CASE STUDY

Topic: Early Prediction Of Diabetes

Team Name: Free Thinkers

Branch : AIML

Team members:

A.Sri Nikhila 20A21A6122

U.Sai Durga 20A21A6158

G.Om Sai 21A25A6103

Y.Manikanta 20A21A6160

Abstract

Diabetes is a common, chronic disease. Prediction of diabetes at an early stage can lead to improved treatment. Data mining techniques are widely used for prediction of disease at an early stage. Diabetes is predicted using significant attributes, and the relationship of the differing attributes is also characterized. Various tools are used to determine significant attribute selection, and for clustering, prediction, and association rule mining for diabetes .Our findings indicate a strong association of diabetes with body mass index (BMI) and with glucose level, which was extracted via the Apriori method. Random forest (RF) and K-means clustering techniques were implemented for the prediction of diabetes. The Random Forest provided a best accuracy of 75.7%, and may be useful to assist medical professionals with treatment decisions.

Introduction

The disease or condition which is continual or whose effects are permanent is a chronic condition. These types of diseases affected quality of life, which is major adverse effect. Diabetes is one of the most acute diseases, and is present worldwide. A major reason of deaths in adults across the globe includes this chronic condition. Chronic conditions are also cost associated. A major portion of budget is spent on chronic diseases by governments and individuals . Research on biological data is limited but with the passage of time enables computational and statistical models to be used for analysis. A sufficient amount of data is also being gathered by healthcare organizations. New knowledge is gathered when models are developed to learn from the observed data using data mining techniques. Data mining is the process of extracting from data and can be utilized to create a decision making process with efficiency in the medical domain . Several data mining techniques have been utilized for disease prediction as well as for knowledge discovery from biomedical data .
Diagnosis of diabetes is considered a challenging problem for quantitative research. Some parameters like A1c , fructosamine, white blood cell count, fibrinogen and hematological indices were shown to be ineffective due to some limitations. Different research studies used these parameters for the diagnosis of diabetes . A few treatments have thought to raise A1C including chronic ingestion of liquor, salicylates and narcotics. Ingestion of vitamin C may elevate A1c when estimated by electrophoresis but levels may appear to diminish when estimated by chromatography . Most studies have suggested that a higher white blood cell count is due to chronic inflammation during hypertension . A family history of diabetes has not been associated with BMI and insulin . However, an increased BMI is not always associated with abdominal obesity . A single parameter is not very effective to accurately diagnose diabetes and may be misleading in the decision making process. There is a need to combine different parameters to effectively predict diabetes at an early stage. Several existing techniques have not provided effective results when different parameters were used for prediction of diabetes .In our study, diabetes is predicted with the assistance of significant attributes, and the association of the differing attributes. We examined the diagnosis of diabetes using RF and K-Means Clustering.

Methods and materials

Dataset

The dataset used in this study, is originally taken from the National Institute of Diabetes and Digestive and Kidney Diseases (**publicly available at: UCI ML Repository**). The main Objective of using this dataset was to predict through diagnosis whether a patient has diabetes, based on certain diagnostic measurements included in the dataset.The type of dataset and problem is a classic supervised binary classification. The Pima Indian Diabetes (PID) dataset having: 9 = 8 + 1 (Class Attribute) attributes, 768 records describing female patients (of which there were 500 negative instances (65.1%) and 268 positive instances (34.9%)). The detailed description of all attributes is given in below table.

Sr. #	Attribute Name	Attribute Description	Mean ± S. D
1	Pregnancies	Number of times a woman got pregnant	3.8 ± 3.3
2	Glucose (mg/dl)	Glucose concentration in oral glucose tolerance test for 120 min	120.8 ± 31 .9
3	Blood Pressure (mmHg)	Diastolic Blood Pressure	69.1 ± 19. 3
4	Skin Thickness (mm)	Fold Thickness of Skin	20.5 ± 15. 9
5	Insulin (mu U/mL)	Serum Insulin for 2 h	79.7 ± 115 .2
6	BMI (kg/m2)	Body Mass Index (weight/(height)^2)	31.9 ± 7.8
7	Diabetes Pedigree Function	Diabetes pedigree Function	0.4 ± 0.3
8	Age	Age (years)	33.2 ± 11. 7
9	Outcome	Class variable (class value 1 for positive 0 for Negative for diabetes)	

