Chapter 8
Machine Learning in Python: Diabetes Prediction Using Machine Learning

Astha Baranwal
VIT University, India

Bhagyashree R. Bagwe
VIT University, India

Vanitha M
VIT University, India

ABSTRACT

Diabetes is a disease of the modern world. The modern lifestyle has led to unhealthy eating habits causing type 2 diabetes. Machine learning has gained a lot of popularity in the recent days. It has applications in various fields and has proven to be increasingly effective in the medical field. The purpose of this chapter is to predict the diabetes outcome of a person based on other factors or attributes. Various machine learning algorithms like logistic regression (LR), tuned and not tuned random forest (RF), and multilayer perceptron (MLP) have been used as classifiers for diabetes prediction. This chapter also presents a comparative study of these algorithms based on various performance metrics like accuracy, sensitivity, specificity, and F1 score.

INTRODUCTION

Diabetes is a disease which happens when the glucose level of the blood becomes high, which eventually leads to other health problems such as heart diseases, kidney disease etc. Several data mining projects have used algorithms to predict diabetes in a patient. Though, in most of these projects, nothing is mentioned about the dangers of diabetes in women post-pregnancies. While data mining has been successfully applied to various fields in human society, such as weather prognosis, market analysis, engineering diagnosis, and customer relationship management, the application in disease prediction and medical data analysis still has room for improvement in accuracy.

DOI: 10.4018/978-1-5225-9902-9.ch008

Machine learning relates closely to Artificial Intelligence (AI) and makes software applications predict outcomes through statistical analysis. The algorithms used allow for reaching an optimal accuracy rate in predicting the output from the input data. Machine learning follows similar processes used in data mining and predictive modeling. They recognize patterns through the data entered and then adjust the actions of the program accordingly.

Machine learning algorithms are categorized as supervised learning and unsupervised learning. Supervised learning requires input data and the desired output data to build a training model. The training model is built by a data analyst or a data scientist. A feedback is then furnished concerning the accuracy of the model and other performance metrics during algorithm training. Revising is done as needed. Once the training phase is completed, the model can predict outcomes for new data. Classification is one of the many data mining tasks. Classification comes under supervised learning which implies that the machine learns through examples in Classification. In classification, every instance from the dataset is classified into a target value. Classification can either be binary or multi-label. Sometimes, one particular instance can also have multiple classes known as multi-class classification. Classification algorithms are majorly used for prediction and come under the category of predictive learning.

Unsupervised learning is used to draw inferences from the input data which do not have any labeled responses. This data is not categorized, labeled or classified into classes. Clustering analysis, one of the most common unsupervised learning method, is used to find hidden patterns in data or to form groups based on the input data.

While machine learning models have been around for decades, they have gained a new momentum with the rise of AI. Deep learning models are now used in most of the advanced AI applications. If these models are implemented for medical uses, they could be revolutionary for the society. Diagnosis of diseases like diabetes would be easier than ever. Machine learning in medical diagnosis applications fall under three classes: Pathology, Oncology and Chatbots. Pathology deals with the diagnosis of diseases with the help of machine learning models created with the data of diagnostic measurements of the patients. Oncology uses deep learning models to determine cancerous tissues in patients. Chatbots designed using AI and machine learning techniques can identify patterns in the symptoms of the patients and suggest a potential diagnosis or it can recommend further courses of action. This chapter falls under the pathological uses of machine learning as the model created will give diagnosis of whether a patient is diabetic or not.

This chapter focuses on implementing machine learning algorithms on the diabetes dataset in python. Python is a great language to support machine learning. It was created by Guido van Rossom. It is powerful, multipurpose, and simple and has an easy to use syntax. The length of a python code is generally relatively short. It is not overly strict. It is a fun language to work with because it lets us focus on the problem rather than the syntax. Python is a general purpose language with applications in a wide range of fields like web development, mathematical computing, graphical user interfaces, etc. These wide range of applications python most suitable for implementing machine learning algorithms. It is possible to implement some machine learning algorithms in python and later deploy the web service for it. Or it is also possible to create a graphical user interface for the deployed web service. This all becomes possible due to the diverse nature of the language. Various python libraries like, NumPy, SciPy, Matplotlib, Keras, Pandas, TensorFlow, and etc support machine learning in python. Scikit-learn library supports almost all the major machine learning classification, clustering and regression models. The Flask python web framework is used to create web apps from APIs created using the machine learning models. This web app can perform real-time predictions for individual and batch data inputs.

Background

About one in every seven U.S. adult citizen has diabetes currently, as per the Centers for Disease Control and Prevention. According to statistics, this rate could skyrocket to as many as one in three by the year 2050. Hence, it is of utmost importance that a proper diabetes prediction system should be built with high prediction accuracy. Several prior studies have proposed various models for the prediction of diabetes on the PIMA Indian Dataset. Accuracy up to 95.42% (proposed by Han Wu et al. 2018) has been obtained by first clustering the data using K-Means algorithm and removing the outliers and then creating the classification models. Patil et al. (2010) proposed a similar model with C4.5 algorithm as the final classifier model. They obtained an accuracy of 92.38%. Kandhasamy et al. (2015) applied machine learning classification techniques like Decision Tree J48, KNN Classifier, Random Forest and Support Vector Machine and obtained 86.46% accuracy. The paper proposed by the Sisodias (2018) applied three classification algorithms: Decision Tree, SVM and Naive Bayes and reported highest accuracy (76.30%) with the Naïve Bayes model.

Linear Discriminant Analysis (LDA) and Adaptive Network Based Fuzzy Inference System (ANFIS): LDA-ANFIS was proposed by Dogantekin et al. (2010) with an accuracy of 84.61%. R. Aishwarya et al. (2013) used Support Vector Machine to build the classification model. Temurtas et al. (2009) used a multilayer neural network structure which was trained by Levenberg–Marquardt (LM) algorithm. The accuracy stated was 82.37%.

Fatima and Pasha (2017) provided a comparative analysis of various machine learning algorithms and techniques for the diagnosis of serious diseases such as diabetes, heart diseases, liver diseases, dengue and hepatitis diseases. K Saravananathan and Velmurugan (2016) in the paper Analyzing Diabetic Data using Classification Algorithms in Data Mining use J48, support vector machine (SVM), classification and regression tree (CART) and K-Nearest Neighbor (KNN) to categorize the dataset that included 10 attributes from 545 patients in Weka. Kavakiotis et al. (2017) discovered that 85% of papers used supervised learning approaches and 15% by unsupervised ones, association rules, more specifically. Support vector machines (SVM) was stated as the most successful and popularly used algorithm. M. P. Gopinath and Murali (2017) provided a comparative study between various classification algorithms, namely Naive Bayes (NB), Support Vector Machine (SVM), Decision Tree (DT), Bayes net, etc, in their paper. Yasodha and Kannan (2011) used WEKA tool to classify the dataset and the 10-fold cross validation is used for evaluation and the results are compared. D. Nanthini and Thangaraju (2015) in their paper propose a method for building a hybrid on the diabetes dataset using three variations of the classification algorithm Decision Tree namely, C4.5, J48 and the FB tree.

Many other papers have reported a classification accuracy ranging from 50% to 80% on the PIMA Indians Diabetes dataset using various machine learning techniques. Research is underway for better accuracy rates using different advanced machine learning and deep learning algorithms and techniques.

MAIN FOCUS OF THE CHAPTER

Issues, Controversies, Problems

Medical diagnosis of Diabetes is possible only after the patient has acquired the disease and not before. This completely cuts down the possibility of prevention of the disease. This is an issue of the medical field. This problem can be sorted by aiding the medical field with the Information Technology sector.

Machine Learning algorithms are used in this chapter to predict if a person will have diabetes or not. This prediction is done based on some other lifestyle attributes of the person. The machine learning task used here is classification. Classification comes under supervised learning. Supervised learning is when the machine learns based on examples, like a human being, using the training dataset. In supervised leaning it is required that the data has a class label or a target value as an independent variable into which the instances will be classified. With respect to the PIMA diabetes dataset, the target variable is the outcome value which predicts whether the person will have diabetes or not. It is for this variable that the classification is carried out. If it is somehow possible to predict in advance if the person is going to contract diabetes or not, some preventive measures can be taken in order to avoid contraction of diabetes.

Thus, in this way, the use of machine learning in the medical field can help solve some critical issues and aid medical science in more ways than expected.

Section-Wise Description

1. **Platform Details:** This section provides a description about the platform used.
   a. Anaconda.
   b. Jupyter
   c. Python
   d. NumPy
   e. Pandas
   f. SciPy
   g. Matplotlib
   h. TensorFlow and Keras
2. **System Architecture:** This section briefly explains how the system works and how it is built using a flowchart and a textual description.
3. **Dataset Description:** This section gives a description about the selected PIMA diabetes dataset. It gives information about the datatype of the attributes along with their explanation.
4. **Data Cleaning:** This section shows how the data was preprocessed and cleaned using Weka. This included various processes like finding missing values and replacing them with suitable values.
5. **Data Visualization:** In this section the data was visualized using the Python Jupyter platform. Data visualization helps us in getting a better understanding of the data.
6. **Random Forest Model (Not Tuned):** This section explains how the basic random forest model is built on the PIMA diabetes dataset using machine learning in python. It also shows the results produced by this classifier.
7. **Random Forest Model (Tuned):** This section explains how the tuned random forest model is built on the PIMA diabetes dataset using machine learning in python. It also shows the results produced by this classifier.

