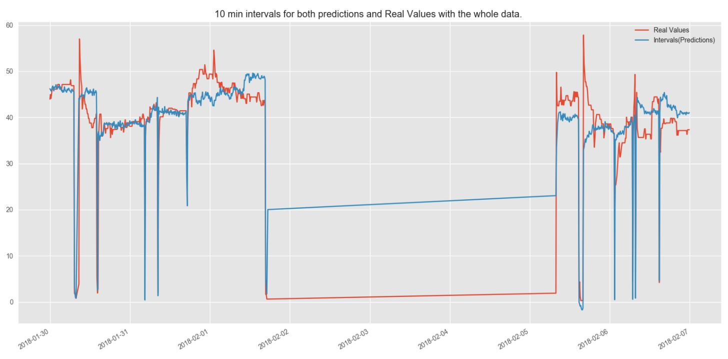


Figure 4.15: 10 min intervals for both Predictions and Real Values with the whole data

Figure 4.15 is generated by taking 10 min interval values for both prediction and real values to see how close they look and observe the model's constancy at 10 min interval of time in the process.

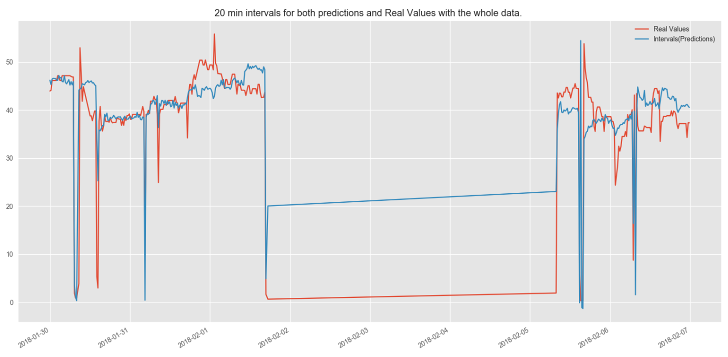


Figure 4.16: 20 min intervals for both Predictions and Real Values with the whole data

Figure 4.16 is generated by taking 20 min interval values for both prediction and real values to see how close they look and inspect like before the model's constancy at 20 min interval of time in the process.

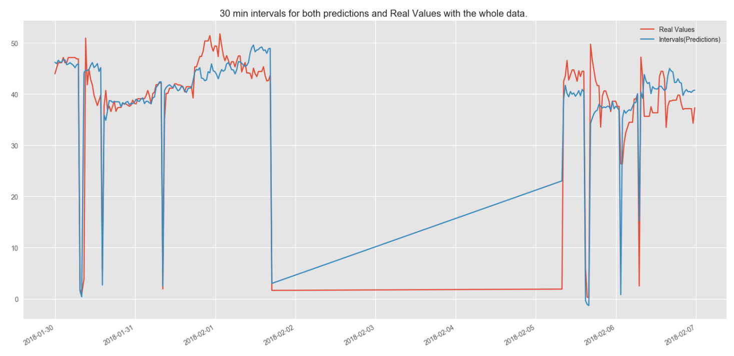


Figure 4.17: 30 min intervals for both predictions and Real Values with the whole data

Figure 4.17 is generated like same as before but taking 30 min interval values for both prediction and real values to see how similar they look and scrutinize like before the model's constancy at 30 min interval of time in the process.

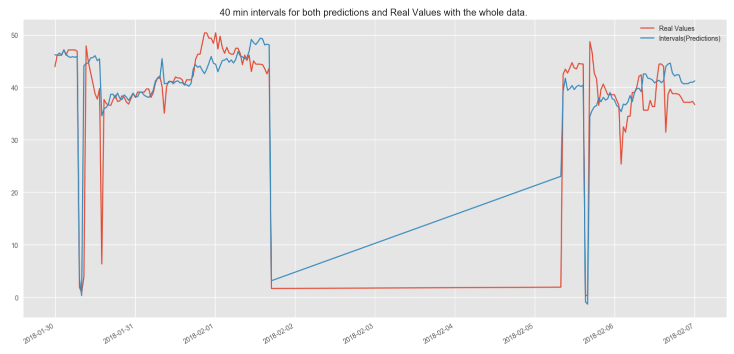


Figure 4.18: 40 min intervals for both Predictions and Real Values with the whole data

Figure 4.18 is generated by taking 40 min interval values for both prediction and real values to see how similar they look and monitor like before the model's constancy at 40 min interval of time in the process.

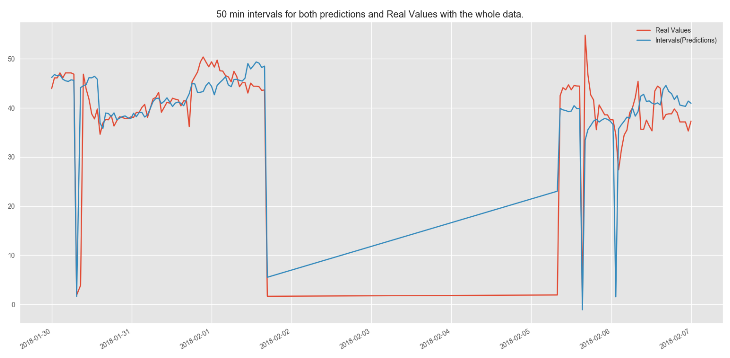


Figure 4.19: 50 min intervals for both Predictions and Real Values with the whole data

Figure 4.19 is generated by taking 50 min interval values for both prediction and real values to observe again their closeness are and monitor like before the model's constancy at 50 min interval of time in the process.

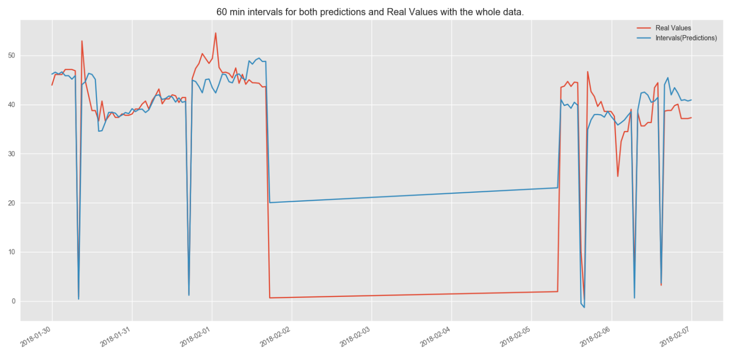


Figure 4.20: 60 min intervals for both Predictions and Real Values with the whole data

At last, in Figure 4.20 an hour interval values for both prediction and real values are taken to monitor finally how close they look and observe like before the model's constancy at 60 min interval of time in the process.

There are several dissimilarities but there are many cases where the prediction is accurate and most of them are pretty close. The distance between each time interval in real and prediction data is much closer to say the model predicts good enough. In several cases, it couldn't predict the increase of or downfall of the curve because of errors which were calculated in several statistical calculations.

4.2.5 Forecasting and Plotting Regression

The final part of the implementation is finally forecasting the values of pressure with Date-Time. To do that the last date of the given data is used, sliced and converted into a feasible format. The prediction is generated using the last 20 percent of the data that has been denoted already before as `x_lately`. Now to set up the dates with forecasted values the forecasted predicted values have been looped through and Date-Time has been assigned for each forecasted pressure value. All values are together kept in an array and later saved in a CSV file to see the values.

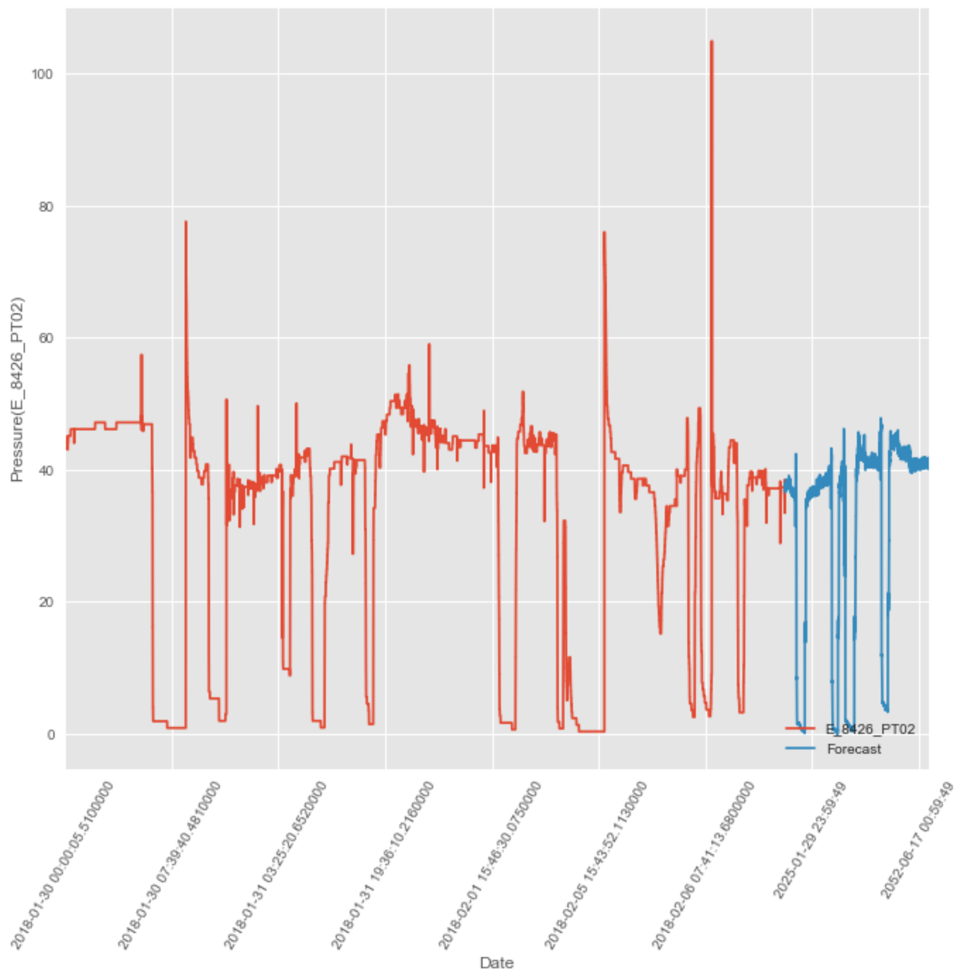
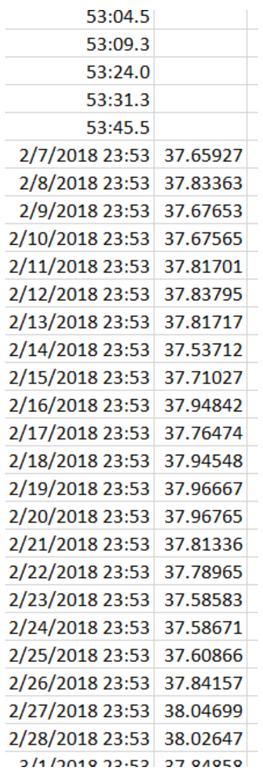


Figure 4.21: Future Prediction Graph with Regression(SVR)

Figure 4.21 plots regression graph where the real values and forecasted values have been put together with proper legends. On the x-axis there are dates and y-axis have the pressure values. The graph shows us the pattern of real values vs. future forecasted values. The future prediction is also performed for a longer period in future.



