INSULIN DOSAGE PREDICTION FOR DIABETIC PATIENTS USING REGRESSION TECHNIQUES

PROJECT REPORT

Submitted

in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

in the faculty of

COMPUTER SCIENCE AND ENGINEERING

By

BATTA RAMA DAKSHNA MURTHY	[R.NO.17021A0537]
PIPPALLA RAVINDRA BABU	[R.NO.17021A0524]
PULIPATI KANAKA DURGA PRASAD	[R.NO.18025A0562]
BALAKA SANTHOSH	[R.NO.17021A0505]
KANCHANAPALLI BALA NAVEEN SAI	[R.NO.17021A0552]

Under the Guidance of

Dr. S. SUREKHA

Assistant Professor

UNIVERSITY COLLEGE OF ENGINEERING (AUTONOMOUS)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

KAKINADA – 533003, A.P., INDIA

2020-2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY COLLEGE OF ENGINEERING (AUTONOMOUS)
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY
KAKINADA – 533003, A.P., INDIA

DECLARATION FROM THE STUDENTS

We hereby declare that the project work described in this thesis, entitled "***Insulin Dosage Prediction For Diabetic Patients Using Regression Techniques***", which is being submitted by us in partial fulfillment of the requirements for the award of degree of ***Bachelor of Technology*** in the faculty of Computer Science And Engineering to the University College of Engineering (Autonomous), Jawaharlal Nehru Technological University Kakinada is the result of investigations carried out by us under the guidance of Dr. S. SUREKHA, Assistant Professor in the Department of Computer Science And Engineering.

The work is original and has not been submitted to any other University or Institute for the award of any degree or diploma.

Place: Kakinada **Signature:**

Date:

BATTA RAMA DAKSHNA MURTHY	[17021A0537]
PIPPALLA RAVINDRA BABU	[17021A0524]
PULIPATI KANAKA DURGA PRASAD	[18025A0562]
BALAKA SANTOSH	[17021A0505]
KANCHANAPALLI BALA NAVEEN SAI	[17021A0552]

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING (AUTONOMOUS)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

KAKINADA – 533003, A.P., INDIA

CERTIFICATE FROM THE SUPERVISOR

This is to certify that the project work entitled "***Insulin Dosage Prediction For Diabetic Patients Using Regression Techniques***", that is being submitted by Batta Rama Dakshna Murthy (17021A0537), Pippalla Ravindra Babu(17021A0524), Pulipati Kanaka Durga Prasad(18025A0562), Balaka Santosh (17021A0505), Kanchanapalli Bala Naveen Sai (17021A0552) in partial fulfillment of the requirements for the award of degree of ***Bachelor of Technology*** in the faculty of Computer Science And Engineering to the Jawaharlal Nehru Technological University Kakinada is a record of Bonafide project work carried out by them under my guidance in the Department of Computer Science And Engineering.

Signature of Supervisor

Dr. S. SUREKHA

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING (AUTONOMOUS)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

KAKINADA – 533003, A.P., INDIA

CERTIFICATE FROM THE HEAD OF DEPARTMENT

This is to certify that the project work entitled "***Insulin Dosage Prediction For Diabetic Patients Using Regression Techniques***", that is being submitted by Batta Rama Dakshna Murthy (17021A0537), Pippalla Ravindra Babu(17021A0524), Pulipati Kanaka Durga Prasad(18025A0562), Balaka Santosh (17021A0505), Kanchanapalli Bala Naveen Sai (17021A0552) in partial fulfillment of the requirements for the award of degree of ***Bachelor of Technology*** in the faculty of Computer Science and Engineering to the Jawaharlal Nehru Technological University Kakinada is a record of Bonafide project work carried out by them at our Department.

Signature of Head of the Department

Dr. D. Haritha

Prof & Head of CSE

ACKNOWLEDGEMENTS

We express our deep gratitude and regards to Dr.S.SUREKHA, internal guide and Assistant Professor, department of Computer Science and Engineering, University College of Engineering (Autonomous), Jawaharlal Nehru Technological University, Kakinada for her encouragement and valuable guidance in bringing shape to this dissertation.

We are grateful to Dr. D. Haritha, Head of the department of Computer Science and Engineering, University College of Engineering (Autonomous), Jawaharlal Nehru Technological University, Kakinada for her encouragement and motivation.

We express special thanks to our beloved Principal, Dr. B. Bala Krishna for providing necessary facilities to carry out the project.

We are thankful to all the Professors and Faculty Members in the department of their teachings and academic support and thanks to Technical Staff and Non-Teaching staff in the department for their support.

BATTA RAMA DAKSHNA MURTHY	**[17021A0537]**
PIPPALLA RAVINDA BABU	**[17021A0524]**
PULIPATI KANAKA DURGA PRASAD	**[18025A0562]**
BALAKA SANTOSH	**[17021A0505]**
KANCHANAPALLI BALA NAVEEN SAI	**[17021A0552]**

ABSTRACT

Diabetes Mellitus, describes a group of metabolic diseases in which the person has high blood glucose (blood sugar), either because insulin production is inadequate, or because the body's cells do not respond properly to insulin, or both. Diabetes is manageable by giving proper dosage of insulin to a patient.

The project automates the diagnosis of diabetes for the patient and it also automatically predicts the amount of insulin to be prescribed to the diabetic patient. Automating the entire process reduces the burden on the healthcare profession and at the same time improves the health of the patient in an effective manner. By using machine learning algorithms, the model can predict whether a person has diabetes or not and also it can predict the amount of insulin need to be taken by the diabetic patient. The model first classifies whether the patient has diabetes or not, then if the patient is suffering from diabetes, it predicts the amount of insulin to be given to that patient using regression. For the classification, PIMA Indian dataset is used. First the data is made into clusters by using Fuzzy C Means clustering then on each cluster the data mining algorithm FP-Growth is applied to find frequent item sets and generate association rules based on the frequent item sets. Based on the generated rules, whether the patient has diabetes or not is classified. The Performance of Association Rule Based Classifier is compared with the conventional classifiers KNN, Decision trees, SVM. If a patient has diabetes, the amount of insulin to be taken is predicted using most popular regression algorithms Linear Regression and Ridge Regression. The Dataset used for regression is UCI Diabetes Dataset.

The performance of the classifiers is evaluated based on Accuracy. The performance of Regressor is evaluated based on Root Mean Squared Error.

Table of Contents

LIST OF FIGURES

CHAPTER – 1
INTRODUCTION

1.1 Introduction

Diabetes Mellitus or commonly known as diabetes is a chronic disease caused by inheritance and/or acquired deficiency in production of insulin by the pancreas. Such a deficiency results in increased concentrations of glucose in the blood, which in turn can damage many of the body's organs, in particular, the blood vessels and nerves. Especially in early stages, it is often mentioned as “challenging to diagnose”.

Diabetes is due to either the pancreas not producing enough insulin, or the cells of the body not responding properly to the insulin produced.

AI is the ability of computer algorithms to approximate conclusions based solely on input data. What distinguishes AI technology from traditional technologies in health care is the ability to gather data, process it and give a well-defined output to the end-user. To gain useful insights and predictions, machine learning models must be trained using extensive amounts of input data.

The primary aim of health-related AI applications is to analyze relationships between prevention or treatment techniques and patient outcomes. AI programs are applied to practices such as diagnosis process, treatment protocol development, drug development, personalized medicine, and patient monitoring and care. AI algorithms can also be used to analyze large amounts of data through electronic health records for disease prevention and diagnosis

The objective of adopting AI in the healthcare sector should be to build trust amongst the patients, deliver better data management, and greater reliability on advanced tools for seamless outcomes. Apt data can be defined as data or information that can be represented to get apt results. AI engines are trained to satisfy the requirements of dealing with the giant datasets along with the smaller ones.

So, both the larger and the smaller healthcare firms can adopt AI for building better trust amongst the patients while educating them to step into an advanced healthcare area. Also, the primary motive of utilizing AI in healthcare is to impart the right knowledge to the patients and healthcare service providers while ensuring that the patients are in right and safe hands.

Hospital goals for the patient with diabetes focus on the management of blood glucose levels to prevent glycemic excursions. Insulin alone is the standard treatment for inpatient diabetes management. Diabetes can be predicted using Machine Learning Algorithms and even the amount of insulin to be given to a diabetic patient can be predicted using Machine Learning Algorithms.

1.2 Introduction to Machine Learning

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

1.2.1 Types of Machine Learning

Machine learning can be classified into 3 types of algorithms.

1) Supervised Learning
2) Unsupervised Learning
3) Reinforcement Learning

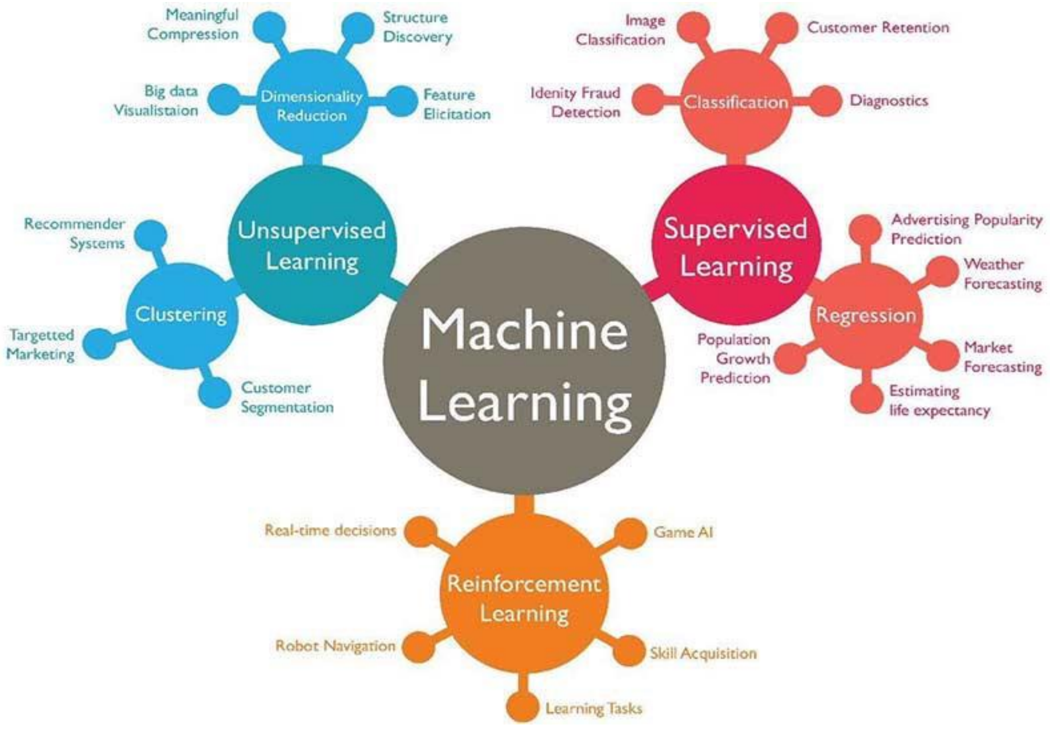


Figure -1.1 Types of Machine Learning

1.2.1.1 Supervised Learning

In Supervised learning, an AI system is presented with data which is labeled, which means that each data tagged with the correct label.

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

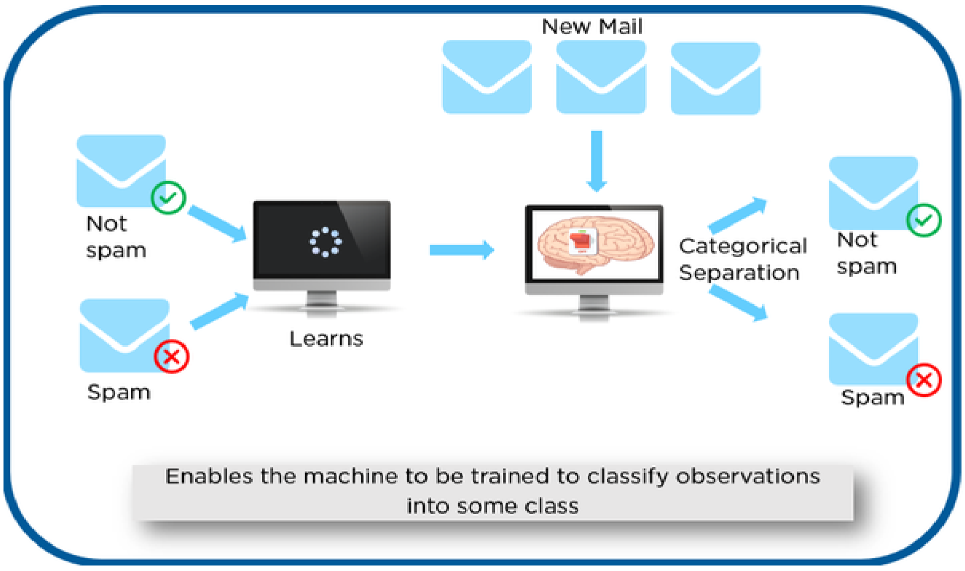

New Mail
Not spam
Spam
Learns
Categorical Separation
Not spam
Spam
Enables the machine to be trained to classify observations into some class

Figure -1.2 Supervised Learning

Types of Supervised learning

- **Classification**: A classification problem is used when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”. In classification we have different algorithms like KNN, Decision Trees, Random Forests, SVM etc.,..

- **Regression**: A regression problem is used when the output variable is a real value, such as “dollars” or “weight”. We have regression algorithms like linear regression, ridge regression, lasso regression etc.,.

1.2.1.2 Unsupervised Learning

In unsupervised learning, an AI system is presented with unlabeled, uncategorized data and the system’s algorithms act on the data without prior training. The output is dependent upon the coded algorithms. Subjecting a system to unsupervised learning is one way of testing AI.

