CHAPTER 1
INTRODUCTION

1.1 MACHINE LEARNING

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to “automatically learn and improve from experience without being explicitly programmed”. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly. Machine Learning is also often referred to as predictive analytics, or predictive modeling. Machine Learning uses programmed algorithms that receive and analyse input data to predict output values within an acceptable range.

1.1.1 Machine Learning Types:

Supervised Learning:

The operator provides the Machine Learning algorithm with a known dataset that includes desired inputs and outputs. While the operator knows the correct answers to the problem, the algorithm identifies patterns in data, learns from observations and makes predictions. The algorithm makes predictions and is corrected by the operator and this process continues until the algorithm achieves a high level of accuracy/performance. While Classification, Regression and Forecasting problems are mainly solved here. Labelled data is used for training here. It is mainly used in Predicting Modelling.

Unsupervised Learning:

The Machine Learning algorithm is left to interpret large data sets and address that data accordingly. The algorithm tries to organise that data in some way to describe its structure. This might mean grouping the data into clusters or arranging it in a way that looks more organised. As it assesses more data, its ability to make decisions on that data gradually improves and becomes more refined. While Clustering and Dimension reduction problems are solved. Unlabeled data is used in unsupervised learning. It is mainly used in Descriptive Modelling.

Semi-supervised Learning:

It is in-between that of Supervised and Unsupervised Learning. Semi-supervised learning is similar to supervised learning, but instead uses both labeled and unlabeled data. Labeled data is essentially information that has meaningful tags so that the algorithm can understand the data, whilst unlabelled data lacks that information. By using this combination, machine learning algorithms can learn to label unlabelled data.

Reinforcement learning:

Reinforcement learning focuses on regimented learning processes, where a machine learning algorithm is provided with a set of actions, parameters and end values. By defining the rules, the machine learning algorithm then tries to explore different options and possibilities, monitoring and evaluating each result to determine which one is optimal. Reinforcement learning teaches the machine trial and error. It learns from past experiences and begins to adapt its approach in response to the situation to achieve the best possible result.

1.1.2 APPLICATIONS OF MACHINE LEARNING

Web search engine: One of the reasons why search engine like google,bingetc work so well is because the system has learnt how to rank pages through a complex learning algorithm.

Photo tagging applications: Be it facebook or any other photo tagging application the ability to tag friends makes it even more happening. It is all possible because of a face recognition algorithm that runs behind the application.

Spam detector: Our mail agent like Gmail or Hotmail does a lot of hardwork for us in classifying the mails and moving the spam mails to the spam folder. This is again achieved by a spam classifier running in the backend of mail application.

CLUSTERING:

Clustering is the most common unsupervised learning technique. It is used for exploratory data analysis to find hidden patterns or grouping in data, Applications for cluster analysis include gene sequence analysis, market research and object recognition. Common algorithms for performing clustering include k-means and k-medoids, hierarchical clustering, Gaussian mixture models ,hidden Markov models, self organizing maps, fuzzy c-means clustering and subtractive clustering.

1.2 DIABATES DATA ANALYSIS :

People in today's world get affected by many diseases that do not have a complete cure. The development of one disease may lead to various other complications. One such disease is Type-2 Diabetes. It is a global health problem. This is the most common type of diabetes usually developed at the age of 40 and older. This increases the risk factors like kidney failure, heart disease, blindness, nerve damage and blood vessel damage. It is predicted from the characteristics of the patients.

Diabetes mellitus (DM) is a chronic disease that is characterized by high blood glucose. Nearly half of all diabetics have household heredity factors, which is one of the most important features of DM. Failure of the pancreas to produce enough insulin and the body's inefficient use insulin are both pathologic causes of DM. There are two types of DM. The pathogenesis of type 1 diabetes mellitus (T1DM) is that the pancreas secretes damaged β-cells, preventing it from lowering blood glucose level in time. Insulin resistance and insulin secretion deficiency are the pathogeneses of type 2 diabetes mellitus (T2DM), which is also called non-insulin dependent DM. In the past 30 years of development in China, with rising number of diabetics, people have started to realize that this chronic disease has deeply impacted every family and everyone’s daily life. There is an ascending trend in the proportion of diabetics in the general population, and the growth rate of male diabetics is higher than that of female diabetics. According to some official statistics, the number of diabetics in China was nearly 110 million in 2017. This means that China has the largest diabetic population in the world.

Type 2 diabetes

Type 2 diabetes is sometimes called non-insulin dependent diabetes or adult-onset diabetes. At least 90% of all cases of diabetes are victims of this type. It strikes a person due to insulin resistance and relative insulin deficiency, either of which may be present at the time that diabetes becomes clinically evident. The diagnosis of type 2 diabetes usually occurs after the age of 40 but can occur earlier, especially in populations with high diabetes prevalence. Type 2 diabetes can remain undetected for many years and the diagnosis is often made from associated complications or incidentally through an abnormal blood or urine glucose test. It is often, but not always, associated with obesity, which itself can cause insulin resistance and lead to elevated blood glucose levels. The normal range of fasting blood glucose level is between 4.0-5.6mmol/L. After consuming a meal, the blood glucose level rises in the blood and can reach up to 7.8mmol/L. Any value higher than these ranges indicates the prevalence of diabetes. After two hours of having a meal, the blood glucose level drops again. There is also a condition called pre-diabetes. It is that state, where the blood glucose level is higher than the normal range but not high enough to be stated as diabetes.

1.3 OBJECTIVE

1. To predict whether a patient has diabetes or not based on the patient's record.
2. To estimate the accuracy of the model.
3. To use the machine learning algorithms to improve the performance of the model.

1.4 CLASSIFICATION:

Classification techniques in data mining are capable of processing a large amount of data. It can be used to predict categorical class labels and classifies data based on training set and class labels and it can be used for classifying newly available data. The term could cover any context in which some decision or forecast is made on the basis of presently available information. Classification procedure is recognized method for repeatedly making such decisions in new situations. Classification techniques predict discrete responses for example whether an email is genuine or spam. Classification models classify input data into categories. Common algorithms for performing classification include support vector machine (SVM), Logistic regression, Decision tree, Random forest, k-nearest neighbor and neural networks.

1.4.1 LOGISTIC REGRESSION ALGORITHM

Logistic Regression can be used for various classification problems such as spam detection. Diabetes prediction, if a given customer will purchase a particular product or will they churn another competitor, whether the user will click on a given advertisement link or not, and many more examples are in the bucket.

Logistic Regression is one of the most simple and commonly used Machine Learning algorithms for two-class classification. It is easy to implement and can be used as the baseline for any binary classification problem. Its basic fundamental concepts are also constructive in deep learning. Logistic regression describes and estimates the relationship between one dependent binary variable and independent variables.

Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. For example, it can be used for cancer detection problems. It computes the probability of an event occurrence.

1.4.2 NAÏVE BAYES CLASSIFIER

Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets.

Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location. Even if these features are interdependent, these features are still considered independently. This assumption simplifies computation, and that's why it is considered as naive. This assumption is called class conditional independence

1.4.3 RANDOM FOREST CLASSIFIER

Random forest is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

Random forests has a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset.

Accuracy:

Accuracy is calculated as the number of all correct predictions divided by the total number of the dataset.

1.5 REGRESSION:

Regression techniques predict continuous responses – for example, changes in temperature or fluctuations in power demand. Typical applications include electricity load forecasting and algorithmic trading. Common regression algorithms includes step wise regression and bagged decision trees, neural networks and adaptive neuro-fuzzy learning.

1.6 PREDICTION:

Prediction or forecast is a statement about the future event. A prediction is often, but not always, based upon experiences or knowledge. It's sometimes based on facts or evidences, but not always. Common algorithms of predictive models are decision tree, k-means clustering and Bayesian interference, linear regression. The most complex area of predictive modeling is the neural network.

CHAPTER 2
LITERATURE SURVEY

In this section the literature survey has been carried out.The main focus is on the loan prediction model and improved accuracy. This gives clear idea for the proposed system.

[1] Jianchao Han, Juan C.Rodriguze, Mohsen Beheshti,proposed “Diabetes Data Analysis and Prediction Model Discovery Using Rapid miner”.Majority of the work has already been carried out in the area of predictive classification for the Pima Indian diabetes datasets. Decision tree using the Rapid miner tool has been used to build the prediction model on this datasets. This model considered that the Plasma Glucose attribute as the main attribute in predicting the disease. It has produced an accuracy of 72%.

[2] Asma A. AlJarullah proposed “Decision tree discovery for the diagnosis of type-2 Diabetes”. Another decision tree model has been built using the Weka’s J48 decision tree classifier on this dataset with an accuracy of 78.18%. Association rule mining has also been implemented on the same datasets. This produced large number of rules from the combinations of attributes.

[3] B.M. Patil, R.C. Joshi, Durga Toshniwal, proposed “Hybrid Prediction Model for Type-2 Diabetic patients”. Other classification methods like neural network approach obtained an accuracy of 75.4%, Bayesian approach achieved an accuracy of 79.5%. A number of classification algorithms on the same dataset have performed classification with accuracies in the range of 59.5-77.7%.

[4] C.Yue, L. Xin, X.Kewen and S Chang, proposed “An Intelligent Diagnosis to Type 2 Diabetes Based on QPSO algorithm and WLS-SVM”. The improvement in the prediction accuracy can also be obtained by using improved

Weighted Least Squares Support Vector Machine (WLS-SVM) based on Quantum Particle swarm Optimization (QPSO) algorithm as stated.

[5] S.M Nuwangi, C.R. Oruthotaarachchi, J.M.P.P Tilakaratna and H.A. Caldera, proposed usage of Association rules and Classification Techniques in Knowledge Extraction of Diabetes" . Two models namely Adaptive Neuro Fuzzy Inference System (ANFIS) and Rough Set Theory based methods have been developed. ANFIS method was found to be reliable as it produced better sensitivity and lesser Root Mean Square Error values than Rough Set Theory model for the dataset. These models have been built from the real dataset with some more additional features like Creatinine, Cholesterol, urine level, etc. These features were also helpful in assessing the risks related to diabetes. Earlier researches have also been carried out in predicting the risks related to the diabetes with some additional features and has revealed different accuracies. Association rule mining and classification techniques were useful in identifying the relationships edema and diabetes, and wheezes and diabetes as stated.

[6] Vijayalakshmi et al. developed a clustering algorithm used for predicting diabetes based on graph b-colouring technique. They implemented, performed experiments, and compared their approach with KNN Classification and K-means clustering. The results showed that the clustering based on graph colouring outperforms the other clustering approaches in terms of accuracy and purity. The proposed technique presented a real representation of clusters by dominant objects that assures the inter cluster disparity in a partitioning and used to evaluate the quality of clusters.

[7] An ensemble model of three classifiers is used on the PIDD by Pujari et al. Classification performance of SVM (support vector machine), discriminant analysis and Bayesian network was investigated individually with the help of gain

chart and response chart for both training and testing set. Results indicated that the ensemble model achieved an accuracy of 76.03% on test data set.

[8] Pradhan et al. proposed a model based on Neural Network and Fuzzy k-Nearest Neighbor Algorithm [8]. They first pre-processed the data by eliminating the records containing the missing values from the Pima Indian Diabetes Dataset. The Fuzzy k-Nearest Neighbor algorithm is used to train the Neural Networks. Finally, the entire training set is used as test set to calculate the classification accuracy.

[9] Han et al. used data mining technique through RapidMiner for diabetes data analysis and diabetes prediction model. A decision tree was used for prediction with 72 % of accuracy. ID3 Algorithm was also used for this purpose which gave 80 % accurate results.

[10] Breault proposed the idea of using rough sets on the PIDD for the first time. He first pre-processed the data and discretized it by making intervals of data. He used the equal frequency binning criteria for the same purpose. Then he created reducts by using Johnson reducer algorithm and classified using the batch classifier with the standard/tuned voting method (RSES). The rules were constructed for each of the 10 randomizations of the PIDD training sets from above. The test sets were classified according to defaults of the naïve Bayes classifier, and the 10 accuracies ranged from 69.6% to 85.5% with a mean of 73.8% and a 95% CI of (71.3%, 76.3%).

CHAPTER 3
SYSTEM ANALYSIS

3.1 EXSITING SYSTEM

Due to its continuously increasing occurrence, more and more families are influenced by diabetes mellitus. Most diabetics know little about their health quality or the risk factors they face prior to diagnosis. In this study, we have proposed a novel model based on data mining techniques for predicting type 2 diabetes mellitus (T2DM). The main problems that we are trying to solve are to improve the accuracy of the prediction model, and to make the model adaptive to more than one dataset. Based on a series of preprocessing procedures, the model is comprised of two parts, the improved K-means algorithm and the logistic regression algorithm. The Pima Indians Diabetes Dataset and the Waikato Environment for Knowledge Analysis toolkit were utilized to compare our results with the results from other researchers. The conclusion shows that the model attained a 3.04% higher accuracy of prediction than those of other researchers. Moreover, our model ensures that the dataset quality is sufficient. To further evaluate the performance of our model, we applied it to two other diabetes datasets. Both experiments' results show good performance. As a result, the model is shown to be useful for the realistic health management of diabetes.