53:04.5	
53:09.3	
53:24.0	
53:31.3	
53:45.5	
2/7/2018 23:53	37.65927
2/8/2018 23:53	37.83363
2/9/2018 23:53	37.67653
2/10/2018 23:53	37.67565
2/11/2018 23:53	37.81701
2/12/2018 23:53	37.83795
2/13/2018 23:53	37.81717
2/14/2018 23:53	37.53712
2/15/2018 23:53	37.71027
2/16/2018 23:53	37.94842
2/17/2018 23:53	37.76474
2/18/2018 23:53	37.94548
2/19/2018 23:53	37.96667
2/20/2018 23:53	37.96765
2/21/2018 23:53	37.81336
2/22/2018 23:53	37.78965
2/23/2018 23:53	37.58583
2/24/2018 23:53	37.58671
2/25/2018 23:53	37.60866
2/26/2018 23:53	37.84157
2/27/2018 23:53	38.04699
2/28/2018 23:53	38.02647
3/1/2018 23:53	37.84858

Figure 4.22: Snippet of forecasted data in CSV

Finally, Figure 4.22 is showing the snippet of forecasted value that has been generated and saved in a CSV file. This snippet of CSV file is shown to observe how our model is predicting future data to see any significant changes in future for pressure.

4.2.6 Final Summary

As these forecasted values are stored in CSV, the concept of this experiment is, after all these prediction analyses the data can be uploaded into the cloud and later it can be fetched by an AR application which will send feedback for possible maintenance in future. It includes not only prediction data but also the prediction analysis graphs that have been generated. This way there will be an interactive platform or interface to show prediction, graph and keep alerted about any future problems which fulfill the goal of this virtual ecosystem combining AR and machine learning.

Chapter 5

Discussion, Summary and Future Directions

All the analyses and implementation show that this experiment has been conducted on Siemens industrial data for a bigger purpose. The data has been analyzed and discovered that it is a regression analysis data which is has been put through different validations to proof that the model is good enough. So, as we finally summarize our implementation, it is obvious that most of the significant implementation have been done on Machine Learning (ML) which includes huge data analysis from Siemens feed production machines. In this feed production machine, the quality of the final product from machines has depended on a parameter that was pressure (`E_8426_PT02`). The data analysis has been done generating several graphs and doing statistical calculations which gave us the concept that regression algorithms can be good. Various tasks done through the process were scattered graphs which show how different components in the machine were reacting against pressure, selection of the feature for the model, modifying features and StatsModel evaluation for the features. Four algorithms (Linear regression, SVR, K NeighborsRegressor, RandomForestRegressor) have been tested one by one by drawing scatter graphs and performing some statistical calculations. The data were divided with KFold cross-validation to perform many validation and statistical analysis before doing any prediction. The prediction algorithm SVR (Support Vector Regression) have been selected and used as it got higher accuracy than any other regression model. With this model, any future prediction has been showing good acceptable results as over-fitting issues have been resolved. Therefore, this model fulfills one part of our concept of this thesis that is Machine Learning (ML).

5.1 Problems Observed:

Several issues that we have faced while building this model. First important fact about the data that has been provided was good but not enough. If the experiment could be conducted on enough data, the prediction or results would have been much better. In the graphs, it was obvious that in some portion of the data the production was down. But still, we tried to get the best out of the data as much as possible. Next problem that has been faced was overfitting. At first, the splitting was done by Scikit-learn's model selection library train-test split which caused over-fitting while the predictions were calculated and graphs showed unacceptable results. Those results did not fit any Machine learning analysis. The accuracy was quite high for each algorithm and RandomForestRegressor was providing higher accuracy. At first, the prediction was done by RandomForestRegressor. But later when the over-fitting issue was discovered, the data split was converted into a cross-validation split. Scikit-Learn's KFold cross-validation splitting library has been used to solve this issue. The reason behind overfitting was the normal data splitting was taking random data from both training and testing sets which led to over-fitting on the process. The error was mainly discovered by seeing that the prediction graphs were predicting before the real data time frame.

5.2 Future Directions:

In the Machine Learning part, the only important task is done prediction of the most significant component that is pressure. By predicting pressure we suggested maintenance. But this part of implementation can be extended more by adding a classification feature or extended neural networks which will be able to detect the exact reasons of problems in the machines. There are some cases when the model is having some exception in predicting for high or low-pressure values. That can be solved also by introducing some classification approach. On the other hand, the model can be trained to detect any current problems that have already occurred. These future works can lead it to a robust system with various Machine Learning (ML) features to analyze issues in an industrial machine and provide maintenance and solution fast and early. As for the AR application part, it can be developed according to the future research works on the Machine Learning part which will include all the possibilities that have been discussed. This will enable the customers or engineers to conduct their task in a sophisticated high tech environment and provide a solution in a faster way. The more features are added to the machine learning parts the more implementation can be added to the AR application to show results, analysis, and conduct feedbacks. According to the implementations, the user interface can be developed in a user-friendly way.

All these future works can lead to a very important and significant application which will both save money and time. It also shows us that many researches can be conducted on the all the technological components, which possess a huge amount of data in Siemens. Like this virtual ecosystem, many excellent ideas or new technologies can be brought into the light for solving problems in relevant field.

5.3 Summary:

With the technological advancement in this era big data analysis, machine learning or any advanced technology like AR or VR have become a significant part of important research works nowadays. This experiment or research is a small approach to contribute to those significant areas. Even if there were issues, they are solved with much better implementation. KFold cross-validation is working like a charm and providing acceptable analysis. The graphs and prediction results are accepted as the overfitting issue is fixed. SVR is achieving the highest accuracy. Tuning is done on SVR to achieve this accuracy. We have also tried to predict the whole data again with our model and the results are quite satisfactory as our model is able to predict most of the data pretty close except some exceptions. The prediction graphs are not predicting anymore before the given time-frame in the data. The next best algorithms that show good accuracy are LinearRegression and RandomForestRegressor. Different time interval graphs show that even in different intervals of time the prediction of the model gives acceptable results which can be improved more. Another important fact has been found out that the longer the period of real data has been used for prediction the accuracy is better for each regression model. This concludes that this model is good enough to do any regression analysis and predict future. Siemens has so many industrial data that can be very useful to do any sort of machine learning algorithm like this. As our model only focused on regression within Siemens any regression type of data can be analyzed with our model. If some improved works are done in future machine learning like this can be a significant and useful part of the automation.

The AR application is merely a concept to show the model's prediction analysis, graphs or results in an interesting way. As many researches are going on in AR within Siemens the goal is here to show that AR and Machine Learning (ML) can be combined together and presented as a total sophisticated application. A simple AR application is developed which presents the results we produced with machine learning. But according to the concept all machine learning analysis can be stored in the cloud and later simple AR application can use it. This concept can help to build a robust application combining cloud, Machine Learning, and AR.

List of Figures

1.1 Proposed System Architecture 4
1.2 AR Application Example 8
1.3 AR Application Example in Field Service 8
1.4 AR Applications different roles applicable 9
1.5 Machine Learning Algorithms Flow Chart [2] 12
1.6 Machine Learning Supervised Process [12] 13

2.1 Machine Learning Supervised Process [14] 18
2.2 One-dimensional linear regression with epsilon intensive band. [18] 22
2.3 Non-linear regression function. [18] 22
2.4 Detailed picture of epsilon band with slack variables and selected data points [18] 23
2.5 KNN Regression 24

3.1 Feed Production Machine(1)(Siemens) 28
3.2 Feed Production Machine(2)(Siemens) 28
3.3 Pair Plots to show different data measure comparing pressure 30
3.4 Pair Plots to show different data measure comparing pressure 30
3.5 Pair Plots to show different data measure comparing pressure 31
3.6 Pressures vs. Predicted Pressures(LinearRegression) 38
3.7 Pressures vs. Predicted Pressures(SVR) 39
3.8 Pressures vs. Predicted Pressures(KNeighborRegression) 39
3.9 Pressures vs. Predicted Pressures(RandomForestRegressor) 40
3.10 K-Fold Cross Validation [26] 43
3.11 Machine Learning Approach 44
3.12 Machine Learning collaboration with AR 45

4.1 Residual Plot using training (blue) and test(green) data (LinearRegression) 48
4.2 Residual Plot using training (blue) and test(green) data (SVR) 49
4.3 Residual Plot using training (blue) and test(green) data (KNeighborsRegressor) 50
4.4 Residual Plot using training (blue) and test(green) data (RandomForestRegressor) 50
4.5 True vs. Predicted Value (LinearRegression) 52
4.6 True vs. Predicted Value (SVR) 53
4.7 True vs. Predicted Value (KNeighborsRegressor) 53
4.8 True vs. Predicted Value (RandomForestRegressor) 54
4.9 Precision 58
4.10 Recall 58

4.11 F1-Score 58
4.12 Confusion Matrix Result 59
4.13 Important Feature Bar Plot 61
4.14 Comparing real data vs. forecast on the same data set 62
4.15 10 min intervals for both Predictions and Real Values with the whole data 63
4.16 20 min intervals for both Predictions and Real Values with the whole data 63
4.17 30 min intervals for both predictions and Real Values with the whole data 64
4.18 40 min intervals for both Predictions and Real Values with the whole data 64
4.19 50 min intervals for both Predictions and Real Values with the whole data 65
4.20 60 min intervals for both Predictions and Real Values with the whole data 65
4.21 Future Prediction Graph with Regression(SVR) 66
4.22 Snippet of forecasted data in CSV 67

List of Tables

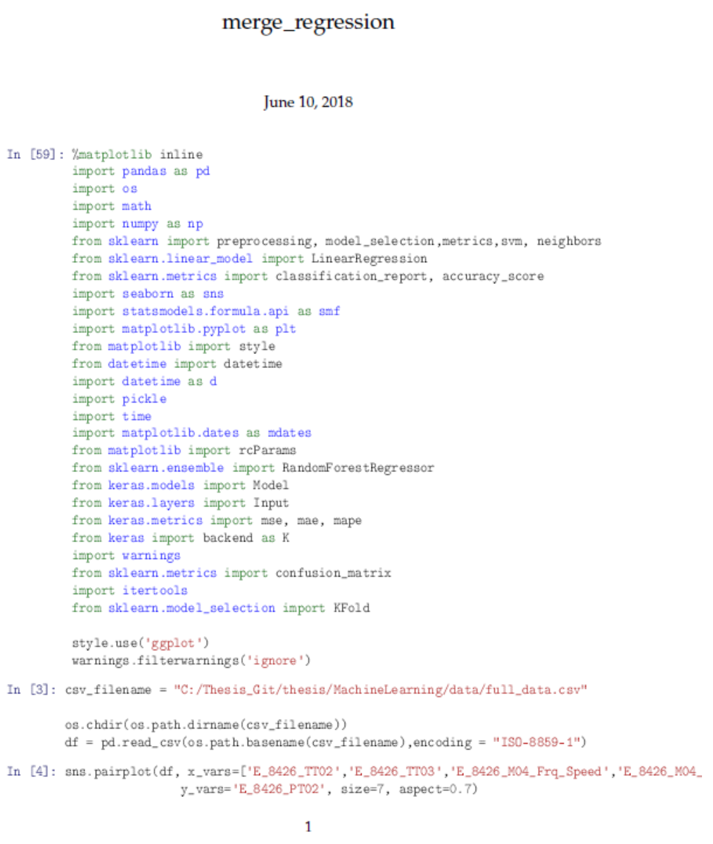
4.1 MSE with x_test, y_test 51
4.2 Accuracy Score For Different Models 55
4.3 RMSE, MAE, MedAE Calculation 57
4.4 Precision, Recall, F1-Score Calculation 59