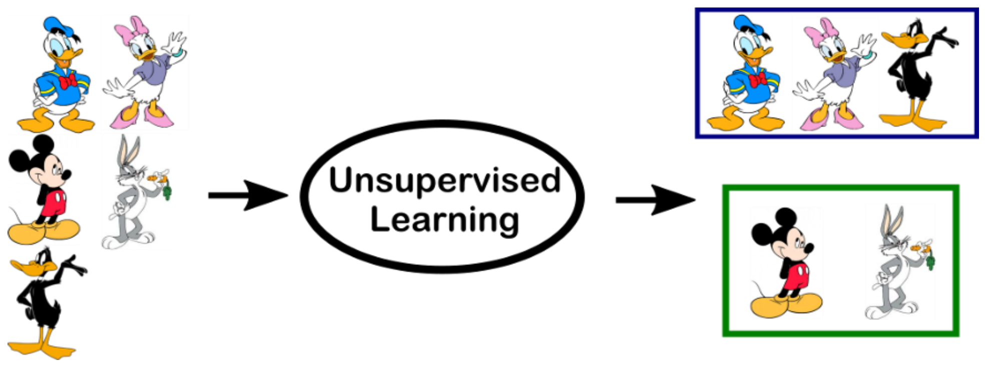


Figure -1.3 Unsupervised Learning

Types of Unsupervised learning

- **Clustering**: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- **Association**: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

Association Rule Mining

Data mining techniques are also used to extract useful information to generate rules. Association rule mining is an important branch to determine the patterns and frequent items used in the dataset. It contains two parts:

1) determine the frequent item set

2) generate rules

Association rule mining plays an important role in medical as well as in commercial data analysis to detect and characterize interesting and important patterns. There are several methods to generate rules from data using association rule mining algorithms such as the Apriori algorithm, Tertius and predictive Apriori algorithms. Mostly, association rule-based algorithms are linked with Apriori, which make it a state-of-the-art algorithm. Apriori works as an iterative method to identify the frequent item set in a given dataset, and to generate important rules from it. To determine the association between two item sets X and Y, there is a need to set the minimum support of that fraction of transactions which contains both X and Y called min support. The other important task is to set the minimum confidence that measures how often items in Y appear in transactions that contain X, known as min confidence, to determine frequent item sets.

Clustering

Clustering or cluster analysis is a machine learning technique, which groups the un labelled dataset. It can be defined as "A way **of grouping** the data points into different clusters, consisting of similar data points".

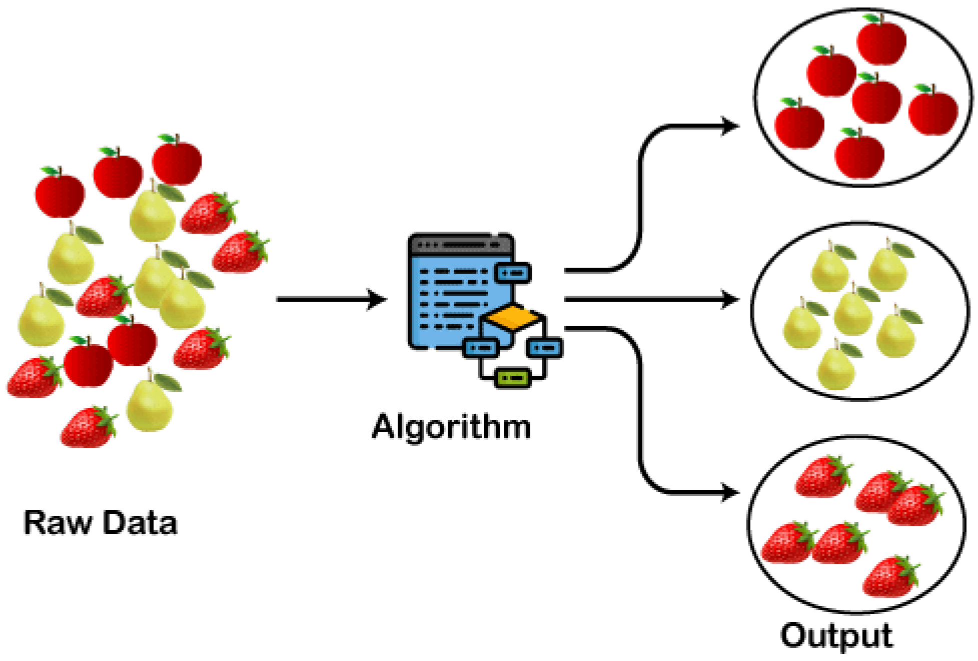


Figure-1.4 Clustering

1.3 About the Disease

Diabetes Mellitus or commonly known as diabetes is a chronic disease caused by inheritance and/or acquired deficiency in production of insulin by the pancreas. Such a deficiency results in increased concentrations of glucose in the blood, which in turn can damage many of the body's organs, in particular, the blood vessels and nerves. Especially in early stages, it is often mentioned as “challenging to diagnose”.

Diabetes is due to either the pancreas not producing enough insulin, or the cells of the body not responding properly to the insulin produced. There are three main types of diabetes mellitus:

- Type 1 diabetes results from failure of the pancreas to produce enough insulin due to loss of beta cells. This form was previously referred to as "insulin-dependent diabetes mellitus" (IDDM) or "juvenile diabetes". The loss of beta cells is caused by an autoimmune response. The cause of this autoimmune response is unknown.

- Type 2 diabetes begins with insulin resistance, a condition in which cells fail to respond to insulin properly. As the disease progresses, a lack of insulin may also develop. This form was previously referred to as "non-insulin-dependent diabetes mellitus" (NIDDM) or "adult-onset diabetes". The most common cause is a combination of excessive body weight and insufficient exercise.

- Gestational diabetes is the third main form, and occurs when pregnant women without a previous history of diabetes develop high blood sugar levels.

World Health Organization (WHO) declares diabetes as a serious and costly disease. Diabetes is a major cause of blindness, kidney failure, heart attacks, stroke, and lower limb amputation in the world. In 2016, an estimated 1.6 million deaths were directly caused by diabetes globally. Another 2.2 million deaths were attributable to high blood glucose in 2012.

Hospital goals for the patient with diabetes focus on the management of blood glucose levels to prevent glycemic excursions. Although there are many medications used for outpatient blood sugar control, especially in T2DM, insulin alone is the standard treatment for inpatient diabetes management.

Diabetes can be predicted using Machine Learning Algorithms and even the amount of insulin to be given to a diabetic patient can be predicted using Machine Learning Algorithms.

1.4 Organization of the Thesis

The thesis has Eight chapters

Chapter 1: Introduction: In this chapter, a brief overview of the project has been given here.

Chapter 2: Literature Review: In this chapter, we discuss about the project domain and the detailed description of existing systems by analysis the literature survey of the existing techniques.

Chapter 3: System Analysis: In this chapter, we discuss about the system analysis as proposed work.

Chapter 4: System Requirements: In this chapter, we discuss about the system requirements used in it.

Chapter 5: System Design: In this chapter, we discuss about the design diagrams of system.

Chapter 6: Implementation And Result Analysis: In this chapter, we discuss about the source code we written in it.

Chapter 7: Evaluation: In this chapter, we discuss about the evaluation of performance of different Algorithms.

Chapter 8: Conclusion And Future Scope: This chapter mainly consists of conclusion for the project and the future scope of the project.

CHAPTER – 2
LITERATURE REVIEW

2.1 Related Work

2.1.1 Classification of Diabetes Mellitus Using Machine Learning Techniques

Author: Amit Kumar Dewangan and Pragati Agrawal

In this work[1], Data mining techniques are used like Multilayer Perceptron, and the Bayesian Net classification techniques and ensemble them for classification of diabetes data: MLP is a development from the simple perceptron in which extra hidden layers (additional to the input and output layers, not connected externally) are added. In this process, more than one hidden layer can be used. The network topology is constrained to be feed forward, i.e., loop-free. Generally, connections are allowed from the input layer to the first (and only possible) hidden layer, from the first hidden layer to the second and so on, until the last hidden layer to the output layer. In this experiment, the hybrid C4.5 and RF, similarly hybrid MLP and Bayes Net classification model to develop a robust model for classification of diabetes data. It is not necessary to improve classification accuracy in each partition. In case of C4.5+RF gives 79.31% of accuracy which is higher than its individuals model. Similarly, MLP+BayesNet, gives 81.89% of accuracy which is higher than its individuals model. In case of both ensemble model 85-15% training-testing partitions play important role for classification of diabetes data. This proposed MLP + Bayes Net gives 81.89% of accuracy which is a robust model for classification of data.

2.1.2 A model for early prediction of diabetes

Author: Talha Mahboob Alam, Muhammad Atif Iqbal

In this work[2], diabetes is predicted using significant attributes, and the relationship of the differing attributes is also characterized. Various tools are used to determine significant attribute selection, and for clustering, prediction, and association rule mining for diabetes. Significant attributes selection was done via the principal component analysis method. The findings indicate a strong association of diabetes with body mass index (BMI) and with glucose level, which was extracted via the Apriori method. Artificial neural network (ANN), random forest (RF) and K-means clustering techniques were implemented for the prediction of diabetes. The ANN technique provided a best accuracy of 75.7%, and may be useful to assist medical professionals with treatment decisions.

2.1.3 Predictive modelling and analytics for diabetes using a machine learning approach

Authors: Harleen Kaur, Vinita Kumari

In the work[3], machine learning techniques are used for Pima Indian diabetes dataset to develop trends and detect patterns with risk factors using R data manipulation tool. To classify the patients into diabetic and non-diabetic five different predictive models were developed and analyzed using *R* data manipulation tool. For this purpose, supervised machine learning algorithms are used namely linear kernel support vector machine (SVM-linear), radial basis function (RBF) kernel support vector machine, *k*-nearest neighbor (*k*-NN), artificial neural network (ANN) and multifactor dimensionality reduction (MDR). The experimental results suggested that all the models achieved good results; SVM-linear model provides best accuracy of 0.89 and precision of 0.88 for prediction of diabetes as compared to other models used. On the other hand, *k*-NN model provided best recall and F1 score of 0.90 and 0.88. Further it can be seen that AUC value of SVM- linear and *k*-NN model is 0.90 and 0.92 respectively. Such a high value of AUC indicates that both SVM- linear and *k*-NN are optimal classifiers for diabetic dataset.

2.1.4 Classification of Diabetes Disease Using Support Vector Machine

Authors: V. Anuja Kumari, R. Chitra

In this work[4], a design of classifiers used for the detection of Diabetes disease with optimal cost and better performance is the need of the time. The Pima Indian diabetic database at the UCI machine learning laboratory has become a standard for testing data mining algorithms to see their prediction accuracy in diabetes data classification. The proposed method uses Support Vector Machine (SVM), a machine learning method as the classifier for diagnosis of diabetes. The machine learning method focuses on classifying diabetes disease from high dimensional medical dataset. The choice of best value of parameters for particular kernel is critical for a given amount of data SVM approach can be successfully used to detect a common disease with simple clinical measurements, without laboratory tests. The experimental results obtained show that a support vector machine can be successfully used for diagnosing diabetes disease.

2.1.5 Classification of Diabetic Patient Data Using Machine Learning Techniques

Authors: Pankaj Pratap Singh, Shitala Prasad, Bhaskar Jyoti Das, Upasana Poddar, Dibarun Roy Choudhury.

This work[5], aims at finding solutions to diagnose the disease by analyzing the patterns found in the dataset through data mining. In addition, the neural network approach is also used for classifying the existing diabetic patient data for predicting the patient's disease based on the trained data that can lead to finding the different levels of diabetes in the patients. It is also compared with the association rule mining-based approach for classification of data to validate the correctly classified cases. The results shows that both rule-based model and neural network give almost same accuracy which shows the utility of these DM approaches. Thus, both approaches are an efficient method for classification of the existing preprocessed medical data. This research analysis also predicts the readmission rates of each diabetes class patients.

2.1.6 Early detection of type 2 diabetes mellitus using machine learning-based prediction models

Authors: Leon Kopitar, Primoz Kocbek, Leona Cilar, Aziz Sheikh, Gregor Stiglic

This work[6], compares machine learning-based prediction models (i.e.; Glmnet, RF, XGBoost, LightGBM) to commonly used regression models for prediction of undiagnosed T2DM. The performance in prediction of fasting plasma glucose level was measured using 100 bootstrap iterations in different subsets of data simulating new incoming data in 6-month batches. With 6 months of data available, a simple regression model performed with the lowest average RMSE of 0.838, followed by RF (0.842), LightGBM (0.846), Glmnet (0.859) and XGBoost (0.881). When more data were added, Glmnet improved with the highest rate (+ 3.4%). The highest level of variable selection stability over time was observed with LightGBM models. These results show no clinically relevant improvement when more sophisticated prediction models were used. Since higher stability of selected variables over time contributes to simpler interpretation of the models, interpretability and model calibration should also be considered in development of clinical prediction models.

2.1.7 Diabetes Mellitus Affected Patients Classification and Diagnosis through Machine Learning Techniques

Authors: Francesco Mercaldo, Vittoria Nardone, Antonella Santone

In this work[7], the proposed model is able to classify patients affected by diabetes using a set of characteristics selected according to World Health Organization criteria. Evaluating real-world data using state of the art machine learning algorithms, and obtaining a precision value equal to 0.770 and a recall equal to 0.775 using the Hoeffding Tree algorithm. Training the model using six different classification algorithms, we obtain a precision equal to 0.757 and the recall of 0.762 after the best feature's selection step.

2.1.8 A comprehensive exploration of the machine learning techniques for diabetes identification

Authors: Sidong Wei, Xuejiao Zhao, Chunyan Miao

In this work[8], a comprehensive exploration of the most popular techniques (e.g.; DNN (Deep Neural Network), SVM (Support Vector Machine), etc.) used to identify diabetes and data preprocessing methods. Basically, these techniques are examined by the accuracy of cross-validation on the Pima Indian data set. Then compare the accuracy of each classifier over several ways of data preprocessors and modify the parameters to improve their accuracy. The best technique has 77.86% accuracy using 10-fold cross-validation and also analyzes the relevance between each feature with the classification result.

2.1.9 A Method for Classification Using Machine Learning Technique for Diabetes

Authors: Aishwarya.R, Gayathri.P, N.Jaisankar

In this work[9], a system is proposed which uses Principal Component Analysis (PCA) for preprocessing techniques. PCA is one of the most promising techniques for preprocessing. Classification of diabetes disease, with non-diabetes and diabetes is done using SVM. The diagnosing of diabetes using SVM with preprocessing PCA shows better accuracy that the existing system. Accuracy of existing system has been 89.07 percentages and proposed system has achieved output of 95 %. Benchmark data is obtained from Pima database. When compare with previous work which has been done without preprocessing the system. Preprocessing has played a key role in classification of diabetes.

2.1.10 Diabetes Disease Prediction Using Machine Learning on Big Data of Healthcare

Authors: Ayman Mir; Sudhir N. Dhage

In this work[10], the main aim is to build a classifier model using WEKA tool to predict diabetes disease by employing Naive Bayes, Support Vector Machine, Random Forest and Simple CART algorithm. The research hopes to recommend the best algorithm based on efficient performance result for the prediction of diabetes disease. Experimental results of each algorithm used on the dataset was evaluated. It is observed that Support Vector Machine performed best in prediction of the disease having maximum accuracy.

CHAPTER – 3
SYSTEM ANALYSIS

3.1 Workflow Process

This Project is divided into 2 phases.

In phase-1 the model predicts Whether a person has diabetes or not using Association Rule Based Classifier and conventional classifiers.