8. **Checking with different Hyper parameter tuning:** This sections tests for random values of the three selected parameters namely, n_tree, max_features and max_depth for hyper parameter tuning.
9. **Logistic Regression (LR):** This section explains Logistic Regression classifier model is built on the PIMA diabetes datasetusing machine learning in python. It also shows the results produced by this classifier.
10. **Multilayer Perceptron:** This section explains how the multilayer perceptron (MLP), a type of Artificial Neural Network (ANN) is built on the PIMA diabetes dataset using machine learning in python. It also shows the results produced by this classifier.
11. **Performance Metrics**: This section gives a detailed information about the various performance metrics used for evaluating and comparing the various classifier models built on the PIMA diabetes dataset.
12. **Comparison between the models using various performance metrics**: This section provides a tabular comparison between the various classifier models built for predicting the outcome on the PIMA diabetes dataset.

SOLUTIONS AND RECOMMENDATIONS

Platform Details

Anaconda

Anaconda is a professional data science platform which we can use as a GUI as well as a console. Anaconda is a Python distribution which brings a lot of useful libraries with it, which are not included in Python standard library like Jupyter, NumPy, Pandas, SciPy, Matplotlib, etc. It can create different Python version environments.

Jupyter

The Jupyter Notebook is an open-source web app that allows the creation and sharing of documents that contain equations, live code, visualizations and narrative text in form of markdowns. It is included in the Anaconda distribution.

WEKA (Waikato Environment for Knowledge Analysis)

Weka is a software that is built in Java and can run on almost every platform. Weka gives provisions for various machine learning algorithms and can be used for data cleaning and data visualization.

Python

Python is a widely used high-level programming language which is interpreted, general-purpose, and dynamic.

The Python libraries used for data analysis, visualization and machine learning are:

NumPy

NumPy is a Python library that adds support for large and multi-dimensional arrays. It supports matrices.

Pandas

Pandas is a Python package designed to make the work with labeled and relational data be very simple and intuitive. It is designed for fast and easy data manipulation, visualization and aggregation.

SciPy

SciPy contains modules for statistics, linear algebra, optimization and integration. The principal functionality of SciPy library is based upon NumPy, and hence its arrays make considerable use of NumPy.

Matplotlib

Matplotlib is a Python library for 2D plotting which produces quality figures in a variety of hardcopy formats and interactive environments across platforms.

Tensorflow and Keras

Tensorflow is a Python Deep Learning library.

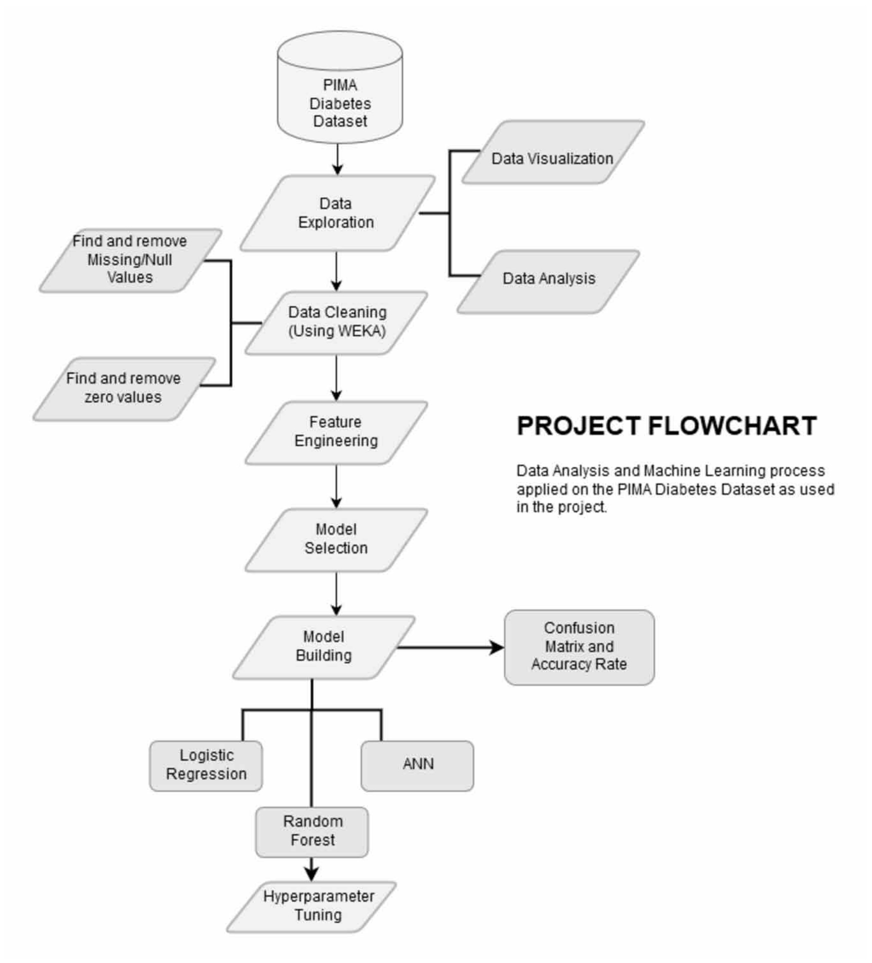
Keras is a simple neural networks library designed for high-level neural networks. It is written in Python and works as a wrapper to Tensorflow.

System Architecture

This chapter uses the PIMA Indians Diabetes Dataset for diabetes prediction by various machine learning models. The dataset is first explored. Data visualization helps to understand the data significance by identifying any useful patterns or abnormalities that are revealed in the data points of the dataset. Data visualization is done and correlation between each and every attribute is examined. The values of attributes are represented by binning for all the instances. The target variable 'Outcome' contains positive (1) and negative (0) values. The count of positive and negative instances can be identified.

The data cleaning involves removal of zero and missing (null) values from the dataset. This is done using the Weka Explorer software. Next, feature engineering is done. Feature engineering involves determination of useful attributes. The attributes which do not contribute much in predicting the outcome accurately are discarded. No attribute is discarded in making the models in this chapter. SkinThickness contributes the least in prediction of diabetes for the dataset but it is considered nonetheless. Logistic Regression (LR), Random Forest (RF) and Multilayer Perceptron (MLP) are selected for model building because they all are powerful machine learning algorithms which can be further hyper parameter tuned to obtain optimal accuracy. Multilayer perceptron classifier is a type of Artificial Neural Network (ANN). Confusion matrix and classification report for each model is generated. The results of the models are then compared based on various performance metrics.

Figure 1. The system architecture


PIMA Diabetes Dataset
Data Exploration
Data Visualization
Data Analysis
Find and remove Missing/Null Values
Data Cleaning (Using WEKA)
Find and remove zero values
Feature Engineering
PROJECT FLOWCHART
Data Analysis and Machine Learning process applied on the PIMA Diabetes Dataset as used in the project.
Model Selection
Model Building
Confusion Matrix and Accuracy Rate
Logistic Regression
ANN
Random Forest
Hyperparameter Tuning

Dataset Description

The dataset used is originally contributed by the National Institute of Diabetes and Digestive and Kidney Diseases. It has 768 instances and 9 attributes. The goal is to predict whether a patient has diabetes based on diagnostic measurements. The datasets consist of several medical predictor (independent) variables and one target (dependent) variable, Outcome.

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.

Data Cleaning

The missing values and the null values are cleaned:

- There are instances with BloodPressure as 0. Blood pressure of a living person can never be 0.
- Plasma glucose levels, even after fasting, can never be 0.

Table 1. Attribute description for the diabetes dataset

Sr. No.	Attribute	Description	Datatype
1	Pregnancies	Number of times pregnant	int64
2	Glucose	Plasma glucose concentration	int64
3	BloodPressure	Diastolic blood pressure (mm Hg)	int64
4	SkinThickness	Triceps skin fold thickness (mm)	int64
5	Insulin	2-Hour serum insulin (mu U/ml)	int64
6	BMI	Body mass index (weight in kg/(height in m)^2)	float64
7	DiabetesPedigreeFunction	Diabetes pedigree function	float64
8	Age	Age of patient (in years)	int64
9	Outcome	Class variable. 1 for diabetic and 0 for non-diabetic.	int64

- Skin Fold Thickness cannot be 0.
- BMI of a person can never be 0 as it is weight/height^2
- Insulin is never 0 in normal cases and there are about 374 instances with 0 insulin.

They are replaced by means of the values of the concerned attributes. The cleaned dataset has 768 instances and 9 attributes.

Data Cleaning of the "Pima Indian Diabetes Dataset" by WEKA Explorer

Mark Missing Values

Attributes such as BloodPressure and BMI (Body Mass Index) have values of zero, which is impossible. These examples of corrupt or missing data must be marked manually. Since zero values are not considered as missing values by Weka, this needs to be fixed first.

Weka marks zero values as missing values by the NumericalCleaner filter. The steps below show how to use this filter to mark the missing values on the Pregnancies,BloodPressure, Glucose, SkinThickness, BMI and Insulin attributes:

The Pima Indians diabetes dataset is loaded in the Weka Explorer.