Appendix A

Source Code

lheadAppendix *Appendix Source Code*

The source code for this thesis is give below as appendix:



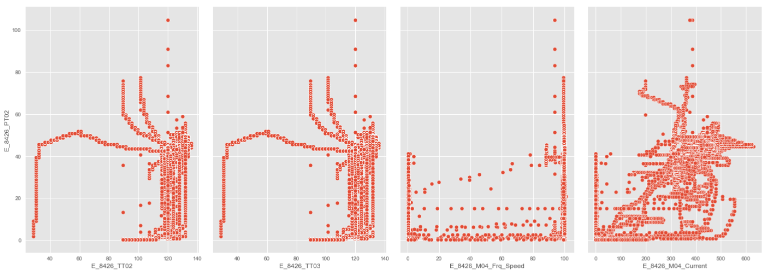
merge_regression

June 10, 2018

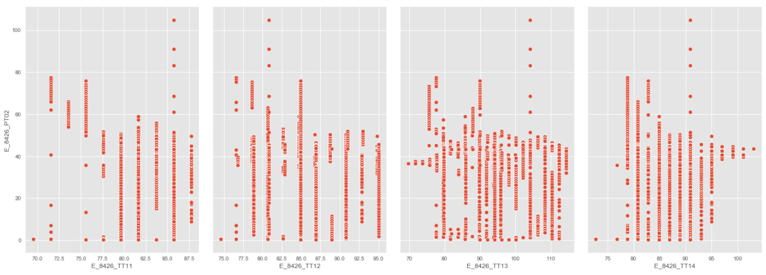
```
In [59]: %matplotlib inline
         import pandas as pd
         import os
         import math
         import numpy as np
         from sklearn import preprocessing, model_selection,metrics,svm, neighbors
         from sklearn.linear_model import LinearRegression
         from sklearn.metrics import classification_report, accuracy_score
         import seaborn as sns
         import statsmodels.formula.api as smf
         import matplotlib.pyplot as plt
         from matplotlib import style
         from datetime import datetime
         import datetime as d
         import pickle
         import time
         import matplotlib.dates as mdates
         from matplotlib import rcParams
         from sklearn.ensemble import RandomForestRegressor
         from keras.models import Model
         from keras.layers import Input
         from keras.metrics import mse, mae, mape
         from keras import backend as K
         import warnings
         from sklearn.metrics import confusion_matrix
         import itertools
         from sklearn.model_selection import KFold

         style.use('ggplot')
         warnings.filterwarnings('ignore')

In [3]: csv_filename = "C:/Thesis_Git/thesis/MachineLearning/data/full_data.csv"

        os.chdir(os.path.dirname(csv_filename))
        df = pd.read_csv(os.path.basename(csv_filename),encoding = "ISO-8859-1")

In [4]: sns.pairplot(df, x_vars=['E_8426_TT02','E_8426_TT03','E_8426_M04_Frq_Speed','E_8426_M04_
                         y_vars='E_8426_PT02', size=7, aspect=0.7)
```

1

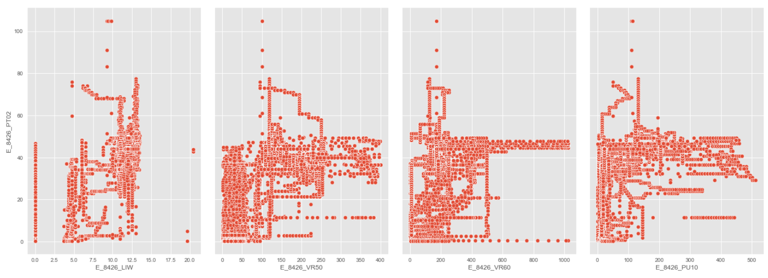
```
Out[4]: <seaborn.axisgrid.PairGrid at 0x8678d0>
```



```
In [5]: sns.pairplot(df, x_vars=['E_8426_TT11',
                                  'E_8426_TT12','E_8426_TT13','E_8426_TT14'],
                        y_vars='E_8426_PT02', size=7, aspect=0.7)

Out[5]: <seaborn.axisgrid.PairGrid at 0x8673c8>
```



```
In [6]: sns.pairplot(df, x_vars=['E_8426_LIW','E_8426_VR50',
                                  'E_8426_VR60','E_8426_PU10'],
                        y_vars='E_8426_PT02', size=7, aspect=0.7)

Out[6]: <seaborn.axisgrid.PairGrid at 0xe744898>
```



```
In [7]: pd.to_datetime(df['DateTime'])

        df = df.set_index(df['DateTime'])

        full_df = df

        df =df[['E_8426_PT02','E_8426_TT02',
              'E_8426_TT03','E_8426_M04_Frq_Speed',
              'E_8426_M04_Current','E_8426_TT11',
              'E_8426_TT12','E_8426_TT13','E_8426_TT14',
              'E_8426_LIW','E_8426_VR50',
              'E_8426_VR60','E_8426_PU10']]

        df['Avg_TTO'] = (df['E_8426_TT02']+ df['E_8426_TT03']) / 2

        df['Avg_TT_Series'] = (df['E_8426_TT11']+ df['E_8426_TT12']
        +df['E_8426_TT13']+df['E_8426_TT13'])/4

        df['PCT_M04'] = (df['E_8426_M04_Current'] - df['E_8426_M04_Frq_Speed']) / df['E_8426_M04

        df = df[['E_8426_PT02', 'Avg_TTO', 'Avg_TT_Series',
                 'PCT_M04','E_8426_LIW','E_8426_VR50',
              'E_8426_VR60','E_8426_PU10']]
```

StatsModels

Issues with R-squared R-squared will always increase as you add more features to the model, even if they are unrelated to the response Selecting the model with the highest R-squared is not a reliable approach for choosing the best linear model.

Solution Adjusted R-squared Penalizes model complexity (to control for overfitting), but it generally under-penalizes complexity.

Better Solution Train/test split or model_selection More reliable estimate of out-of-sample error Better for choosing which of your models will best generalize to out-of-sample data.There is extensive functionality for cross-validation in scikit-learn, including automated methods for searching different sets of parameters and different models

Importantly, cross-validation can be applied to any model, whereas the methods described above only apply to linear models

```
In [8]: ### STATSMODELS ###
        # create a fitted model with all the features
        lm1 = smf.ols(formula='E_8426_PT02 ~ Avg_TT0 + Avg_TT_Series + PCT_M04 + E_8426_LIW + E_
                      data=df).fit()

        # print the coefficients
        print('R-squared value for the model:',lm1.rsquared)

R-squared value for the model: 0.865464375161
```

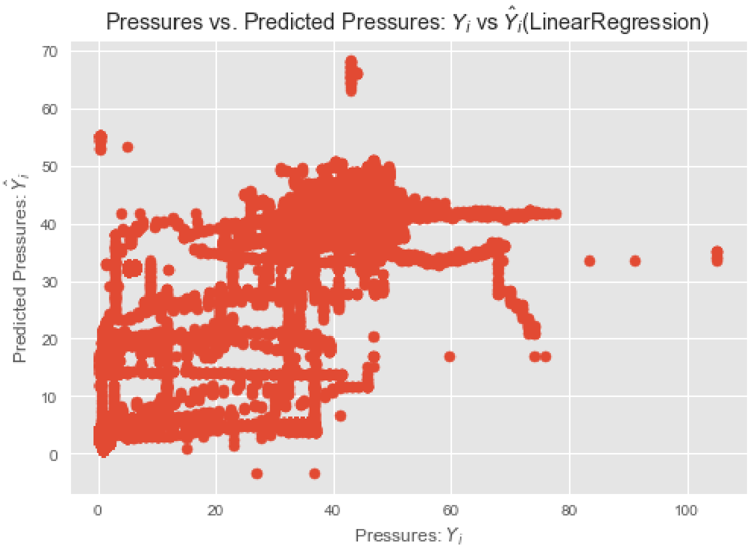
Getting Ready the data for implementing Scikit-learn. Forcast improvement for longer period.

```
In [9]: forcast_col = 'E_8426_PT02'
        df.fillna(-99999, inplace=True)
        forcast_out = int(math.ceil(0.2*len(df)))

In [10]: # df['label'] = df[forcast_col].shift(-forcast_out)

In [11]: x = np.array(df.drop(['E_8426_PT02'],1))
         x = preprocessing.scale(x)

         #slicing the data in multiple chunks to make prediction faster
         x_slice_1 = x[:10000]
         x_slice_2 = x[10000:20000]
         x_slice_3 = x[20000:30000]
         x_slice_4 = x[30000:40000]
         x_slice_5 = x[40000:50000]
         x_slice_6 = x[50000:60000]
         x_slice_7 = x[60000:67453]



         x_lately= x[-forcast_out:]


         df.dropna(inplace=True)

         y = np.array(df['E_8426_PT02'])

         lm = LinearRegression()
         lm.fit(x,y)

         kn = neighbors.KNeighborsRegressor()
         kn.fit(x,y)
        vm = svm.SVR()
        vm.fit(x,y)

        rfg = RandomForestRegressor()
        rfg.fit(x,y)

        print('Estimated intercept coefficient:', lm.intercept_)
        print('Number of coefficient:', len(lm.coef_))

        # print("10 Predicted Pressure LinearRegression:", lm.predict(x)[0:10])
        # print("10 Predicted Pressure KNeighborsRegression:", kn.predict(x)[0:10])
        # print("10 Predicted Pressure SVR:", vm.predict(x)[0:10])
        # print("10 Predicted Pressure RandomForestRegressor:", rfg.predict(x)[0:10])
```

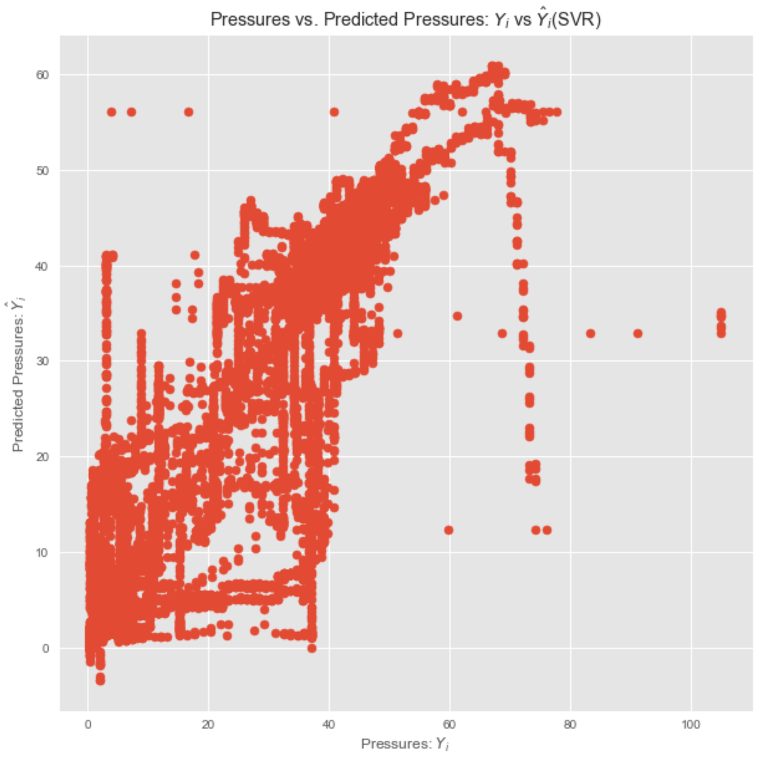
```
Estimated intercept coefficient: 33.0650547749
Number of coefficient: 7
```

Scatter plot to compare true pressures and the predicted pressures.