The Workflow of this phase is shown in figure-3.1,

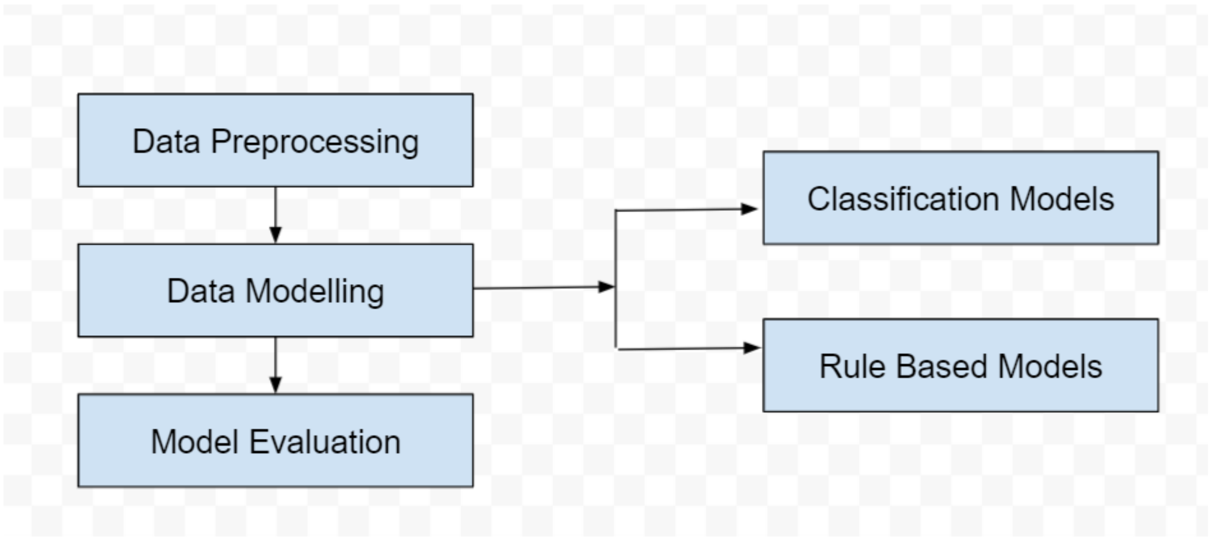


Figure-3.1 Proposed Workflow for Classification

In Phase-2 the model predicts the Insulin dosage required for the diabetic patient using most popular Regression Algorithms, Linear Regression and Ridge Regression.

The workflow of this phase is shown in figure-3.2,

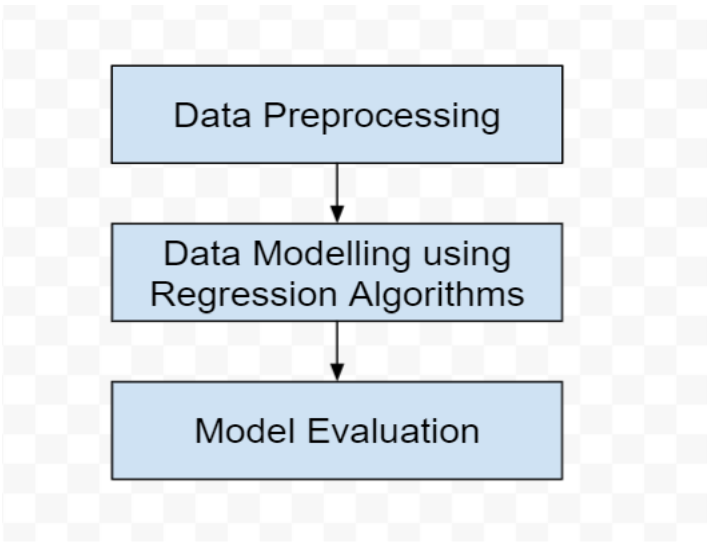


Figure-3.2 Proposed Workflow for Regression

3.2 Workflow for Classification model

3.2.1 Dataset

Pima Indians Diabetes dataset is used in this model. This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage. The datasets consist of several medical predictor variables and one target variable, outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

3.2.2 Data preprocessing

3.2.2.1 Description of the features

Accurate data of these features is important for building good machine learning models. Now before working on any dataset, we must know about the data and what each feature mean.
Now let’s look at each feature of dataset.

1)Pregnancies-Number of times pregnant

2)Glucose-Plasma glucose concentration 2 hours in an oral glucose tolerance test

3)BloodPressure-Diastolic blood pressure (mm Hg)

4)SkinThickness-Triceps skin fold thickness (mm)

5)Insulin-2-Hour serum insulin (mu U/ml)

6)BMI-Body mass index (weight in kg/(height in m)^2)

7)DiabetesPedigreeFunction-Diabetes pedigree function

8)Age-Age (years)

9)Outcome-Class variable (0 or 1) 268 of 768 are 1, the others are 0

3.2.2.2 Handling Missing Values

In real world data, there are some instances where a particular element is absent because of various reasons, such as, corrupt data, failure to load the information, or incomplete extraction.

We have different ways to handle those kinds of missing values,

1)Filling with mean/median/mode: This strategy can be applied on a feature which has numeric data like the age of a person or the ticket fare. We can calculate the mean, median or mode of the feature and replace it with the missing values.

2)Deleting Rows: This method commonly used to handle the null values. Here, we either delete a particular row if it has a null value for a particular feature and a particular column if it has more than 70-75% of missing values.

3)Assigning a Unique Category: A categorical feature will have a definite number of possibilities, such as gender, for example. Since they have a definite number of classes, we can assign another class for the missing values.

4)Predicting the Missing Values: Using the features which do not have missing values, we can predict the nulls with the help of a machine learning algorithm. This method may result in better accuracy, unless a missing value is expected to have a very high variance.

In this project if a record contains a missing value, we removed that record because in medical field accurate data is important and assumptions about data will result in error in predictions which might cause severe issues for the patients.

3.2.2.3 Detecting and Removing the Outliers

An Outlier is a data-item/object that deviates significantly from the rest of the (so-called normal) objects. They can be caused by measurement or execution errors. The analysis for outlier detection is referred to as outlier mining.

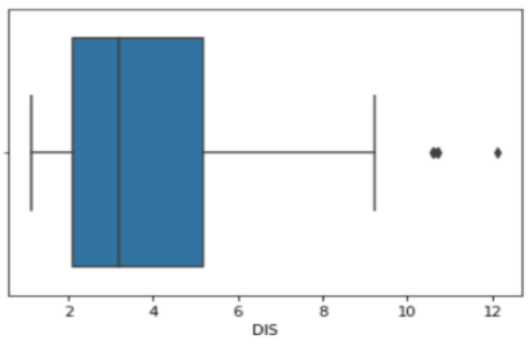
Outliers can be detected using visualization, implementing mathematical formulas on the dataset, or using the statistical approach. Different ways to find the outliers is as follows,

1. Visualization Using Box Plot: It captures the summary of the data effectively and efficiently with only a simple box and whiskers. Boxplot summarizes a sample data using 25th, 50th, and 75th percentiles. One can just get insights (quartiles, median, and outliers) into the dataset by just looking at its boxplot.

The Box Plot is shown in Figure-3.4,

```
# Box Plot
import seaborn as sns
sns.boxplot(df_boston['DIS'])
```

Figure-3.3 Box Plot Code

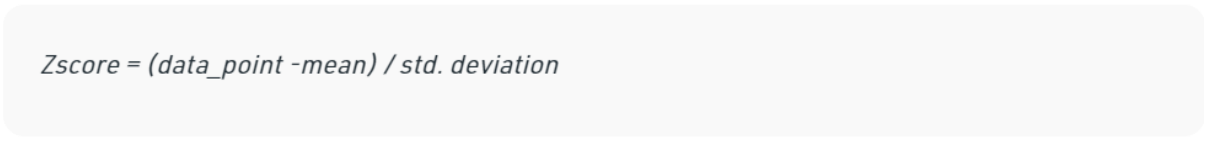


Boxplot- DIS column

Figure-3.4 Box Plot

2. Z-score: Z- Score is also called a standard score. This value/score helps to understand that how far is the data point from the mean. And

after setting up a threshold value one can utilize z score values of data points to define the outliers.
The formula for Z-Score is as follows,



```
Zscore = (data_point -mean) / std. deviation
```

Figure-3.5 Z-score

3.2.2.4 Normalization

Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information. For machine learning, every dataset does not require normalization. It is required only when features have different ranges.

For example, assume your input dataset contains one column with values ranging from 0 to 1, and another column with values ranging from 10,000 to 100,000. The great difference in the scale of the numbers could cause problems when you attempt to combine the values as features during modelling. Normalization avoids these problems by creating new values that maintain the general distribution and ratios in the source data while keeping values within a scale applied across all numeric columns used in the model.

Standard Scaler: In this project we used Standard scalar, The idea behind Standard Scaler is that it will transform your data such that its distribution will have a mean value 0 and standard deviation of 1.

In case of multivariate data, this is done feature-wise (in other words independently for each column of the data).

Given the distribution of the data, each value in the dataset will have the mean value subtracted, and then divided by the standard deviation of the whole dataset (or feature in the multivariate case).

Core of standard scalar method: The main idea is to **normalize or standardize** i.e., μ = 0 and σ = 1 your features/variables/columns of X, individually, before applying any machine learning model.

3.2.2.5 Feature Selection

Feature selection is the process of reducing the number of input variables when developing a predictive model. It is desirable to reduce the number of input variables to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model.

The Feature selection methods we used in this project are as follows,

1. Removing the features which has very low variance: If the **variance** is **low** or close **to** zero, then a **feature** is approximately constant and will not improve the performance of the model. In that case, it should be removed. ... Hence, the **feature** will not help the performance of the model **to** predict the target. Therefore, it should be removed.

2. Finding important features using Random Forest Classifier: We can use the Random Forest algorithm for feature importance implemented in scikit-learn as the *RandomForestRegressor* and *RandomForestClassifier* classes. After being fit, the model provides a *feature_importances_* property that can be accessed to retrieve the relative importance scores for each input feature. This approach can also be used with the bagging and extra trees algorithms.

when training a tree, it is possible to compute how much each feature decreases the impurity. The more a feature decreases the impurity, the more important the feature is. In random forests, the impurity decrease from each feature can be averaged across trees to determine the final importance of the variable.

3.2.2.6 Data Preprocessing for Rule based Classifier:

Feature binning: Binning or discretization is used for the transformation of a continuous or numerical variable into a categorical feature. Binning of continuous variable introduces non-linearity and tends to improve the performance of the model. It can be also used to identify missing values or outliers. We used Unsupervised binning in this project it is a category of binning that transforms a numerical or continuous variable into categorical bins **without considering the target class label into account**.

Feature Encoding: Feature Encoding is used for the transformation of a categorical feature into a numerical variable. Most of the ML algorithms cannot handle categorical variables and hence it is important to do feature encoding.

We used the get_dummies to do feature Encoding. get_dummies implement the one hot encoding. One hot encoding technique splits the category each to a column. It creates k different columns each for a category and replaces one column with 1 rest of the columns is 0.

Then we convert the one hot encoding into binary format. Now, the data is suitable for rule mining algorithms.

3.2.3 Data Modelling

The process of modeling means training a machine learning algorithm to predict the labels from the features, tuning it for the business need, and validating it on holdout data. The output from modeling is a trained model that can be used for inference, making predictions on new data points. The objective of machine learning is not a model that does well on training data, but one that demonstrates it satisfies the business need and can be deployed on live data.

significantly depends on the value of hyperparameters. Note that there is no way to know in advance the best values for hyperparameters so ideally, we need to try all possible values to know the optimal values. Doing this manually could take a considerable amount of time and resources and thus we use GridSearchCV to automate the tuning of hyperparameters.

Fuzzy C Means: Fuzzy logic principles can be used to cluster multidimensional data, assigning each point a *membership* in each cluster center from 0 to 100 percent. This can be very powerful compared to traditional hard-threshold clustering where every point is assigned a crisp, exact label. This algorithm works by assigning membership to each data point corresponding to each cluster center on the basis of distance between the cluster center and the data point. More the data is near to the cluster center the more is its membership towards the particular cluster center. Clearly, summation of membership of each data point should be equal to one. After each iteration membership and cluster centers are updated.

<u>Algorithmic steps for Fuzzy c-means clustering:</u>

Let X = {x_1, x_2, x_3 …, x_n} be the set of data points and V = {v_1, v_2, v_3 …, v_c} be the set of centers.

1) Randomly select *'c'* cluster centers.

2) Calculate the fuzzy membership 'μ_{ij}' using:

$$\mu_{ij} = 1 / \sum_{k=1}^{c} (d_{ij} / d_{ik})^{(2/m-1)}$$

Figure – 3.6 Fuzzy Membership Formula

3) Compute the fuzzy centers 'v_j' using:

$$v_j = \left(\sum_{i=1}^{n} (\mu_{ij})^m x_i\right) / \left(\sum_{i=1}^{n} (\mu_{ij})^m\right), \forall j = 1, 2, ...c$$

Figure – 3.7 Fuzzy Centers Formula

4) Repeat step 2) and 3) until the minimum *'J'* value is achieved or $||U^{(k+1)} - U^{(k)}|| < \beta$.

where,

'k' is the iteration step.

'β' is the termination criterion between [0, 1].

'$U = (\mu_{ij})_{n*c}$' is the fuzzy membership matrix.

'J' is the objective function.