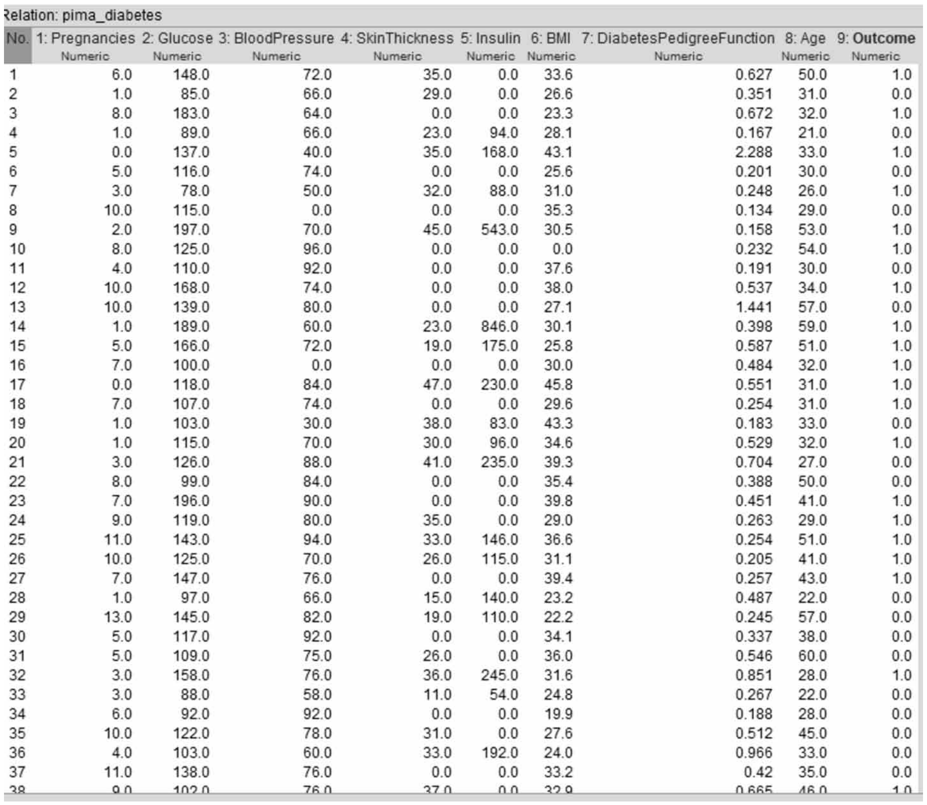
NumericalCleaner filter is selected. It is under unsupervized.attribute.NumericalCleaner. This filter is further configured.

TheattributeIndicies is set to 1-6. This is the indices of the attributes with 0 values. The minThreshold is set to 0.1E-8 (close to zero), which is the minimum value allowed for the attribute. The minDefault is set to NaN, which is unknown and will replace values below the threshold.

The filter configuration is applied on the dataset. The transformation for the BloodPressure attribute is shown for each step.

Notice that the attribute values that were formally set to zero are now marked as Missing.

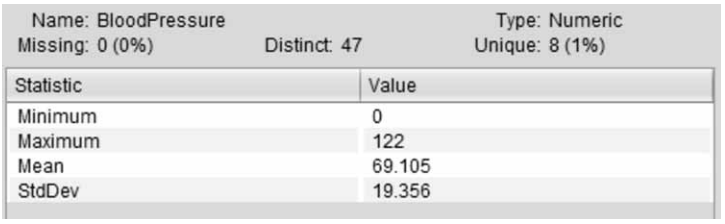
Figure 2. The initial dataset



Relation: pima_diabetes

No.	1: Pregnancies Numeric	2: Glucose Numeric	3: BloodPressure Numeric	4: SkinThickness Numeric	5: Insulin Numeric	6: BMI Numeric	7: DiabetesPedigreeFunction Numeric	8: Age Numeric	9: **Outcome** Numeric
1	6.0	148.0	72.0	35.0	0.0	33.6	0.627	50.0	1.0
2	1.0	85.0	66.0	29.0	0.0	26.6	0.351	31.0	0.0
3	8.0	183.0	64.0	0.0	0.0	23.3	0.672	32.0	1.0
4	1.0	89.0	66.0	23.0	94.0	28.1	0.167	21.0	0.0
5	0.0	137.0	40.0	35.0	168.0	43.1	2.288	33.0	1.0
6	5.0	116.0	74.0	0.0	0.0	25.6	0.201	30.0	0.0
7	3.0	78.0	50.0	32.0	88.0	31.0	0.248	26.0	1.0
8	10.0	115.0	0.0	0.0	0.0	35.3	0.134	29.0	0.0
9	2.0	197.0	70.0	45.0	543.0	30.5	0.158	53.0	1.0
10	8.0	125.0	96.0	0.0	0.0	0.0	0.232	54.0	1.0
11	4.0	110.0	92.0	0.0	0.0	37.6	0.191	30.0	0.0
12	10.0	168.0	74.0	0.0	0.0	38.0	0.537	34.0	1.0
13	10.0	139.0	80.0	0.0	0.0	27.1	1.441	57.0	0.0
14	1.0	189.0	60.0	23.0	846.0	30.1	0.398	59.0	1.0
15	5.0	166.0	72.0	19.0	175.0	25.8	0.587	51.0	1.0
16	7.0	100.0	0.0	0.0	0.0	30.0	0.484	32.0	1.0
17	0.0	118.0	84.0	47.0	230.0	45.8	0.551	31.0	1.0
18	7.0	107.0	74.0	0.0	0.0	29.6	0.254	31.0	1.0
19	1.0	103.0	30.0	38.0	83.0	43.3	0.183	33.0	0.0
20	1.0	115.0	70.0	30.0	96.0	34.6	0.529	32.0	1.0
21	3.0	126.0	88.0	41.0	235.0	39.3	0.704	27.0	0.0
22	8.0	99.0	84.0	0.0	0.0	35.4	0.388	50.0	0.0
23	7.0	196.0	90.0	0.0	0.0	39.8	0.451	41.0	1.0
24	9.0	119.0	80.0	35.0	0.0	29.0	0.263	29.0	1.0
25	11.0	143.0	94.0	33.0	146.0	36.6	0.254	51.0	1.0
26	10.0	125.0	70.0	26.0	115.0	31.1	0.205	41.0	1.0
27	7.0	147.0	76.0	0.0	0.0	39.4	0.257	43.0	1.0
28	1.0	97.0	66.0	15.0	140.0	23.2	0.487	22.0	0.0
29	13.0	145.0	82.0	19.0	110.0	22.2	0.245	57.0	0.0
30	5.0	117.0	92.0	0.0	0.0	34.1	0.337	38.0	0.0
31	5.0	109.0	75.0	26.0	0.0	36.0	0.546	60.0	0.0
32	3.0	158.0	76.0	36.0	245.0	31.6	0.851	28.0	1.0
33	3.0	88.0	58.0	11.0	54.0	24.8	0.267	22.0	0.0
34	6.0	92.0	92.0	0.0	0.0	19.9	0.188	28.0	0.0
35	10.0	122.0	78.0	31.0	0.0	27.6	0.512	45.0	0.0
36	4.0	103.0	60.0	33.0	192.0	24.0	0.966	33.0	0.0
37	11.0	138.0	76.0	0.0	0.0	33.2	0.42	35.0	0.0
38	9.0	102.0	76.0	37.0	0.0	32.9	0.665	46.0	1.0

Figure 3. The BloodPresuure attribute before data cleaning



Name: BloodPressure Type: Numeric
Missing: 0 (0%) Distinct: 47 Unique: 8 (1%)

Statistic	Value
Minimum	0
Maximum	122
Mean	69.105
StdDev	19.356

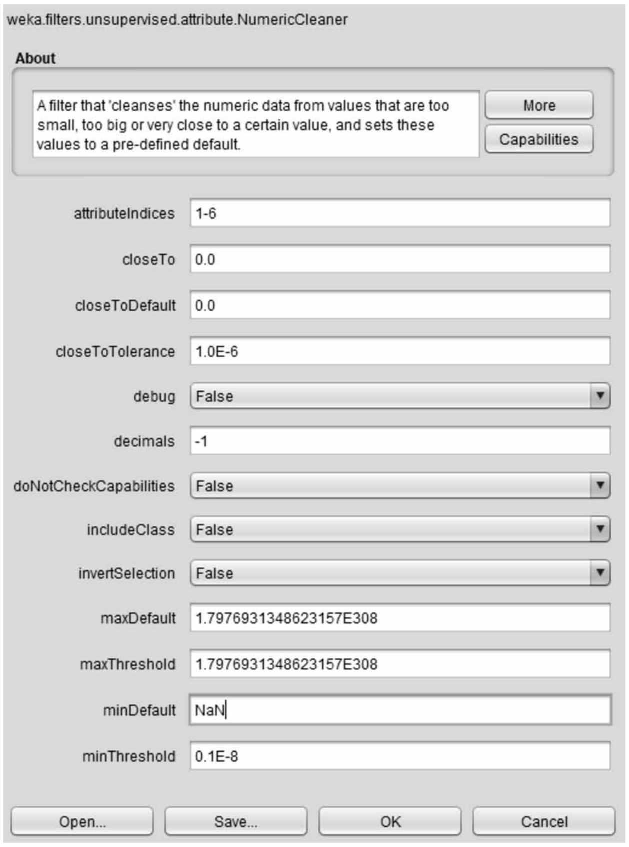
Remove Missing Data

Now that missing values are marked in the data, they need to be handled. An easy way to deal with missing data in the dataset is to remove those data points that have one or more missing values.

This can be done in Weka using the RemoveWithValues filter.

However, this leads to loss in some data instances. It is advisable to impute the missing values instead of removing these instances altogether from the dataset.

Figure 4. Configuring the NumericalCleaner filter



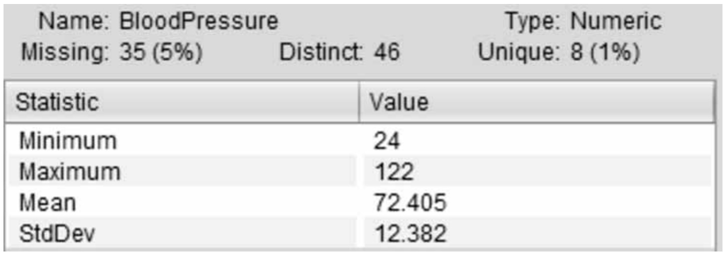
weka.filters.unsupervised.attribute.NumericCleaner

About

A filter that 'cleanses' the numeric data from values that are too small, too big or very close to a certain value, and sets these values to a pre-defined default.

attributeIndices	1-6
closeTo	0.0
closeToDefault	0.0
closeToTolerance	1.0E-6
debug	False
decimals	-1
doNotCheckCapabilities	False
includeClass	False
invertSelection	False
maxDefault	1.7976931348623157E308
maxThreshold	1.7976931348623157E308
minDefault	NaN
minThreshold	0.1E-8

Figure 5. The BloodPressure attribute after applying NumericalCleaner



Name: BloodPressure Type: Numeric
Missing: 35 (5%) Distinct: 46 Unique: 8 (1%)

Statistic	Value
Minimum	24
Maximum	122
Mean	72.405
StdDev	12.382

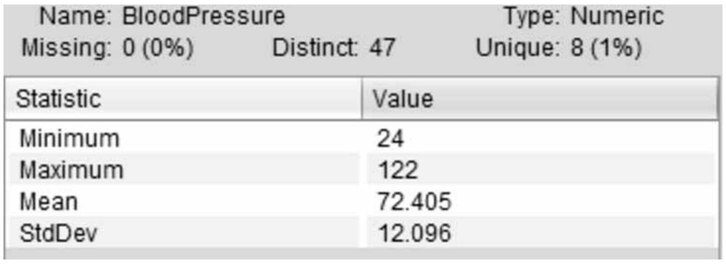
Impute Missing Values

It is not necessary to remove the instances with missing values; the missing values can be replaced instead with some other value.