Points to be noted:

1. Hybrid Prediction Model
2. Data Mining
3. Diabetes Mellitus

3.1.1 DISADAVANTAGES

1. Achieved a Low accuracy.
2. Time consuming process

3.2 PROPOSED SYSTEM

In our proposed work, the Pima Indian Diabetes Dataset is obtained. The Pima Indian Diabetes Dataset consists of information on patients. In order to lower the morbidity and reduce the influence of DM, it is vital for us to focus on a high-risk group of people with DM. According to the definitions of groups with a high risk of DM are as follows:

1. Age ≥ 45 and seldom exercising
2. BMI ≥ 24kg/m2
3. Impaired glucose tolerance (IGT) or impaired fasting glucose (IFG)
4. Family history of DM
5. Lower high-density lipoprotein cholesterol or hypertriglyceridemia (HTG)
6. Hypertension or cardiovascular and cerebrovascular disease
7. Gestation female whose age ≥30

The above kinds of high risk group of people has been taken to predict the accuracy. It is used to compare the result to attain higher accuracy. Naïve bayes algorithm is depending upon likelihood and probability it is fast and stable to data changes. Naive bayes is used to train the dataset.

Logistic Regression, calculates the relationship of each feature and weights them based on their impact on results. Random forest algorithm is an ensemble algorithm, fits multiple trees with subset of data and averages tree result to improve performance and control over-fitting. The main problems is to solve are to improve the higher accuracy of prediction model for Type 2 Diabetes Mellitus. To further evaluate the performance of the model, Logistics and Random forest algorithm techniques are used. Both the algorithms show the good performance that the data set quality is sufficient. 99% of accuracy has been achieved using the logistics and Random forests algorithm. The model is shown to be useful for realistic health management of diabetes.

3.2.1ADVANTAGES

1. Reliable and high accuracy(99%).
2. Probable loss will be reduced.
3. Fast processing system.

CHAPTER 4
SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

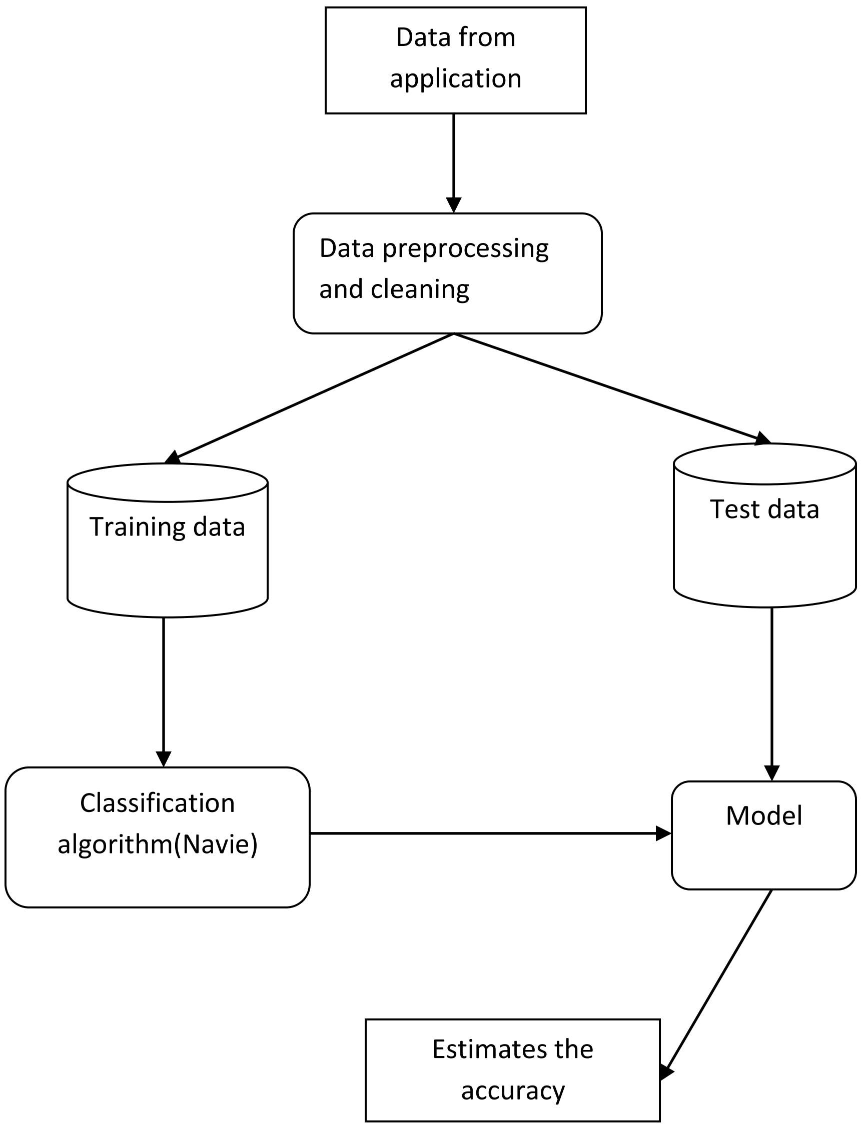


Figure 4.1: Architecture of the proposed system

The above Figure 4.1 shows the architecture diagram of the proposed system. Here the training data is taken and normalized. The model is trained by using Naïve bayes classifier .After that the training and test data is tested using classifiers to give the model accuracy which is used to indicate the model performance.

4.2 MODULE DESCRIPTION

1. Data Selection
2. Data Pre-Processing
3. Features Selection
4. Building Classification Model
5. Predicting Class Labels of Test Dataset
6. Evaluating Predictions

4.2.1 DATASET SELECTION

Data is a piece of information that should be collected carefully so that the collected information is useful. Data collection is the process of gathering information or processing information that is used for obtaining outcomes in experiments. The data collected for learnig process may contain missing values, noise or inconsistency. This leads to produce inconsistent information from the learning process. To improve the quality of data and consequently the results, the collected data is to be pre processed so as to improve the efficiency of data mining process. The dataset after selecting and understanding is loaded into Jupyter notebook.

4.2.2 DATA PREPROCESSING

The quality of the data, to a large extent, affects the result of prediction. This means that data preprocessing plays an important role in the model . In this study, we have selected some appropriate methods to optimize the original dataset.

First, we have analyzed each attribute’s medical implication and its correlation to DM. We determined that the number of pregnancies has little connection with DM. Therefore, we transformed this numeric attribute into a nominal attribute. The value 0 indicates non-pregnant and 1 indicates pregnant. The complexity of the dataset was reduced by this process. Second, there are some missing and incorrect values in the dataset due to errors or deregulation. Most of the inaccurate experimental results were caused by these meaningless values. For example, in the original dataset, the values of diastolic blood pressure and body mass index could not be 0, which indicates that the real value was missing. To reduce the influence of meaningless values, we used the means from the training data to replace all missing values.

4.2.3 DATA CLEANING

The main aim of Data Cleaning is to identify and remove errors & duplicate data, in order to create a reliable dataset. This improves the quality of the training data for analytics and enables accurate decision-making. Needless to say, data cleansing is a time-consuming process and most data scientists spend an enormous amount of time in enhancing the quality of the data. However, there are various methods to identify and classify data for data cleansing.

Methods for Identifying and Classifying Data Errors

There are mainly two distinct techniques, namely Qualitative and Quantitative techniques to classify data errors. Qualitative techniques involve rules, constraints, and patterns to identify errors. On the other hand, Quantitative techniques employ statistical techniques to identify errors in the trained data. Once the errors are identified using these techniques, they can be rectified by making changes in the script or by human intervention, or with a combination of both.

Data Cleaning consists of two basic stages, first is error identification and second is error solving. For any data cleaning activity, the first step is to identify the anomalies. No matter which technique you employ to analyze errors, your technique should involve three questions that should be addressed:

i. What types of errors to identify?
ii. How to identify these errors?
iii. Where to identify these errors?

Answering these questions will help you clean the data and improve the quality of your machine learning data. Apart from this, there are certain best practices that can be used for data error identification and data cleaning. Let’s take a look at things to consider when it comes to data cleaning.

4.2.4 Steps for test and train dataset

The model is initially fit on a training dataset, that is a set of examples used to fit the parameters (e.g. weights of connections between neurons in artificial neural networks) of the model.The model (e.g. a neural net or a naive Bayes classifier) is trained on the training dataset using a supervised learning method (e.g. gradient descent or stochastic gradient descent). In practice, the training dataset often consist of pairs of an input vector (or scalar) and the corresponding output vector (or scalar), which is commonly denoted as the target (or label). The current model is run with the training dataset and produces a result, which is then compared with the target, for each input vector in the training dataset. Based on the result of the comparison and the specific learning algorithm being used, the parameters of the model are adjusted. The model fitting can include both variable selection and parameter estimation.

Successively, the fitted model is used to predict the responses for the observations in a second dataset called the validation dataset. The validation dataset provides an unbiased evaluation of a model fit on the training dataset while tuning the model's hyper parameters (e.g. the number of hidden units in a neural network). Validation datasets can be used for regularization by early stopping: stop training when the error on the validation dataset increases, as this is a sign of overfitting to the training dataset. This simple procedure is complicated in practice by the fact that the validation dataset's error may fluctuate during training, producing multiple local minima. This complication has led to the creation of many ad-hoc rules for deciding when overfitting has truly begun.

Finally, the test dataset is a dataset used to provide an unbiased evaluation of a final model fit on the training dataset. When the data in the test dataset has never been used in training (for example in cross-validation), the test dataset is also called a holdout dataset.

4.2.5 Test dataset

A test dataset is a dataset that is independent of the training dataset, but that follows the same probability distribution as the training dataset. If a model fit to the training dataset also fits the test dataset well, minimal overfitting has taken place . A better fitting of the training dataset as opposed to the test dataset usually points to overfitting.

A test set is therefore a set of examples used only to assess the performance (i.e. generalization) of a fully specified classifier.

4.2.6 Training dataset

A training dataset is a dataset of examples used for learning, that is to fit the parameters (e.g., weights) of, for example, a classifier.

Most approaches that search through training data for empirical relationships tend to overfit the data, meaning that they can identify and exploit apparent relationships in the training data that do not hold in general.

In machine learning, the study and construction of algorithms that can learn from and make predictions on data is a common task. Such algorithms work by making data-driven predictions or decisions through building a mathematical model from input data.

The data used to build the final model usually comes from multiple datasets. In particular, three data sets are commonly used in different stages of the creation of the model.

4.3MODEL BUILDING

4.3.1TRAINING DATA

DATASET DESCRIPTION:

No.of instances	= 768
No.of attributes	= 9
Test data	=30%
Training data	=70%

ATTRIBUTES:

i. Number of times pregnant (preg)
ii. Plasma glucose concentration at 2 hours in an oral glucose tolerance test (plas)
iii. Diastolic blood pressure (pres)

iv. Triceps skin fold thickness (skin)
v. 2-hour serum insulin (insu)
vi. Body mass index (bmi)
vii. Diabetes pedigree function (pedi)
viii. Age (age)
ix. Class variable (class)

4.4 APPLY ML ALGORITHMS

4.4.1 NAÏVE BAYES CLASSIFICATION

Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets.

Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location. Even if these features are interdependent, these features are still considered independently. This assumption simplifies computation, and that's why it is considered as naive. This assumption is called class conditional independence.

4.4.2 LOGISTIC REGRESSION

Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. For example, it can be used for cancer detection problems. It computes the probability of an event occurrence.

It is a special case of linear regression where the target variable is categorical in nature. It uses a log of odds as the dependent variable. Logistic

Regression predicts the probability of occurrence of a binary event utilizing a logit function.

4.4.3 RANDOM FOREST CLASSIFIER

Random forest is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

Random forest has a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset.

4.4.4 EXPERIMENTAL RESULTS

The data obtained from Patient application is used to analyze the accuracy of the proposed scheme for diabetes prediction. Also notice that the accuracy of the proposed scheme gradually increases with the growth of the size of the training dataset**.**

CHAPTER 5

SYSTEM SPECIFICATION

5.1 SOFTWARE REQUIREMENTS

Operating system	-Windows 10
Tools	-Anaconda Navigator3.7
Coding Language	-Python
Editor	-Jupyter Notebook

5.2 HARDWARE REQUIREMENTS

System processor	- Pc or laptop with core i3 and more
RAM	- 4GB
Hard Disk	- 500GB

5.3 ABOUT THE SOFTWARE

ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments and channels without the need to use command line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, Mac OS and Linux. Anaconda is free and easy to install, and it offers free community support.