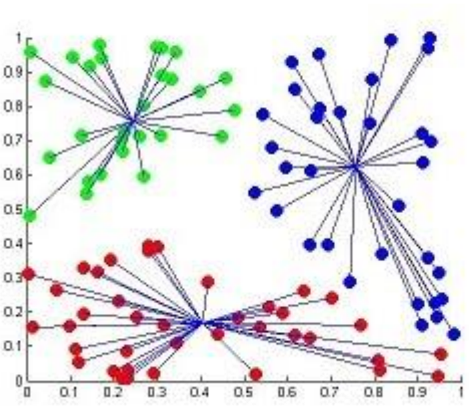


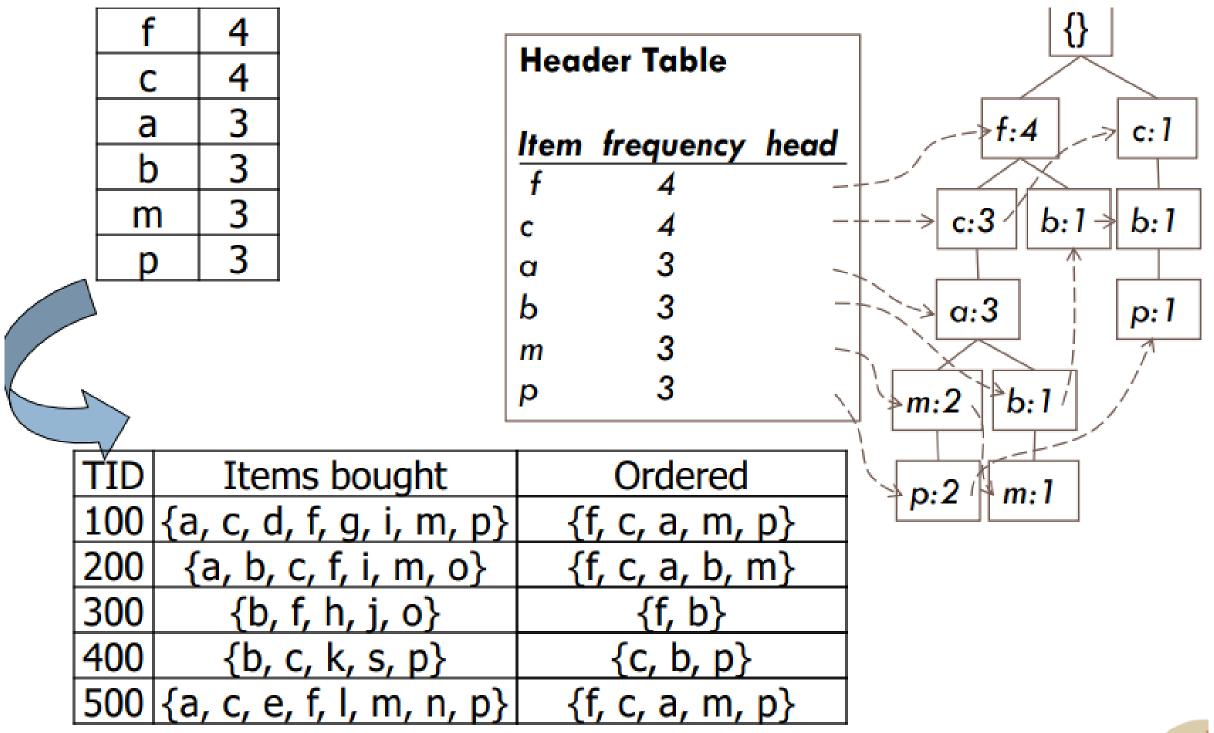
Figure – 3.8 Fuzzy C Means Results

FP Growth: FP tree is the core concept of the whole FP Growth algorithm. Briefly speaking, the FP tree is the compressed representation of the itemset database. The tree structure not only reserves the itemset in DB but also keeps track of the association between item sets. The tree is constructed by taking each itemset and mapping it to a path in the tree one at a time. The whole idea behind this construction is that, more frequently occurring items will have better chances of sharing items. We then mine the tree recursively to get the frequent pattern. Pattern growth, the name of the algorithm, is achieved by concatenating the frequent pattern generated from the conditional FP trees.

Stage 1: FP tree construction

Step 1: Cleaning and sorting

For each transaction, we first remove the items that are below the minimum support. Then, we sort the items in frequency support descending order.



f	4
c	4
a	3
b	3
m	3
p	3

Header Table

Item	frequency	head
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	

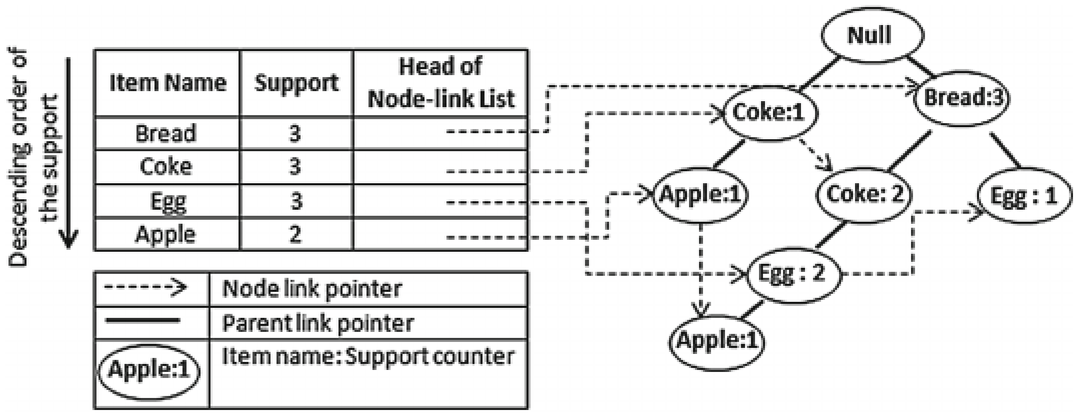
TID	Items bought	Ordered
100	{a, c, d, f, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, i, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, c, e, f, l, m, n, p}	{f, c, a, m, p}

Figure – 3.9 Cleaning and Sorting in Fp-Growth

Step 2: Construct FP tree, header table with cleaned itemsets

Loop through the cleaned itemsets, map it to the tree one at a time. If any item already exists in the branch, they share the same node and the count is incremented. Else, the item will be on the new branch created.

The header table is also constructed during this procedure. There's a linked list for each unique item in the transactions. With the linked list, we can find the occurrence of the item on the tree in a short period of time without traversing the tree.



Item Name	Support	Head of Node-link List
Bread	3	
Coke	3	
Egg	3	
Apple	2	

----->	Node link pointer
———	Parent link pointer
Apple:1	Item name: Support counter

Figure – 3.10 FP-Tree and Header Table

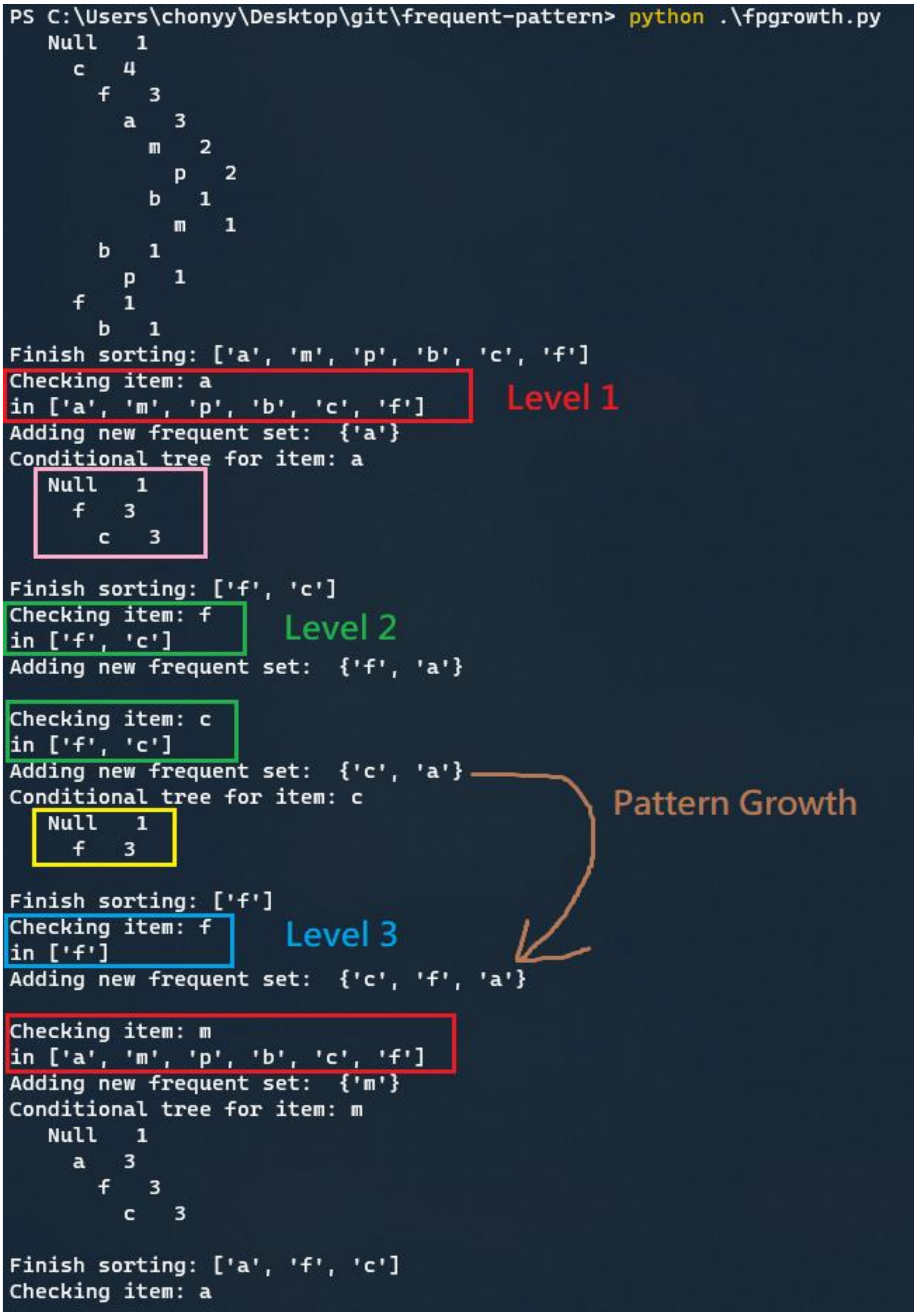
Stage 2: Mine the main tree and conditional FP trees

Step 1: Divide the main FP tree into conditional FP trees

Staring from each frequent 1-pattern, we create conditional pattern bases with the set of prefixes in the FP tree. Then, we use those pattern bases to construct conditional FP trees with the exact same method in Stage 1.

Step 2: Mine each conditional trees recursively

The frequent patterns are generated from the conditional FP Trees. One conditional FP tree is created for one frequent pattern. The recursive function we used to mine the conditional trees is close to depth-first search. It makes sure that there is no more trees can be constructed with the remaining items before moving on.



```
PS C:\Users\chonyy\Desktop\git\frequent-pattern> python .\fpgrowth.py
   Null   1
     c   4
       f   3
         a   3
           m   2
             p   2
           b   1
             m   1
       b   1
         p   1
     f   1
       b   1
Finish sorting: ['a', 'm', 'p', 'b', 'c', 'f']
Checking item: a
in ['a', 'm', 'p', 'b', 'c', 'f']
Adding new frequent set:  {'a'}
Conditional tree for item: a
   Null   1
     f   3
       c   3

Finish sorting: ['f', 'c']
Checking item: f
in ['f', 'c']
Adding new frequent set:  {'f', 'a'}

Checking item: c
in ['f', 'c']
Adding new frequent set:  {'c', 'a'}
Conditional tree for item: c
   Null   1
     f   3

Finish sorting: ['f']
Checking item: f
in ['f']
Adding new frequent set:  {'c', 'f', 'a'}

Checking item: m
in ['a', 'm', 'p', 'b', 'c', 'f']
Adding new frequent set:  {'m'}
Conditional tree for item: m
   Null   1
     a   3
       f   3
         c   3

Finish sorting: ['a', 'f', 'c']
Checking item: a
```

Figure – 3.11 Mining FP-Tree

Same level in same color. The algorithm workflow is listed below

1. Checking the first 1-frequent pattern ‘a’
2. Get the conditional FP tree for ‘a’ in pink
3. Mine the pink tree and go deeper to level 2
4. Check ‘f’ on the pink tree and found no more tree can be constructed
5. Check ‘c’ on the pink tree
6. Mine the yellow tree and go deeper to level 3

In this project Algorithms Used for predicting diabetes using Classification are as follows,

K-Nearest Neighbors: K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. K-nearest neighbors (KNN) algorithm uses ‘feature similarity’ to predict the values of new data points which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification.

Algorithm:

Input: Dtrain, K, xq (query point)

Output: yq (class to which out query point belongs to i.e. +ve or -ve)

- k_pts=[]

```
• for each xi in Dtrain:
    o compute di (xi, xq)
• sort di's in non-decreasing order
• append k xi's in k_pts
• cnt_pts=0, cnt_neg=0
• for each xi in k_pts:
    o if yi ==+ve:
        ▪ cnt_pts+=1
    o else:
        ▪ cnt_neg+=1
• if cnt_pst>cnt_neg:
    o yq= +ve
• else:
    o ya= -ve
```

Decision Trees: Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A decision tree is a flowchart-like structure in which each internal node represents a test on a feature, each leaf node represents a class label (decision taken after computing all features) and branches represent conjunctions of features that lead to those class labels. The paths from root to leaf represent classification rules. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

Algorithm: Generate decision tree. Generate a decision tree from the training tuples of data partition, D.

Input:

- Data partition, D, which is a set of training tuples and their associated class labels;

- attribute_list, the set of candidate attributes
- Attribute_selection method, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a splitting-attribute and possibly, either a split-point or splitting subset.

Output: A decision tree.

Method:

```
• create a node N;
• if tuples in D are all of the same class, C, then
    o return N as a leaf node labeled with the class C
• if attribute_list is empty then
    o return N as a leaf node labeled with the majority class in D;
• apply Attribute_selection method(D, attribute_list) to find the
  "best" splitting criterion;
• label node N with splitting_criterion;
• if splitting attribute is discrete-valued and multiway splits allowed
  then
    o attribute_list ← attribute_list - splitting_attribute;
• for each outcome of splitting-criterion
    o let D, be the set of data tuples in D satisfying outcome j; //
      a partition
    o if D; is empty then
        ▪ attach a leaf labeled with the majority class in D to
          node N;
    o else attach the node returned by Generate decision-tree(Dj,
      attribute_list)to node N;
  endfor
• return N;
```

Support Vector Machine: "Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features you have) with the value of each feature being the value of a particular coordinate. Then, perform classification by finding the hyper-plane that differentiates the two classes very well. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. The objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

Algorithm: Steps for SVM Classification

• Prepare the pattern matrix
• Select the kernel function to use
• Select the parameter of the kernel function and the
value of C

- You can use the values suggested by the SVM software, or you can set apart a validation set to determine the values of the parameter

• Execute the training algorithm and obtain the ai
• Unseen data can be classified using the a; and the support vectors

GridSearchCV: It is the process of performing hyperparameter tuning to determine the optimal values for a given model. The performance of a model significantly depends on the value of hyperparameters. Note that there is no way to know in advance the best values for hyperparameters so ideally, we need to try all possible values to know the optimal values. Doing this manually could take a considerable amount of time and resources and thus we use GridSearchCV to automate the tuning of hyperparameters.

Random Forest: Random Forest is a supervised learning algorithm. The "forest" it builds is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result. Random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result. Random forest can be used for both classification and regression problems.

The Working process can be explained in the below steps:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

The algorithms used for predicting diabetes using Rule Based Model is as follows,

3.2.4 Evaluation criterion for classification model

Machine learning model accuracy is the measurement used to determine which model is best at identifying relationships and patterns between variables in a dataset based on the input, or training, data. The better a model can generalize to ‘unseen’ data, the better predictions and insights it can produce, which in turn deliver more business value. In this project Accuracy is used to measure the performance of the model.