This is called imputing missing values.

It is not uncommon to impute missing values with the mean of the numerical distribution of an attribute. This can be done easily in Weka using the ReplaceMissingValues filter.

Figure 6. The BloodPressure attribute after ReplaceMissingValues filter



Name: BloodPressure Type: Numeric
Missing: 0 (0%) Distinct: 47 Unique: 8 (1%)

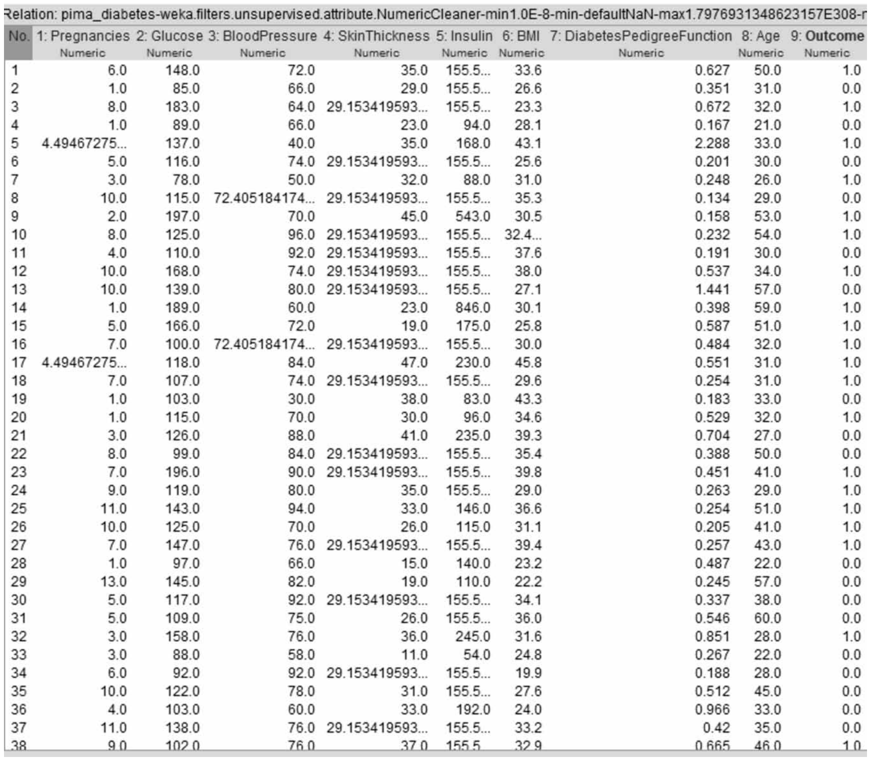
Statistic	Value
Minimum	24
Maximum	122
Mean	72.405
StdDev	12.096

Missing values can be imputed by theReplaceMissingValues filter. It is under the unsupervized.attribute.ReplaceMissingValues path.

The filter is applied on appropriate attributes.

The missing values are gone. They are replaced by means of the values of concerned attributes.

Figure 7. The final dataset after data cleaning



Relation: pima_diabetes-weka.filters.unsupervised.attribute.NumericCleaner-min1.0E-8-min-defaultNaN-max1.7976931348623157E308-r

No.	1: Pregnancies Numeric	2: Glucose Numeric	3: BloodPressure Numeric	4: SkinThickness Numeric	5: Insulin Numeric	6: BMI Numeric	7: DiabetesPedigreeFunction Numeric	8: Age Numeric	9: **Outcome** Numeric
1	6.0	148.0	72.0	35.0	155.5...	33.6	0.627	50.0	1.0
2	1.0	85.0	66.0	29.0	155.5...	26.6	0.351	31.0	0.0
3	8.0	183.0	64.0	29.153419593...	155.5...	23.3	0.672	32.0	1.0
4	1.0	89.0	66.0	23.0	94.0	28.1	0.167	21.0	0.0
5	4.49467275...	137.0	40.0	35.0	168.0	43.1	2.288	33.0	1.0
6	5.0	116.0	74.0	29.153419593...	155.5...	25.6	0.201	30.0	0.0
7	3.0	78.0	50.0	32.0	88.0	31.0	0.248	26.0	1.0
8	10.0	115.0	72.405184174...	29.153419593...	155.5...	35.3	0.134	29.0	0.0
9	2.0	197.0	70.0	45.0	543.0	30.5	0.158	53.0	1.0
10	8.0	125.0	96.0	29.153419593...	155.5...	32.4...	0.232	54.0	1.0
11	4.0	110.0	92.0	29.153419593...	155.5...	37.6	0.191	30.0	0.0
12	10.0	168.0	74.0	29.153419593...	155.5...	38.0	0.537	34.0	1.0
13	10.0	139.0	80.0	29.153419593...	155.5...	27.1	1.441	57.0	0.0
14	1.0	189.0	60.0	23.0	846.0	30.1	0.398	59.0	1.0
15	5.0	166.0	72.0	19.0	175.0	25.8	0.587	51.0	1.0
16	7.0	100.0	72.405184174...	29.153419593...	155.5...	30.0	0.484	32.0	1.0
17	4.49467275...	118.0	84.0	47.0	230.0	45.8	0.551	31.0	1.0
18	7.0	107.0	74.0	29.153419593...	155.5...	29.6	0.254	31.0	1.0
19	1.0	103.0	30.0	38.0	83.0	43.3	0.183	33.0	0.0
20	1.0	115.0	70.0	30.0	96.0	34.6	0.529	32.0	1.0
21	3.0	126.0	88.0	41.0	235.0	39.3	0.704	27.0	0.0
22	8.0	99.0	84.0	29.153419593...	155.5...	35.4	0.388	50.0	0.0
23	7.0	196.0	90.0	29.153419593...	155.5...	39.8	0.451	41.0	1.0
24	9.0	119.0	80.0	35.0	155.5...	29.0	0.263	29.0	1.0
25	11.0	143.0	94.0	33.0	146.0	36.6	0.254	51.0	1.0
26	10.0	125.0	70.0	26.0	115.0	31.1	0.205	41.0	1.0
27	7.0	147.0	76.0	29.153419593...	155.5...	39.4	0.257	43.0	1.0
28	1.0	97.0	66.0	15.0	140.0	23.2	0.487	22.0	0.0
29	13.0	145.0	82.0	19.0	110.0	22.2	0.245	57.0	0.0
30	5.0	117.0	92.0	29.153419593...	155.5...	34.1	0.337	38.0	0.0
31	5.0	109.0	75.0	26.0	155.5...	36.0	0.546	60.0	0.0
32	3.0	158.0	76.0	36.0	245.0	31.6	0.851	28.0	1.0
33	3.0	88.0	58.0	11.0	54.0	24.8	0.267	22.0	0.0
34	6.0	92.0	92.0	29.153419593...	155.5...	19.9	0.188	28.0	0.0
35	10.0	122.0	78.0	31.0	155.5...	27.6	0.512	45.0	0.0
36	4.0	103.0	60.0	33.0	192.0	24.0	0.966	33.0	0.0
37	11.0	138.0	76.0	29.153419593...	155.5...	33.2	0.42	35.0	0.0
38	9.0	102.0	76.0	37.0	155.5	32.9	0.665	46.0	1.0

Data Visualization

Importing the libraries and the dataset:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
diabetes = pd.read_csv('pima_diabetes.csv')
```

Visualize the distribution of positive and negative instances for the Outcome attribute in the diabetes dataset:

```
diabetes.groupby('Outcome').size()
sns.countplot(diabetes['Outcome'],label="Count")
```

It can be identified that out of the 768 persons, 500 are labeled as 0 (Non-Diabetic) and 268 as 1 (Diabetic)