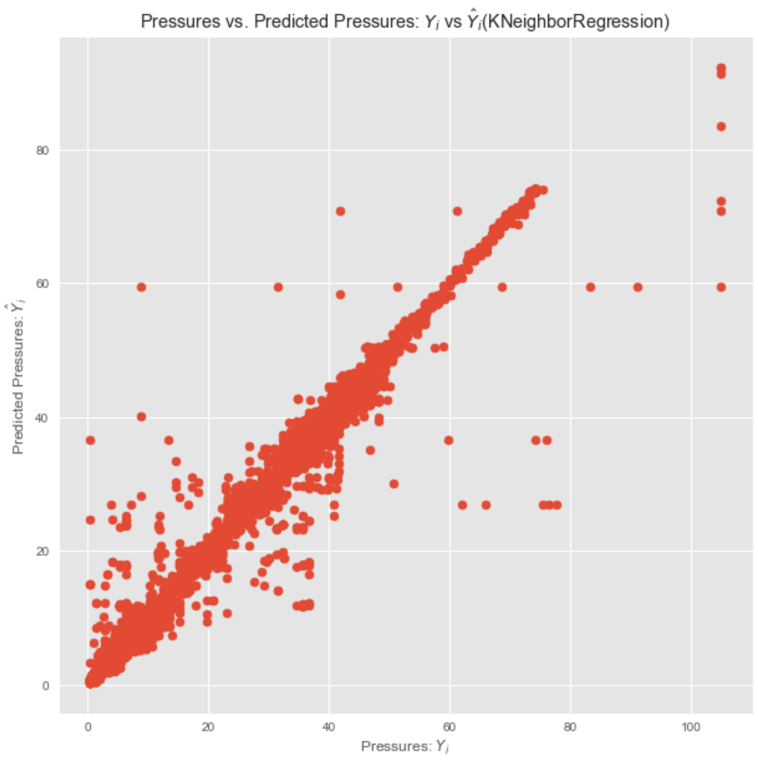
```
In [12]: plt.scatter(y, lm.predict(x))
         plt.xlabel("Pressures: $Y_i$")
         plt.ylabel("Predicted Pressures: $\hat{Y}_i$")
         plt.title("Pressures vs. Predicted Pressures: $Y_i$ vs $\hat{Y}_i$(LinearRegression)")
         plt.rcParams["figure.figsize"] = (10,10)
         plt.show()
```



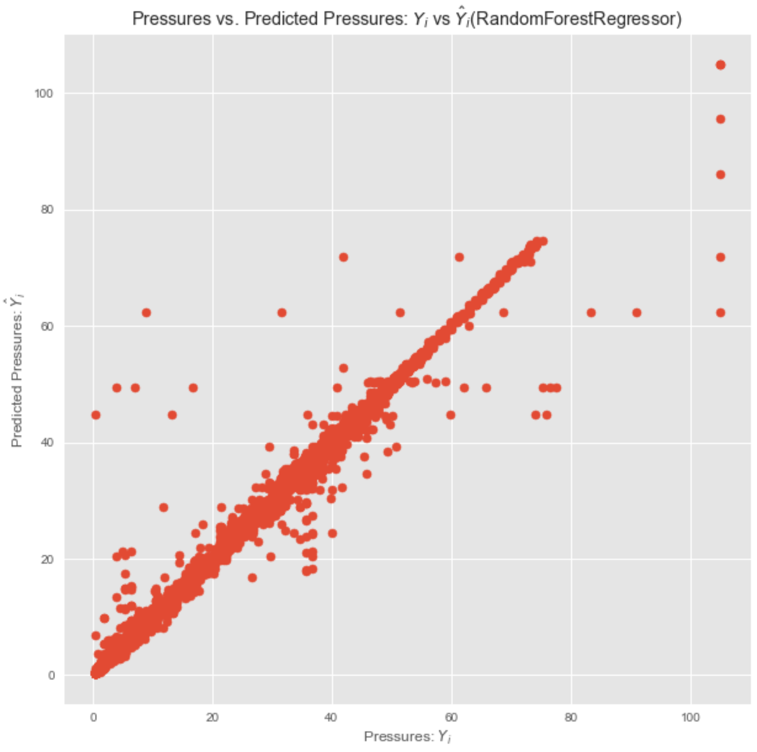
```
In [13]: plt.scatter(y, vm.predict(x))
         plt.xlabel("Pressures: $Y_i$")
         plt.ylabel("Predicted Pressures: $\hat{Y}_i$")
         plt.title("Pressures vs. Predicted Pressures: $Y_i$ vs $\hat{Y}_i$(SVR)")
         plt.rcParams["figure.figsize"] = (10,10)
         plt.show()
```



```
In [14]: plt.scatter(y, kn.predict(x))
         plt.xlabel("Pressures: $Y_i$")
         plt.ylabel("Predicted Pressures: $\hat{Y}_i$")
plt.title("Pressures vs. Predicted Pressures: $Y_i$ vs $\hat{Y}_i$(KNeighborRegression)
plt.rcParams["figure.figsize"] = (10,10)
plt.show()
```



```
In [15]: plt.scatter(y, rfg.predict(x))
         plt.xlabel("Pressures: $Y_i$")
         plt.ylabel("Predicted Pressures: $\hat{Y}_i$")
         plt.title("Pressures vs. Predicted Pressures: $Y_i$ vs $\hat{Y}_i$(RandomForestRegresso
         plt.rcParams["figure.figsize"] = (10,10)
         plt.show()
```



```
In [16]: # x_train, x_test, y_train, y_test = model_selection.train_test_split(x, y, test_size=(

         x= x[:-forcast_out]

         kf = KFold(n_splits=2) # Define the split - into 2 folds
         folds = kf.get_n_splits(x) # returns the number of splitting iterations in the cross-va

         print("Folds:", folds)

         # KFold(n_splits=6, random_state=None, shuffle=False)
         #
         for train_index, test_index in kf.split(x):
             print("TRAIN:", train_index, "TEST:", test_index)
```

```
            x_train, x_test = x[train_index], x[test_index]
            y_train, y_test = y[train_index], y[test_index]


        clf = LinearRegression(n_jobs=-1)
        clf.fit(x_train,y_train)

        with open('linearregression.pickle', 'wb') as f:
            pickle.dump(clf, f)

        pickle_in = open('linearregression.pickle', 'rb')

        clf = pickle.load(pickle_in)
```

```
Folds: 2
TRAIN: [26981 26982 26983 ..., 53958 53959 53960] TEST: [    0     1     2 ..., 26978 26979 2698
TRAIN: [    0     1     2 ..., 26978 26979 26980] TEST: [26981 26982 26983 ..., 53958 53959 5396
```

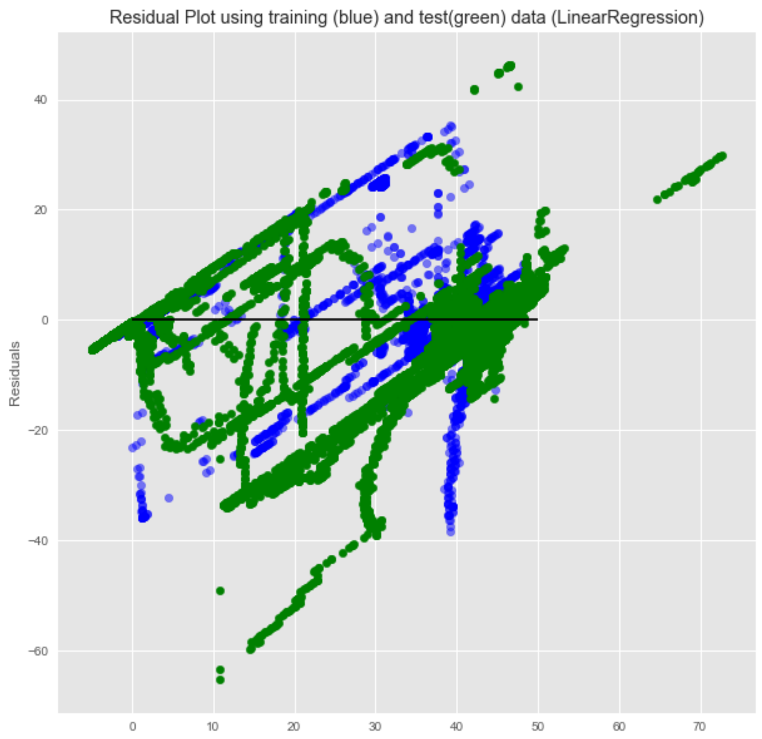
```
In [17]: kng = neighbors.KNeighborsRegressor()
         kng.fit(x_train,y_train)

         with open('KNeighborsRegressor.pickle', 'wb') as f:
             pickle.dump(kng, f)

         pickle_in = open('KNeighborsRegressor.pickle', 'rb')

         kng = pickle.load(pickle_in)

In [18]: vm = svm.SVR(kernel='linear')
         vm.fit(x_train,y_train)

         with open('SVR.pickle', 'wb') as f:
             pickle.dump(vm, f)

         pickle_in = open('SVR.pickle', 'rb')

         vm = pickle.load(pickle_in)

In [19]: n_trees = 2000
         rfg = RandomForestRegressor()
         rfg.fit(x_train,y_train)

         with open('RandomForestRegressor.pickle', 'wb') as f:
             pickle.dump(rfg, f)

         pickle_in = open('RandomForestRegressor.pickle', 'rb')

         rfg = pickle.load(pickle_in)
```

```
In [ ]:
```

"Residual plots are a good way to visualize the errors in your data.If you have done a good job then your data should be randomly scattered around line zero. If you see structure in your data, that means your model is not capturing some thing. May be there is a interaction between 2 variables that you are not considering, or may be you are measuring time dependent data. If you get some structure in your data, you should go back to your model and check whether you are doing a good job with your parameters."