Preparing the data

Data preprocessing

In real-world data there can be missing values and/or noisy and inconsistent data. If data quality is low then no quality results may be found. It is necessary to preprocess the data to achieve quality results. Cleaning, integration, transformation, reduction, and discretization of data are applied to preprocess the data. It is important to make the data more appropriate for data mining and analysis with respect to time, cost, and quality.

Data cleaning

Data cleaning consists of filling the missing values and removing noisy data. Noisy data contains outliers which are removed to resolve inconsistencies. In our dataset, glucose, blood Pressure, skin thickness, insulin, and BMI have some zero (0) values. Thus, all the zero values were replaced with the median value of that attribute.

Data reduction

Data reduction obtains a reduced representation of the dataset that is much smaller in volume yet produces the same (or almost the same) result. Dimensionally reduction has been used to reduce the number of attributes in a dataset . The principal component analysis method was used to extract significant attributes from a complete dataset. Glucose, BMI, diastolic blood pressure and age were significant attributes in the dataset.

Data transformation

Data transformation consists of smoothing, normalization, and aggregation of data . For the smoothing of data, the binning method has used. The attribute of age has been useful to classify in five categories, as shown in table below.

Age(Years)	Age Bins
≤30	Youngest
31–40	Younger
41–50	Middle aged
51–60	Older
≥61	Oldest

Table. Binning of age.

Glucose	Glucose Bins
≤60	Very Low
61–80	Low
81–140	Normal
141–180	Early Diabetes
≥181	Diabetes

Table. Binning of glucose.

Blood Pressure	Diastolic Blood Pressure Bins
<61	Very low
61–75	Low
75–90	Normal
91–100	High
>100	Hypertension

Table. Binning of diastolic blood pressure.

BMI	BMI Bins
<19	Starvation
19–24	Normal
25–30	Overweight
31–40	Obese
>40	Very Obese

Table. Binning of BMI.

For the completion of the preprocessing task, selection of significant attributes and transformation of significant attributes into bins are done after data cleaning. The preprocessed dataset visualization is shown in Fig. 1.