```
def plot_corr(df, size=11):
corr = df.corr()
    fig, ax = plt.subplots(figsize=(size, size))
ax.matshow(corr)
plt.xticks(range(len(corr.columns)), corr.columns)
plt.yticks(range(len(corr.columns)), corr.columns)
plot_corr(diabetes)
```

Figure 8. Positive and negative instances for the Outcome attribute in the diabetes dataset

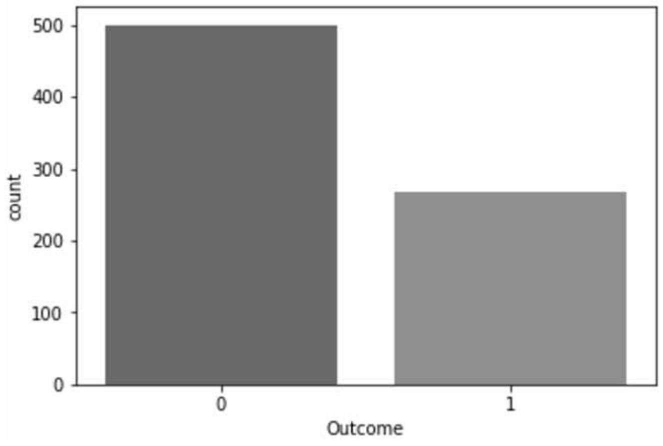
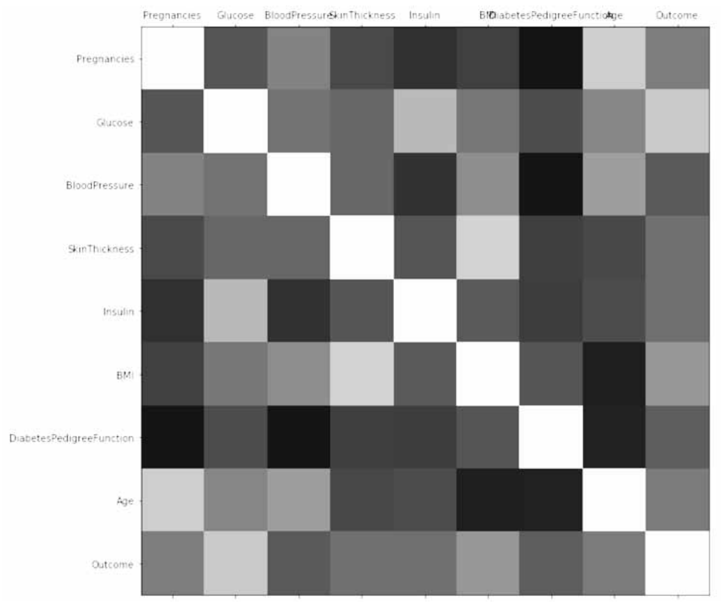


Figure 9. The correlation between various attributes of the diabetes dataset



In the above visualization plot, lighter color represents maximum correlation and darker color represents minimum correlation. We can see none of the variable have proper correlation with any of the other variables.

Binning of the instances in dataset as per attributes:

```
diabetes.hist(figsize=(12,8),bins=20)
```

Model Selection: Comparison of Calibration of Classifiers

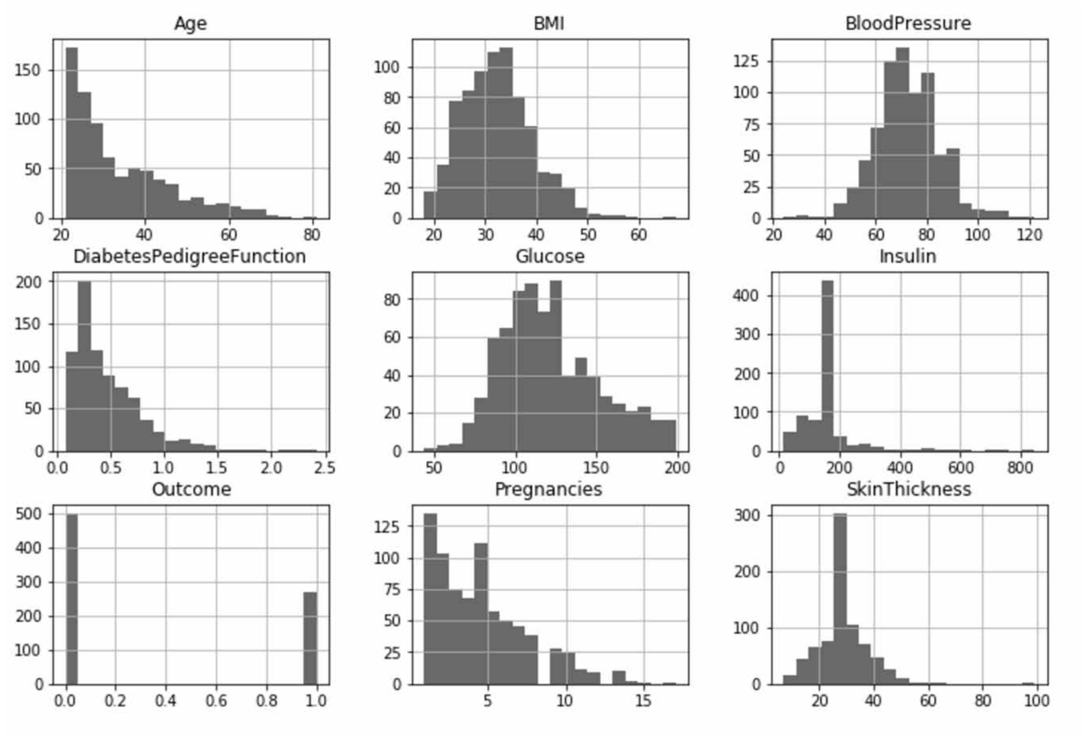
Model selection is the phase where the model with the best calibration and reliability estimate for the data set at hand is selected.

The "Classification Accuracy (Testing Accuracy)" of a given set of classification models is calculated with their default parameters to predict which model can perform better with the PIMA diabetes data set.

Seven classifiers namely K-Nearest Neighbors, Support Vector Classifier, Logistic Regression, Decision Tree, Gaussian Naive Bayes and Random Forest and Multilayer Perceptron are to be the contenders for the best classifier.

```
models = []

models.append(('KNN', KNeighborsClassifier()))
models.append(('SVC', SVC()))
models.append(('LR', LogisticRegression()))
models.append(('DT', DecisionTreeClassifier()))
models.append(('GNB', GaussianNB()))
```

Figure 10. The values of attributes in the diabetes dataset represented by binning



```
models.append(('RF', RandomForestClassifier()))
models.append(('ANN', MLPClassifier()))

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify = diabetes.
Outcome, random_state=0)

names = []
scores = []

for name, model in models:
model.fit(X_train, y_train)
y_pred = model.predict(X_test) #prediction of models
scores.append(accuracy_score(y_test, y_pred))
names.append(name)

tr_split = pd.DataFrame({'Name': names, 'Score': scores})
print(tr_split)

axis = sns.barplot(x = 'Name', y = 'Score', data = tr_split)
```

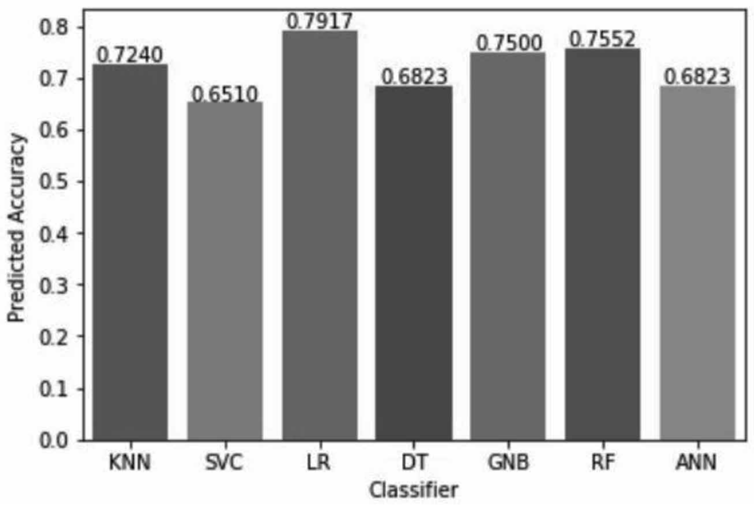
```
axis.set(xlabel='Classifier', ylabel='Predicted Accuracy')

for p in axis.patches:
    height = p.get_height()
axis.text(p.get_x() + p.get_width()/2, height + 0.005, '{:1.4f}'.
format(height), ha="center")

plt.show()
```

Logistic Regression is the best contender based on the plot obtained. Random forest gives a decent performance. ANN or the Multilayer Perceptron (MLP) classifier can use standard scalar to increase the accuracy.

Random Forest Model (Not Tuned)

```
from sklearn import model_selection

# random forest model creation
rfc = RandomForestClassifier()
rfc.fit(X_train,y_train)

# predictions
rfc_predict = rfc.predict(X_test)

from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report, confusion_matrix
rfc_cv_score = cross_val_score(rfc, X, y, cv=10, scoring='roc_auc')
```

Figure 11. Predicted Accuracy vs contender Classifiers

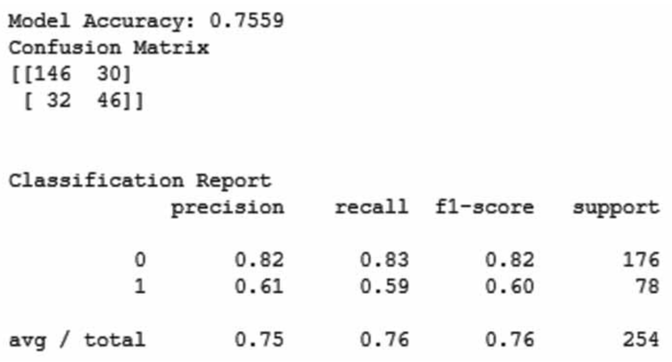


```
print("Model Accuracy: {0:.4f}".format(metrics.accuracy_score(y_test, rfc_pre-
dict)))

print("Confusion Matrix")
print(confusion_matrix(y_test, rfc_predict))
print('\n')

print("Classification Report")
print(classification_report(y_test, rfc_predict))
print('\n')
```

Random Forest Model (Tuned): Hyper Parameter Tuning the Random Forest Model

```
from sklearn.model_selection import RandomizedSearchCV

# number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num =
10)]

# number of features at every split
max_features = ['auto', 'sqrt']

# max depth
max_depth = [int(x) for x in np.linspace(100, 500, num = 11)]
max_depth.append(None)

# create random grid
random_grid = {
 'n_estimators': n_estimators,
 'max_features': max_features,
```

Figure 12. Classification report for the Random Forest (not tuned) model



```
Model Accuracy: 0.7559
Confusion Matrix
[[146  30]
 [ 32  46]]