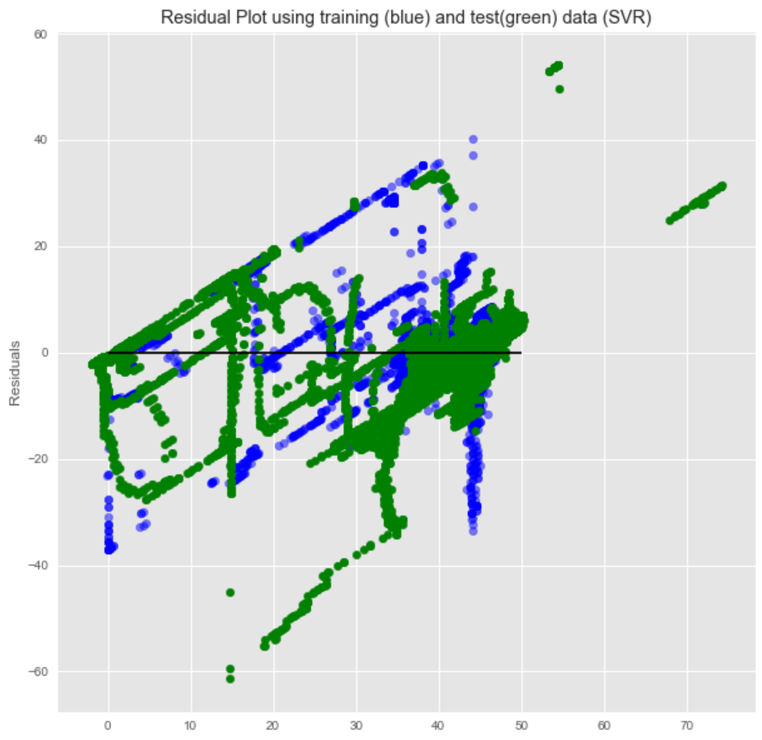
```
In [20]: pred_train = clf.predict(x_train)
         pred_test = clf.predict(x_test)
         print("Mean squared error for training and test data:")
         print("Fit a model X_train, and calculate MSE with Y_train:",np.mean(y_train-clf.predic
         print("Fit a model X_train, and calculate MSE with X_test, Y_test:",np.mean(y_test-clf.

         plt.scatter(clf.predict(x_train), clf.predict(x_train) - y_train, c='b', s=40, alpha=0.
         plt.scatter(clf.predict(x_test), clf.predict(x_test) - y_test, c='g', s=40)
         plt.hlines(y=0, xmin=0, xmax =50)
         plt.rcParams["figure.figsize"] = (10,10)
         plt.title('Residual Plot using training (blue) and test(green) data (LinearRegression)'
         plt.ylabel('Residuals')
         plt.show()
```

```
Mean squared error for training and test data:
Fit a model X_train, and calculate MSE with Y_train: 1.016066549e-25
Fit a model X_train, and calculate MSE with X_test, Y_test: 1.72083640817
```



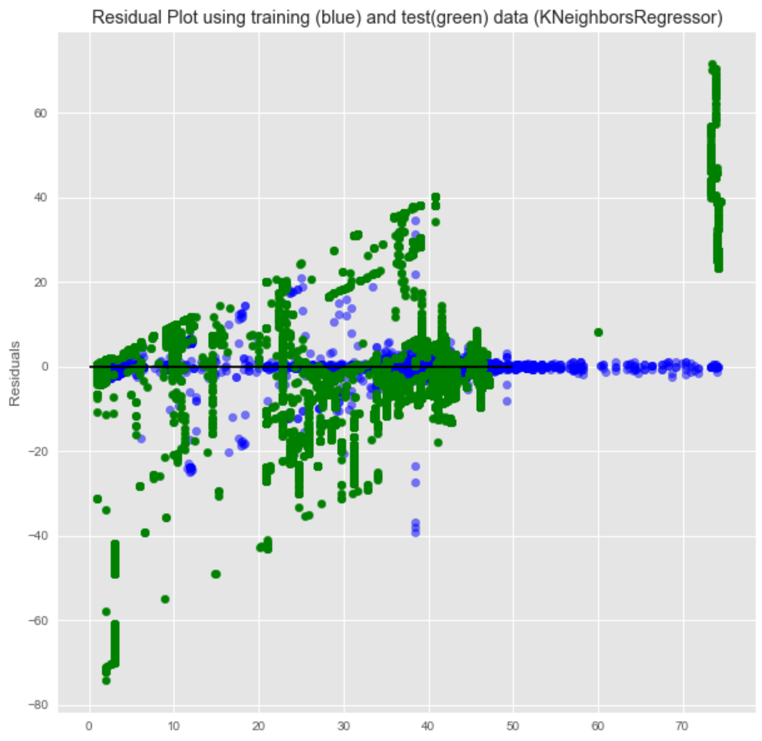
```
In [21]: pred_train_vm = vm.predict(x_train)
         pred_test_vm = vm.predict(x_test)
         print("Mean squared error for training and test data:")
         print("Fit a model X_train, and calculate MSE with Y_train:",np.mean(y_train-vm.predict
         print("Fit a model X_train, and calculate MSE with X_test, Y_test:",np.mean(y_test-vm.p

         plt.scatter(vm.predict(x_train), vm.predict(x_train) - y_train, c='b', s=40, alpha=0.5)
         plt.scatter(vm.predict(x_test), vm.predict(x_test) - y_test, c='g', s=40)
         plt.hlines(y=0, xmin=0, xmax =50)
         plt.rcParams["figure.figsize"] = (10,10)
         plt.title('Residual Plot using training (blue) and test(green) data (SVR)')
         plt.ylabel('Residuals')
         plt.show()
```

```
Mean squared error for training and test data:
Fit a model X_train, and calculate MSE with Y_train: 0.132587316774
Fit a model X_train, and calculate MSE with X_test, Y_test: 1.61179027145
```



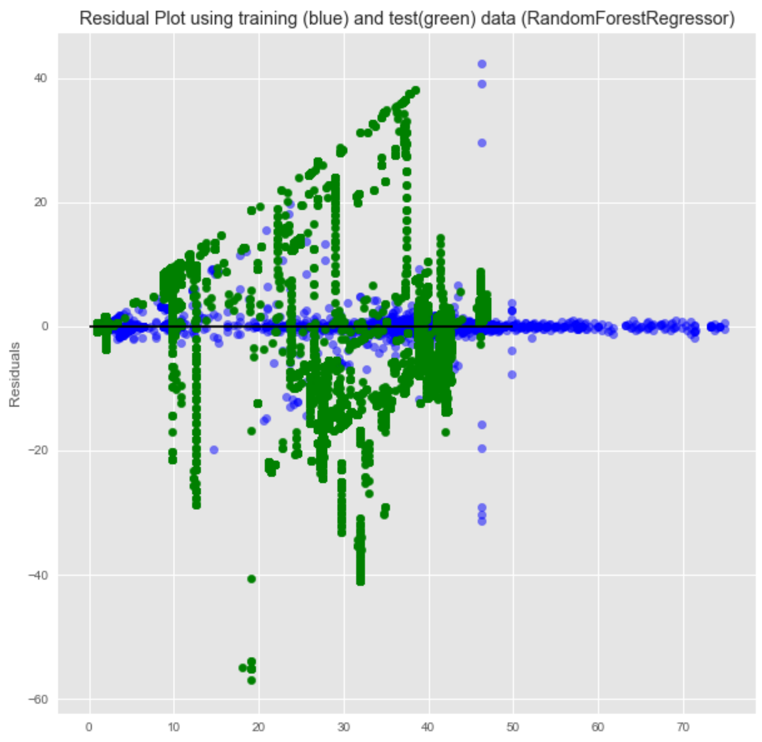
```
In [22]: pred_train_kng = kng.predict(x_train)
         pred_test_kng = kng.predict(x_test)
         print("Mean squared error for training and test data:")
         print("Fit a model X_train, and calculate MSE with Y_train:",np.mean(y_train-kng.predic
         print("Fit a model X_train, and calculate MSE with X_test, Y_test:",np.mean(y_test-kng.

         plt.scatter(kng.predict(x_train), kng.predict(x_train) - y_train, c='b', s=40, alpha=0.
         plt.scatter(kng.predict(x_test), kng.predict(x_test) - y_test, c='g', s=40)
         plt.hlines(y=0, xmin=0, xmax =50)
plt.rcParams["figure.figsize"] = (10,10)
plt.title('Residual Plot using training (blue) and test(green) data (KNeighborsRegresso
plt.ylabel('Residuals')
plt.show()
```

```
Mean squared error for training and test data:
Fit a model X_train, and calculate MSE with Y_train: 3.60903509106e-05
Fit a model X_train, and calculate MSE with X_test, Y_test: 5.91559859264
```



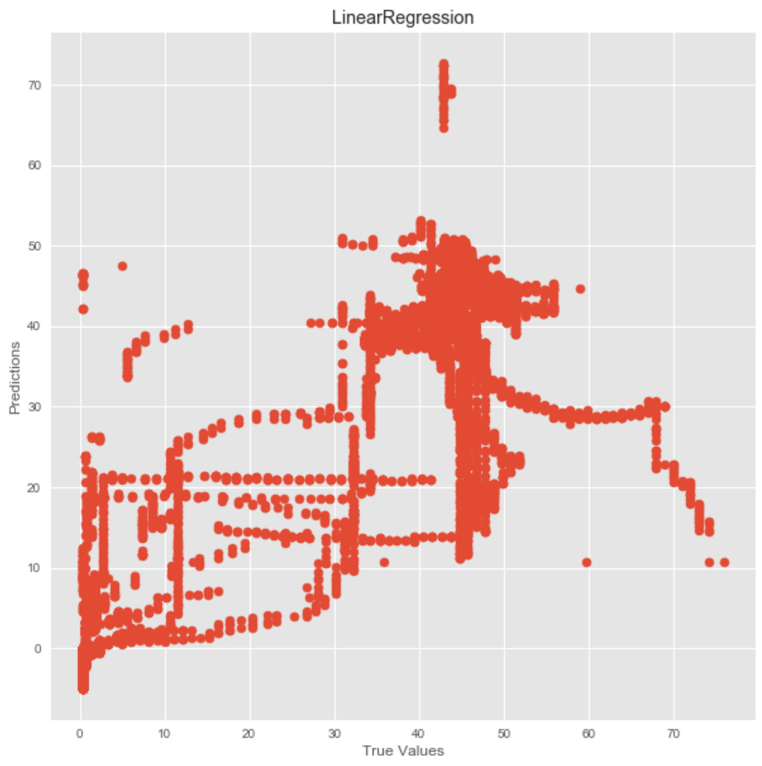
```
In [23]: pred_train_rfg = rfg.predict(x_train)
         pred_test_rfg = rfg.predict(x_test)
         print("Mean squared error for training and test data:")
         print("Fit a model X_train, and calculate MSE with Y_train:",np.mean(y_train-rfg.predic
        print("Fit a model X_train, and calculate MSE with X_test, Y_test:",np.mean(y_test-rfg.

        plt.scatter(rfg.predict(x_train), rfg.predict(x_train) - y_train, c='b', s=40, alpha=0.
        plt.scatter(rfg.predict(x_test), rfg.predict(x_test) - y_test, c='g', s=40)
        plt.hlines(y=0, xmin=0, xmax =50)
        plt.rcParams["figure.figsize"] = (10,10)
        plt.title('Residual Plot using training (blue) and test(green) data (RandomForestRegres
        plt.ylabel('Residuals')
        plt.show()
```

```
Mean squared error for training and test data:
Fit a model X_train, and calculate MSE with Y_train: 9.79181189302e-06
Fit a model X_train, and calculate MSE with X_test, Y_test: 4.94582563098
```