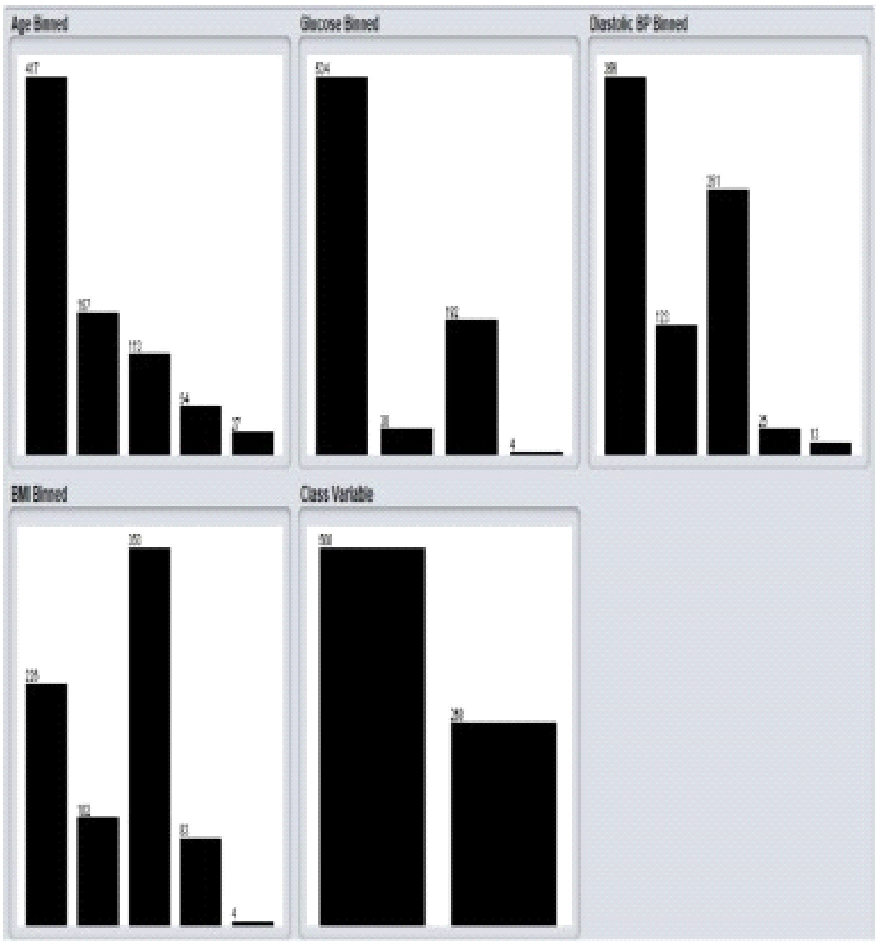


Fig 1: preprocessed dataset visualization

Association rule mining

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

i. determine the frequent item set
ii. generate rules

Association rule mining plays an important role in medical as well as in commercial data analysis to detect and characterize interesting and important patterns. There are several methods to generate rules from data using association rule mining algorithms such as the Apriori algorithm, Tertius and predictive Apriori algorithms. Mostly, association rule-based algorithms are linked with Apriori, which make it a state-of-the-art algorithm. Apriori works as an iterative method to identify the frequent item set in a given dataset, and to generate important rules from it. To determine the association between two item sets X and Y, there is a need to set the minimum support of that fraction of transactions which contains both X and Y called minsupp. The other important task is to set the minimum confidence that measures how often items in Y appear in transactions that contain X, known as minconf, to determine frequent item sets. There were only 268 patients with diabetes in dataset, so only those instances were used to generate rules among them. To develop rules from a given dataset, set minimum support as 0.25 and minimum confidence as 0.9 to generate the following three different rules. Best rules are shown in Table.

Rule#1. If **(**BMI = Obesity) → Class = Yes
Rule#2. If **(**Glucose = Diabetes) → Class = Yes
Rule#3. If (Glucose = Diabetes ∩ BMI = Obesity) → Class = Yes

Table. Association Rules using Apriori.

Modeling

Random forest (RF)

The random forest method is a flexible, fast, and simple machine learning algorithm which is a combination of tree predictors. Random forest produces satisfactory results most of the time. It is difficult to improve on its performance, and it can also handle different types of data including numerical, binary, and nominal. Random forest builds multiple decision trees and aggregates them to achieve more suitable and accurate results. It has been used for both classification and regression. Classification is a major task of machine learning. It has the same hyper parameters as the decision tree or bagging classifier. The fact behind random forest is the overlapping of random trees, and it can be analyzed easily. Suppose if seven random trees have provided the information related to some variable, among them four trees agree and the remaining three disagree. On the basis of majority voting, the machine learning model is constructed based on probabilities. In random forest, a random subset of attributes gives more accurate results on large datasets, and more random trees can be generated by fixing a random threshold for all attributes, instead of finding the most accurate threshold. This algorithm also solves the overfitting issue.

K- means clustering

Clustering is the process of grouping similar objects together on the basis of their characteristics. It is an unsupervised learning technique, in which we determine the natural grouping of instances given for unlabeled data. The clusters are similar to each other. However, the objects of one cluster are different from the objects of other clusters. In clustering, intra clustering similarity between objects is high and inter cluster similarity of objects is low. There are many type of clustering, such as partitioning and Hierarchal clustering but in this study, the k-Means clustering method was used. K-Means clustering is relatively simple to implement and understandable, and works on numerical data, in which K is represented as centers of clusters. Taking the distance of each datapoint from the center it assigns each instance to a cluster, and moves cluster centers by taking the means of all the data points in a cluster and repeating until the cluster center stops moving.