Classification Report
             precision    recall  f1-score   support

          0       0.82      0.83      0.82       176
          1       0.61      0.59      0.60        78

avg / total       0.75      0.76      0.76       254
```

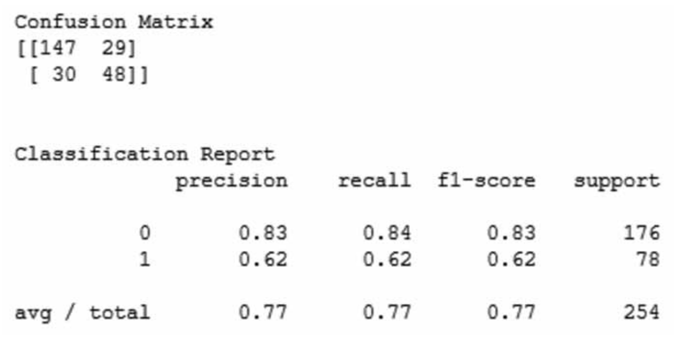
```
 'max_depth': max_depth
 }

# Random search of parameters
rfc_random = RandomizedSearchCV(estimator = rfc, param_distributions = random_
grid, n_iter = 100, cv = 3, verbose=2, random_state=42, n_jobs = -1)

# Fit the model
rfc_random.fit(X_train, y_train.ravel())

# print results
print(rfc_random.best_params_)
```

The results are: 'n_estimators' = 1200; 'max_features' = 'auto'; 'max_depth': 180. Now these parameters can be plugged back into the model to see whether it improved the performance.

Checking With Different Hyper Parameter Tuning

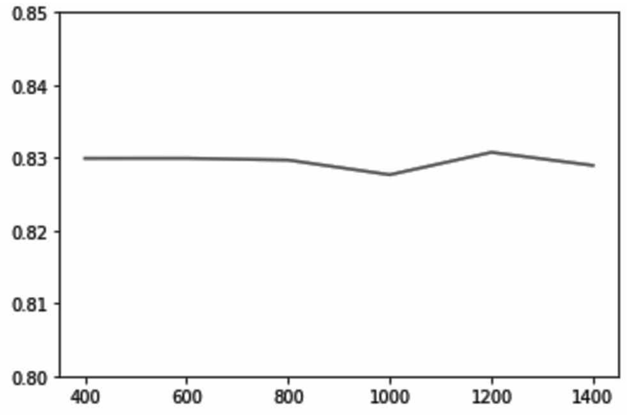
```
import numpy
from matplotlib import pyplot
x = numpy.array([400,600,800,1000,1200,1400])
y = numpy.array([0.8299330484330485, 0.829940170940171, 0.8296937321937321,
0.8276780626780628, 0.8307606837606837, 0.8289601139601139])

fig = pyplot.figure()
ax = fig.add_subplot(111)
ax.set_ylim(0.80,0.85)
pyplot.plot(x,y)
pyplot.show()
```

Figure 13. Classification report for the random forest (tuned) model



```
Confusion Matrix
[[147  29]
 [ 30  48]]


Classification Report
             precision    recall  f1-score   support

          0       0.83      0.84      0.83       176
          1       0.62      0.62      0.62        78

avg / total       0.77      0.77      0.77       254
```

Figure 14. Area Under the Curve (AUC) score versus the number of trees (n_tree) in the Random Forest



Clearly, at n_tree = 1200; max_features = ‘auto’; max_depth: 180, the AUC score is highest and this, hence, will give the highest accuracy rate

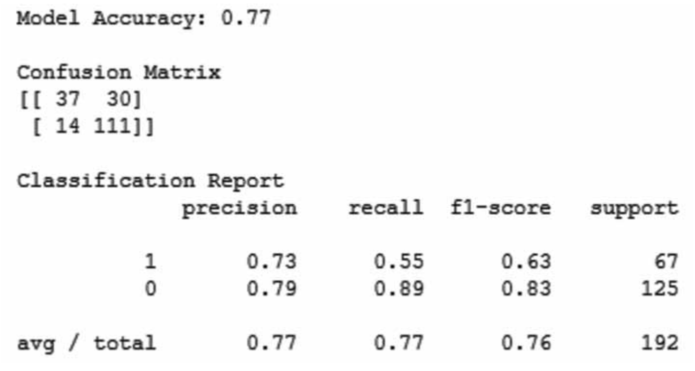
Logistic Regression

```
from sklearn.linear_model import LogisticRegression

diab_lr_model = LogisticRegression(C=0.7, random_state=52)
diab_lr_model.fit(X_train, y_train.ravel())
lr_test_predict = diab_lr_model.predict(X_test)

print("Model Accuracy: {0:.2f}".format(metrics.accuracy_score(y_test, lr_test_
predict)))
print("")
print("Confusion Matrix")
print(metrics.confusion_matrix(y_test, lr_test_predict, labels=[1, 0]))
print("")
print("Classification Report")
print(metrics.classification_report(y_test, lr_test_predict, labels=[1, 0]))
```

Multilayer Perceptron

```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(diabetes.loc[:, diabetes.
columns != 'Outcome'], diabetes['Outcome'], stratify=diabetes['Outcome'], ran-
dom_state=66)
from sklearn.preprocessing import StandardScaler
```

Figure 15. Classification report for the Logistic Regression (LR) model



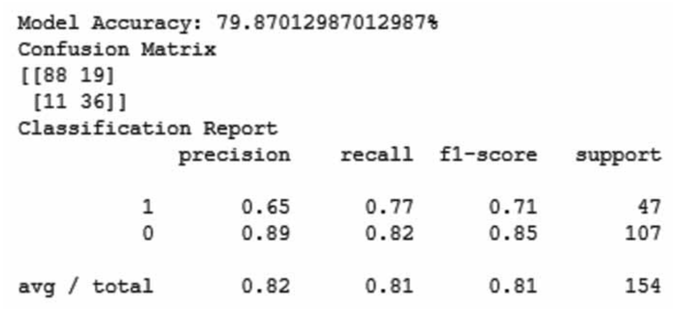
```
Model Accuracy: 0.77

Confusion Matrix
[[ 37  30]
 [ 14 111]]

Classification Report
             precision    recall  f1-score   support

          1       0.73      0.55      0.63        67
          0       0.79      0.89      0.83       125

avg / total       0.77      0.77      0.76       192
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)

mlp = MLPClassifier(random_state=0)
mlp.fit(X_train_scaled, y_train)

print("Accuracy on test set: {:.4f}".format(mlp.score(X_test_scaled, y_test)))
```

Performance Metrics

Performance metrics play an important role in the evaluation of the various machine learning models built. Choice of performance metrics determine how the machine learning models are compared and measured.This chapter compares the models based on the four performance metrics namely – Accuracy, Specificity, Sensitivity and the F1 score.

Figure 16. Classification report for the Multilayer Perceptron (MLP) classifier



```
Model Accuracy: 79.87012987012987%
Confusion Matrix
[[88 19]
 [11 36]]
Classification Report
             precision    recall  f1-score   support

          1       0.65      0.77      0.71        47
          0       0.89      0.82      0.85       107

avg / total       0.82      0.81      0.81       154
```

A confusion matrix is a performance metric table that is used to analyze the performance of a classification model (or “classifier”) on a set of test data for which the true target attribute values are already known. It is also called as the error matrix.Python has confusion matrix and classification report functions under the sklearn.metrics library. Confusion matrix contains:

True Positive: The instances which are classified as positive and are actually positive.
True Negative: The instances which are classified as negative and are actually negative.
False Positive: The instances which are classified as positive but are actually negative.
False Negative: The instances which are classified as negative but are actually positive.

Sensitivity (True Positive Rate)

If a person has diabetes, how often will the classifier be able to predict it?

It is the ratio of true positives to the sum of the true positive and false negative. A highly sensitive test helps rule out diabetes. If the test is highly sensitive and the test result is test result is negative, it becomes nearly certain that the particular person doesn’t have diabetes.

Sensitivity = True Positive / (True Positive + False Negative)

Specificity (True Negative Rate)

If a person doesn’t have diabetes, how often will the classifier be able to predict it?

It is the ratio of true negatives to the sum of the true negatives and false positives. A highly sensitive test helps rule in diabetes. If the test is highly specific and the test result is test result is positive, it becomes nearly certain that the particular person has diabetes.

Specificity = True Negative / (True Negative + False Positive)

Accuracy

It is a metric used to predict the correctness of a machine learning model. The model is trained using the train data and a classifier is built. The test data is used to cross validate the classifier model. The percentage of correctly classified instances is termed as Accuracy.

F1 Score

F1 score is a combination function of precision and recall. It is used when we need to seek a balance between precision and recall.

Precision = True Positives / (True Positives + False Positives)

Here, the denominator (True Positives + False Positives) is the total predicted positives.

Table 2. Comparison between various classifiers based on performance metrics

Algorithm	Accuracy	Specificity	Sensitivity	F1-Score
Random Forest (before hyper parameter tuning)	0.7559	0.8295	0.5897	0.5974
Random Forest (after hyper parameter tuning)	0.7677	0.8352	0.6153	0.6193
Logistic Regression	0.7789	0.7872	0.7255	0.6271
Artificial Neural Networks (ANN)	0.7987	0.8224	0.7659	0.7058

Recall = True Positives / (True Positives + False Negatives)

Here, the denominator (True Positives + False Negatives) is the total actual positives.

F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

Accuracy vs F1 Score

In cases where there is an uneven class distribution i.e. there are large number of actual negatives, F1 Score might prove to be better than accuracy. Accuracy only checks the correctness of a model while F1 score strikes a balance between precision and recall.

The model architecture is defined by several parameters. These parameters are referred to as hyper parameters. The process of searching for an ideal model architecture for optimal accuracy score is referred to as hyper parameter tuning. In this chapter, hyper parameter tuning of Random Forest model is done to give a better accuracy. At n_tree = 1200; max_features = 'auto'; max_depth: 180, the AUC (Area Under the Curve) score is highest and this, hence, gives the highest accuracy rate.

After building the models, Multilayer Perceptron (MLP) model gives the highest accuracy at 79.87%. Random Forest model was focused on to increase accuracy by Hyper-parameter Tuning. However, the highest accuracy obtained from it, i.e. 76.38%, could not exceed even the Logistic Regression model's accuracy (77.89%).

The neural network model built gives a better accuracy than LR and Random Forest classifier models. Diabetes is a huge threat to the modern world and devising an algorithm that tackles or even reduces its adversity can be a great help to diabetic patients.

Comparison Between the Models Using Various Performance Metrics