Accuracy: Accuracy is one metric for evaluating classification models. Informally, **accuracy** is the fraction of predictions our model got right.

Formally, accuracy has the following definition:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Figure - 3.12 Accuracy Score

For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Figure - 3.13 Accuracy Score For Binary Classification

3.3 Workflow for Regression model

3.3.1 Dataset

In this project we used diabetes dataset from uci repository. Diabetes patient records were obtained from two sources: an automatic electronic recording device and paper records. The automatic device had an internal clock to timestamp events, whereas the paper records only provided "logical time" slots (breakfast, lunch, dinner, bedtime). For paper records, fixed times were assigned to breakfast (08:00), lunch (12:00), dinner (18:00), and bedtime (22:00). Thus, paper records have fictitious uniform recording times whereas electronic records have more realistic time stamps.

3.3.2 Data Preprocessing

In the dataset used for Regression Models there are many missing values with inconsistent data. So, we handled missing values by removing them and removed outliers in the Data Preprocessing step.

3.3.3 Data Modelling

The algorithms used to predict the amount of insulin to be given to a diabetic patient are,

Linear Regression: Linear Regression is a machine learning algorithm based on **supervised learning**. It performs a **regression task**. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.

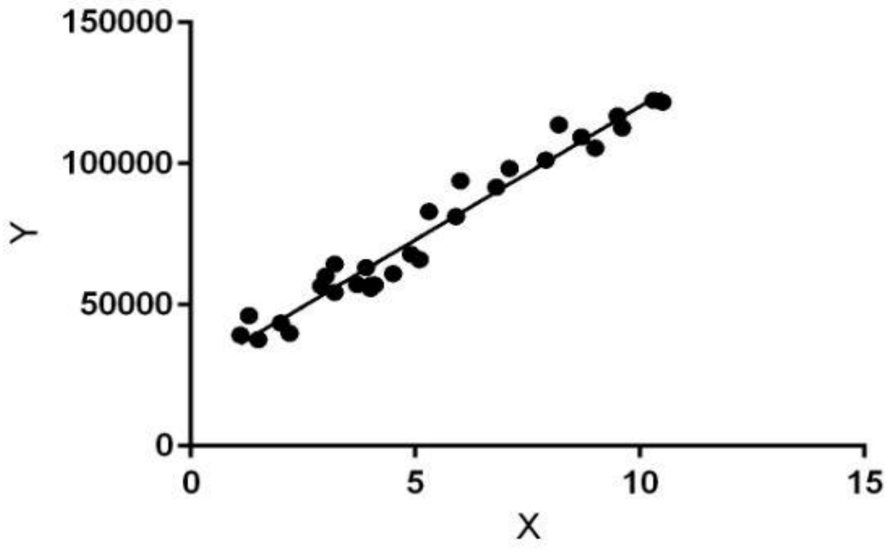


Figure – 3.14 Regression Line

Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression. In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

Hypothesis function for Linear Regression :

$$y = \theta_1 + \theta_2.x$$

While training the model we are given:

x: input training data (univariate – one input variable(parameter))

y: labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ_1 and θ_2 values.

θ_1: intercept

θ_2: coefficient of x

Once we find the best θ_1 and θ_2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

Cost function(J) of Linear Regression is the Root Mean Squared Error (RMSE) between predicted y value (pred) and true y value (y).

Gradient Descent:

To update θ_1 and θ_2 values to reduce Cost function (minimizing RMSE value) and achieving the best fit line the model uses Gradient Descent. The idea is to start with random θ_1 and θ_2 values and then iteratively updating the values, reaching minimum cost.

Ridge Regression: Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs L2 regularization. When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values to be far away from the actual values.

The cost function for ridge regression:

Min(||Y – X(theta)||^2 + λ||theta||^2)

Lambda is the penalty term. λ given here is denoted by an alpha parameter in the ridge function. So, by changing the values of alpha, we are controlling the penalty term. Higher the values of alpha, bigger is the penalty and therefore the magnitude of coefficients is reduced.

- It shrinks the parameters. Therefore, it is used to prevent multicollinearity
- It reduces the model complexity by coefficient shrinkage

For any type of regression machine learning models, the usual regression equation forms the base which is written as: ***Y* = *XB* + *e***

Where Y is the dependent variable, X represents the independent variables, B is the regression coefficients to be estimated, and e represents the errors are residuals.

Once we add the lambda function to this equation, the variance that is not evaluated by the general model is considered. After the data is ready and identified to be part of L2 regularization, there are steps that one can undertake.

3.3.4 Evaluation criterion for regression model

While data preparation and training a machine learning model is a key step in the machine learning pipeline, it's equally important to measure the performance of this trained model. How well the model generalizes on the unseen data is what defines adaptive vs non-adaptive machine learning models.

Root mean squared error: 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 spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. The formula of RMSE is as follows,

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

Figure - 3.15 Root Mean Square Error

CHAPTER – 4
SOFTWARE AND HARDWARE REQUIREMENTS

4.1 Hardware Requirements

As the project involves dealing with a small dataset, less hardware specifications are enough to run this project. We used an Intel Core I5 processor with 8GB RAM and 1TB storage working with more than 2.4 GHz processing power to implement this project.

❖ RAM	:	8GB
❖ Hard Disk	:	500GB
❖ Processor	:	Core i5 or Higher
❖ Speed	:	2.1Ghz

4.2 Software Requirements

We implemented the project in Google colab using Python Programming Language. We have Imported different libraries provided by Python to implement this project.

❖ Operating System	: Windows (Any version)
❖ Programming Language	: Python
❖ Tool/IDE	: Google Colab
❖ Libraries	: Sklearn, Matplotlib, Numpy, Pandas, Mlxtend, SkFuzzy

CHAPTER – 5
SYSTEM DESIGN

5.1 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. The data is collected from the user which is then preprocessed and sent it to classification model This model will predict whether the person has diabetes or not. If the person has diabetes. Then additional data required for insulin prediction is collected and sent to regressor and the results are sent to the person.

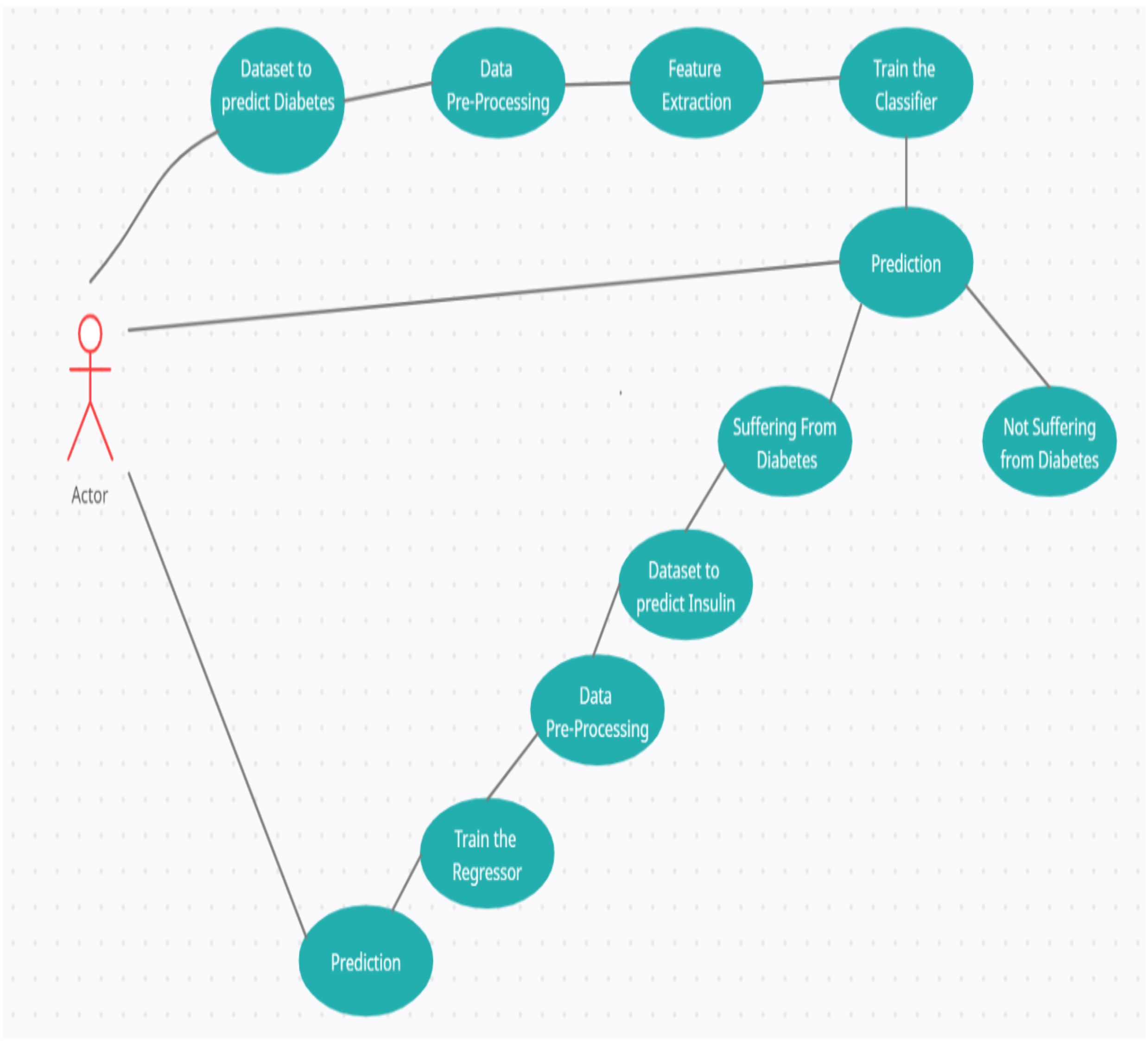


Figure – 5.1 Use Case Diagram

5.2 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration, and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. Different Classifiers are trained on the dataset and results are predicted for patients and is they have diabetes the amount of insulin to be given to them is predicted using Regression models.

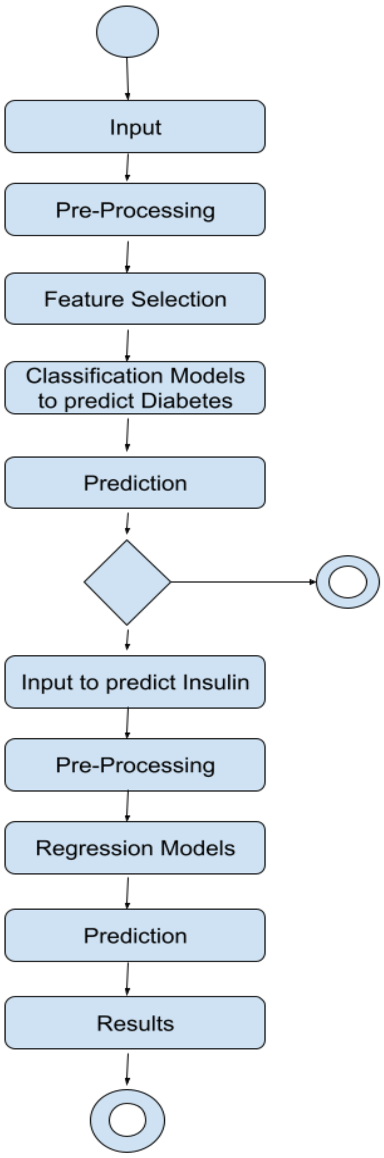


Figure – 5.2 Activity Diagram

5.3 Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams. The Raw data is collected from the user which is then preprocessed and sent it to different Classifiers. This model will predict whether the person has diabetes or not. If the person has diabetes. Then additional data required for insulin prediction is collected and sent to regressor and the results are sent to the person.

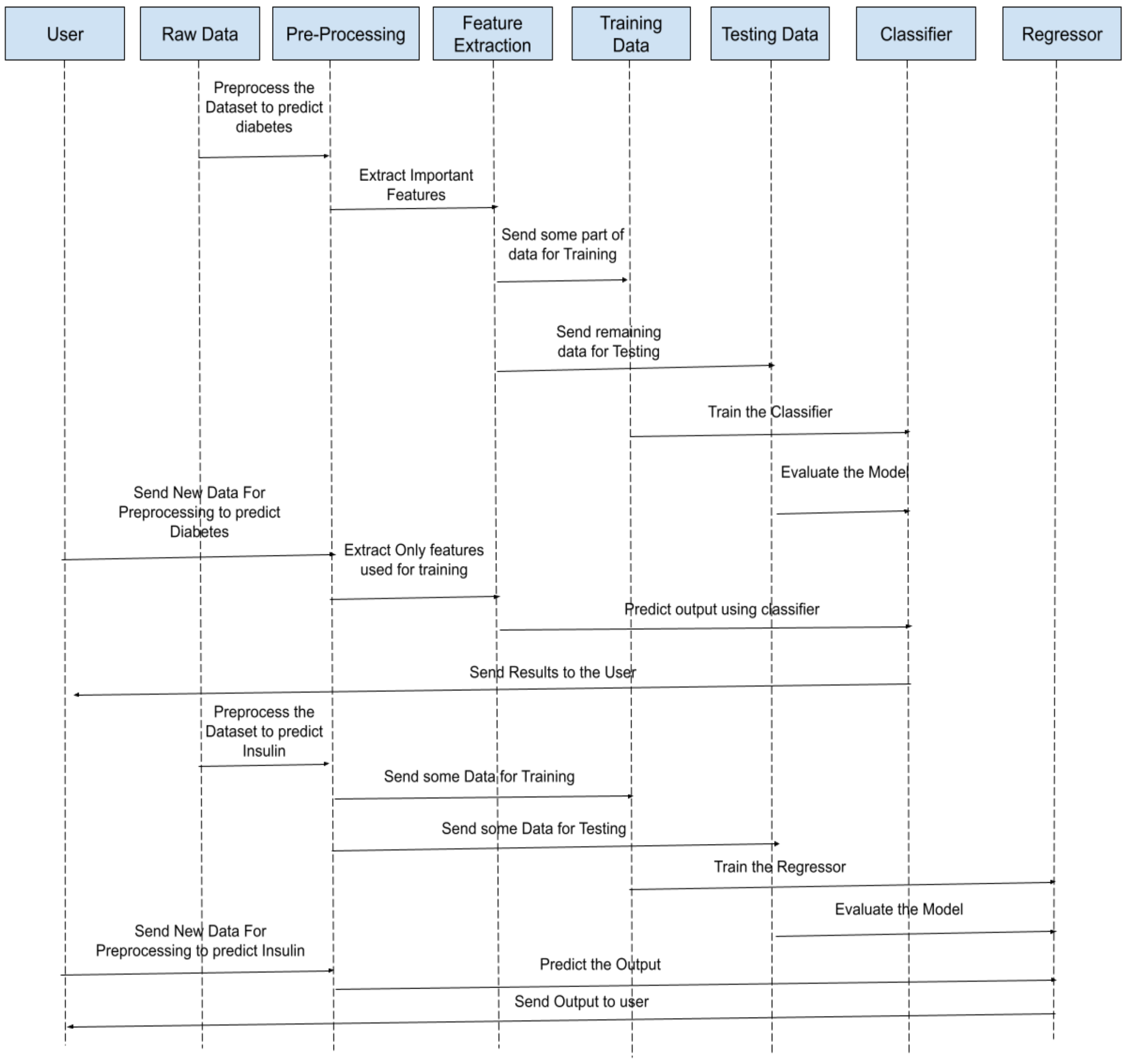

User
Raw Data
Pre-Processing
Feature Extraction
Training Data
Testing Data
Classifier
Regressor
Preprocess the Dataset to predict diabetes
Extract Important Features
Send some part of data for Training
Send remaining data for Testing
Train the Classifier
Evaluate the Model
Send New Data For Preprocessing to predict Diabetes
Extract Only features used for training
Predict output using classifier
Send Results to the User
Preprocess the Dataset to predict Insulin
Send some Data for Training
Send some Data for Testing
Train the Regressor
Evaluate the Model
Send New Data For Preprocessing to predict Insulin
Predict the Output
Send Output to user

Figure – 5.3 Sequence Diagram

5.4 Data Flow Diagram

A data flow diagram (DFD) of this project illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on the flow of Information, where data comes from, where it goes and how it gets stored. After collecting the raw data it is preprocessed and on this preprocessed data feature extraction will be applied then the model will be trained on this data. If the person has diabetes then the model collects further data required for insulin dosage prediction and predicts the insulin dosage required for the patient.