Applications Available in Navigator

i. JupyterLab
ii. Jupyter Notebook
iii. QTConsole
iv. Spyder
v. VSCode

vi. GlueViz
vii. Orange 3 App
viii. Rstudio

JUPYTER NOTEBOOK

A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

A Jupyter Notebook can be converted to a number of open standard output formats (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through "Download As" in the web interface, via the nbconvert library or "jupyter nbconvert" command line interface in a shell. To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

Jupyter Notebook provides a browser-based REPL built upon a number of popular open-source libraries:

i. IPython
ii. ØMQ
iii. jQuery
iv. Bootstrap (front-end framework)
v. MathJax

Jupyter Notebook can connect to many kernels to allow programming in many languages. By default Jupyter Notebook ships with the IPython kernel. As of the 2.3 release(October 2014), there are currently 49 Jupyter-compatible kernels for as many programming languages, including Python, R, Julia and Haskell.

The Notebook interface was added to IPython in the 0.12 release (December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0). Jupyter Notebook is similar to the notebook interface of other programs such as Maple, Mathematica, and SageMath, a computational interface style that originated with Mathematica in the 1980s.

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, Numerical simulation, statistical modeling, data visualization, machine learning and much more. To support interactive data science and scientific computing across all programming languages.

THE PYTHON LANGUAGE

Python is an interoperated, high level, general-purpose programming language with dynamic semantics. Its high-level in data structures, combined with dynamic typing and binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect the existing components together. Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. There is no compilation step, edit-test-debug cycle is incredibly fast. The python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

FEATURES

i. Interpreted
ii. Platform Independent
iii. Free and open source
iv. Redistributable
v. Embeddable
vi. Robust
vii. Rich Library Support

SCIKIT – LEARN

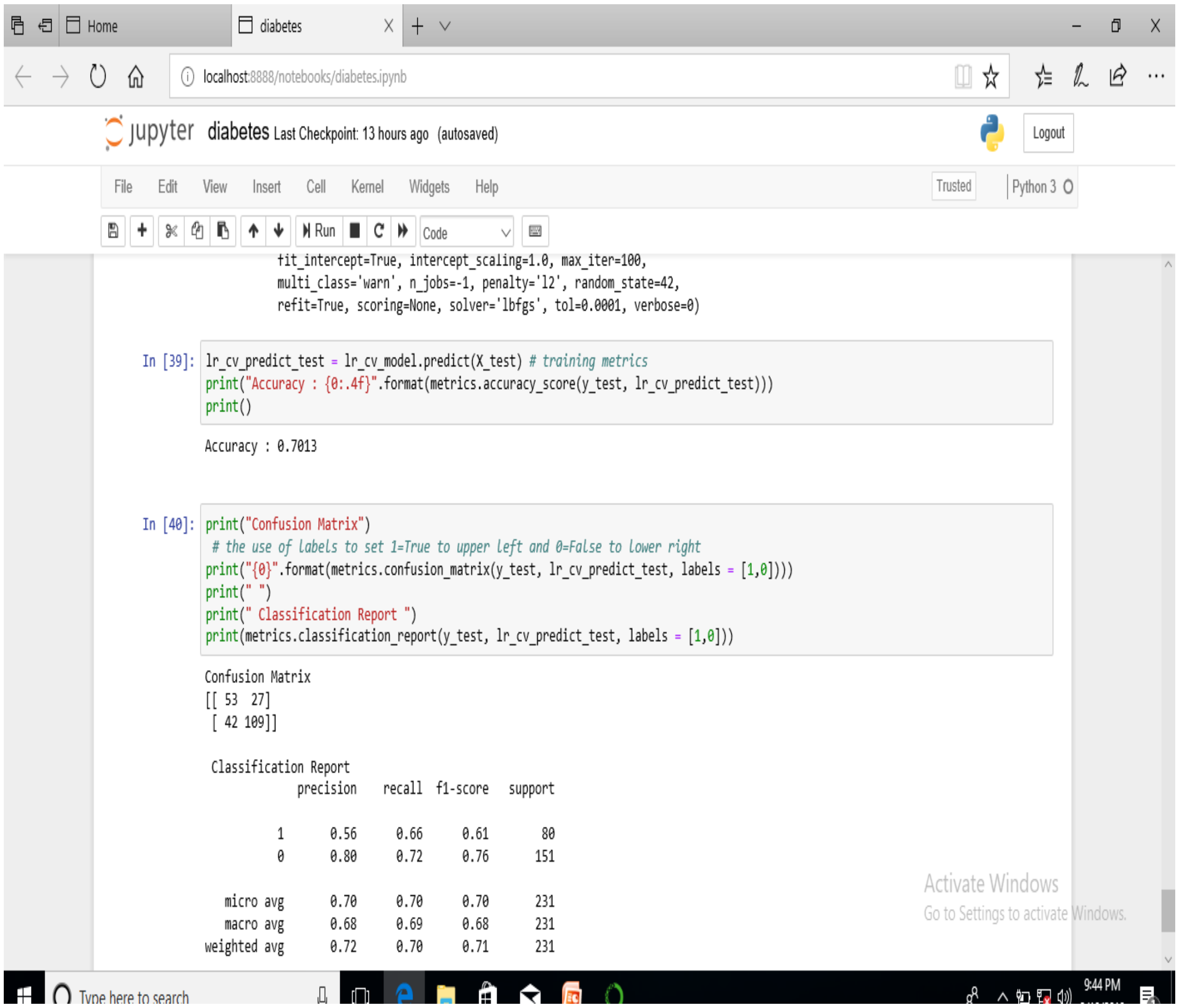
Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface In Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

Simple and efficient tools for data mining and data analysis. Accessible to everybody, and reusable in various contexts. Built on Numpy, Scipy, and Matplotlib.It is open source, commercially usable-BSD license.

Scikit –learn (formerly scikits.learn) is a free software machine learning library for the python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the python numerical and scientific libraries Numpy and Scipy.

CHAPTER 6

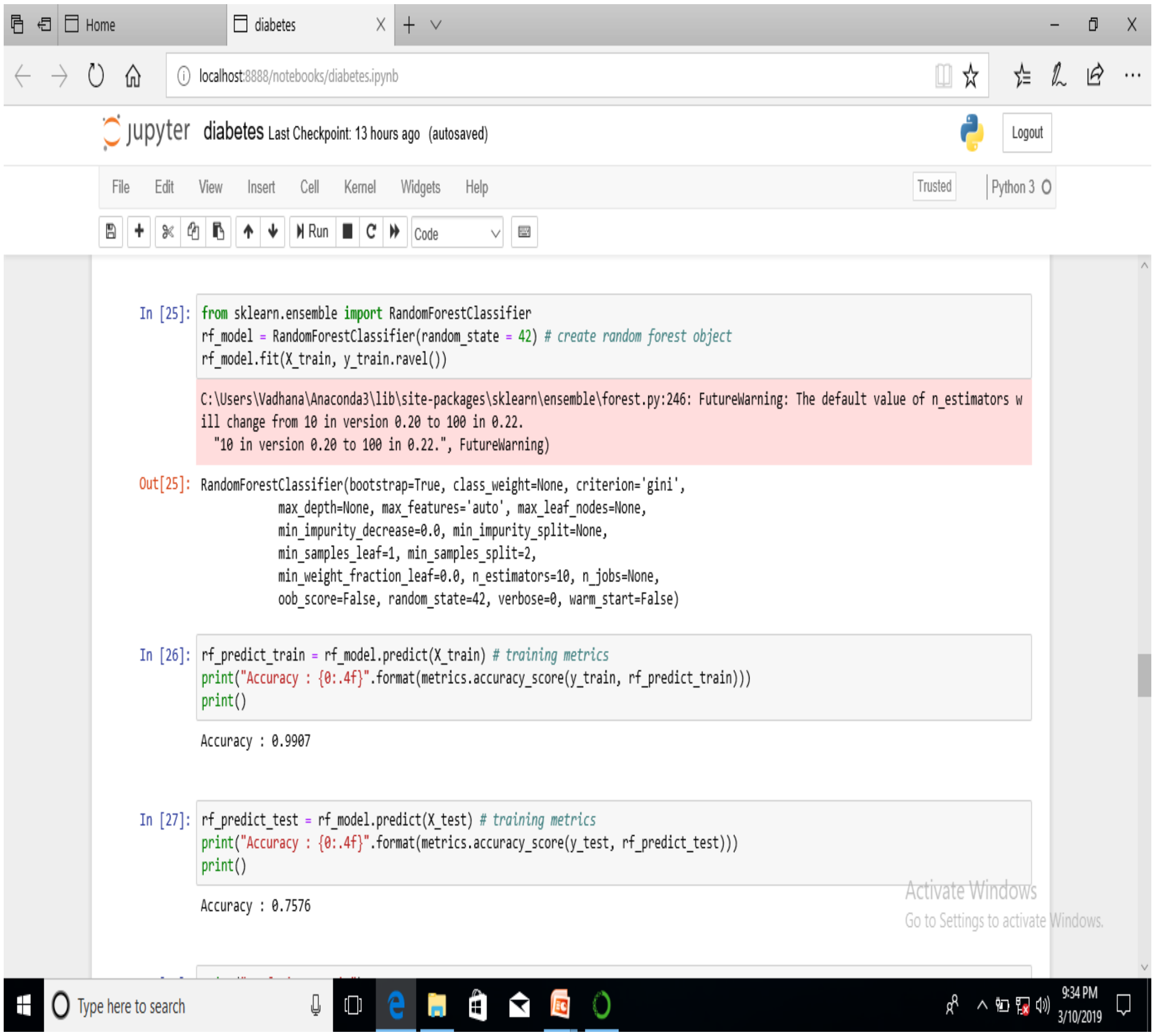
RESULT&DISCUSSION


Home
diabetes
localhost:8888/notebooks/diabetes.ipynb
jupyter diabetes Last Checkpoint: 13 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Activate Windows
Go to Settings to activate Windows.
Type here to search
9:44 PM

```
                fit_intercept=True, intercept_scaling=1.0, max_iter=100,
                multi_class='warn', n_jobs=-1, penalty='l2', random_state=42,
                refit=True, scoring=None, solver='lbfgs', tol=0.0001, verbose=0)

In [39]: lr_cv_predict_test = lr_cv_model.predict(X_test) # training metrics
         print("Accuracy : {0:.4f}".format(metrics.accuracy_score(y_test, lr_cv_predict_test)))
         print()

         Accuracy : 0.7013

In [40]: print("Confusion Matrix")
          # the use of labels to set 1=True to upper left and 0=False to lower right
         print("{0}".format(metrics.confusion_matrix(y_test, lr_cv_predict_test, labels = [1,0])))
         print(" ")
         print(" Classification Report ")
         print(metrics.classification_report(y_test, lr_cv_predict_test, labels = [1,0]))

         Confusion Matrix
         [[ 53  27]
          [ 42 109]]

          Classification Report
                       precision    recall  f1-score   support

                    1       0.56      0.66      0.61        80
                    0       0.80      0.72      0.76       151

            micro avg       0.70      0.70      0.70       231
            macro avg       0.68      0.69      0.68       231
         weighted avg       0.72      0.70      0.71       231
```

Fig 6.1 Logistics regression

Fig 6.1 shows the result of logistics regression.The accuracy of logistics regression range in 70% .


Home
diabetes
localhost:8888/notebooks/diabetes.ipynb
jupyter diabetes Last Checkpoint: 13 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Activate Windows
Go to Settings to activate Windows.
Type here to search
9:34 PM
3/10/2019

```
In [25]: from sklearn.ensemble import RandomForestClassifier
         rf_model = RandomForestClassifier(random_state = 42) # create random forest object
         rf_model.fit(X_train, y_train.ravel())

         C:\Users\Vadhana\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The default value of n_estimators w
         ill change from 10 in version 0.20 to 100 in 0.22.
           "10 in version 0.20 to 100 in 0.22.", FutureWarning)

Out[25]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                     max_depth=None, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
                     oob_score=False, random_state=42, verbose=0, warm_start=False)

In [26]: rf_predict_train = rf_model.predict(X_train) # training metrics
         print("Accuracy : {0:.4f}".format(metrics.accuracy_score(y_train, rf_predict_train)))
         print()

         Accuracy : 0.9907

In [27]: rf_predict_test = rf_model.predict(X_test) # training metrics
         print("Accuracy : {0:.4f}".format(metrics.accuracy_score(y_test, rf_predict_test)))
         print()

         Accuracy : 0.7576
```

Fig 6.2 Random forest

Fig 6.2 shows the result of random forest.The accuracy of logistics regression range in 99% . Random forest show the higher accuracy to predict the type 2 diabetes mellitus patient.