```
models = []
models.append((‘LR’, 0.7789))
models.append((‘ANN (not tuned)’, 0.7857))
models.append((‘ANN (tuned)’, 0.8052))
models.append((‘Ensemble’, 0.7512))
models.append((‘Hybrid’, 0.8377))
names = []
scores = []
```

```
for name, accuracy in models:
scores.append(accuracy)
names.append(name)
tb_split = pd.DataFrame({'Name': names, 'Score': scores})
print(tb_split)

axis = sns.barplot(x = 'Name', y = 'Score', data = tb_split, palette="GnBu_d")
axis.set(xlabel='CLASSIFIER', ylabel='ACCURACY SCORE')
sns.set(rc={'figure.figsize':(15,8)}, font_scale=1.5)
sns.set(font_scale=1.5)
for p in axis.patches:
    height = p.get_height()
axis.text(p.get_x() + p.get_width()/2, height + 0.005, '{:1.4f}'.
format(height), ha="center")
plt.show()
```

Clearly, accuracy of the Multilayer Perceptron (MLP) classifier is the highest at 79.87%, followed by the Logistic Regression model at 77.08%. Surprisingly, the Random forest did not perform better than the other two even after hyperparameter tuning the parameters. The accuracy of the Random Forest model increased only by 1.18% approximately after hyper parameter tuning.

The Logistic Regression model gives the lowest specificity rate at 78.72%. The best specificity rate is given by the Random Forest (hyperparameter tuned) model at 83.52%. The specificity determines how often a given classifier will be able to correctly predict if a person does not actually have diabetes.

The Random Forest (not tuned) model gives the lowest sensitivity rate at 58.97%. The best sensitivity rate is given by the Multilayer Perceptron (MLP) classifier model at 76.59%. The sensitivity determines how often a given classifier will be able to correctly predict if a person actually has diabetes.

The Logistic Regression model gives the lowest F1 score rate at 59.74%. The best F1 score is given by the Multilayer Perceptron (MLP) classifier model at 70.58%. The F1 score establishes a balance between the precision and recall values of a model.

Figure 17. Comparison between the accuracy score for each classifier

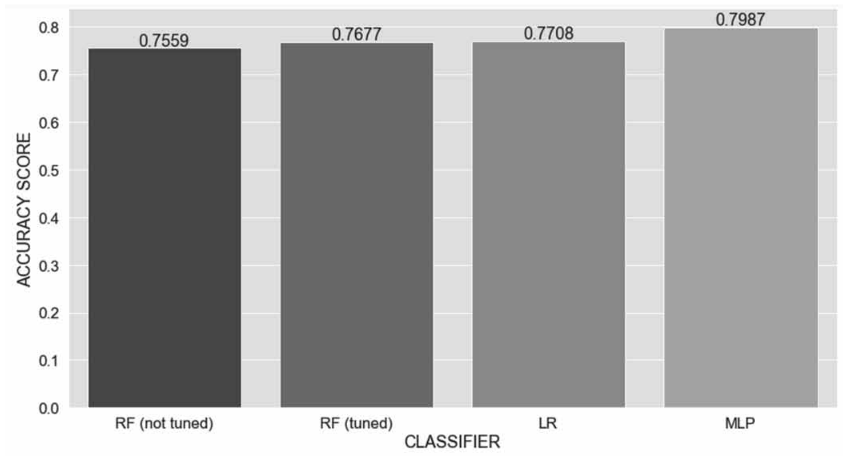


Figure 18. Comparison between the specificity for each classifier

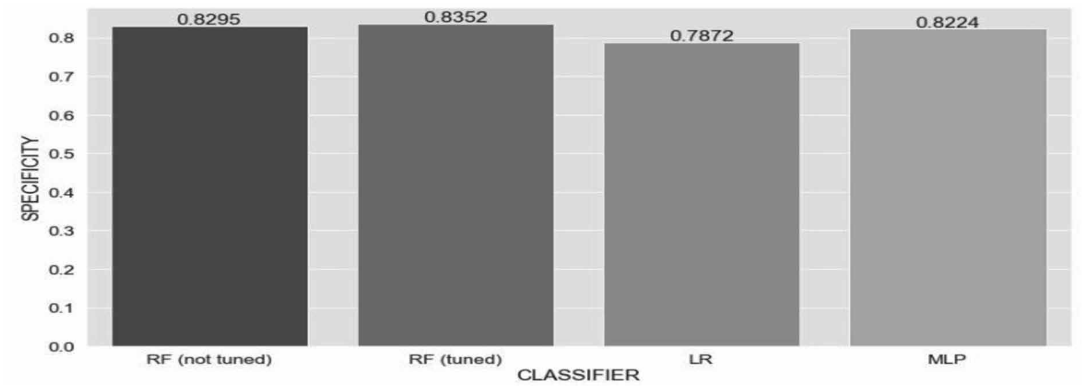


Figure 19. Comparison between the sensitivity for each classifier

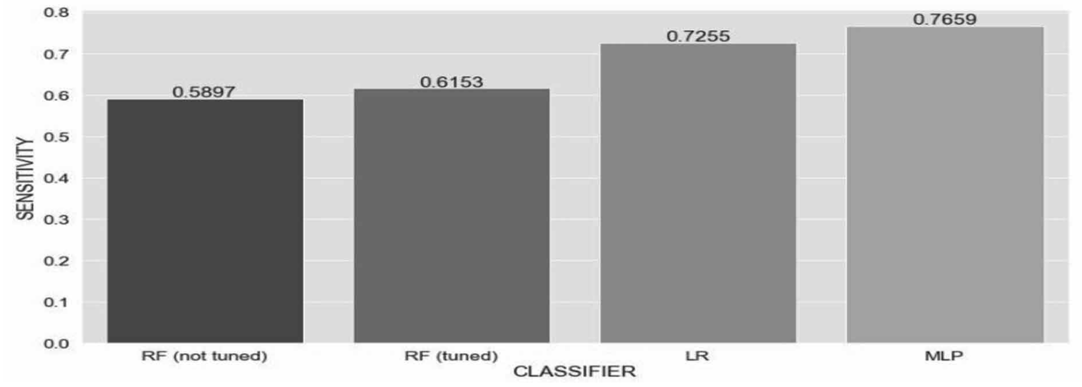
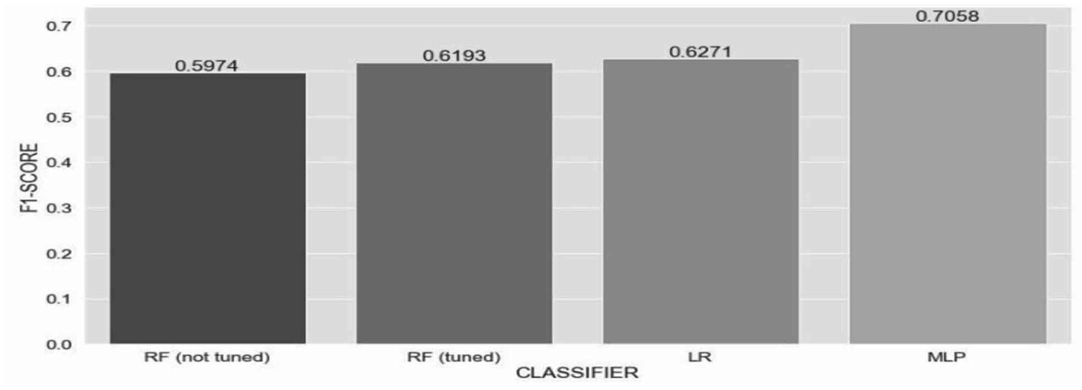


Figure 20. Comparison between the F1 score for each classifier



FUTURE RESEARCH DIRECTIONS

As a field of machine learning is progressing and improving day-by-day, researchers are diving into hybrid and ensemble modeling techniques. These complex hybrid and ensemble models are built by combining the traditional algorithms in a way that yields better performance metrics. Hybrid models are built in sequential blocks. The first block usually involves feature selection and is followed by blocks that contain various traditional classifiers. These classifiers can be further hyper parameter tuned. The ensemble model basically works on the concept of bagging. The prediction is made using multiple traditional algorithms and the class with the maximum number of votes is selected as the final prediction. Machine learning algorithms are giving rise to complex deep learning algorithms. Deep learning is a broader field of the machine learning family. It deals with learning data representations and pattern detection. The future neural network APIs in python libraries like PyTorch or Keras will just require a range of layers or parameters that are wanted in a customized neural network. Hyperparameter tuning by methods like Grid Search, Trial &Error, Random Search, Bayesian Optimization or Genetic Algorithm Optimization will increase the accuracy rate of these models. Batch size, epoch, activation function, number of hidden layers, units in each hidden layers, etc can result in distinct complex artificial neural networks.

Python is growing as the go-to language for data science, machine learning and is used even in big data. As the machine learning algorithms evolve, python is becoming the language of choice for most machine learning and data science professionals.

There are some limitations of the dataset taken in the chapter too which can be overcome to create an even more efficient and accurate diabetes prediction system/model. Considering the diabetes dataset taken is for women only, the complete scope of diabetes cannot be covered. Data for children, men, young and old people are not present in the PIMA diabetes dataset. There might be other risk factors that the dataset did not consider. Important factors like family history (i.e. if diabetes runs in the family), gestational diabetes, smoking; metabolic syndrome, inactive lifestyles, dietary patterns etc are not considered as attributes in the dataset. A proper model for prediction would need more data gathered from various sources to make the machine learning models more accurate. This can be attained by the collection of diabetes data from multiple sources, integrating these data and then generating a model from each dataset. The prediction model can be used as an API (Application Program Interface) for a Flask (python web framework) based web app which will give real-time prediction results of the entered data of a patient.

CONCLUSION

This chapter provides insights into various machine learning techniques used for classification. The code snippets demonstrate the use of python to build classifiers, hyper parameter tune them and visualize the data. It predicts the diabetes outcome based on other attributes. The classifiers like Logistic Regression (LR), Random Forest (RF) and Multilayer Perceptron (MLP) are used. It uses performance metrics like sensitivity, specificity, accuracy and F1 score to compare various classifiers and decide which gives the best results.

Python is one of the most suited languages to implement machine learning. It is easy and has a simple syntax to support diverse actions. Various python libraries make implementing machine learning a very easy process. The scikit-learn python library provides for almost all the major machine learning techniques and algorithms. Jupyter renders elegant graphical representations and code structures with results. Mark ups and headings provide further efficiency in presenting the work and code in a structured way. This chapter uses various python functions and libraries to implement the machine learning algorithms in order to build prediction models for the PIMA Indians Diabetes dataset.

The process of building a classifier starts with procuring data to train the model. The dataset selected for building classifiers in this chapter is the PIMA diabetes dataset. This dataset has attributes like number of pregnancies, glucose, blood pressure, skin thickness, insulin levels, the body mass index (BMI), the diabetes pedigree function and age. This dataset also has an independent or target variable namely outcome which predicts whether a person will have diabetes or not. Once the data is procured, the next step is data preprocessing. Data preprocessing includes, data cleaning, data reduction, data integration, data transformation, feature selection, dimensionality reduction, discretization and generating concept hierarchies. Data in the real world is dirty meaning it is incomplete, noisy and inconsistent. This makes data preprocessing inevitable. Low quality data is bound to produce low quality results. Thus, before building the classifier in this chapter, the data is preprocessed. It is made sure that all the attributes have the correct data types. Next, the missing values are found and replaced with some suitable values. Once the data is preprocessed, it is ready for the process of classification. Here, three classification algorithms are used, namely, Random Forest (RF), Logistic Regression (LR) and Multilayer Perceptron (MLP). Before the application of every algorithm, the dataset is divided into train and test dataset using percentage split technique for cross validation. Next, the algorithms are applied on the dataset and certain performance metrics like accuracy, F1 score, sensitivity and specificity are used to evaluate the performance of the algorithms.