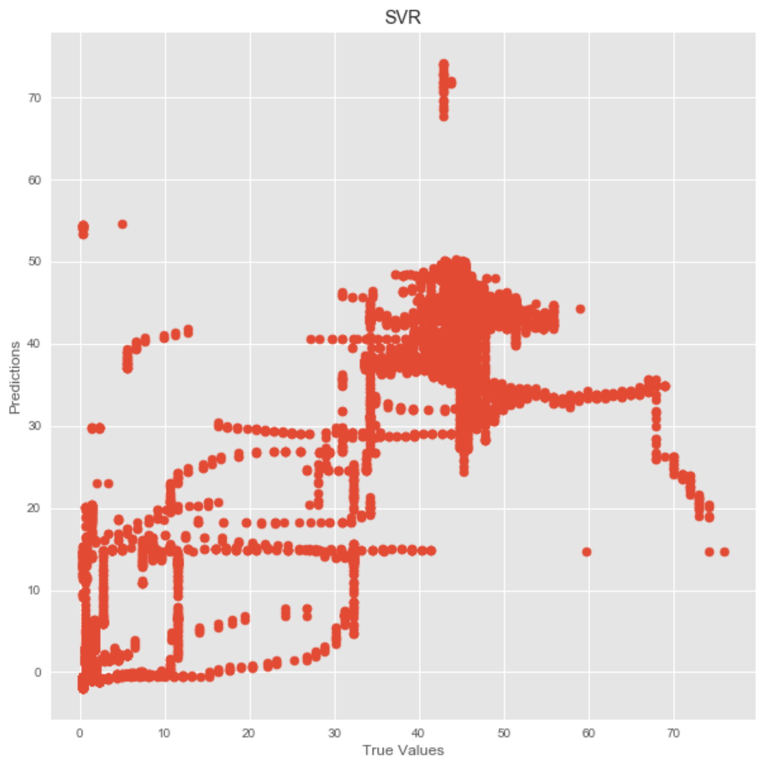
```
In [24]: accuracy = clf.score(x_test,y_test)
         print("Root Mean Squared Error (RMSE) is the square root of the mean of the squared err
         print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test, pred_test)))
         print("The Mean Absolute Error: %.2f " % metrics.mean_absolute_error(y_test, pred_test)
         print("The Median Absolute Error: %.2f " % metrics.median_absolute_error(y_test, pred_t
         print("Accuracy Scikit-learn(LinearRegression):",accuracy)

         plt.scatter(y_test, pred_test)
         plt.title("LinearRegression")
         plt.xlabel("True Values")
         plt.ylabel("Predictions")
         plt.show()
```

```
Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors(LinearRegres
RMSE: 7.4798213597
The Mean Absolute Error: 4.82
The Median Absolute Error: 3.73
Accuracy Scikit-learn(LinearRegression): 0.836039535805
```



```
In [25]: accuracy = vm.score(x_test,y_test)
         print("Root Mean Squared Error (RMSE) is the square root of the mean of the squared err
         print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test, pred_test_vm)))
         print("The Mean Absolute Error: %.2f " % metrics.mean_absolute_error(y_test, pred_test_
         print("The Median Absolute Error: %.2f " % metrics.median_absolute_error(y_test, pred_t
         print("Accuracy Scikit-learn:",accuracy)

         plt.scatter(y_test, pred_test_vm)
         plt.title("SVR")
         plt.xlabel("True Values")
         plt.ylabel("Predictions")
         plt.show()
```

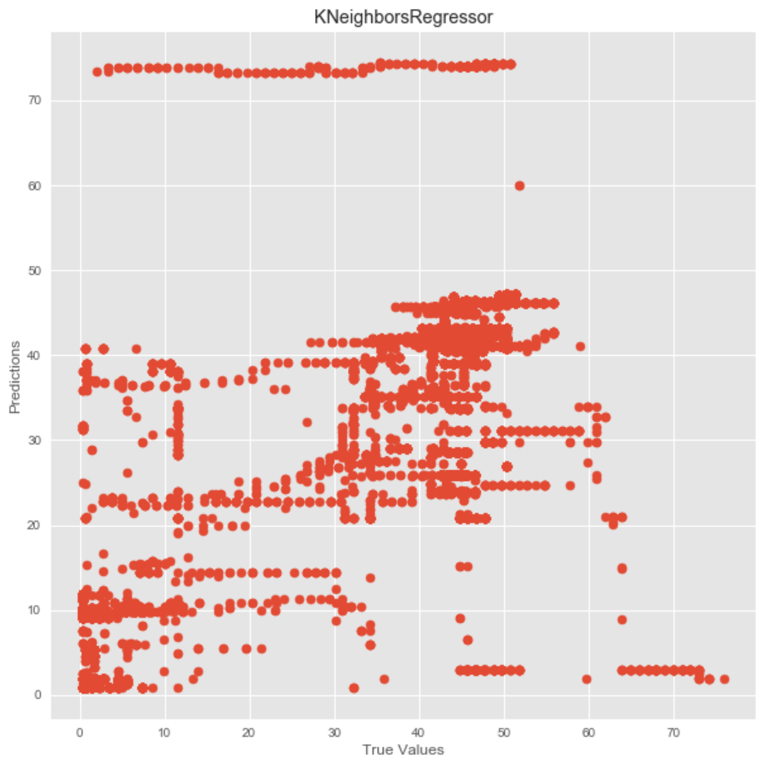
```
Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors(SVR):
RMSE: 6.53935204454
The Mean Absolute Error: 4.27
The Median Absolute Error: 3.37
Accuracy Scikit-learn: 0.8746783443
```



```
In [26]: accuracy = kng.score(x_test,y_test)
         print("Root Mean Squared Error (RMSE) is the square root of the mean of the squared err
         print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test, pred_test_kng)))
         print("The Mean Absolute Error: %.2f " % metrics.mean_absolute_error(y_test, pred_test_
         print("The Median Absolute Error: %.2f " % metrics.median_absolute_error(y_test, pred_t
         print("Accuracy Scikit-learn:",accuracy)
```

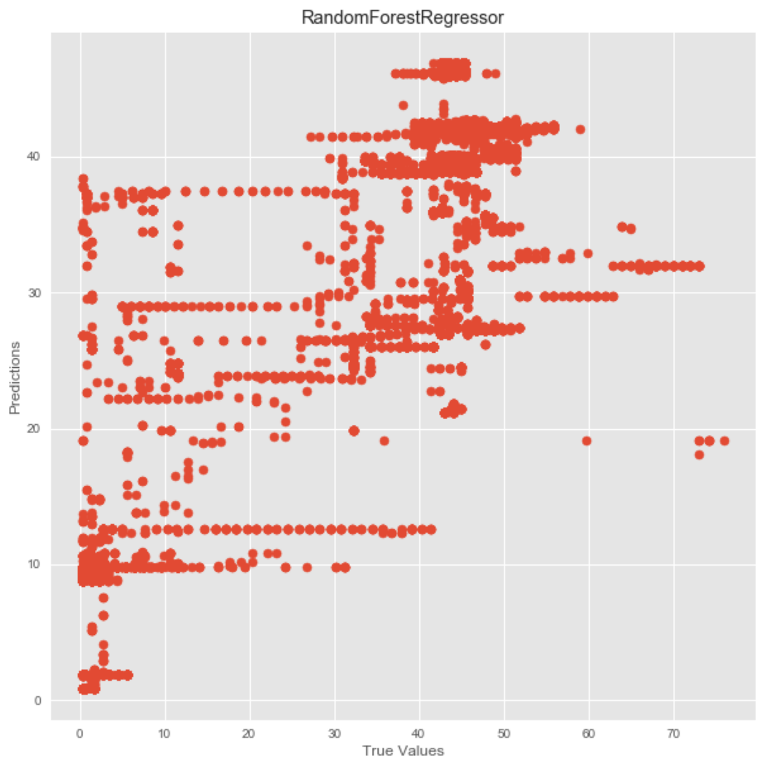
```
        plt.scatter(y_test, pred_test_kng)
        plt.title("KNeighborsRegressor")
        plt.xlabel("True Values")
        plt.ylabel("Predictions")
        plt.show()
```

```
Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors(KNeighborsRe
RMSE: 10.9931452553
The Mean Absolute Error: 6.62
The Median Absolute Error: 3.36
Accuracy Scikit-learn: 0.645839093481
```



```
In [27]: accuracy = rfg.score(x_test,y_test)
         print("Root Mean Squared Error (RMSE) is the square root of the mean of the squared err
         print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test, pred_test_rfg)))
         print("The Mean Absolute Error: %.2f " % metrics.mean_absolute_error(y_test, pred_test_
         print("The Median Absolute Error: %.2f " % metrics.median_absolute_error(y_test, pred_t
         print("Accuracy Scikit-learn:",accuracy)

         plt.scatter(y_test, pred_test_rfg)
         plt.title("RandomForestRegressor")
         plt.xlabel("True Values")
         plt.ylabel("Predictions")
         plt.show()
```

```
Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors(RandomForest
RMSE: 7.80753931268
The Mean Absolute Error: 5.26
The Median Absolute Error: 3.63
Accuracy Scikit-learn: 0.821357393364
```

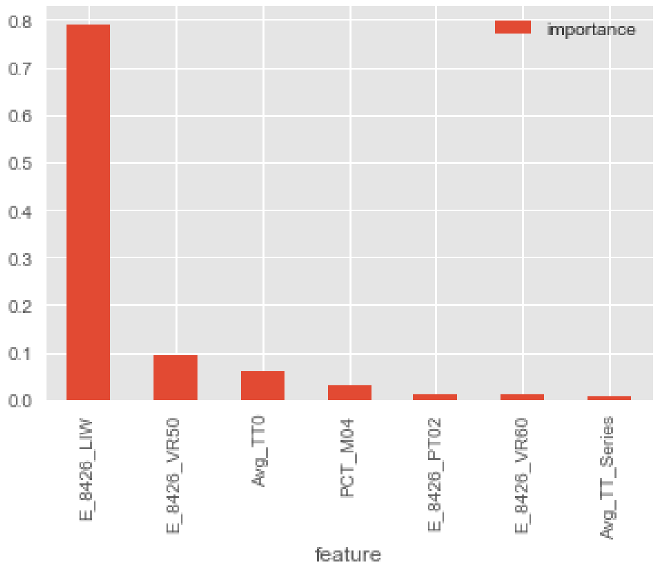


```
In [66]: col = []
         for i in df.columns:
             col.append(i)
         col = col[0:7]

         importances = pd.DataFrame({'feature':col,'importance':np.round(rfg.feature_importances
         importances = importances.sort_values('importance',ascending=False).set_index('feature'
         print (importances)
         importances.plot.bar()
```

```
          importance
feature
E_8426_LIW         0.792
```

```
E_8426_VR50         0.094
Avg_TT0             0.060
PCT_M04             0.029
E_8426_PT02         0.010
E_8426_VR60         0.010
Avg_TT_Series       0.006
```

```
Out[66]: <matplotlib.axes._subplots.AxesSubplot at 0x7352ef0>
```