CHAPTER 7

CONCLUSION

This approach can help hospital in predicting the future of diabetes patients. Using this approach hospitals can reduce the number of diabetes patients. Using the naïve bayes built the prediction model and some other ML algorithms (logistic regression, random forest) had been used in this approach. Among all those, we achieved the higher accuracy of **(99%)** while using the NAÏVE BAYES algorithm than in the other two ML algorithms. Data Mining techniques are very useful to the hospital for better predictions and acquiring new patients , fraud detection in real time, providing segment based products, analysis of the patients, risk management.

FUTURE WORKS

For future work, it is necessary to bring in hospital’s real and latest patients’ data for continuous training and optimization of our proposed model. The quantity of the dataset should be large enough for training and predicting. Some advanced algorithms and models should be applied in the study of DM. Grading forecasting standards are also necessary for potential diabetes patients. Developing a series of rules and standards is a valid method to prevent people from developing DM. Based on that, a more effective model for predicting DM and grading potential patients is presented. This will help to lower the growth rate of diabetes and eventually decrease the risk of developing DM.

CHAPTER 8

APPENDIXES

APPENDIX 1 : SOURCE CODE

Import Libraries

```
In [1]:
import pandas as pd                     # pandas is a dataframe library
import matplotlib.pyplot as plt            # matplotlib.pyplot plot data
import numpy as np                      # numpy provides N-dim object support

# do ploting line instead of in a separate window
%matplotlib inline
```

Load and review data

```
In [2]: df = pd.read_csv("./data/pima-data.csv")
In [3]: df.shape
In [4]: df.head(5)
```

Cleaning the Data : Check for null values in data frame

```
In [6]: df.isnull().values.any()

def plot_corr(df, size=11):
"""
function plots a graphical coreleastion matrix for each pair of column in different
dataframe .
Input :
df : pandas dataframe
corr = df.corr() # data frame corelation function
```

```
fig, ax = plt.subplots(figsize =(size,size))
ax.matshow(corr) # color code the rectanges by corelation value
plt.xticks (range(len(corr.columns)), corr.columns) # draw x ticks mark
plt.yticks (range(len(corr.columns)), corr.columns) # draw y ticks mark
```

Splitting the data

70% for training , 30% for testing

In [17]:

```
from sklearn.cross_validation import train_test_split
feature_col_names = ['num_preg', 'glucose_conc' , 'diastolic_bp', 'thickness',
'insulin', 'bmi', 'diab_pred', 'age']
predicted_class_names = ['diabetes']
X = df[feature_col_names].values # predictor feature coloumns ( 8 X m )
y = df[predicted_class_names].values # predicted class ( 1= true , 0=false ) column
split_test_size = 0.30
X_train , X_test , y_train , y_test = train_test_split(X, y, test_size = split_test_size,
random_state = 42)
```

Naive Bayes algorithim

In [22]:

```
from sklearn.naive_bayes import GaussianNB
# Create Gaussian naive bayes model object and train it with the data
nb_model = GaussianNB()
nb_model.fit(X_train, y_train.ravel())
nb_predict_train = nb_model.predict(X_train)

# import the performance metrices library
```

```
from sklearn import metrics
# Accuracy
print("Accuracy : {0:.4f}".format(metrics.accuracy_score(y_train,
nb_predict_train)))
print()
In [24]: # predict values using the Texting data
nb_predict_text = nb_model.predict(X_test)
# import the performance metrices library
from sklearn import metrics
# Accuracy
print("Accuracy : {0:.4f}".format(metrics.accuracy_score(y_test,
nb_predict_text)))
print()
```

Random Forest algorithim

In [26]:

```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(random_state = 42) # create random forest
object
rf_model.fit(X_train, y_train.ravel())
```

Predicting Training data

In [27]:

```
rf_predict_train = rf_model.predict(X_train)
# training metrics
print("Accuracy : {0:.4f}".format(metrics.accuracy_score(y_train,
rf_predict_train)))
print()
```

Predicting Test data

In [28]:

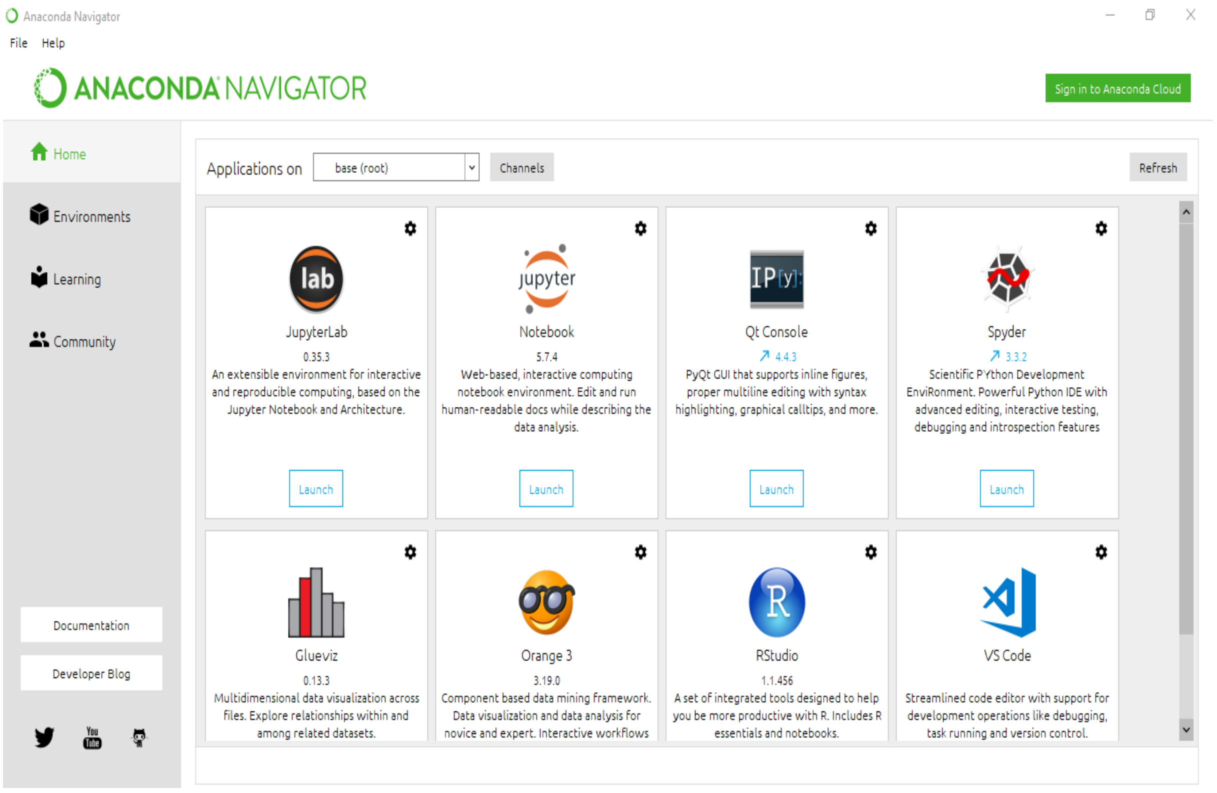
```
rf_predict_test = rf_model.predict(X_test)
# training metrics
print("Accuracy : {0:.4f}".format(metrics.accuracy_score(y_test, rf_predict_test)))
print()
```

Logistic Regression

In [30]:

```
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression(C=0.7, random_state=42) # create random forest
object
lr_model.fit(X_train, y_train.ravel())
```

APPENDIX 2: SCREENSHOT



JupyterLab
0.35.3
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
5.7.4
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Qt Console
4.4.3
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
3.3.2
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

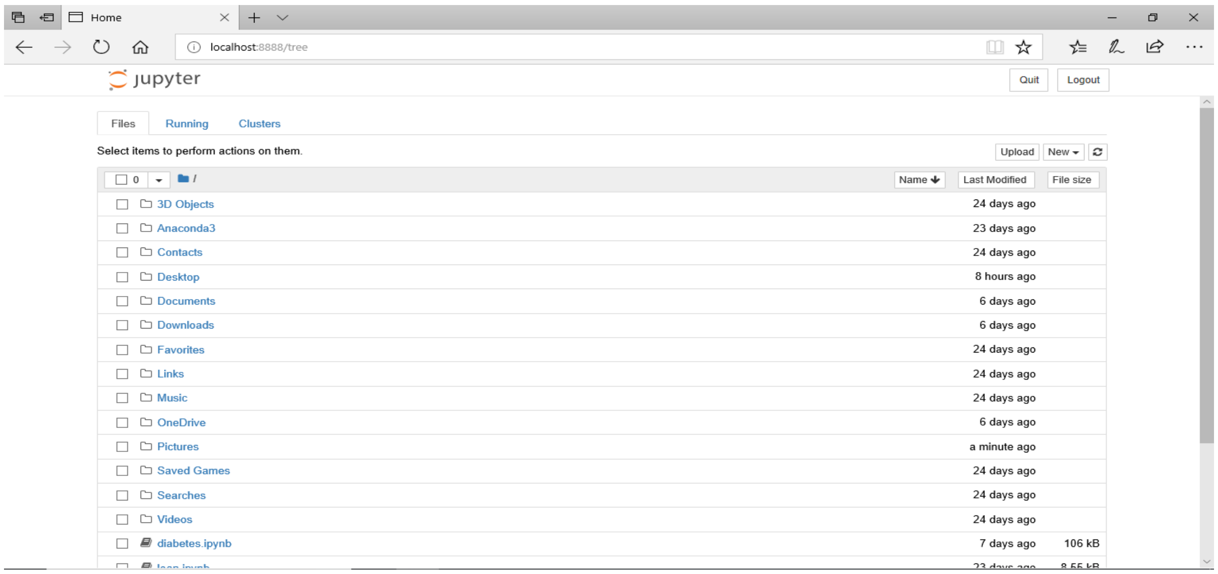
Glueviz
0.13.3
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3
3.19.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

VS Code
Streamlined code editor with support for development operations like debugging, task running and version control.

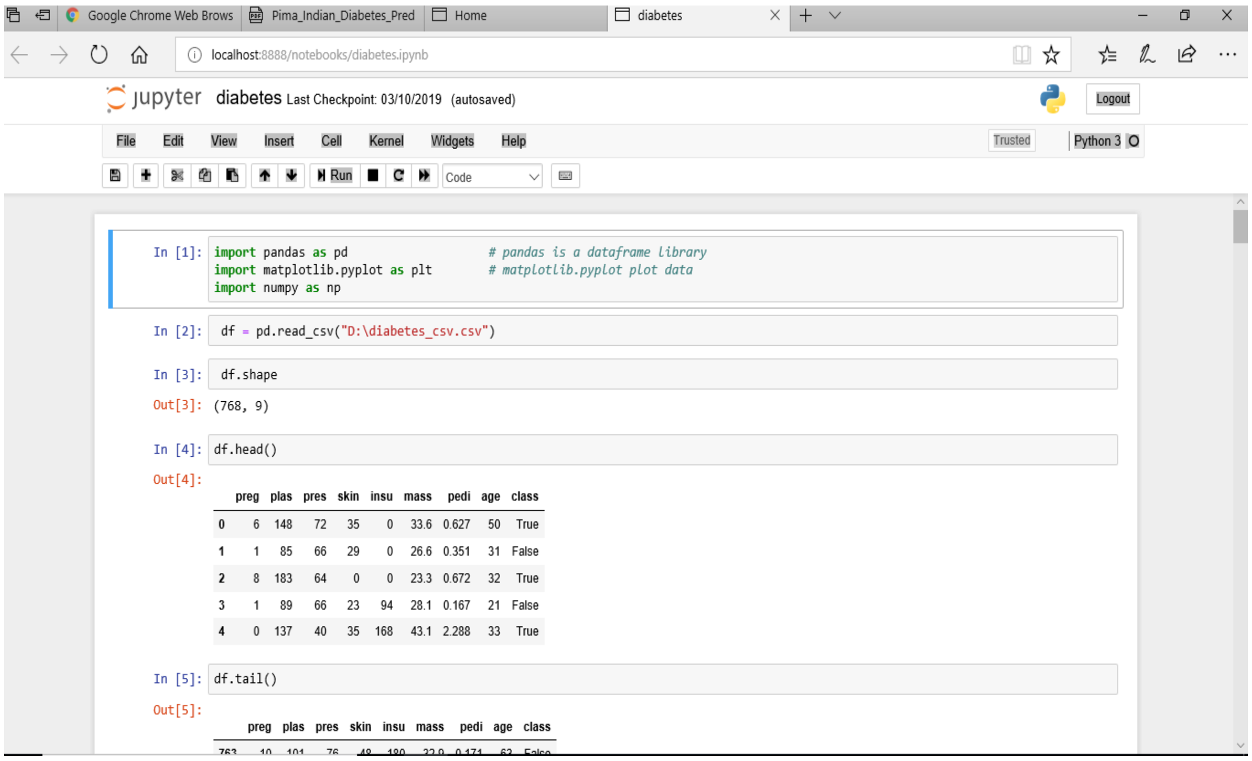
ANACONDA NAVIGATOR



Select items to perform actions on them.

