DIABETES DISEASE PREDICTION
MACHINE LEARNING

REPORT BY,

SHILPA J SHETTY

DATA SCIENCE INTERN AT

EXPOSYS DATA LABS

NISHMA NAYANA

DATA SCIENCE INTERN AT

EXPOSYS DATA LABS

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. In this project, 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, sampling and for prediction. Significant attributes selection was done via the principal component analysis method. Our findings indicate a strong association of diabetes with body mass index (BMI) and with glucose level. Gradient Boost, XGBoost, AdaBoost and random forest (RF) techniques were implemented for the prediction of diabetes. The Gradient Boost technique provided a best accuracy and recall score of 94.00%, and may be useful to assist medical professionals with treatment decisions.

1)Introduction

Diabetes is a disease that occurs when your blood glucose, also called blood sugar, is too high. Blood glucose is your main source of energy and comes from the food you eat. Insulin, a hormone made by the pancreas, helps glucose from food get into your cells to be used for energy.The disease or condition which is continual or whose effects are permanent is a chronic condition. These types of diseases affect quality of life, which is a major adverse effect. Diabetes is one of the most acute diseases, and is present worldwide. A major reason for deaths in adults across the globe includes this chronic condition. Chronic conditions are also cost associated. A major portion of the budget is spent on chronic diseases by governments and individuals . The worldwide statistics for diabetes in the year 2013 revealed around 382 million individuals had this ailment around the world. It was the fifth leading cause of death in women and eight leading cause of death for both sexes in 2012. Higher income countries have a high probability of diabetes. In 2017, approximately 451 million adults were treated with diabetes worldwide. It is projected that in 2045, almost 693 million patients with diabetes will exist around the globe and half of the population will be undiagnosed. In addition, 850 million USD were spent on patients with diabetes in 2017. 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.

2)Methods

2.1) 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 [29]). 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. Many limitations were faced during the selection of the occurrences from the bigger 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%)).

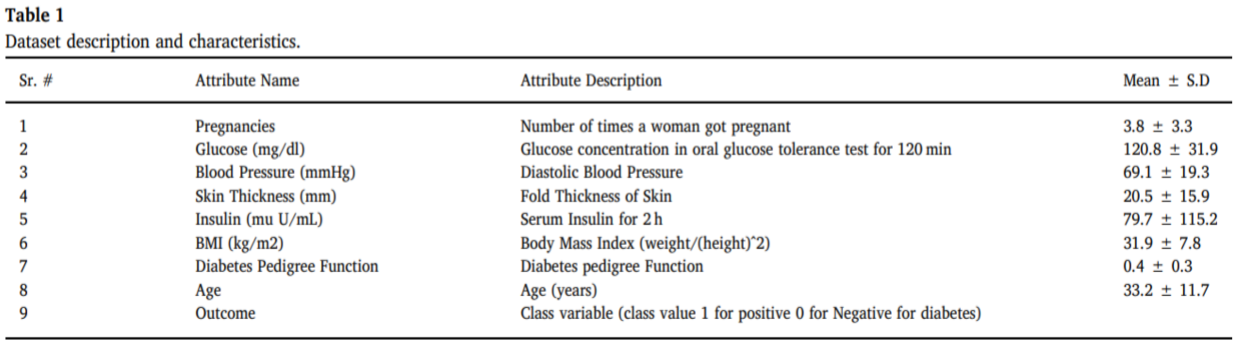


Table 1
Dataset description and characteristics.

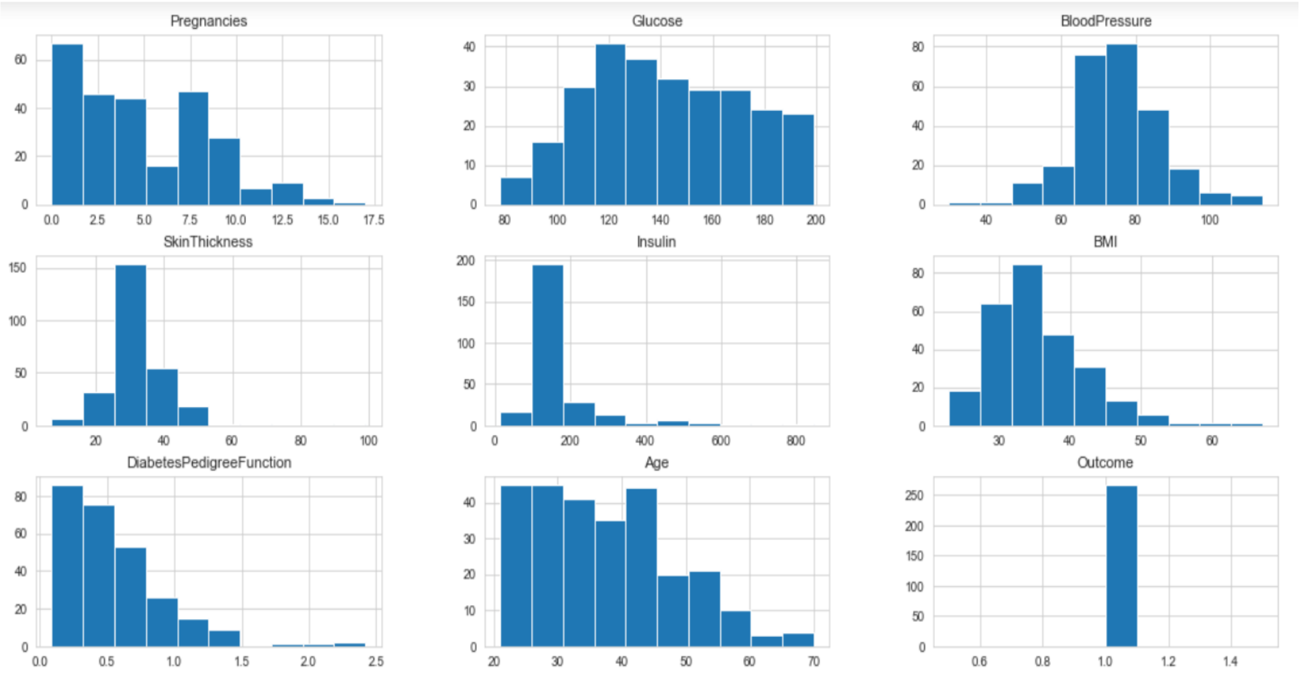
Sr. #	Attribute Name	Attribute Description	Mean ± S.D
1	Pregnancies	Number of times a woman got pregnant	3.8 ± 3.3
2	Glucose (mg/dl)	Glucose concentration in oral glucose tolerance test for 120 min	120.8 ± 31.9
3	Blood Pressure (mmHg)	Diastolic Blood Pressure	69.1 ± 19.3
4	Skin Thickness (mm)	Fold Thickness 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)ˆ2)	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)	