Testing confusion matrix and classification report on true values and predicted values to determine some statistical results

```
In [51]: y_regression = y_test.astype(np.int).reshape((-1, 1))
         pred_test_value = pred_test_vm.astype(np.int).reshape((-1, 1))
         matrix = metrics.confusion_matrix(y_regression, pred_test_value)
         report = metrics.classification_report(y_regression, pred_test_value)
         binary = np.array(matrix)
         print(matrix)
         print(report)
```

```
[[   0    0    0 ...,    0    0    0]
 [1506 1085  164 ...,    0    0    0]
[  32  431  246 ...,    0    0    0]
...,
[   0    0    0 ...,    0    0    0]
[   0    0    0 ...,    0    0    0]
[   0    0    0 ...,    0    0    0]]
             precision    recall  f1-score   support

         -1       0.00      0.00      0.00         0
          0       0.50      0.35      0.42      3076
          1       0.45      0.13      0.21      1856
          2       0.00      0.00      0.00       484
          3       0.00      0.00      0.00       105
          4       0.00      0.00      0.00       222
          5       0.00      0.00      0.00       121
          6       0.00      0.00      0.00        67
          7       0.00      0.00      0.00       100
          8       0.00      0.00      0.00        67
          9       0.00      0.00      0.00        39
         10       0.00      0.00      0.00        73
         11       0.28      0.06      0.09       142
         12       0.00      0.00      0.00        20
         13       0.00      0.00      0.00         9
         14       0.00      0.00      0.00        12
         15       0.00      0.00      0.00         9
         16       0.04      0.08      0.06        24
         17       0.00      0.00      0.00        17
         18       0.00      0.00      0.00        18
         19       0.00      0.00      0.00        11
         20       0.00      0.00      0.00        20
         21       0.00      0.00      0.00         9
         22       0.00      0.00      0.00        14
         23       0.00      0.00      0.00        13
         24       0.00      0.00      0.00        17
         25       0.00      0.00      0.00         9
         26       0.11      0.25      0.15        20
         27       0.00      0.00      0.00        16
         28       0.00      0.00      0.00        31
         29       0.02      0.09      0.03        23
         30       0.00      0.00      0.00        63
         31       0.00      0.00      0.00        41
         32       0.00      0.00      0.00       161
         33       0.00      0.00      0.00       109
         34       0.02      0.02      0.02       200
         35       0.00      0.00      0.00       184
         36       0.03      0.21      0.06       124
         37       0.02      0.34      0.04       115
         38       0.31      0.27      0.29       730
         39       0.00      0.01      0.00       529
         40       0.03      0.09      0.05       687
         41       0.44      0.25      0.32      1361
         42       0.00      0.00      0.00      1341
         43       0.02      0.01      0.01      2669
         44       0.04      0.02      0.03      3757
         45       0.26      0.18      0.21      2736
         46       0.29      0.13      0.18      1578
         47       0.01      0.00      0.01      1123
         48       0.00      0.00      0.00       642
         49       0.00      0.00      0.00       921
         50       0.00      0.00      0.00       739
         51       0.00      0.00      0.00       160
         52       0.00      0.00      0.00        53
         53       0.00      0.00      0.00        26
         54       0.00      0.00      0.00        78
         55       0.00      0.00      0.00        39
         56       0.00      0.00      0.00         3
         57       0.00      0.00      0.00         7
         58       0.00      0.00      0.00        12
         59       0.00      0.00      0.00         7
         60       0.00      0.00      0.00         9
         61       0.00      0.00      0.00         4
         62       0.00      0.00      0.00         5
         63       0.00      0.00      0.00        10
         64       0.00      0.00      0.00         8
         65       0.00      0.00      0.00        10
         66       0.00      0.00      0.00        10
         67       0.00      0.00      0.00        20
         68       0.00      0.00      0.00         7
         69       0.00      0.00      0.00        11
         70       0.00      0.00      0.00         0
         71       0.00      0.00      0.00        13
         72       0.00      0.00      0.00        13
         73       0.00      0.00      0.00        15
         74       0.00      0.00      0.00         5
         75       0.00      0.00      0.00         1

avg / total       0.17      0.11      0.13     26980
```

```
In [30]: forcast_set = vm.predict(x_lately)

         y_test_slice= y_test[-forcast_out:]
         #print(forcast_set, accuracy, forcast_out)
         df['Forecast'] = np.nan

         last_date = df.iloc[-1].name
```

```
        last_date= last_date[:19]
        dt=time.mktime(datetime.strptime(last_date, '%Y-%m-%d %H:%M:%S').timetuple())
```

Plotting Test values and Forcasted values to see the behavior of the prediction with real data

```
In [31]: def data_difference(data, dates, interval=10*60):
             diff = []
             new_time = []
             flag = time.mktime(datetime.strptime(str(dates[0]), '%Y-%m-%d %H:%M:%S').timetuple(
             diff.append(0)
             new_time.append(flag)
             for idx, d in enumerate(dates):
                 if idx > 0:
                     d = time.mktime(datetime.strptime(str(d), '%Y-%m-%d %H:%M:%S').timetuple())
                     if d - flag >= interval:
                         flag = d
                         diff.append(idx)
                         new_time.append(d)
             new_data = [data[i] for i in diff]
             return new_time,new_data
```

Adding forcasted values to CSV for one day:

```
In [32]: one_day = 86400
         next_unix = dt + one_day
         for i in forcast_set:
             next_date = d.datetime.fromtimestamp(next_unix)
             next_unix += one_day
             df.loc[next_date] = [np.nan for _ in range(len(df.columns)-1)] + [i]

         forcast_val = df['Forecast']
         df['Forecast'].to_csv('forecast.csv',sep=',', encoding='utf-8')

In [34]: def forcast_all():
             for i in range(0,len(df)):
                 last_date_loop= df.iloc[i].name
                 last_date_loop= last_date_loop[:19]
             #    print(last_date_loop)
                 dt=time.mktime(datetime.strptime(last_date_loop, '%Y-%m-%d %H:%M:%S').timetuple

                 #last_unix = last_date.timestamp()
                 one_day = 86400
                 next_unix = dt + one_day

                 for i in forcast_set:
                     next_date = d.datetime.fromtimestamp(next_unix)
                     next_unix += one_day
                     df.loc[next_date] = [np.nan for _ in range(len(df.columns)-1)] + [i]

                 df['Forecast'].to_csv('forcast_all.csv',sep=',', encoding='utf-8')
```

```
In [35]: def test(preds, y_regression):
             print("preds: {}, {}".format(preds.dtype, preds.shape))
             print("y_regression: {}, {}".format(y_regression.dtype, y_regression.shape))

             print('manual result: mse=%f' % np.mean(np.square(y_regression - preds)))

             a = mse(y_regression, preds)
             b = mae(y_regression, preds)
             c = mape(y_regression, preds)
             f = K.function([], [a, b, c])
             #print('backend result: mse={}, mae={}, mape={}'.format(*f([])))

             x = Input(preds.shape[1:])
             m = Model(x, x)
             m.compile(loss='mse', optimizer='rmsprop', metrics=['mae', 'mape'])
             scores = m.evaluate(preds, y_regression, batch_size=32, verbose=0)

             print('\nevaluate result: mse={}, mae={}, mape={}'.format(*scores))

In [36]: #Prediction function to generate predictions for diffrent slice with RandomForestRegres
         def prediction_func(inputVal):
             forecast_val= vm.predict(inputVal)
             return forecast_val

         #predicting each slice of preprocessing with the prediction func
         first_predict = np.array(prediction_func(x_slice_1))
         second_predict = np.array(prediction_func(x_slice_2))
         third_predict = np.array(prediction_func(x_slice_3))
         fourth_predict = np.array(prediction_func(x_slice_4))
         fifth_predict = np.array(prediction_func(x_slice_5))
         sixth_predict = np.array(prediction_func(x_slice_6))
         seventh_predict = np.array(prediction_func(x_slice_7))

         #merging the predictions
         merge_1 = np.concatenate([first_predict,second_predict,third_predict])
         merge_2 = np.concatenate([fourth_predict,fifth_predict,sixth_predict])

         #total merged prediction values
         final_merge = np.concatenate([merge_1,merge_2,seventh_predict])

In [37]: def plot_comparison_graph():
             print('Generating Plot...')
             datetime = []
             for data in full_df['DateTime']:
                 datetime.append(data)
             value = []
             for data in full_df['E_8426_PT02']:
                 value.append(data)
```

```
            dates = [pd.to_datetime(d, infer_datetime_format=True) for d in datetime]
            plt.plot(dates,value,dates,final_merge)
            plt.fill_between(dates, value, final_merge, color='grey', alpha='0.7')
            plt.legend(['E_8426_PT02', 'Forecast'])
            plt.gcf().autofmt_xdate()
            plt.show()

In [38]: def plot_partial_comparison_graph():
            print('Generating Plot...')
            datetime = []
            for data in full_df['DateTime']:
                datetime.append(data)
            value = []
            for data in full_df['E_8426_PT02']:
                value.append(data)

            sliced_value = value[:66777]
            forecast_val = df['Forecast']
            new_forecast_val = forecast_val[-forcast_out:]
            for val in new_forecast_val:
                sliced_value.append(val)

            dates = [pd.to_datetime(d, infer_datetime_format=True) for d in datetime]
            plt.plot(dates,value,dates,sliced_value)
            plt.fill_between(dates, value, sliced_value, color='grey', alpha='0.7')
            plt.legend(['E_8426_PT02', 'Forecast'])
            plt.gcf().autofmt_xdate()
            plt.show()

In [39]: def plot_regression_graph():
            print('\nGenerating plot...')
            fig = plt.figure(figsize=(10,10))
            fig.set_tight_layout(False)
            ax = fig.add_axes([0.1, 0.2, 0.85, 0.75])
            # format of the labels
            hfmt = mdates.DateFormatter('%H:%M')
            ax.xaxis.set_major_formatter(hfmt)
            fig.autofmt_xdate(rotation=90, ha='center')
            df['E_8426_PT02'].plot()
            df['Forecast'].plot()
            plt.legend(loc=4)
            plt.xlabel('Date')
            plt.ylabel('Pressure(E_8426_PT02)')
            plt.xticks(rotation=60)
            plt.show()

In [40]: def plot_difference_all_data_graph():
            print('Generating Plot...')
```

```
datetime = []
for data in full_df['DateTime']:
    datetime.append(data[:19])

value = []
for data in full_df['E_8426_PT02']:
    value.append(data)

date_inv,val_inv= data_difference(final_merge,datetime)

date_inv_all, val_inv_all = data_difference(value, datetime)

final_dates = []
final_dates_all=[]

for timeval in date_inv:
    dts = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(timeval))
    final_dates.append(dts)

for timevalall in date_inv_all:
    dts_all = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(timevalall))
    final_dates_all.append(dts_all)

total_dates= [pd.to_datetime(d, infer_datetime_format=True) for d in final_dates_al
dat_val = [pd.to_datetime(d, infer_datetime_format=True) for d in final_dates]

plt.plot(total_dates,val_inv_all,dat_val,val_inv)
plt.gcf().autofmt_xdate()
plt.title('10 min intervals for both predictions and Real Values with the whole dat
plt.legend(['Real Values','Intervals(Predictions)'])
plt.show()
```

```
In [41]: #forcast_all()
```