Name	Last Modified	File size
3D Objects	24 days ago	
Anaconda3	23 days ago	
Contacts	24 days ago	
Desktop	8 hours ago	
Documents	6 days ago	
Downloads	6 days ago	
Favorites	24 days ago	
Links	24 days ago	
Music	24 days ago	
OneDrive	6 days ago	
Pictures	a minute ago	
Saved Games	24 days ago	
Searches	24 days ago	
Videos	24 days ago	
diabetes.ipynb	7 days ago	106 kB

JUPYTER FILES



localhost:8888/notebooks/diabetes.ipynb

jupyter diabetes Last Checkpoint: 03/10/2019 (autosaved)

```
In [1]: import pandas as pd               # pandas is a dataframe library
        import matplotlib.pyplot as plt   # matplotlib.pyplot plot data
        import numpy as np
```

```
In [2]: df = pd.read_csv("D:\diabetes_csv.csv")
```

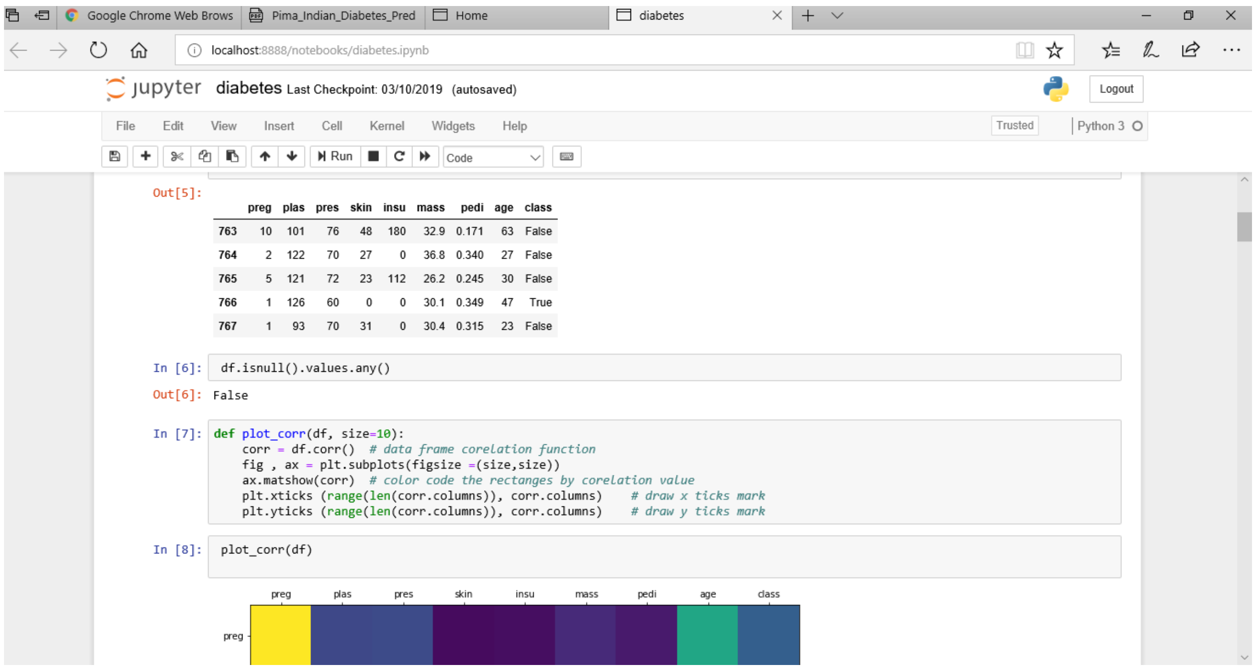
```
In [3]: df.shape
Out[3]: (768, 9)
```

```
In [4]: df.head()
Out[4]:
```

	preg	plas	pres	skin	insu	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	True
1	1	85	66	29	0	26.6	0.351	31	False
2	8	183	64	0	0	23.3	0.672	32	True
3	1	89	66	23	94	28.1	0.167	21	False
4	0	137	40	35	168	43.1	2.288	33	True

```
In [5]: df.tail()
Out[5]:
```

IMPORT LIBRARIES



localhost:8888/notebooks/diabetes.ipynb

jupyter diabetes Last Checkpoint: 03/10/2019 (autosaved)

```
Out[5]:
```

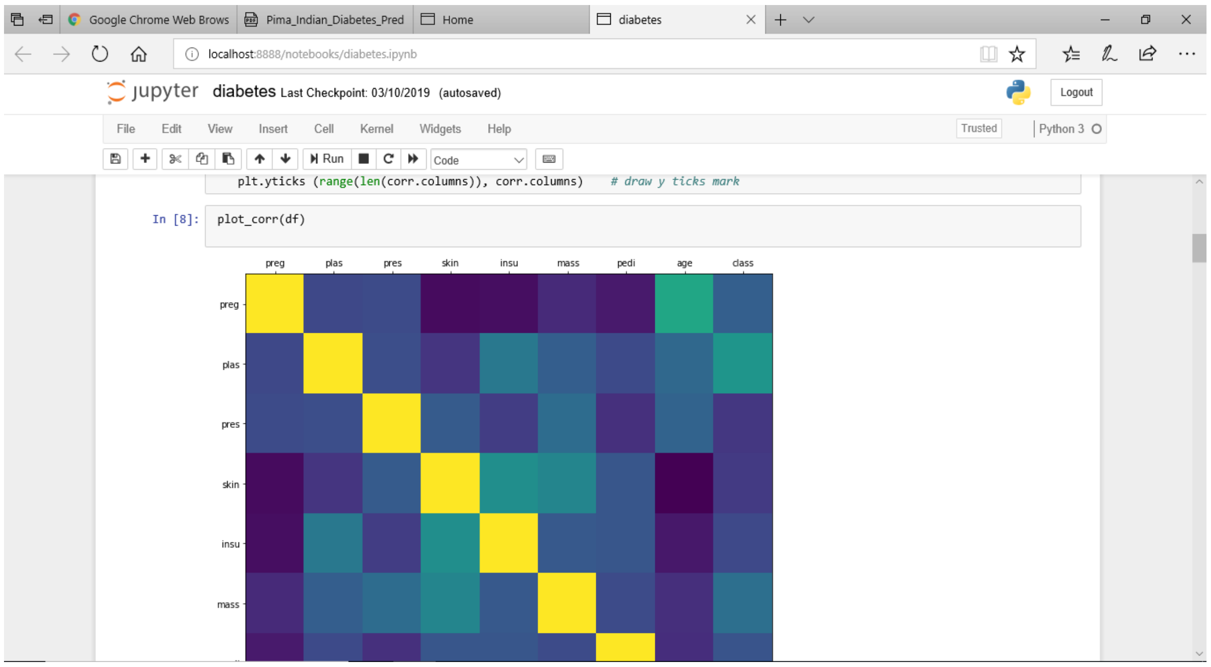
	preg	plas	pres	skin	insu	mass	pedi	age	class
763	10	101	76	48	180	32.9	0.171	63	False
764	2	122	70	27	0	36.8	0.340	27	False
765	5	121	72	23	112	26.2	0.245	30	False
766	1	126	60	0	0	30.1	0.349	47	True
767	1	93	70	31	0	30.4	0.315	23	False

```
In [6]: df.isnull().values.any()
Out[6]: False
```

```
In [7]: def plot_corr(df, size=10):
            corr = df.corr()  # data frame corelation function
            fig , ax = plt.subplots(figsize =(size,size))
            ax.matshow(corr)  # color code the rectanges by corelation value
            plt.xticks (range(len(corr.columns)), corr.columns)    # draw x ticks mark
            plt.yticks (range(len(corr.columns)), corr.columns)    # draw y ticks mark
```

```
In [8]: plot_corr(df)
```

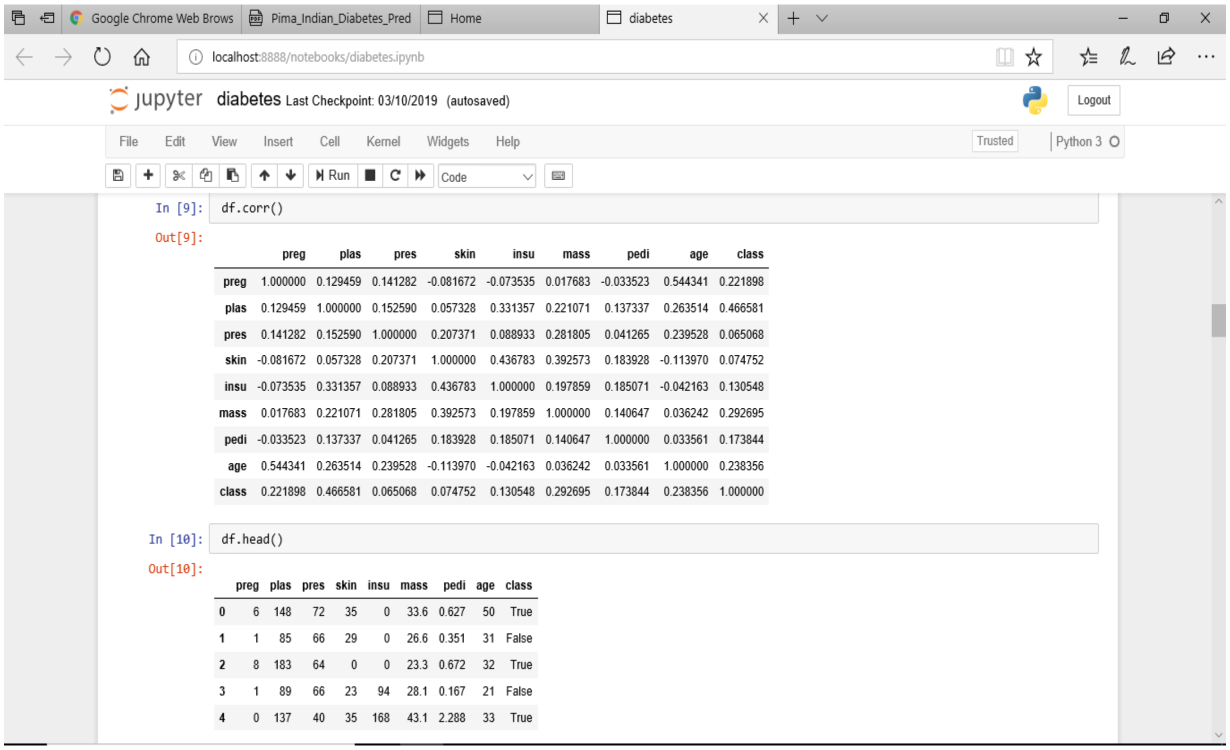
CHECK FOR NULL VALUES



```
plt.yticks (range(len(corr.columns)), corr.columns)    # draw y ticks mark
```

In [8]:
```
plot_corr(df)
```

CORRELATION MATRIX



In [9]:
```
df.corr()
```

Out[9]:

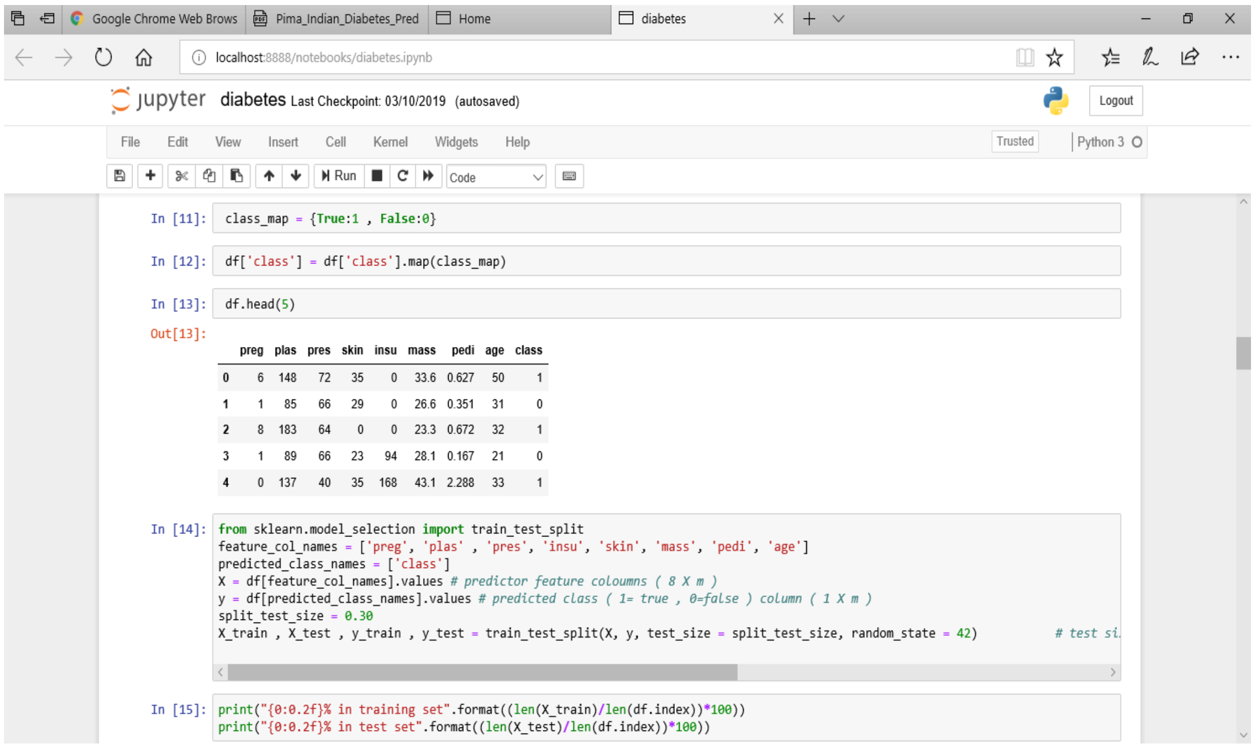
	preg	plas	pres	skin	insu	mass	pedi	age	class
preg	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
plas	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
pres	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
skin	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
insu	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
mass	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
pedi	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
class	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