Results

The random forest method provided an accuracy of 74.7%,and K-means clustering method has given 73.6% accuracy. In this work different steps were taken. The proposed approach uses different classification and ensemble methods and implemented using python. These methods are standard Machine Learning methods used to obtain the best accuracy from data. In this work we see that random forest classifier achieves better compared to others. Overall we have used best Machine Learning techniques for prediction and to achieve high performance accuracy. Figure shows the result of these Machine Learning methods. Here feature played important role in prediction is presented for random forest algorithm. The sum of the importance of each feature playing major role for diabetes have been plotted, where X-axis represents the importance of each feature and Y-Axis the names of the features.

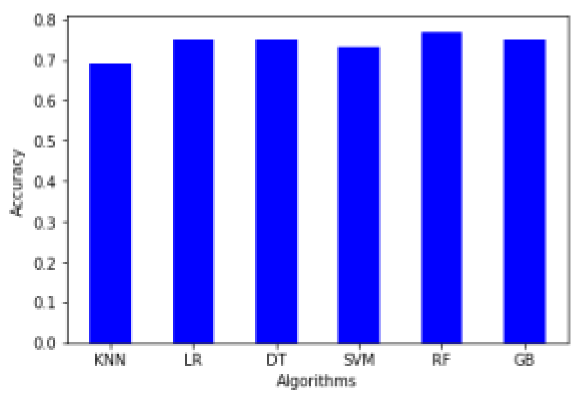


Figure: Accuracy Result of Machine learning methods

Here feature played important role in prediction is presented for random forest algorithm. The sum of the importance of each feature playing major role for diabetes have been plotted, where X-axis represents the importance of each feature and Y-Axis the names of the features.

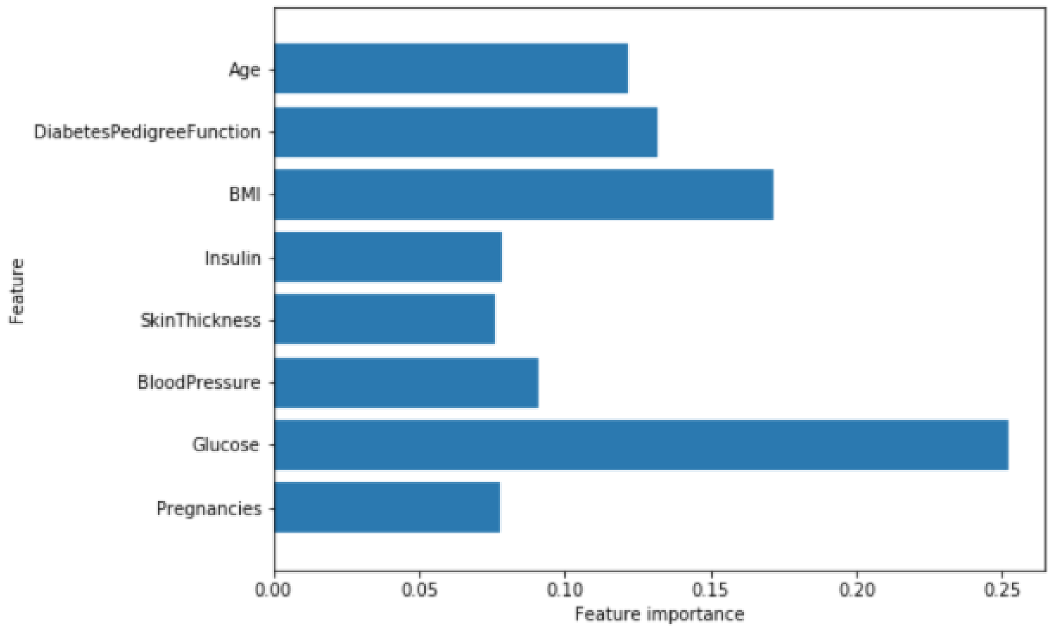


Figure: Feature Importance Plot for Random Forest

Early Prediction Of Diabetes

In [69]:
```
import pandas as pd
```

In [70]:
```
import numpy as np
```

In [71]:
```
import seaborn as sns
```

In [72]:
```
%matplotlib inline
```

In [73]:
```
import matplotlib.pyplot as plt
```

In [74]:
```
from sklearn.model_selection import train_test_split
```

In [75]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [76]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [77]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