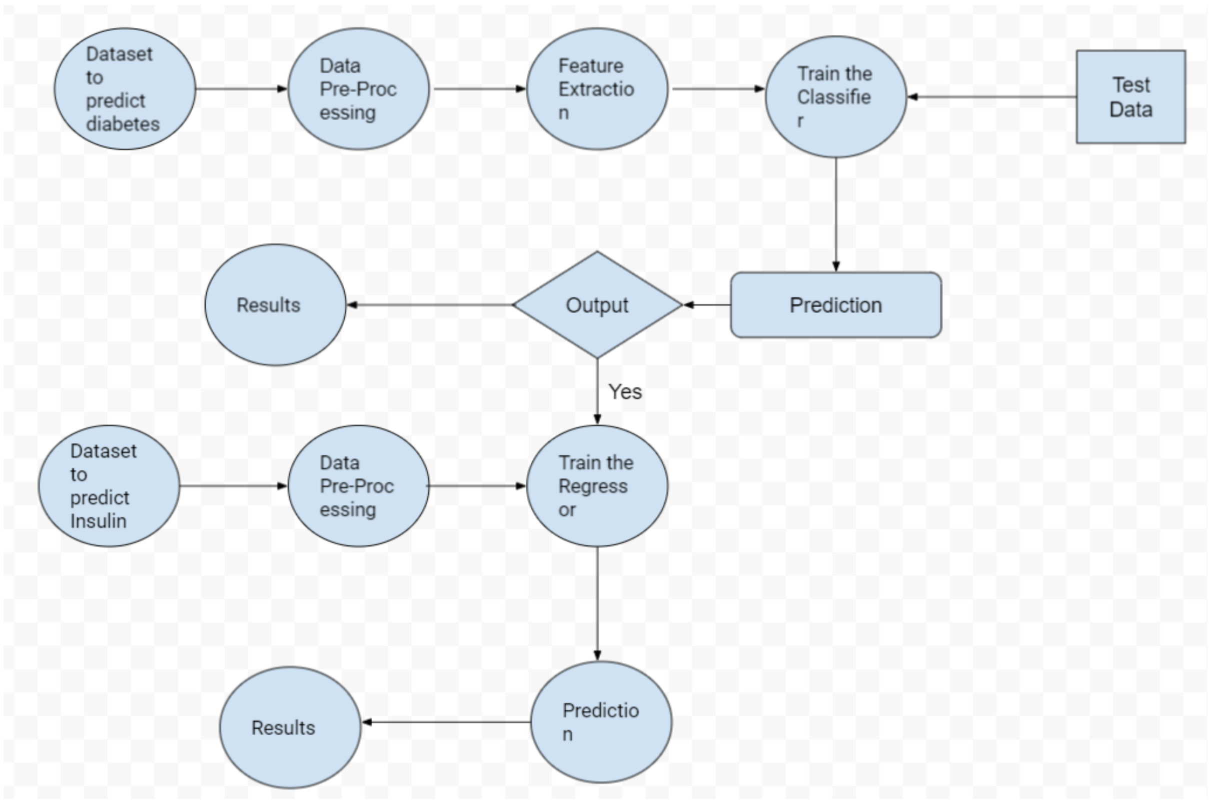


Figure – 5.4 Data Flow Diagram

CHAPTER – 6 IMPLEMENTATION AND RESULT ANALYSIS

6.1Required Libraries:

Numpy: NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. All the array operations in python can be done using Numpy

Pandas: Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on the top of the NumPy library. Pandas is fast and it has high-performance & productivity for users. The manipulation of dataset and cleaning of the dataset can be done using Pandas.

Matplotlib: Matplotlib is a plotting library for the python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

Sklearn: Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machines, random forests, and k-neighbors, and it also supports Python numerical and scientific libraries like NumPy and SciPy and its core algorithms are written in Cython for even better performance.

Mlxtend.frequent_patterns: This package contains the implementations of fp growth and association rules and it’s useful for generating the rules based on which we classify.

Scikit-fuzzy: This is an implementation of fuzzy c means clustering and it's useful for the clustering of the dataset.

Steps of Implementation: -

i. Installing required packages
ii. Importing required libraries
iii. Loading dataset
iv. Handling missing values
v. Handling outliers
vi. Normalization
vii. Feature Importance using Random Forests
viii. Training and testing using classifiers
ix. Loading data again
x. Loading data again
xi. Normalization and Clustering
xii. Binning and Encoding
xiii. Frequent Item sets Generation and Association rules finding
xiv. Testing the rules
xv. Loading and Cleaning data for regression
xvi. Training and Testing
xvii. Source code
xviii. Result Analysis

6.2 Installing required packages

pip is the standard package manager for Python. We can use pip to install additional packages that are not available in the Python standard library.

In this project we used scikit-fuzzy and mlxtend packages which helps in doing fuzzy c means clustering and data mining algorithms respectively.

6.3 Importing Libraries

Import statement can be used to import the libraries or packages or modules required to implement the logic.

we can import libraries or, packages and modules. we can also import specific objects from a package or module. There are generally two types of import syntax.

When you use the first one, you import the resource directly.

import numpy

6.4 Loading Dataset in Google Colab

First, we need to upload the file from local computer to cloud and it can be done by using

```
from google.colab import files
uploaded = files.upload()
```

Then load the dataset into the working notebook

```
import io
df = pd.read_csv(io.BytesIO(uploaded['diabetes.csv']))
```

6.5 Handling missing values

In this project as the data is related to medical records we can't assume or fill the missing values with other values. So, we removed the rows having missing values.

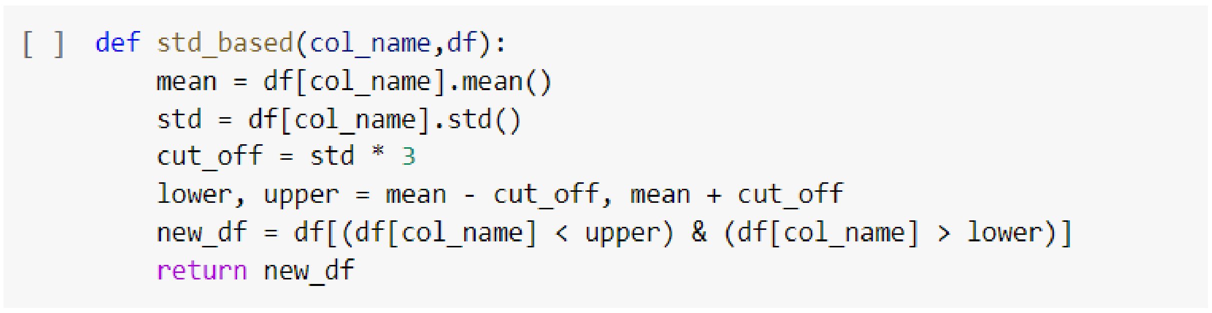
In this project the missing values are filled with zeros so we removed them.

df = df[(df[col[i]] > 0)]

6.6 Handling Outliers

In this project z-score and box plots are used for removing the outliers in the data. Z score is an important concept in statistics. Z score is also called standard score. This score helps to understand if a data value is greater or smaller than mean and how far away it is from the mean. More specifically, Z score tells how many standard deviations away a data point is from the mean.

Z score = (x -mean) / std. deviation

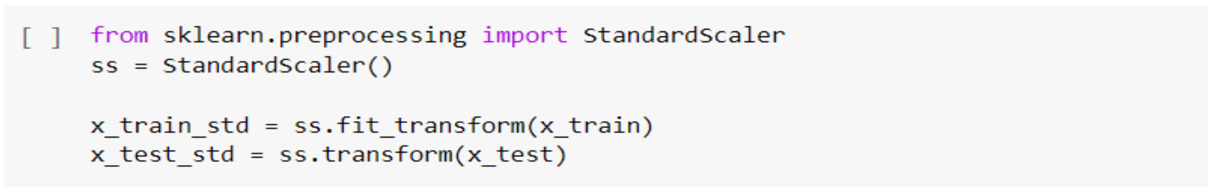


```
[ ]  def std_based(col_name,df):
         mean = df[col_name].mean()
         std = df[col_name].std()
         cut_off = std * 3
         lower, upper = mean - cut_off, mean + cut_off
         new_df = df[(df[col_name] < upper) & (df[col_name] > lower)]
         return new_df
```

Figure – 6.1 Z-Score Code

6.7 Normalization

In this project Standard Scalar is used for normalizing the given data.

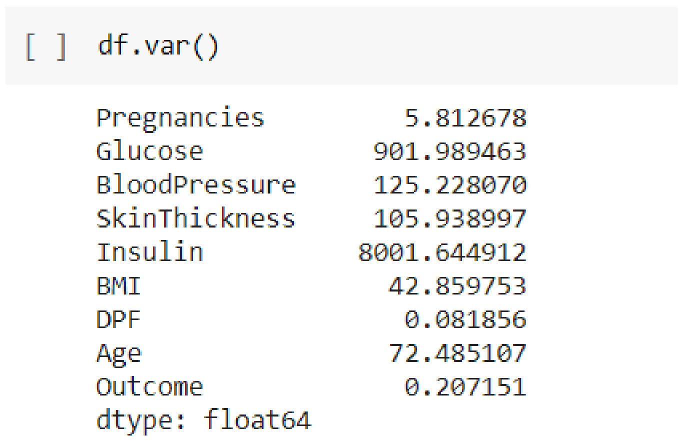


```
[ ]  from sklearn.preprocessing import StandardScaler
     ss = StandardScaler()

     x_train_std = ss.fit_transform(x_train)
     x_test_std = ss.transform(x_test)
```

Figure – 6.2 Normalization

6.8 Finding Feature Importance

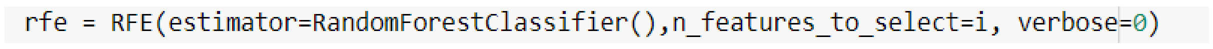
In this project we removed features having very low variance.



```
[ ]  df.var()

     Pregnancies          5.812678
     Glucose            901.989463
     BloodPressure      125.228070
     SkinThickness      105.938997
     Insulin           8001.644912
     BMI                 42.859753
     DPF                  0.081856
     Age                 72.485107
     Outcome              0.207151
     dtype: float64
```

Figure – 6.3 Variance of Features

Next, we found the important features using random forest



```
rfe = RFE(estimator=RandomForestClassifier(),n_features_to_select=i, verbose=0)
```

Figure – 6.4 Recursive Feature Elimination

Important Features are 'Glucose', 'SkinThickness', 'Insulin', 'BMI', 'Age'.

6.9 Training and Testing using Classifiers

Training is a step where an Algorithm is given a dataset as input and the algorithm gets trained based on the combinations of inputs and outputs of the provided dataset so that it can predict the output for a given new input.

We can split our Dataset into 2 parts using train_test_split() method available in Sklearn library easily.

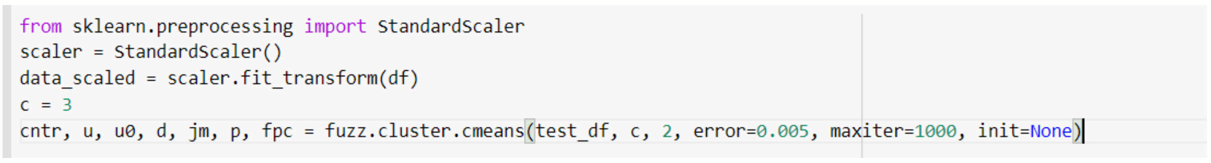
In this project we have trained on different algorithms such as K-Nearest Neighbors, Support Vector Machines, Decision Trees, XGBoost, voting classifier and Random Forests by using Grid Search CV which helps in finding optimal parameters for the algorithm.

6.10 Reloading Dataset

We again reload the data for doing the rule-based classification.