In [10]:
```
df.head()
```

Out[10]:

	preg	plas	pres	skin	insu	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	True
1	1	85	66	29	0	26.6	0.351	31	False
2	8	183	64	0	0	23.3	0.672	32	True
3	1	89	66	23	94	28.1	0.167	21	False
4	0	137	40	35	168	43.1	2.288	33	True

VALUES OF CORRELATION MATRIX



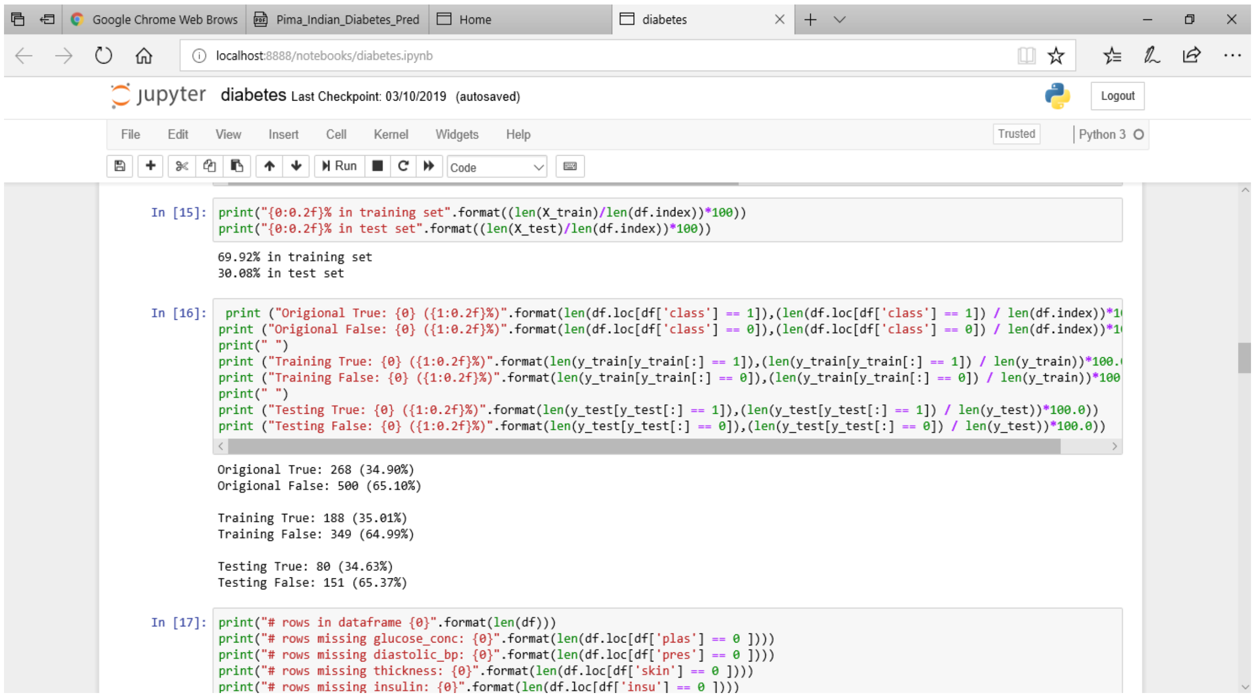
Google Chrome Web Brows
Pima_Indian_Diabetes_Pred
Home
diabetes
localhost:8888/notebooks/diabetes.ipynb
jupyter diabetes Last Checkpoint: 03/10/2019 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [11]:  class_map = {True:1 , False:0}

In [12]:  df['class'] = df['class'].map(class_map)

In [13]:  df.head(5)

Out[13]:
```

	preg	plas	pres	skin	insu	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
In [14]:  from sklearn.model_selection import train_test_split
          feature_col_names = ['preg', 'plas' , 'pres', 'insu', 'skin', 'mass', 'pedi', 'age']
          predicted_class_names = ['class']
          X = df[feature_col_names].values # predictor feature coloumns ( 8 X m )
          y = df[predicted_class_names].values # predicted class ( 1= true , 0=false ) column ( 1 X m )
          split_test_size = 0.30
          X_train , X_test , y_train , y_test = train_test_split(X, y, test_size = split_test_size, random_state = 42)            # test si

In [15]:  print("{0:0.2f}% in training set".format((len(X_train)/len(df.index))*100))
          print("{0:0.2f}% in test set".format((len(X_test)/len(df.index))*100))
```

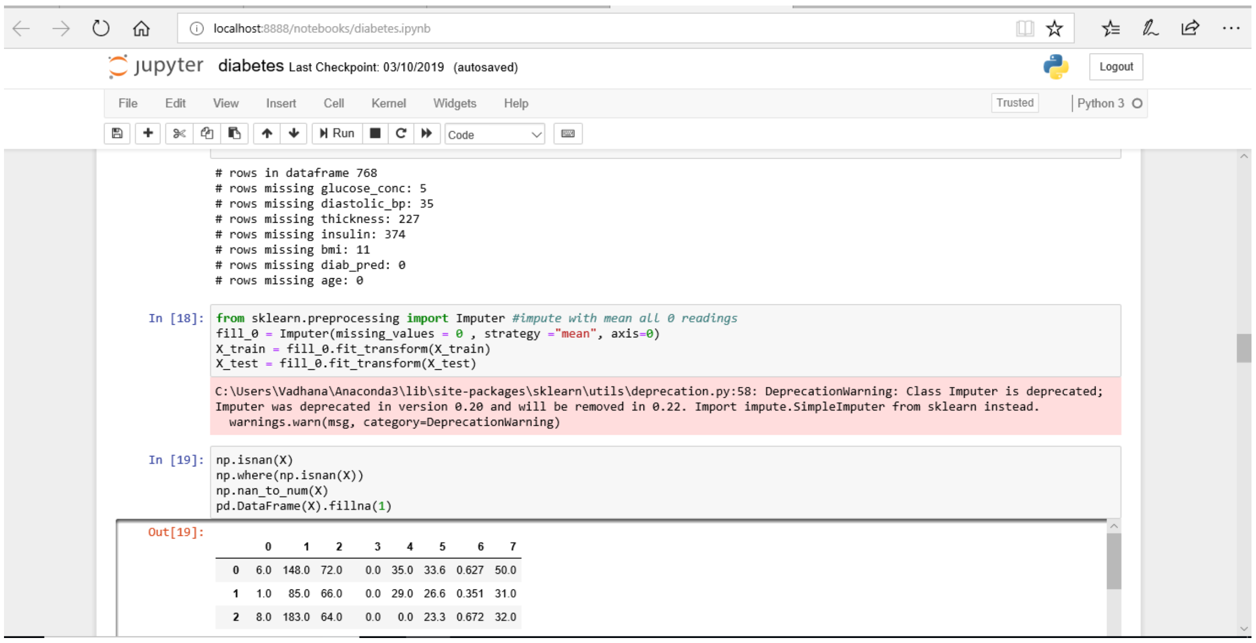
CHECK DATA TYPES



Google Chrome Web Brows
Pima_Indian_Diabetes_Pred
Home
diabetes
localhost:8888/notebooks/diabetes.ipynb
jupyter diabetes Last Checkpoint: 03/10/2019 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [15]:  print("{0:0.2f}% in training set".format((len(X_train)/len(df.index))*100))
          print("{0:0.2f}% in test set".format((len(X_test)/len(df.index))*100))

          69.92% in training set
          30.08% in test set

In [16]:   print ("Origional True: {0} ({1:0.2f}%)".format(len(df.loc[df['class'] == 1]),(len(df.loc[df['class'] == 1]) / len(df.index))*1
          print ("Origional False: {0} ({1:0.2f}%)".format(len(df.loc[df['class'] == 0]),(len(df.loc[df['class'] == 0]) / len(df.index))*1
          print(" ")
          print ("Training True: {0} ({1:0.2f}%)".format(len(y_train[y_train[:] == 1]),(len(y_train[y_train[:] == 1]) / len(y_train))*100.
          print ("Training False: {0} ({1:0.2f}%)".format(len(y_train[y_train[:] == 0]),(len(y_train[y_train[:] == 0]) / len(y_train))*100
          print(" ")
          print ("Testing True: {0} ({1:0.2f}%)".format(len(y_test[y_test[:] == 1]),(len(y_test[y_test[:] == 1]) / len(y_test))*100.0))
          print ("Testing False: {0} ({1:0.2f}%)".format(len(y_test[y_test[:] == 0]),(len(y_test[y_test[:] == 0]) / len(y_test))*100.0))

          Origional True: 268 (34.90%)
          Origional False: 500 (65.10%)

          Training True: 188 (35.01%)
          Training False: 349 (64.99%)

          Testing True: 80 (34.63%)
          Testing False: 151 (65.37%)

In [17]:  print("# rows in dataframe {0}".format(len(df)))
          print("# rows missing glucose_conc: {0}".format(len(df.loc[df['plas'] == 0 ])))
          print("# rows missing diastolic_bp: {0}".format(len(df.loc[df['pres'] == 0 ])))
          print("# rows missing thickness: {0}".format(len(df.loc[df['skin'] == 0 ])))
          print("# rows missing insulin: {0}".format(len(df.loc[df['insu'] == 0 ])))
```

SPLITTING THE DATA


localhost:8888/notebooks/diabetes.ipynb

jupyter diabetes Last Checkpoint: 03/10/2019 (autosaved)