In [78]:
```
from sklearn import metrics
```

In [79]:
```
data=pd.read_csv("C:/Users/Admin/Desktop/case study2022 aiml/diabetes.csv")
```

In [80]:
```
data.head(10)
```

Out[80]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
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
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

In [81]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
Pregnancies                 768 non-null int64
Glucose                     768 non-null int64
BloodPressure               768 non-null int64
SkinThickness               768 non-null int64
Insulin                     768 non-null int64
BMI                         768 non-null float64
DiabetesPedigreeFunction    768 non-null float64
Age                         768 non-null int64
Outcome                     768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [82]:
```
data.describe()
```

Out[82]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

In [83]:
```
data.isnull().head()
```

Out[83]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False

In [84]:
```
data.isnull().sum()
```

Out[84]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [85]:
```
data_copy = data.copy(deep = True)
```

In [86]:
```
data_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] =data_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']
```

In [87]:
```
print(data_copy.isnull().sum())
```

```
Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
dtype: int64
```

In [88]:
```
data_copy['Glucose'].fillna(data_copy['Glucose'].mean(), inplace = True)
```

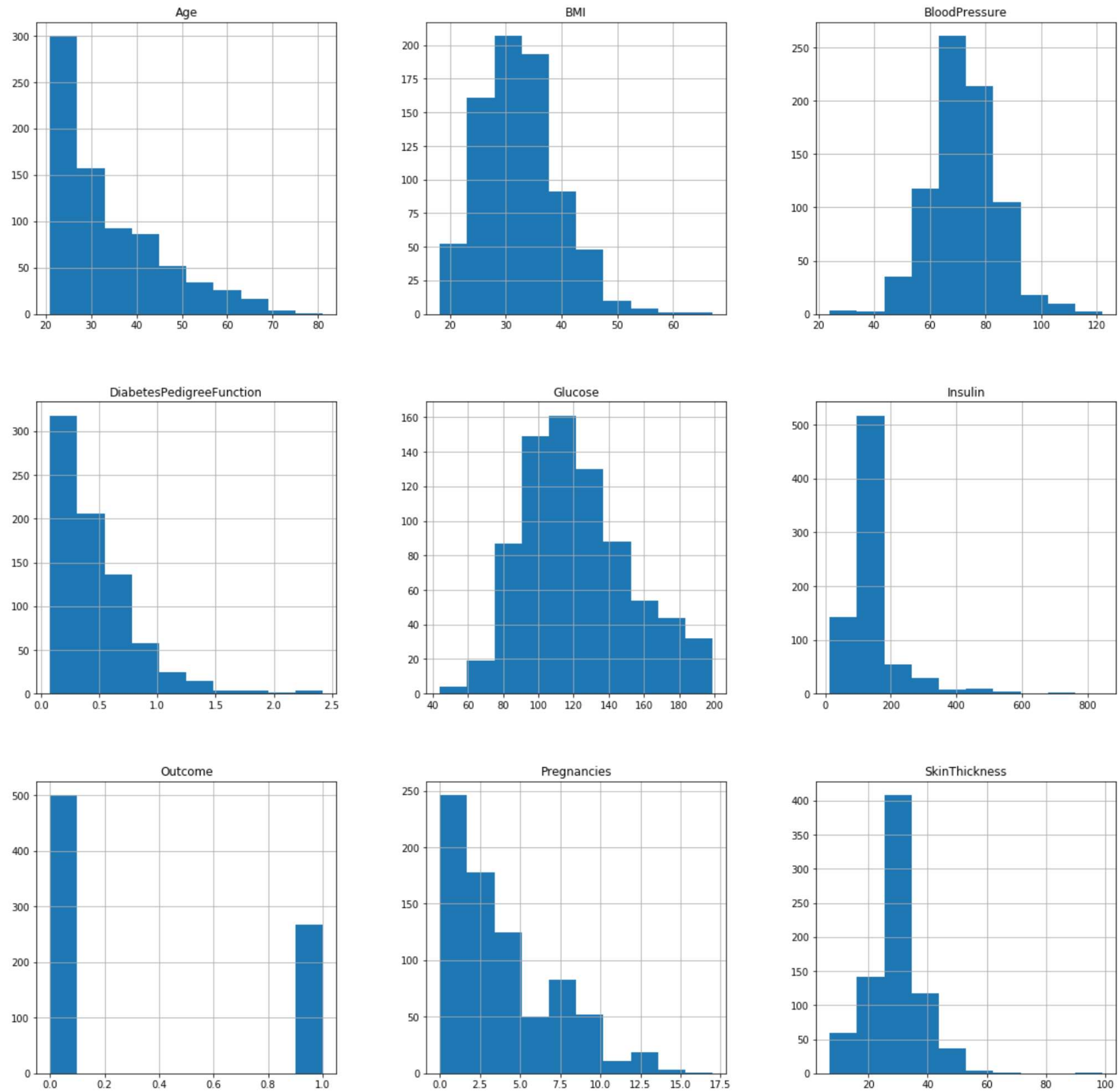
In [89]:
```
data_copy['BloodPressure'].fillna(data_copy['BloodPressure'].mean(), inplace = True)
```