Evaluating MSE manually and Keras Evaluation result for test and prediction

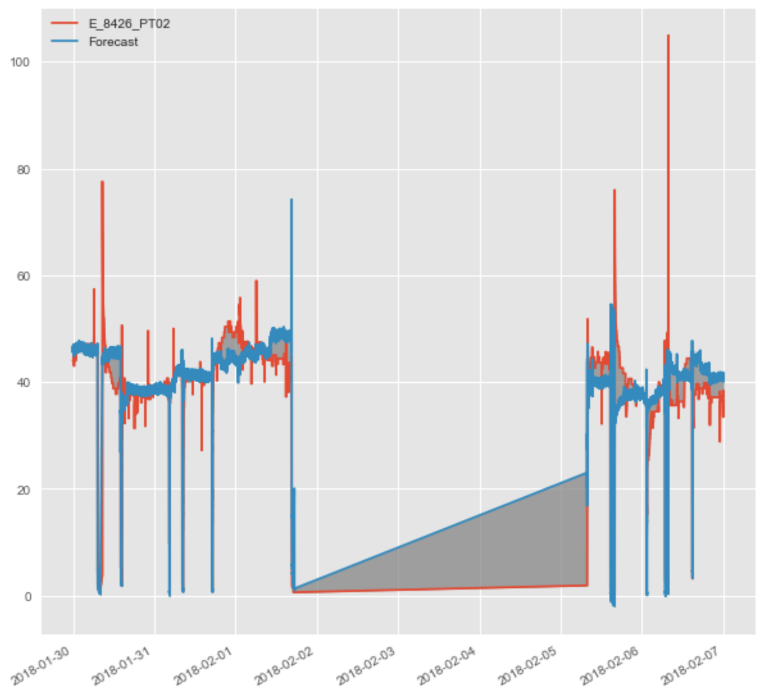
```
In [42]: y_test_shape = y_test.astype(np.float32).reshape((-1, 1))
         pred_test_shape = pred_test_rfg.astype(np.float32).reshape((-1, 1))
         test(pred_test_shape,y_test_shape)
```

```
preds: float32, (26980, 1)
y_regression: float32, (26980, 1)
manual result: mse=60.957672

evaluate result: mse=60.95767021972639, mae=5.263369308019939, mape=124.05776606168104
```

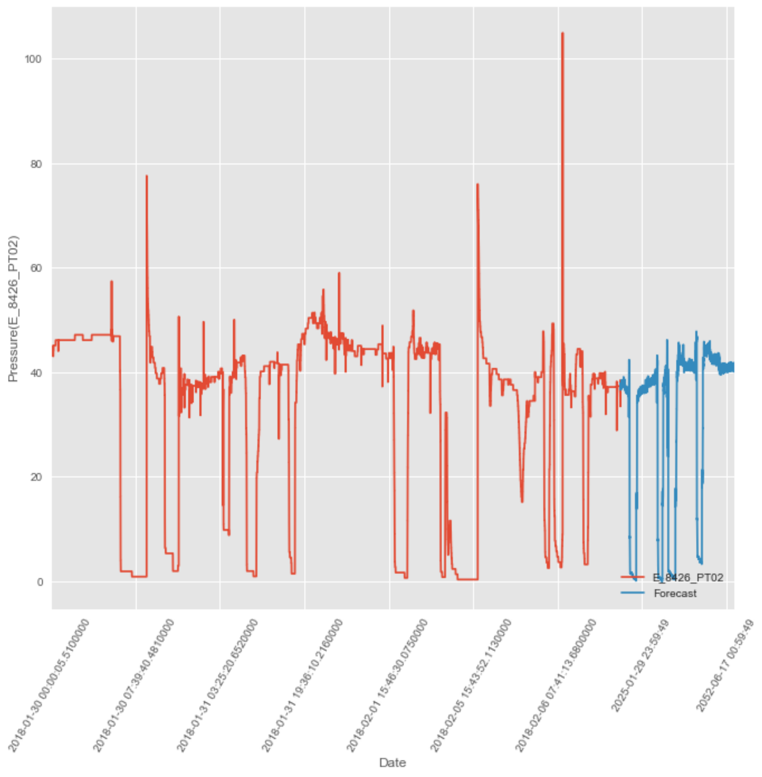
Evaluating MSE manually and Keras Evaluation result for test and forcasted prediction

```
In [43]: y_regression = y_test_slice.astype(np.float32).reshape((-1, 1))
         pred_test_value = forcast_set.astype(np.float32).reshape((-1, 1))
         test(pred_test_value,y_regression)

preds: float32, (13491, 1)
y_regression: float32, (13491, 1)
manual result: mse=541.165100

evaluate result: mse=541.1650903849621, mae=16.290467164062445, mape=2003.677352569957
```

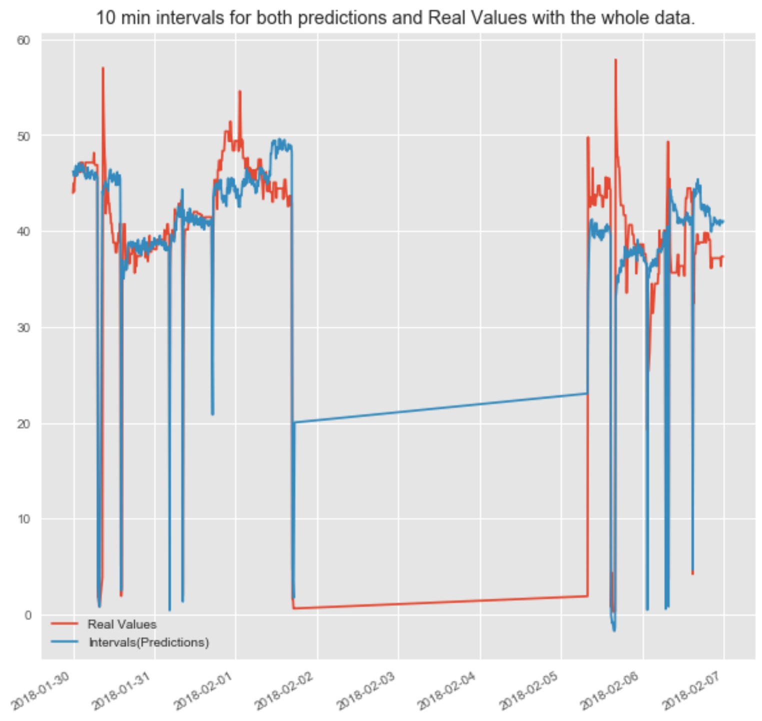
```
In [44]: #generating prediction vs. forecast for whole data set
         plot_comparison_graph()

Generating Plot...
```



```
In [45]: #plotting comparison graph with real and predicted data the portion used for future pre
         # plot_partial_comparison_graph()

In [46]: plot_regression_graph()


Generating plot...
```



```
In [47]: plot_difference_all_data_graph()

Generating Plot...
```


10 min intervals for both predictions and Real Values with the whole data.
60
50
40
30
20
10
0
Real Values
Intervals(Predictions)
2018-01-30
2018-01-31
2018-02-01
2018-02-02
2018-02-03
2018-02-04
2018-02-05
2018-02-06
2018-02-07

```
In [ ]:
```

Bibliography

[1] Ghina Dandachi, Ammar Assoum, Bachar ElHassan, and Fadi Dornaika. Machine learning schemes in augmented reality for features detection. pages 101–105, 05 2015.

[2] CHENGWEI XIAO. Using machine learning for exploratory data analysis and predictive models on large dataset. doi: https://brage.bibsys.no/xmlui/handle/11250/299600.

[3] Billinghurst M. Gamper H. et al. Ajanki, A. An augmented reality interface to contextual information. *Virtual Reality*, 15(161):1–15, 2011. ISSN 1434.

[4] G. A. Giraldi R. Silva, J. C. Oliveira. Introduction to augmented reality. doi: http://lncc.br/jauvane/papers/RelatorioTecnicoLNCC-2503.pdf.

[5] Machine learning is computer science vs. statistics. URL `https://en.wikipedia.org/wiki/Machine_learning`.

[6] Machine Learning. Machine Learning Wikipedia. URL `https://www.quora.com/How-much-of-machine-learning-is-computer-science-vs-statistics-1`.

[7] Borko Furht. *Handbook of Augmented Reality*. Springer-Science + Business Media, Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Glades Road 777, 33431 Boca Raton, Florida, USA, 2011. ISBN 978-1-4614-0063-9.

[8] The Ultimate Guide to Augmented Reality (AR) Technology. URL `http://www.realitytechnologies.com/augmented-reality`.

[9] Igor Kononenko. Machine learning for medical diagnosis: History, state of the art and perspective. doi: https://pdfs.semanticscholar.org/77cf/2d8a174c5a9a6b41c44203695c1d7f83f391.pdf.

[10] Pazzani M.J. Billsus D. Webb, G.I. User modeling and user-adapted interaction. page 11:19, 2001. ISSN 1573-1391.

[11] Ayon Dey. User modeling and user-adapted interaction. 2016.

[12] Taiwo Ayodele. Types of machine learning algorithms. 02 2010.

[13] Types of regression. URL https://www.r-bloggers.com/15-types-of-regression-you-should-know/.

[14] Regression techniques. URL https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/.

[15] Diff regression techniques. URL https://www.listendata.com/2018/03/regression-analysis.html.

[16] Linear models and regression analysis. URL http://home.iitk.ac.in/~shalab/regression/Chapter1-Regression-Introduction.pdf.

[17] Support vector machine wikipedia, . URL https://en.wikipedia.org/wiki/Support_vector_machine.

[18] Support vector machine regression. URL http://kernelsvm.tripod.com/.

[19] Oliver Kramer. Unsupervised k-nearest neighbor regression. 09 2011.

[20] K nearest neighbors - regression. URL http://www.saedsayad.com/k_nearest_neighbors_reg.htm.

[21] Gerard Biau. Analysis of a random forests model. *Journal of Machine Learning Research*, 13(1063-1095), 2012.

[22] Feature selecton wikipedia, . URL https://en.wikipedia.org/wiki/Feature_selection.

[23] Stats Model Documentation . URL http://www.statsmodels.org/stable/index.html.

[24] Evaluating a linear regression model. URL http://www.ritchieng.com/machine-learning-evaluate-linear-regression-model/.

[25] Predictive modeling. URL https://www.mathworks.com/discovery/predictive-modeling.html.

[26] Cross validation. URL https://www.cs.cmu.edu/~schneide/tut5/node42.html.

[27] Scikiit Learn. Model persistence:. URL http://scikit-learn.org/stable/modules/model_persistence.html.

[28] Statistics. Mean Squared Error , . URL http://www.statisticshowto.com/mean-squared-error/.

[29] Statistics. RMSE: Root Mean Square Error , . URL http://www.statisticshowto.com/rmse/.

[30] Wikipedia. Mean absolute error . URL https://en.wikipedia.org/wiki/Mean_absolute_error.

[31] MAPT. computing-mse-and-median-absolute-error. URL https://www.packtpub.com/mapt/book/big_data_and_business_intelligence/9781785282287/10/ch10lvl1sec136/computing-mse-and-median-absolute-error.

[32] Confusion Matrix. Simple Guide to confusion matrix terminology. URL http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/.

[33] Precision and Recall. Calculation of Precision and Recall. URL https://en.wikipedia.org/wiki/Precision_and_recall.

[34] F1 Score. Calculation of F1 Score. URL https://en.wikipedia.org/wiki/F1_score.

[35] Keras. Keras Documentation. URL https://keras.io/.

[36] Keras Git. Keras Git Hub issues. URL https://github.com/keras-team/keras/issues/5140.