```
# rows in dataframe 768
# rows missing glucose_conc: 5
# rows missing diastolic_bp: 35
# rows missing thickness: 227
# rows missing insulin: 374
# rows missing bmi: 11
# rows missing diab_pred: 0
# rows missing age: 0
```

In [18]:
```
from sklearn.preprocessing import Imputer #impute with mean all 0 readings
fill_0 = Imputer(missing_values = 0 , strategy ="mean", axis=0)
X_train = fill_0.fit_transform(X_train)
X_test = fill_0.fit_transform(X_test)
```

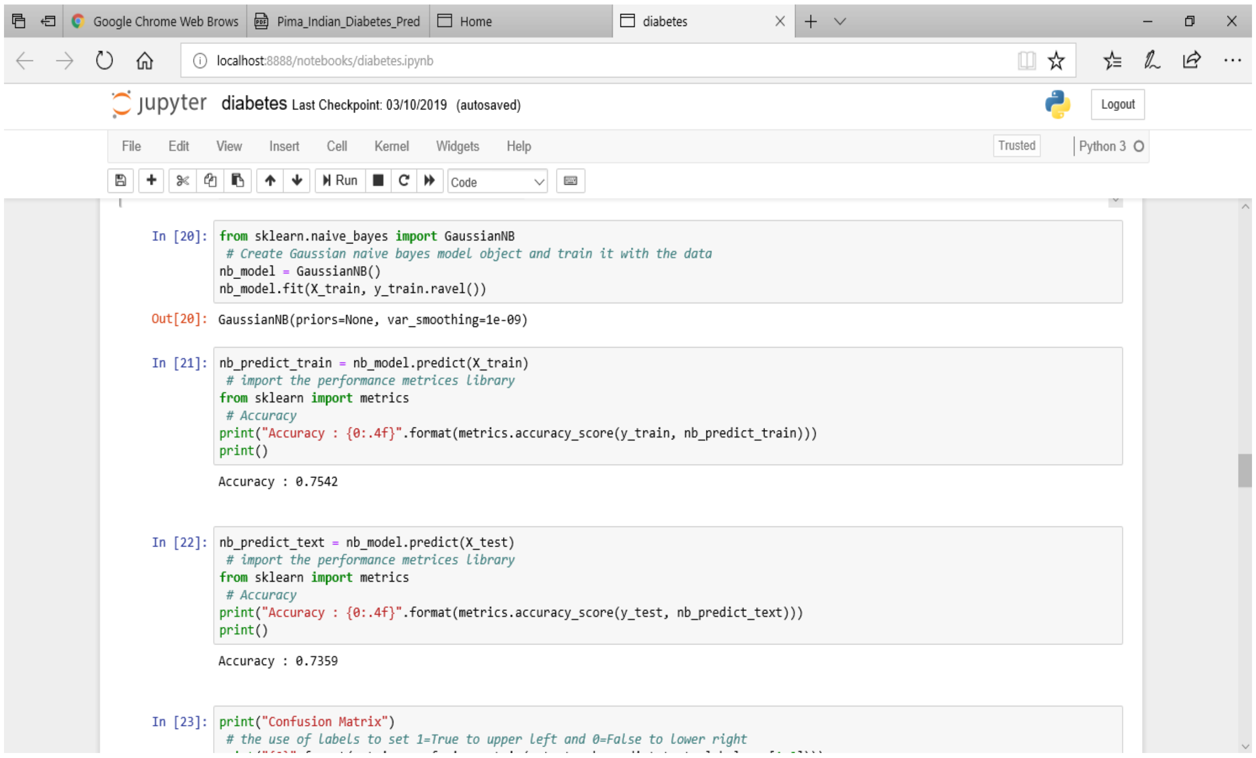
```
C:\Users\Vadhana\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning: Class Imputer is deprecated;
Imputer was deprecated in version 0.20 and will be removed in 0.22. Import impute.SimpleImputer from sklearn instead.
  warnings.warn(msg, category=DeprecationWarning)
```

In [19]:
```
np.isnan(X)
np.where(np.isnan(X))
np.nan_to_num(X)
pd.DataFrame(X).fillna(1)
```

Out[19]:

	0	1	2	3	4	5	6	7
0	6.0	148.0	72.0	0.0	35.0	33.6	0.627	50.0
1	1.0	85.0	66.0	0.0	29.0	26.6	0.351	31.0
2	8.0	183.0	64.0	0.0	0.0	23.3	0.672	32.0

REPLACE THE MISSING VALUES BY MEAN VALUES


localhost:8888/notebooks/diabetes.ipynb

jupyter diabetes Last Checkpoint: 03/10/2019 (autosaved)

In [20]:
```
from sklearn.naive_bayes import GaussianNB
 # Create Gaussian naive bayes model object and train it with the data
nb_model = GaussianNB()
nb_model.fit(X_train, y_train.ravel())
```

Out[20]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [21]:
```
nb_predict_train = nb_model.predict(X_train)
 # import the performance metrices library
from sklearn import metrics
 # Accuracy
print("Accuracy : {0:.4f}".format(metrics.accuracy_score(y_train, nb_predict_train)))
print()
```

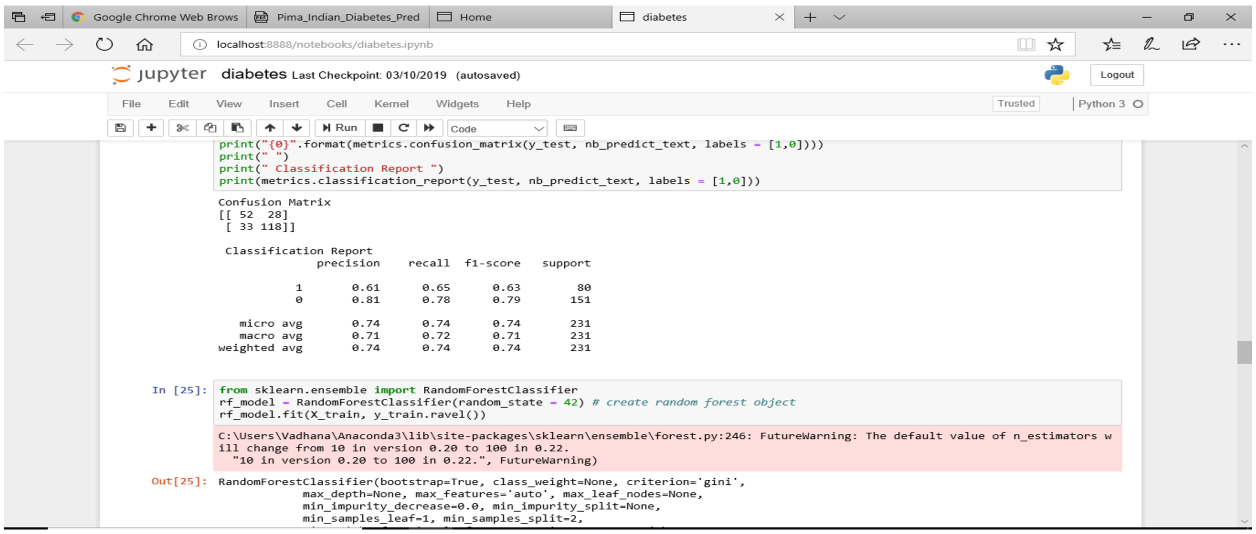
```
Accuracy : 0.7542
```

In [22]:
```
nb_predict_text = nb_model.predict(X_test)
 # import the performance metrices library
from sklearn import metrics
 # Accuracy
print("Accuracy : {0:.4f}".format(metrics.accuracy_score(y_test, nb_predict_text)))
print()
```

```
Accuracy : 0.7359
```

In [23]:
```
print("Confusion Matrix")
 # the use of labels to set 1=True to upper left and 0=False to lower right
```

PERFORMANCE ON TRAINING DATA IN NAIVE BAYES


localhost:8888/notebooks/diabetes.ipynb
jupyter diabetes Last Checkpoint: 03/10/2019 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
print("{0}".format(metrics.confusion_matrix(y_test, nb_predict_text, labels = [1,0])))
print(" ")
print(" Classification Report ")
print(metrics.classification_report(y_test, nb_predict_text, labels = [1,0]))
```

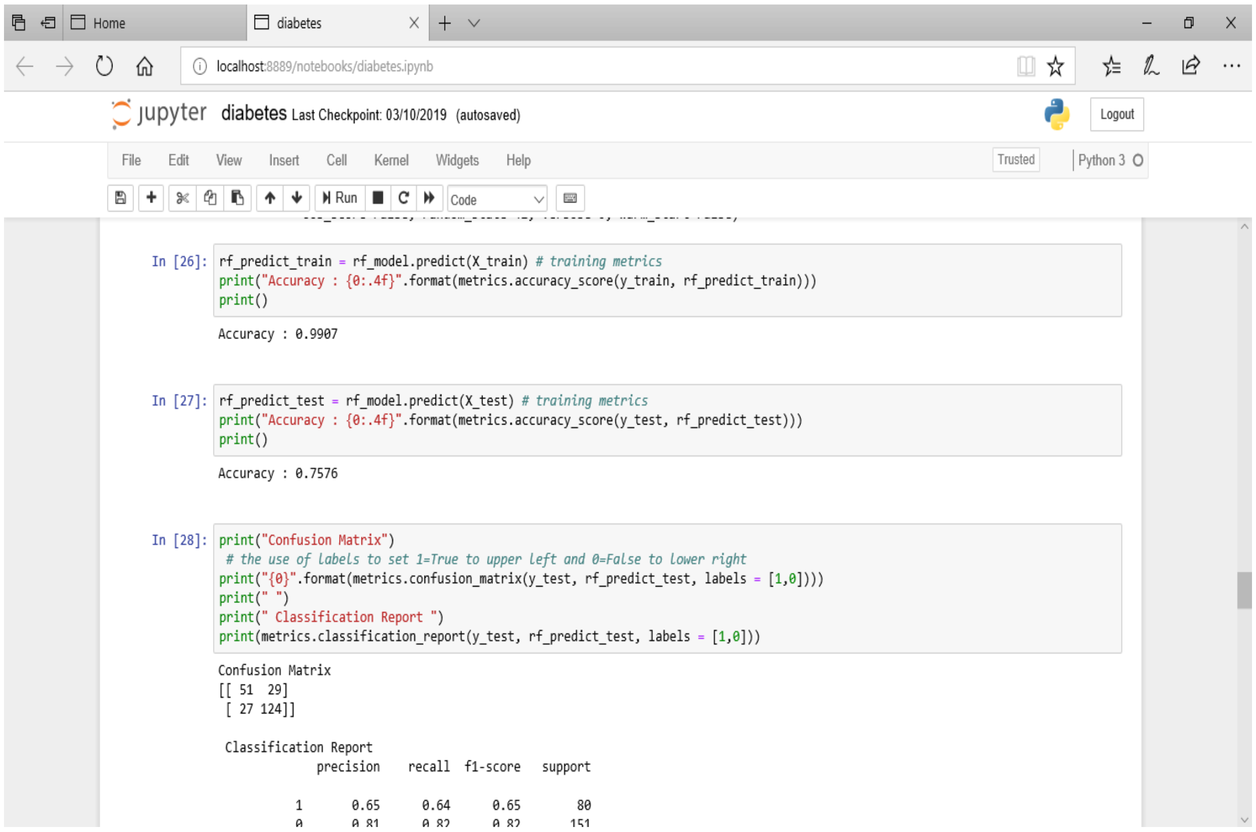
```
Confusion Matrix
[[ 52  28]
 [ 33 118]]

 Classification Report
              precision    recall  f1-score   support

           1       0.61      0.65      0.63        80
           0       0.81      0.78      0.79       151

   micro avg       0.74      0.74      0.74       231
   macro avg       0.71      0.72      0.71       231
weighted avg       0.74      0.74      0.74       231
```

```
In [25]: from sklearn.ensemble import RandomForestClassifier
         rf_model = RandomForestClassifier(random_state = 42) # create random forest object
         rf_model.fit(X_train, y_train.ravel())
```

```
C:\Users\Vadhana\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The default value of n_estimators w
ill change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
Out[25]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                     max_depth=None, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
```

CONFUSION MATRIX OF NAÏVE BAYES


Home
diabetes
localhost:8889/notebooks/diabetes.ipynb
jupyter diabetes Last Checkpoint: 03/10/2019 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [26]: rf_predict_train = rf_model.predict(X_train) # training metrics
         print("Accuracy : {0:.4f}".format(metrics.accuracy_score(y_train, rf_predict_train)))
         print()
```

```
Accuracy : 0.9907
```

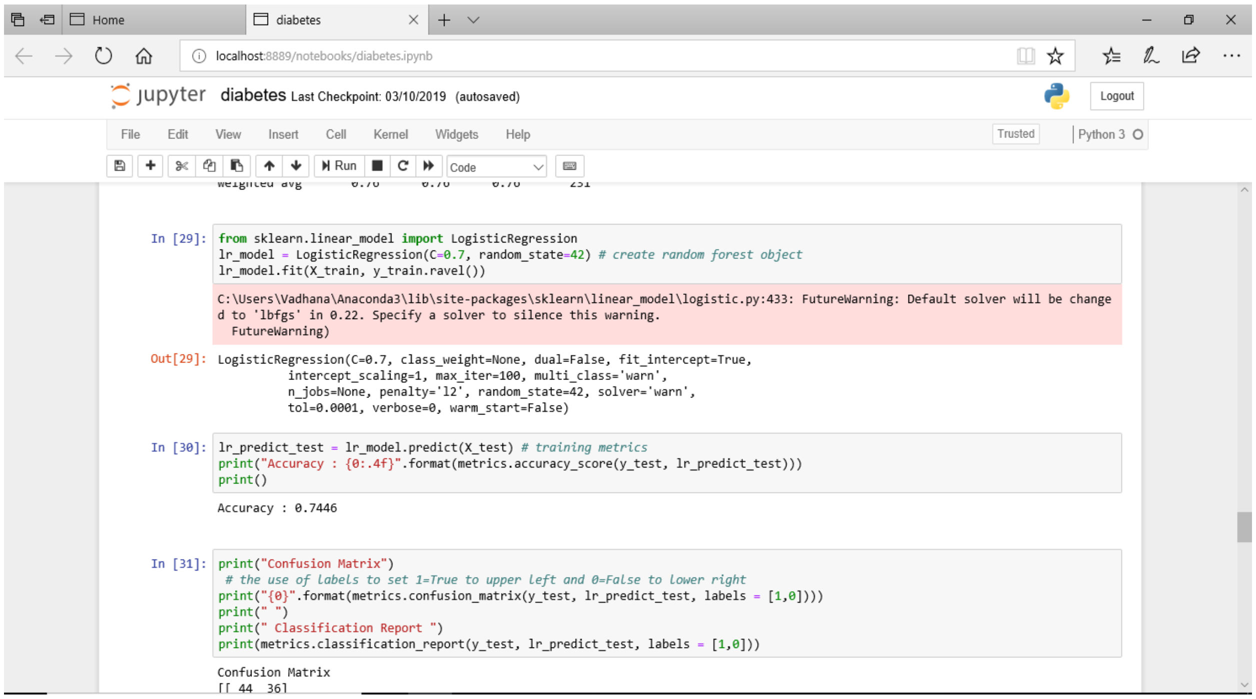
```
In [27]: rf_predict_test = rf_model.predict(X_test) # training metrics
         print("Accuracy : {0:.4f}".format(metrics.accuracy_score(y_test, rf_predict_test)))
         print()
```