In [90]:
```
data_copy['SkinThickness'].fillna(data_copy['SkinThickness'].median(),inplace=True)
```

In [91]:
```
data_copy['Insulin'].fillna(data_copy['Insulin'].median(), inplace = True)
```

In [92]:
```
data_copy['BMI'].fillna(data_copy['BMI'].median(), inplace = True)
```

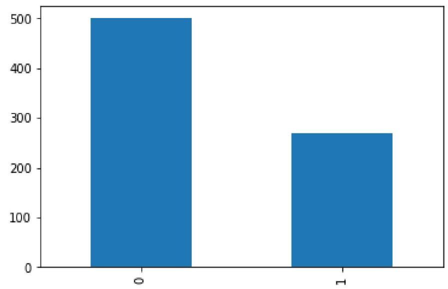
In [93]:
```
data_copy.hist(figsize=(20,20))
```
Out[93]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001B0EAB7FF98>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001B0EAC32208>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001B0EAC086A0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001B0EA543A90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001B0EA6C3080>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001B0EA6F3630>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001B0EA723BE0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001B0EA7621D0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001B0EA762208>]],
      dtype=object)
```



In [94]:
```
print(data.Outcome.value_counts())
```

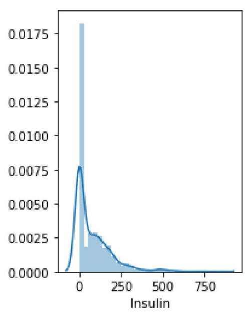
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [95]:
```
p=data.Outcome.value_counts().plot(kind="bar")
```



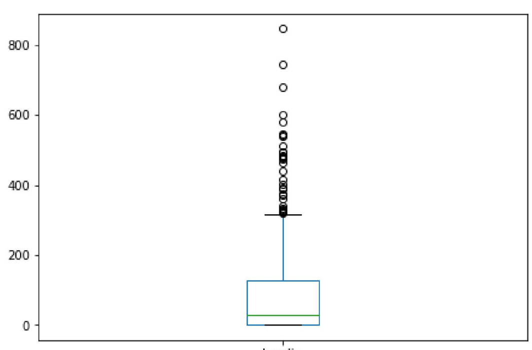
In [96]:
```
plt.subplot(121), sns.distplot(data['Insulin'])
```

Out[96]:
```
(<matplotlib.axes._subplots.AxesSubplot at 0x1b0ec13fcf8>,
 <matplotlib.axes._subplots.AxesSubplot at 0x1b0ec13fcf8>)
```

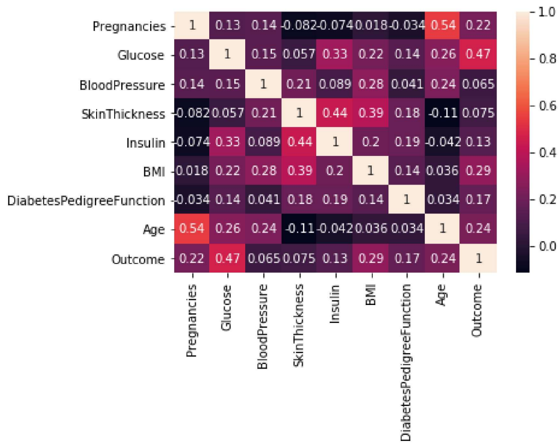


In [97]:
```
plt.subplot(122), data['Insulin'].plot.box(figsize=(16,5))
```

Out[97]:
```
(<matplotlib.axes._subplots.AxesSubplot at 0x1b0ec1c0748>,
 <matplotlib.axes._subplots.AxesSubplot at 0x1b0ec1c0748>)
```