6.11 Normalization and Clustering

After loading the data, we first done the normalization using standard scalar and then we applied the fuzzy c means clustering to get the clusters.



```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data_scaled = scaler.fit_transform(df)
c = 3
cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(test_df, c, 2, error=0.005, maxiter=1000, init=None)
```

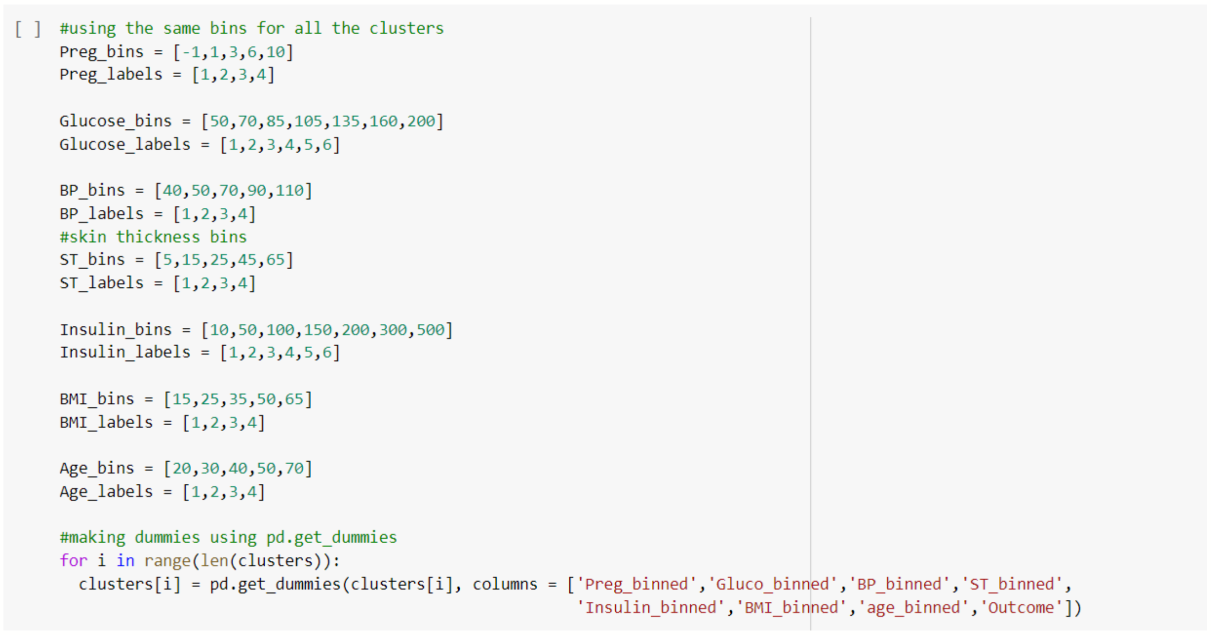
Figure – 6.5 Normalization & Clustering

And then by using sse scores we found the optimal number of clusters and did the clustering based on that.

6.12 Binning and Encoding

After clustering we make each cluster into a separate data frame and on each data frame binning is applied and then the categorical features are encoded into one hot vector and then we changed the values into Boolean.

Binning Ranges:

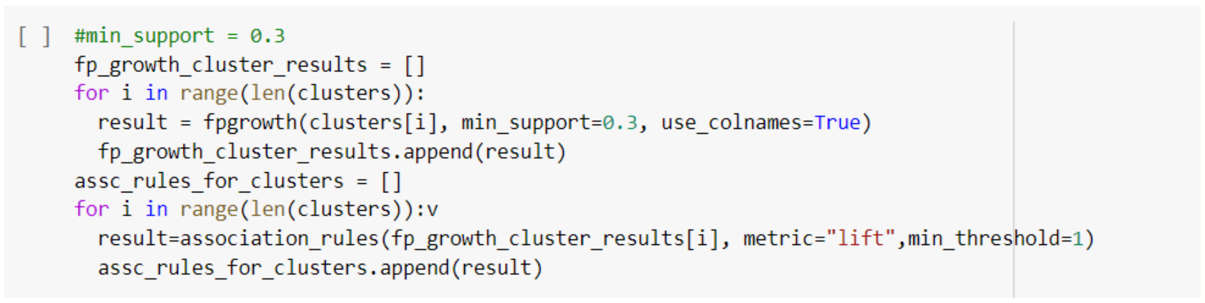


```
[ ]  #using the same bins for all the clusters
     Preg_bins = [-1,1,3,6,10]
     Preg_labels = [1,2,3,4]

     Glucose_bins = [50,70,85,105,135,160,200]
     Glucose_labels = [1,2,3,4,5,6]

     BP_bins = [40,50,70,90,110]
     BP_labels = [1,2,3,4]
     #skin thickness bins
     ST_bins = [5,15,25,45,65]
     ST_labels = [1,2,3,4]

     Insulin_bins = [10,50,100,150,200,300,500]
     Insulin_labels = [1,2,3,4,5,6]

     BMI_bins = [15,25,35,50,65]
     BMI_labels = [1,2,3,4]

     Age_bins = [20,30,40,50,70]
     Age_labels = [1,2,3,4]

     #making dummies using pd.get_dummies
     for i in range(len(clusters)):
       clusters[i] = pd.get_dummies(clusters[i], columns = ['Preg_binned','Gluco_binned','BP_binned','ST_binned',
                                                           'Insulin_binned','BMI_binned','age_binned','Outcome'])
```

Figure – 6.6 Binning & Encoding

6.13 Frequent Item sets Generation and Association rules Finding

After making data suitable for Item sets generation, we used fp growth to generate frequent item sets then by using them we generated the association rules.



```
[ ]  #min_support = 0.3
     fp_growth_cluster_results = []
     for i in range(len(clusters)):
       result = fpgrowth(clusters[i], min_support=0.3, use_colnames=True)
       fp_growth_cluster_results.append(result)
     assc_rules_for_clusters = []
     for i in range(len(clusters)):v
       result=association_rules(fp_growth_cluster_results[i], metric="lift",min_threshold=1)
       assc_rules_for_clusters.append(result)
```

Figure – 6.7 Frequent Item Sets Generation

6.14 Testing the rules

After generating the association rules, rules containing Outcome as their consequent are selected as zero rules and one rules respectively. Zero and one rules containing at least two items are selected for the criteria. Considering the 14 zero rules for testing, the model gave the accuracy of 83%.

6.15 Loading and Cleaning data for Regression

If the person has diabetes, then we predict the amount of insulin that should be given to a person using regression models.

First, we load the data into workspace then we remove the missing data and outliers manually.

df = df[(df['Regular insulin dose'] < 16) & (df['Regular insulin dose'] > 3)]

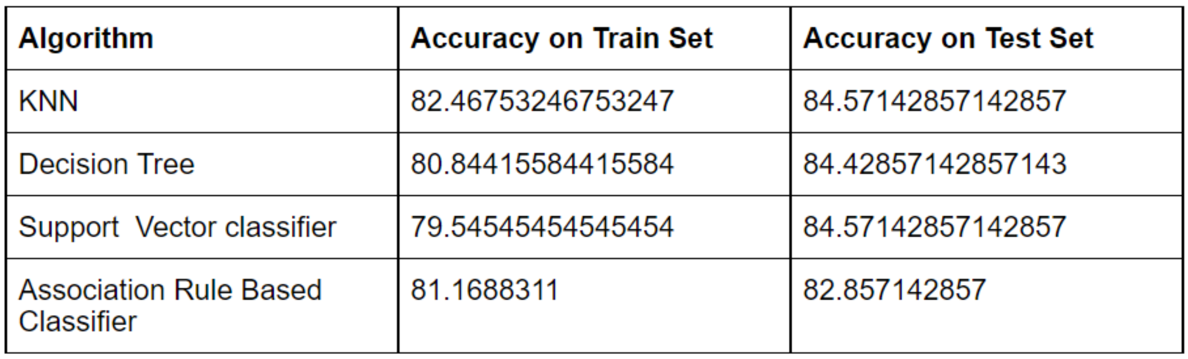
6.16 Training and Testing

After preparing the data, Model is trained on regression algorithms linear regression and ridge regression. Ridge regression includes regularization term to the linear regression loss function thereby decreasing overfitting.

6.17 Result Analysis

In this project for regression, we used RMSE for the evaluation of algorithms performance.

The accuracies of different classification algorithms are shown in Figure-6.8,



Algorithm	Accuracy on Train Set	Accuracy on Test Set
KNN	82.46753246753247	84.57142857142857
Decision Tree	80.84415584415584	84.42857142857143
Support Vector classifier	79.54545454545454	84.57142857142857
Association Rule Based Classifier	81.1688311	82.857142857

Figure – 6.8 Accuracies

The performance of our model using various regression algorithms

Linear Regression:

RMSE on train 2.812487786963059

RMSE on test 2.8513293135077356

Ridge Regression:

RMSE on train 2.81265786963059

RMSE on test 2.855463931350773

6.18 Source Code

Importing the required libraries

```
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
%matplotlib inline
sns.set_style('darkgrid')
```

Loading the data

```
import io
df = pd.read_csv(io.BytesIO(uploaded['diabetes.csv']))
```

Preprocessing: Handling missing values, outliers and removing unimportant features and making data ready for modelling

Checks if there are any missing values

```
df.isnull().sum()
```

Removing the rows with missing values

```
for i in range(1,8):
  df = df[(df[col[i]] >  0)]
```

The model uses Standard deviation method to eliminate outliers

```
def std_based(col_name,df):
    mean = df[col_name].mean()
    std = df[col_name].std()
    cut_off = std * 3
    lower, upper = mean - cut_off, mean + cut_off
    new_df = df[(df[col_name] < upper) & (df[col_name] > lower)]
    return new_df
```

Removing the outliers using std_based function

```
df = std_based('Pregnancies',df)
df  = std_based('BloodPressure',df)
df = std_based("SkinThickness",df)
df = std_based('DPF',df)
df = std_based('Age',df)
df = std_based('Insulin',df)
```

Data Modelling:

Train Test splitting

```
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.10,
random_state = 0)

print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

Scaling is done to give equal importance to every feature

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()

x_train_std = ss.fit_transform(x_train)
x_test_std = ss.transform(x_test)
```

Fuzzy function for clustering

```
#second parameter centres c = 3
c = 3
cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(test_df, c, 2, error=0
.005, maxiter=1000, init=None)
```

SSE is used to find the optimal number of clusters for the Fuzzy C Means clustering

```
SSEs = [0 for i in range(12)]
for ncentres in range(2,12):
  cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(test_df, ncentres, 2
, error=0.005, maxiter=1000, init=None,seed=537)
  print('             ',ncentres,"clusters")

  clusters = np.argmax(u, axis=0)
  frame = pd.DataFrame(df)
  frame['clusters'] = clusters
  clustered_array = frame.to_numpy()

  #finding sse for each cluster
  for cl in range(ncentres):
    for j in d[cl][clusters == cl]:
      SSEs[ncentres] += j**2

  for i in range(ncentres):
    temp = pd.DataFrame(clustered_array[clusters == i][:,:-
1], columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThicknes
s','Insulin','BMI', 'Age', 'Outcome',])
    print('\t',i,'th cluster')
    print(temp['Outcome'].value_counts())
  print('----------------------------------')
```

Optimal number of clusters is decided as 6 with minimum SSE

```
c = 6
cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(test_df, c, 2, error=0
.005, maxiter=1000, init=None,seed=537)
```

Binning is applied to convert continuous values to categorical values

```
#using the same bins for all the clusters
Preg_bins = [-1,1,3,6,10]
Preg_labels = [1,2,3,4]

Glucose_bins = [50,70,85,105,135,160,200]
Glucose_labels = [1,2,3,4,5,6]

BP_bins = [40,50,70,90,110]
BP_labels = [1,2,3,4]
#skin thickness bins
ST_bins = [5,15,25,45,65]
ST_labels = [1,2,3,4]

Insulin_bins = [10,50,100,150,200,300,500]
Insulin_labels = [1,2,3,4,5,6]

BMI_bins = [15,25,35,50,65]
BMI_labels = [1,2,3,4]

Age_bins = [20,30,40,50,70]
Age_labels = [1,2,3,4]

#applying binning
for i in range(len(clusters)):
  clusters[i]['Preg_binned'] = pd.cut(clusters[i]['Pregnancies'], bins
= Preg_bins, labels = Preg_labels)
  clusters[i]['Gluco_binned'] = pd.cut(clusters[i]['Glucose'], bins = G
lucose_bins, labels = Glucose_labels)
  clusters[i]['BP_binned'] = pd.cut(clusters[i]['BloodPressure'], bins
= BP_bins, labels = BP_labels)
  clusters[i]['ST_binned'] = pd.cut(clusters[i]['SkinThickness'], bins
= ST_bins, labels = ST_labels)
  clusters[i]['Insulin_binned'] = pd.cut(clusters[i]['Insulin'], bins =
 Insulin_bins, labels = Insulin_labels)
  clusters[i]['BMI_binned'] = pd.cut(clusters[i]['BMI'], bins = BMI_bin
s, labels =BMI_labels)
  clusters[i]['age_binned'] = pd.cut(clusters[i]['Age'], bins = Age_bin
s, labels = Age_labels)
```

Categorical encoding using dummies

```
#making dummies using pd.get_dummies
for i in range(len(clusters)):
  clusters[i] = pd.get_dummies(clusters[i], columns = ['Preg_binned','G
luco_binned','BP_binned','ST_binned','Insulin_binned','BMI_binned','age
_binned','Outcome'])

!pip install mlxtend --upgrade
from mlxtend.frequent_patterns import fpgrowth
from mlxtend.frequent_patterns import association_rules
```

Frequent Itemsets and association rules are generated

```
#min_support = 0.3
fp_growth_cluster_results = []
for i in range(len(clusters)):
  result = fpgrowth(clusters[i], min_support=0.3, use_colnames=True)
  fp_growth_cluster_results.append(result)
assc_rules_for_clusters = []
for i in range(len(clusters)):v
  result=association_rules(fp_growth_cluster_results[i], metric="lift",
min_threshold=1)
  assc_rules_for_clusters.append(result)
```

Selecting the rules with outcome as consequent

```
antecedents_0 = []
antecedents_1 = []

for i in range(len(clusters)):
  condition_0 = assc_rules_for_clusters[i]['consequents'] == consequent
_0
```

```
  condition_1 = assc_rules_for_clusters[i]['consequents'] == consequent
_1
  ante_0 = assc_rules_for_clusters[i][condition_0]
  print(ante_0[['antecedents','consequents']])
  antecedents_0.append(ante_0['antecedents'])

  ante_1 = assc_rules_for_clusters[i][condition_1]
  print(ante_1[['antecedents','consequents']])
  antecedents_1.append(ante_1['antecedents'])
```

Combining the zero and one rules from different clusters respectively

```
zero_rules = set()
one_rules = set()
for ante in antecedents_0:
  for rule in ante:
    zero_rules.add(rule)

for ante in antecedents_1:
  for rule in ante:
    one_rules.add(rule)
```

Classification method

```
#function which returns tue or false whether a datapoint gives outcome
0 or not
def check_outcome(datapoint):
  for rule in selected_zero_rules:
    result = True
    #print("rule",rule)
    for bin in rule:
      index = bins_indexing[bin]
      result &= (lower_limit[bin] <= datapoint[index]) and (upper_limit
[bin] > datapoint[index]) #indexes in array
    if result:
      return 0

      #for 1 rules
  for rule in selected_one_rules:
    result = True
    for bin in rule:
      index = bins_indexing[bin]
      result &= (lower_limit[bin] <= datapoint[index]) and (upper_limit
[bin] > datapoint[index]) #indexes in array
    if result:
      return 1
  return 0
```