```
Accuracy : 0.7576
```

```
In [28]: print("Confusion Matrix")
          # the use of labels to set 1=True to upper left and 0=False to lower right
         print("{0}".format(metrics.confusion_matrix(y_test, rf_predict_test, labels = [1,0])))
         print(" ")
         print(" Classification Report ")
         print(metrics.classification_report(y_test, rf_predict_test, labels = [1,0]))
```

```
Confusion Matrix
[[ 51  29]
 [ 27 124]]

 Classification Report
              precision    recall  f1-score   support

           1       0.65      0.64      0.65        80
           0       0.81      0.82      0.82       151
```

ACCURACY PREDICTED USING RANDOM FOREST


Home
diabetes
localhost:8889/notebooks/diabetes.ipynb
jupyter diabetes Last Checkpoint: 03/10/2019 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
weighted avg       0.76      0.76      0.76       231
```

```
In [29]: from sklearn.linear_model import LogisticRegression
         lr_model = LogisticRegression(C=0.7, random_state=42) # create random forest object
         lr_model.fit(X_train, y_train.ravel())
```

```
C:\Users\Vadhana\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be change
d to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

```
Out[29]: LogisticRegression(C=0.7, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='warn',
                   n_jobs=None, penalty='l2', random_state=42, solver='warn',
                   tol=0.0001, verbose=0, warm_start=False)
```

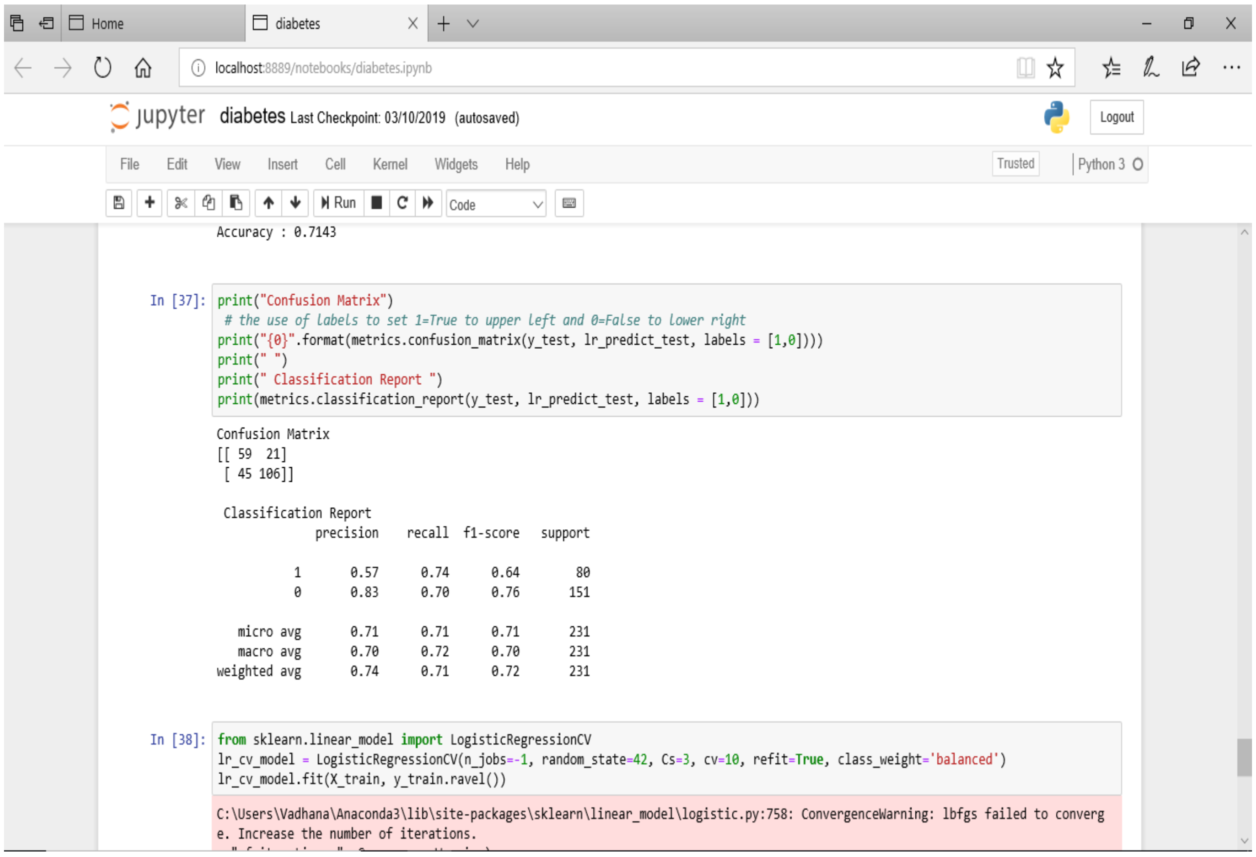
```
In [30]: lr_predict_test = lr_model.predict(X_test) # training metrics
         print("Accuracy : {0:.4f}".format(metrics.accuracy_score(y_test, lr_predict_test)))
         print()
```

```
Accuracy : 0.7446
```

```
In [31]: print("Confusion Matrix")
          # the use of labels to set 1=True to upper left and 0=False to lower right
         print("{0}".format(metrics.confusion_matrix(y_test, lr_predict_test, labels = [1,0])))
         print(" ")
         print(" Classification Report ")
         print(metrics.classification_report(y_test, lr_predict_test, labels = [1,0]))
```

```
Confusion Matrix
[[ 44  36]
```

ACCURACY PREDICTED USING LOGISTIC REGRESSION


Home
diabetes
localhost:8889/notebooks/diabetes.ipynb
jupyter diabetes Last Checkpoint: 03/10/2019 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
Accuracy : 0.7143
```

```
In [37]: print("Confusion Matrix")
          # the use of labels to set 1=True to upper left and 0=False to lower right
         print("{0}".format(metrics.confusion_matrix(y_test, lr_predict_test, labels = [1,0])))
         print(" ")
         print(" Classification Report ")
         print(metrics.classification_report(y_test, lr_predict_test, labels = [1,0]))
```

```
Confusion Matrix
[[ 59  21]
 [ 45 106]]

 Classification Report
              precision    recall  f1-score   support

           1       0.57      0.74      0.64        80
           0       0.83      0.70      0.76       151

   micro avg       0.71      0.71      0.71       231
   macro avg       0.70      0.72      0.70       231
weighted avg       0.74      0.71      0.72       231
```

```
In [38]: from sklearn.linear_model import LogisticRegressionCV
         lr_cv_model = LogisticRegressionCV(n_jobs=-1, random_state=42, Cs=3, cv=10, refit=True, class_weight='balanced')
         lr_cv_model.fit(X_train, y_train.ravel())
```

```
C:\Users\Vadhana\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:758: ConvergenceWarning: lbfgs failed to converg
e. Increase the number of iterations.
```

CONFUSION MATRIX FOR LOGISTIC REGRESSION

REFERENCES

[1] Jianchao Han, Juan C.Rodriguze, Mohsen Beheshti, “Diabetes Data Analysis and Prediction Model Discovery Using Rapid miner”, IEEE, Second International Conference on Future Generation Communication and Networking,pp.96-99,2008.

[2] Asma A. AlJarullah, “Decision tree discovery for the diagnosis of type-2 Diabetes”, IEEE, 2011 Internaional Conference on innovations in information technology, pp. 303-307, 2011.

[3] B.M. Patil, R.C. Joshi, Durga Toshniwal, “Hybrid Prediction Model for Type-2 Diabetic patients”, Expert Systems with Applications, Science direct, pp. 8102-8108,2010.

[4] C.Yue, L. Xin, X.Kewen and S Chang, “An Intelligent Diagnosis to Type 2 Diabetes Based on QPSO algorithm and WLS-SVM”, IEEE, International Symposium on Intelligent Information Technology Application Workshops,2008.

[5] S.M Nuwangi, C.R. Oruthotaarachchi, J.M.P.P Tilakaratna and H.A. Caldera, „Usage of Association rules and Classification Techniques in Knowledge Extraction of Diabetes‟ IEEE, pp.372-377,2010

[6] Vijayalakshmi, D., Thilagavathi, K.: An Approach for Prediction of Diabetic Disease by Using b-Colouring Technique in Clustering Analysis. In: International Journal of Applied Mathematical Research, 1 (4) pp. 520-530 Science Publishing Corporation www.sciencepubco.com/index.php/IJAMR (2012). 18.Pujari, P., [7]. Vishwavidyalaya, G.G.: Ensemble Data Mining Model for Classification of Pima Indian Diabetes Data set.

[8] . Pradhan, M., Sahu, R.K.: Predict the onset of diabetes disease using Artificial Neural Network. Intl J Comp Sci & Emerging Tech. 2:303-11 (2011)

[9]. Han, J., Rodriguze, J.C., Beheshti, M.: Diabetes data analysis and prediction model discovery using RapidMiner. Second International Conference on Future Generation Communication and Networking.96-9 (2008)

[10]. Breault, J.L.: Data Mining Diabetic Databases: Are Rough Sets a Useful Addition?

[11] Yirui Guo. Application of artificial neural network to predict individual risk of type 2 diabetes mellitus. Journal of Zhengzhou University, 2014.49(3):180-183.

[12] Shuaishuai Li, Enke Zhang, Min Li and Wei Pan, Research on the Effectiveness of Application of Diabetes ManagementAPP, China Medical Devices, 2015. Vol 30. No.08.

[13] Ms. K Sowjanya, MobDBTest: A machine learning based system for predicting diabetes risk using mobile devices. 2015 IEEE International Advance Computing Conference (IACC).

[14] Gang Shi, Shanshan Liu and Ding Ye, Design and Implementation of Diabetes Risk Assessment Model Based On MobileThings, 2015 7th International Conference on Information Technology in Medicine and Education.

[15] Juntao Wang and Xiaolong Su, An improved K-Means clustering algorithm, 2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN).

[16] Yanhui Sun, Liying Fang and Pu Wang, Improved k-means clustering based on Efros distance for longitudinal data, 2016Chinese Control and Decision Conference (CCDC).

[17] Shunye Wang, Improved K-means clustering algorithm based on the optimized initial centroids, 2013 3rd International Conference on Computer Science and Network Technology (ICCSNT).

[18] Phattharat Songthung and Kunwadee Sripanidkulchai, Improving Type 2 Diabetes Mellitus Risk Prediction Using Classification, 2016 13th I nternational Joint Conference on Computer Science and Software Engineering (JCSSE).

[19] Omprakash Chandrakar, Dr. Jatinderkumar R. Saini. Development of Indian Weighted Diabetic Risk Score (IWDRS) using Machine Learning Techniques for Type-2 Diabetes, ACM COMPUTE '16, October 21- 23, 2016, Gandhinagar, India.

[20] Longfei Han, Senlin Luo. An Intelligible Risk Stratification Model based on Pairwise and Size Constrained Kmeans, 2016 IEEE Journal of Biomedical and Health Informatics.

[21] Aruna Pavate and Nazneen Ansari, Risk Prediction of Disease Complications in Type 2 Diabetes Patients Using Soft Computing Techniques, 2015 Fifth International Conference on Advances in Computing and Communications.

[22] Naganna Chetty, An Improved Method for Disease Prediction using Fuzzy Approach, 2015 Second International Conference on Advances in Computing and Communication Engineering.

[23] Saad Masood Butt and Karla.FelixNavarro, Using Mobile Technology to improve Nutritional Information of Diabetic Patient’s, New Advances in Information Systems and Technologies(2016).

[24] Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). Machine learning, neural and statistical classification. Ellis Horwood.

[25] Karim M. Orabi1, Yasser M. Kamal, and Thanaa M. Rabah. Early Predictive System for Diabetes Mellitus Disease. ICDM 2016, LNAI 9728, pp. 420–427, 2016.

[26] Guojun, G., Chaoqu, M. and Jianhong, W.. Data clustering theory algorithm and application (1st Ed.). ASA-SIAM.M (2007).

[27] https://en.wikipedia.org/wiki/K-means.

[28] Humar, K. and Novruz, A., Design of a hybrid system for the diabetes and heart diseases. Expert Systems with Applications, 35, 82–89, 2008.

[29] Rojalina Priyadarshini, Nilamadhab Dash and Rachita Mishra, A Novel approach to Predict Diabetes Mellitus using Modified Extreme Learning Machine.

[30] Murari Devakannan Kamalesh, Predicting the Risk of Diabetes Mellitus to Subpopulations Using Association Rule Mining. The International Conference on Soft Computing Systems 2016.