The results show that the Multilayer Perceptron (MLP) performs the best with an accuracy of 79.87%, exceeding the second best performer – Logistic Regression (LR) by almost two percent. The Multilayer Perceptron (MLP) model gave the highest F1 score as well.

ACKNOWLEDGMENT

This work was supported by Vanitha M, Associate Professor, School of Information Technology and Engineering, Vellore Institute of Technology, Vellore. This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

REFERENCES

Aishwarya, R., Gayathri, P., & Jaisankar, N. (2013). A Method for Classification Using Machine Learning Technique for Diabetes. *IACSIT International Journal of Engineering and Technology*, *5*(3), 2903–2908.

Dogantekin, E., Dogantekin, A., Avci, D., & Avci, L. (2010, July). An intelligent diagnosis system for diabetes on Linear Discriminant Analysis and Adaptive Network Based Fuzzy Inference System: LDA-ANFIS. *Digital Signal Processing*, *20*(4), 1248–1255. doi:10.1016/j.dsp.2009.10.021

Fatima, M., & Pasha, M. (2017). Survey of Machine Learning Algorithms for Disease Diagnostic. *Journal of Intelligent Learning Systems and Applications*, *9*(1), 1–16. doi:10.4236/jilsa.2017.91001

Gopinath, M. P., & Murali, S. (2017). Comparative study on Classification Algorithm for Diabetes Data set. *International Journal of Pure and Applied Mathematics*, *117*(7), 47–52.

Hayashi, Y., & Yukita, S. (2016). Rule extraction using Recursive-Rule extraction algorithm with J48graft combined with sampling selection techniques for the diagnosis of type 2 diabetes mellitus in the Pima Indian dataset. *Informatics in Medicine Unlocked*, *2*, 92–104. doi:10.1016/j.imu.2016.02.001

Kavakiotis, I., Tsave, O., Salifoglou, A., Maglaveras, N., Vlahavas, I., & Chouvarda, I. (2017). Machine Learning and Data Mining Methods in Diabetes Research. *Computational and Structural Biotechnology Journal*, *15*, 104–116. doi:10.1016/j.csbj.2016.12.005 PMID:28138367

Nanthini, D., & Thangaraju, P. (2015, August). A Hybrid Classification Model For Diabetes Dataset Using Decision Tree. *International Journal of Emerging Technologies and Innovative Research*, *2*(8), 3302–3308.

Patil, B. M., Joshi, R. C., & Toshniwal, D. (2010, December). Hybrid prediction model for Type-2 diabetic patients. *Expert Systems with Applications*, *37*(12), 8102–8108. doi:10.1016/j.eswa.2010.05.078

Pradeep Kandhasamy, J., & Balamurali, S. (2015). Performance Analysis of Classifier Models to Predict Diabetes Mellitus. *Procedia Computer Science*, *47*, 45–51. doi:10.1016/j.procs.2015.03.182

Ramezani, Maadi, & Khatami. (2018). Analysis of a Population of Diabetic Patients Databases in Weka Tool. *International Journal of Scientific and Engineering Research, 2*(5).

Saravananathan & Velmurugan. (2016). Analyzing Diabetic Data using Classification Algorithms in Data Mining. *Indian Journal of Science and Technology, 9*(43).

Sisodia, D., & Sisodia, D. S. (2018). Prediction of Diabetes using Classification Algorithms. *Procedia Computer Science*, *132*, 1578–1585. doi:10.1016/j.procs.2018.05.122

Temurtas, H., Yumusak, N., & Temurtas, F. (2009, May). A comparative study on diabetes disease diagnosis using neural networks. *Systems with Applications*, *36*(4), 8610–8615. doi:10.1016/j.eswa.2008.10.032

Wu, H., Yang, S., Huang, Z., He, J., & Wang, X. (2018). Type 2 diabetes mellitus prediction model based on data mining. *Informatics in Medicine Unlocked*, *10*, 100–107. doi:10.1016/j.imu.2017.12.006

Yasodha & Kannan. (2011). A novel hybrid intelligent system with missing value imputation for diabetes diagnosis. *Alexandria Engineering Journal, 57*(3).

ADDITIONAL READING

Dey, A. (2016). Machine Learning Algorithms: A Review. *International Journal of Computer Science and Information Technologies*, *7*(3), 1174–1179.

Pedregosa, F., Varoquaux, G., Gramfort, A., & Michel, V. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*.

Simon, A., Deo, M. S., Selvam, V., & Babu, R. (2015). An Overview of Machine Learning and its Applications. *International Journal of Electrical Sciences and Engineering*, *1*(1), 22–24.

Van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array:a structure for efficient numerical computation. *Computing in Science & Engineering*, *11*, 2011.

Zito, T., Wilbert, N., Wiskott, L., & Berkes, P. (2008). Modular toolkit for Data Processing (MDP*): A Python data processing framework. Frontiers in Neuroinformatics*, *2*, 2008. doi:10.3389/neuro.11.008.2008 PMID:19169361

KEY TERMS AND DEFINITIONS

Accuracy: It is a metric used to predict the correctness of a machine learning model. The model is trained using the train data and a classifier is built. The test data is used to cross validate the classifier model. The percentage of correctly classified instances is termed as accuracy.

Area Under the Curve (AUC) Score: Area under the curve (AUC) is a binary classification metric. It considers all the possible thresholds. Different threshold values result in distinct true positive/false positive rates. As the threshold is decreased, more true positives (but also more false positives) instances are discovered.

F1 Score: F1 score is a combination function of precision and recall. It is used when we need to seek a balance between precision and recall.

Hyper-Parameter Tuning: The model architecture is defined by several parameters. These parameters are referred to as hyper parameters. The process of searching for an ideal model architecture for optimal accuracy score is referred to as hyper parameter tuning.

Logistic Regression: Logistic regression is a classification algorithm that comes under supervised learning and is used for predictive learning. Logistic regression is used to describe data. It works best for dichotomous (binary) classification.

Multilayer Perceptron: Multilayer perceptron falls under artificial neural networks (ANN). It is a feed forward network that consists of a minimum of three layers of nodes- an input layer, one or more hidden layers and an output layer. It uses a supervised learning technique, namely, back propagation for training. Its main advantage is that it has the ability to distinguish data that is not linearly separable.

Random Forest: Random forest is a supervised learning algorithm and is used for classification and regression. It is an ensemble learning method that operates by constructing multiple decision trees and merges them together to obtain an accurate and stable prediction. Generally, it produces great results even without hyper parameter tuning.

Recall: Recall is the ratio of true positives to the sum of true positives and false negatives.

Sensitivity/Precision: It is the ratio of true positives to the sum of the true positive and false negative.

Specificity: It is the ratio of true negatives to the sum of the true negatives and false positives.

Supervised Learning: Machine learning is broadly classified into two: supervised learning and unsupervised learning. In supervised learning, the machine learns from examples. Historical or train data is needed which is given as an input to the machine and a classifier model is formed. A supervised algorithm also needs a target value. On the contrary, unsupervised learning algorithms need neither the train data nor the target value.