Conventional classification algorithms

KNN

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV

knn = KNeighborsClassifier()

param_grid = {'n_neighbors':[5,10,15,25,30,50]}

grid_knn = GridSearchCV(knn,param_grid,scoring='accuracy',cv = 10,refit
 = True)
grid_knn.fit(x_train_std,y_train)
print("Best Score ==> ", grid_knn.best_score_)
print("Tuned Paramerers ==> ",grid_knn.best_params_)
print("Accuracy on Train set ==> ", grid_knn.score(x_train_std,y_train)
)
print("Accuracy on Test set ==> ", grid_knn.score(x_test_std,y_test))
```

Decision Trees

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()

param_grid = {'criterion':['gini','entropy'],'max_depth':np.arange(2,10
),'min_samples_leaf':[0.2,0.4,0.6,0.8,0.9,1]}

grid_dtc = GridSearchCV(dtc,param_grid,scoring='accuracy',cv = 10,refit
 = True)
grid_dtc.fit(x_train_std,y_train)
print("Best Score ==> ", grid_dtc.best_score_)
print("Tuned Paramerers ==> ",grid_dtc.best_params_)
print("Accuracy on Train set ==> ", grid_dtc.score(x_train_std,y_train)
)
print("Accuracy on Test set ==> ", grid_dtc.score(x_test_std,y_test))
```

Support Vector Machine

```
from sklearn.svm import SVC

svc = SVC(probability=True)
```

```
param_grid = {'kernel':['rbf','linear'],'C':[0.01,0.1,1,0.001],'gamma':
[0.1,0.01,0.2,0.4]}

grid_svc = GridSearchCV(svc,param_grid,scoring='accuracy',cv = 10,refit
 = True)
grid_svc.fit(x_train_std,y_train)
print("Best Score ==> ", grid_svc.best_score_)
print("Tuned Paramerers ==> ",grid_svc.best_params_)
print("Accuracy on Train set ==> ", grid_svc.score(x_train_std,y_train)
)
print("Accuracy on Test set ==> ", grid_svc.score(x_test_std,y_test))
```

Predicting Amount of Insulin Dosage
Importing the required libraries

Scaling the data that we want to use for regression

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df = scaler.fit_transform(df)
```

Splitting the dataset

```
x_train, x_test, y_train, y_test = train_test_split(df, label,
test_size=0.25,random_state=537)
```

Training linear regression model

```
lr = LinearRegression()
lr.fit(X_train,y_train)
test_pre = lr.predict(X_test)
train_pre = lr.predict(X_train)
print('rmse on train',rmse_CV_train(lr).mean())
print('rmse on test',rmse_CV_test(lr).mean())
```

Training ridge regression model

```
ridge = RidgeCV(alphas = [0.01, 0.03, 0.06, 0.1, 0.3, 0.6, 1, 3, 6, 10,
30, 60])
ridge.fit(X_train,y_train)
alpha = ridge.alpha_
print('best alpha',alpha)
print("Try again for more precision with alphas centered around " +
str(alpha))
ridge = RidgeCV(alphas = [alpha * .6, alpha * .65, alpha * .7, alpha *
.75, alpha * .8, alpha * .85,
                          alpha * .9, alpha * .95, alpha, alpha * 1.05,
alpha * 1.1, alpha * 1.15,
                          alpha * 1.25, alpha * 1.3, alpha * 1.35,
alpha * 1.4],cv = 5)
ridge.fit(X_train, y_train)
alpha = ridge.alpha_
print("Best alpha :", alpha)
print("Ridge RMSE on Training set :", rmse_CV_train(ridge).mean())
print("Ridge RMSE on Test set :", rmse_CV_test(ridge).mean())
y_train_rdg = ridge.predict(X_train)
y_test_rdg = ridge.predict(X_test)
```

Plot for linear regression with regularization

```
plt.scatter(y_train_rdg, y_train_rdg - y_train, c = "blue",  label =
"Training data")
plt.scatter(y_test_rdg, y_test_rdg - y_test, c = "black", marker = "v",
label = "Validation data")
plt.title("Linear regression with Ridge regularization")
plt.xlabel("Predicted values")
plt.ylabel("Residuals")
plt.legend(loc = "upper left")
plt.hlines(y = 0, xmin = 10.5, xmax = 13.5, color = "red")
plt.show()


# Plot predictions - Real values
plt.scatter(y_train_rdg, y_train, c = "blue",  label = "Training data")
plt.scatter(y_test_rdg, y_test, c = "black",  label = "Validation
data")
plt.title("Linear regression with Ridge regularization")
plt.xlabel("Predicted values")
plt.ylabel("Real values")
plt.legend(loc = "upper left")
plt.plot([10.5, 13.5], [10.5, 13.5], c = "red")
plt.show()
```

CHAPTER – 7
EVALUATION

7.1 Evaluation of Models

While training a model is a key step, how the model generalizes on unseen data is an equally important aspect that should be considered in every machine learning pipeline. We need to know whether it actually works and, consequently, if we can trust its predictions.

The above issues can be handled by evaluating the performance of a machine learning model. Model evaluation aims to estimate the generalization accuracy of a model on future (unseen/out-of-sample) data.

In this project we have used Accuracy Score for classification algorithms and Root Mean Square Error for Regression Algorithms.

7.2 Accuracy Score

We have used Association Rule based classifier for the prediction of the diabetes. During Clustering the optimal number of clusters can be found by using Elbow Rule of SSE scores.

The graphical representation of SSE scores with various clusters is shown in Figure-7.1:

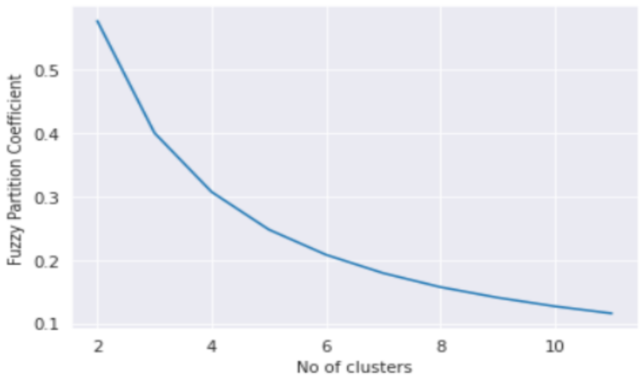
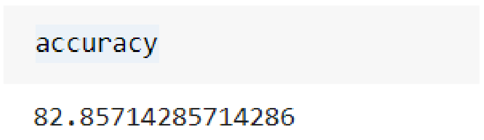


Figure - 7.1 SSE Scores

By using this we have found optimal number of clusters are 6.

After generating the association rules, rules containing Outcome as their consequent are selected as zero rules and one rules respectively. Zero and one rules containing at least two items are selected for the criteria. Considering the 14 zero rules for testing, the model gave the accuracy of 83%.



```
accuracy

82.85714285714286
```

Figure – 7.2 Association Rule Accuracy

We have compared the accuracy of Association Rule Based classifier with conventional classifiers. The different conventional classifiers we have used in this project and their Training and Testing accuracies are as follows,

Algorithm	**Accuracy on Train Set**	**Accuracy on Test Set**
KNN	82.46753246753247	84.57142857142857
Decision Tree	80.84415584415584	84.42857142857143
Support Vector classifier	79.54545454545454	84.57142857142857
Association Rule Based Classifier	81.1688311	82.857142857

Figure - 7.3 Training and Testing Accuracies

Though we got slightly higher accuracy for conventional classifiers, our Association Rule Based classifiers performing very well by giving an accuracy of 83% which is a lot better than many benchmark models.

7.3 Root Mean Square Error

We have used Root Mean Square Error for measuring the accuracy of Regression algorithms which are used for the prediction of Insulin.

The different algorithms we have used for prediction of Insulin and their performances are as follows,

The accuracy of Linear Regression Algorithm is shown in Figure-7.4,

```
rmse on train 2.812487786963059
rmse on test 2.8513293135077356
```

Figure - 7.4 Linear Regression RMSE

The accuracy of Ridge Algorithm is shown in Figure-7.5,

```
best alpha 30.0
Try again for more precision with alphas centered around 30.0
Best alpha : 42.0
Ridge RMSE on Training set : 2.811903397634329
Ridge RMSE on Test set : 2.85401136350229
```

Figure – 7.5 Ridge RMSE

The graphical representation of the performance of Linear Regression with Ridge Regularization is shown in Figure-7.6 and 7.7,

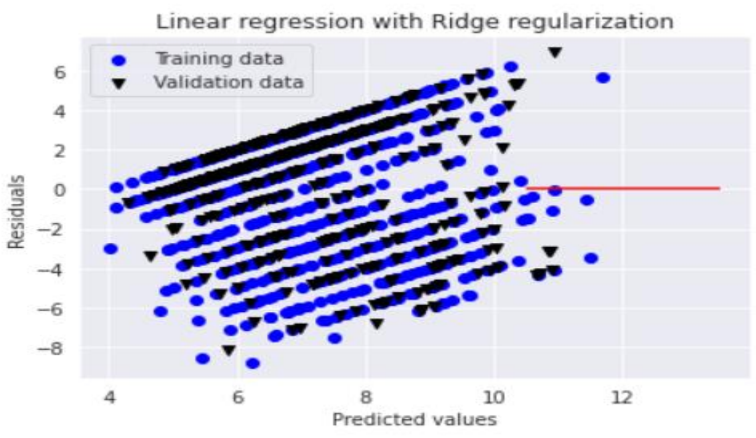


Figure - 7.6 Linear Regression with Ridge Regularization

The Figure-7.6 shows the residuals (predicted - actual) that we got for each data point against the predicted value. Using this graph, we can find how well we are predicting the outcome.

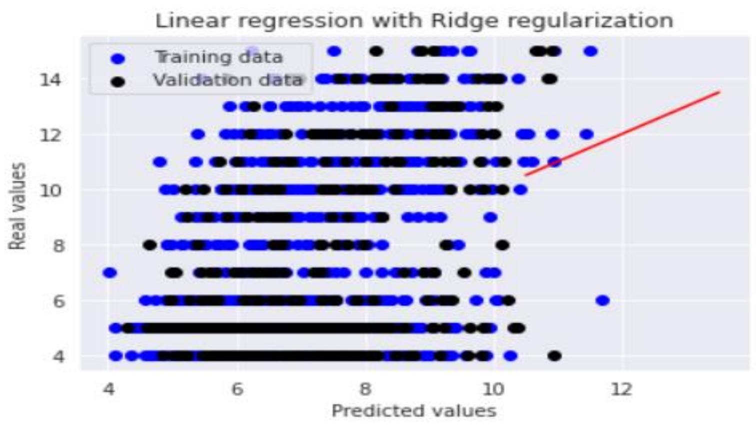


Figure – 7.7 Linear Regression with Ridge Regularization-2

The Figure-7.7 shows the plot between the actual values vs predicted values that we got for each data point. Using this graph, we can find how well we are predicting the outcome.

CHAPTER – 8 CONCLUSION AND FUTURE SCOPE

CONCLUSION AND FUTURE SCOPE

In this project we have addressed the classification of Diabetes disease as well as Prediction of Insulin Dosage using machine learning. We have used Pima Indian dataset for classification and Machine UCI diabetes dataset for Regression. We have performed different preprocessing techniques and generated clusters and, on each cluster, using FP-Growth algorithm we have generated the rules then based on these rules Association Rule Based Classifier is built. We have Compared the accuracy of Association Rule based classifier with Conventional classifiers like KNN, SVM, Decision Trees, Random Forest.

If a person is suffering from diabetes, we have built Regression models using algorithms Linear Regression and Ridge Regression and predicted the Insulin Dosage to be given to a diabetic patient.

8.1 Conclusion

From this experiment it is concluded that the Association Rule based classifier gives the accuracy around 83%. The Accuracies of Conventional Classifiers is for KNN 84%, SVM 85%.

Even though the conventional classifiers got slightly higher accuracies they are Black Box techniques where we do not know the actual reason for the prediction of the output. The Association Rule Based classifier predicts the output based on the rules which were generated from clusters. So, we can know the reason behind the prediction of output. The Root Mean Square Error for algorithms used for Regression for the prediction of Insulin Dosage are for Linear Regression 2.851 and for Ridge Regression 2.854.

8.2 Future Scope

Selection of suitable techniques for data cleaning along with proper classification algorithms will lead to the development of prediction systems that give enhanced accuracy. In future an intelligent system may be developed that can lead to selection of proper treatment methods for a patient diagnosed with diabetes disease. White Box techniques which specifies the reason for prediction have great scope in medical field. The future scope of the paper is the prediction of diabetes by using advanced techniques and algorithms in less time and with high accuracy.

REFERENCES

[1] Dewangan, Amit K.,and Pragati Agrawal."**Classification of Diabetes Mellitus Using Machine Learning Techniques”,** International Journal of Engineering and Applied Sciences, vol. 2, no. 5, May. 2015.

[2] Talha Mahboob Alam, Muhammad Atif Iqbal "**A model for early prediction of diabetes",** Informatics in Medicine Unlocked January 2019

[3] Kaur, H. and Kumari, V. (2020), "**Predictive modelling and analytics for diabetes using a machine learning approach",** *Applied Computing and Informatics*, 28 July 2020

[4] V. Anuja Kumari, R.Chitra "**Classification Of Diabetes Disease Using Support Vector Machine",** International Journal of Engineering Research and Applications (IJERA) Vol. 3, Issue 2, March -April 2013

[5] Singh P.P., Prasad S., Das B., Poddar U., Choudhury D.R. (2018) "**Classification of Diabetic Patient Data Using Machine Learning Techniques",** Ambient Communications and Computer Systems. Advances in Intelligent Systems and Computing, vol 696. Springer, Singapore. 21 March 2018

[6] Leon Kopitar, Primoz Kocbek, Leona Cilar, Aziz Sheikh, Gregor Stiglic "**Early detection of type 2 diabetes mellitus using machine learning-based prediction models",** Scientific Reports july 2020

[7] Francesco Mercaldo, Vittoria Nardone, Antonella Santone "**Diabetes Mellitus Affected Patients Classification and Diagnosis through Machine Learning Techniques**" Procedia Computer Science, 1st September 2017

[8] S. Wei, X. Zhao and C. Miao, "**A comprehensive exploration to the machine learning techniques for diabetes identification,**"2018 IEEE 4th World Forum on Internet of Things (WF-IoT), 2018

[9] Aishwarya.R, Gayathri.P, N.Jaisankar "**A Method for Classification Using Machine Learning Technique for Diabetes ",** International Journal of Healthcare Information Systems and Informatics July 2015

[10] A. Mir and S. N. Dhage, "**Diabetes Disease Prediction Using Machine Learning on Big Data of Healthcare,**" Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018