```
In [98]: p=sns.heatmap(data.corr(),annot=True)
```



Building the model

```
In [99]: X = data.drop('Outcome', axis=1)
```

```
In [100]: y = data['Outcome']
```

```
In [101]: from sklearn.model_selection import train_test_split
```

```
In [102]: X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.33,random_state=7)
```

```
In [103]: knn=KNeighborsClassifier(n_neighbors=15)
```

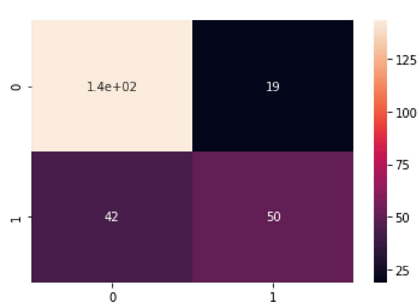
```
In [104]: knn.fit(X_train,y_train)
Out[104]: KNeighborsClassifier(n_neighbors=15)
```

```
In [105]: y_pred=knn.predict(X_test)
```

```
In [106]: y_pred
Out[106]: array([0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
                 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0,
                 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
                 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
                 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
                 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
                 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
                 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1,
                 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0,
                 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0,
                 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
                 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0], dtype=int64)
```

```
In [107]: cm=confusion_matrix(y_test,y_pred)
```

```
In [108]: sns.heatmap(cm,annot=True)
Out[108]: <matplotlib.axes._subplots.AxesSubplot at 0x1b0ec3b2f98>
```



Random Forest Classifier

In [109]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [110]:
```
rfc = RandomForestClassifier(n_estimators=200)
```

In [111]:
```
rfc.fit(X_train, y_train)
```

Out[111]:
```
RandomForestClassifier(n_estimators=200)
```

In [112]:
```
from sklearn import metrics
```

In [113]:
```
predictions = rfc.predict(X_test)
```

In [114]:
```
print("Accuracy_Score =", format(metrics.accuracy_score(y_test, predictions)))
```

```
Accuracy_Score = 0.7874015748031497
```

In [115]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

In [116]:
```
print(confusion_matrix(y_test, predictions))
```

```
[[138  24]
 [ 30  62]]
```

In [117]:
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.82      0.85      0.84       162
           1       0.72      0.67      0.70        92

    accuracy                           0.79       254
   macro avg       0.77      0.76      0.77       254
weighted avg       0.79      0.79      0.79       254
```

In [118]:
```
rfc.predict([[0,137,40,35,168,43.1,2.228,33]])
```

Out[118]:
```
array([1], dtype=int64)
```

In [119]:
```
rfc.predict([[8,125,96,0,0,0.0,0.232,54]])
```

Out[119]:
```
array([0], dtype=int64)
```

In [120]:
```
rfc.predict([[7,139,90,45,170,45.2,3.789,35]])
```

Out[120]:
```
array([1], dtype=int64)
```