2.2) 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 .

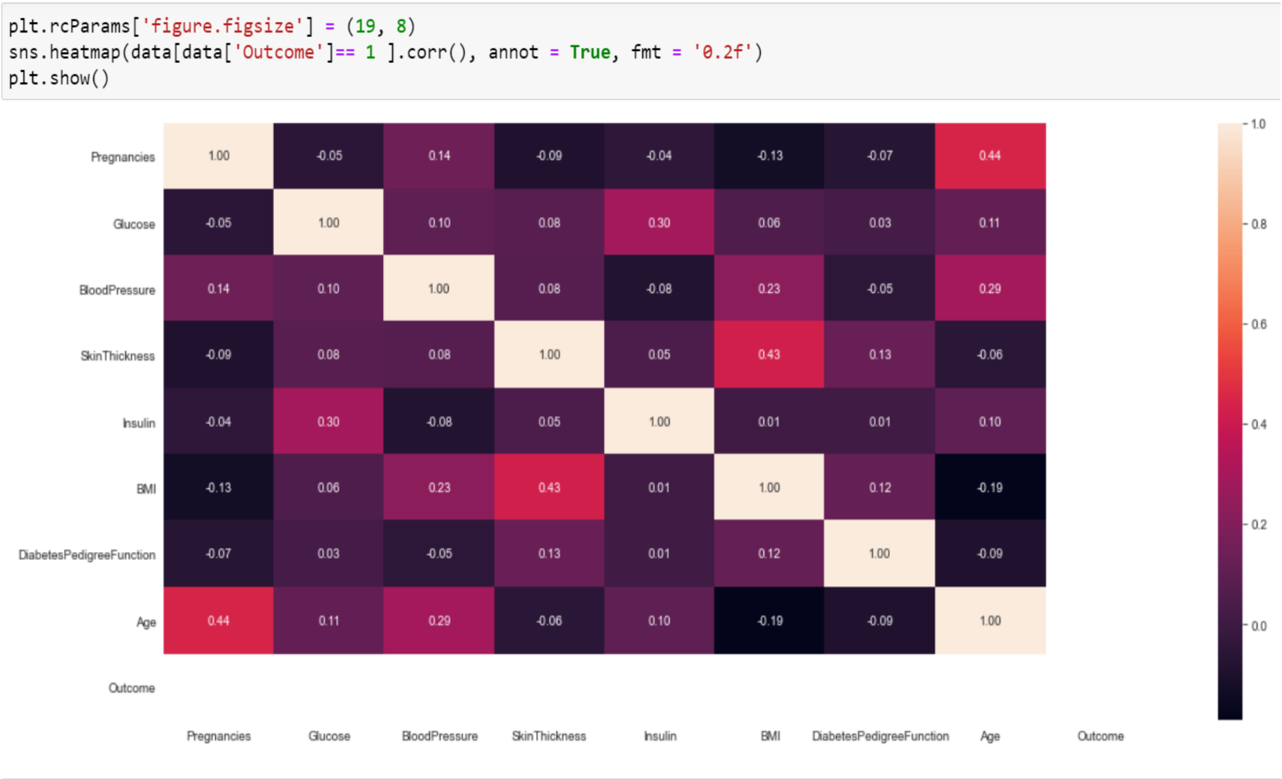
2.2.1. Data cleaning

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 mode value of that attribute with respect to the corresponding output. The distribution of the data with respect to the outcome 1 which is diabetic is shown in the figure below.



2.2.2. 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. Dimensionality reduction has been used to reduce the number of attributes in a dataset . Glucose, BMI, diastolic blood pressure and age were significant attributes in the dataset. The correlation between the attributes is shown in the figure below. At the beginning pregnancies attribute values felt kind of misleading which then ended up helping in increasing the recall so we have not particularly removed any attribute.

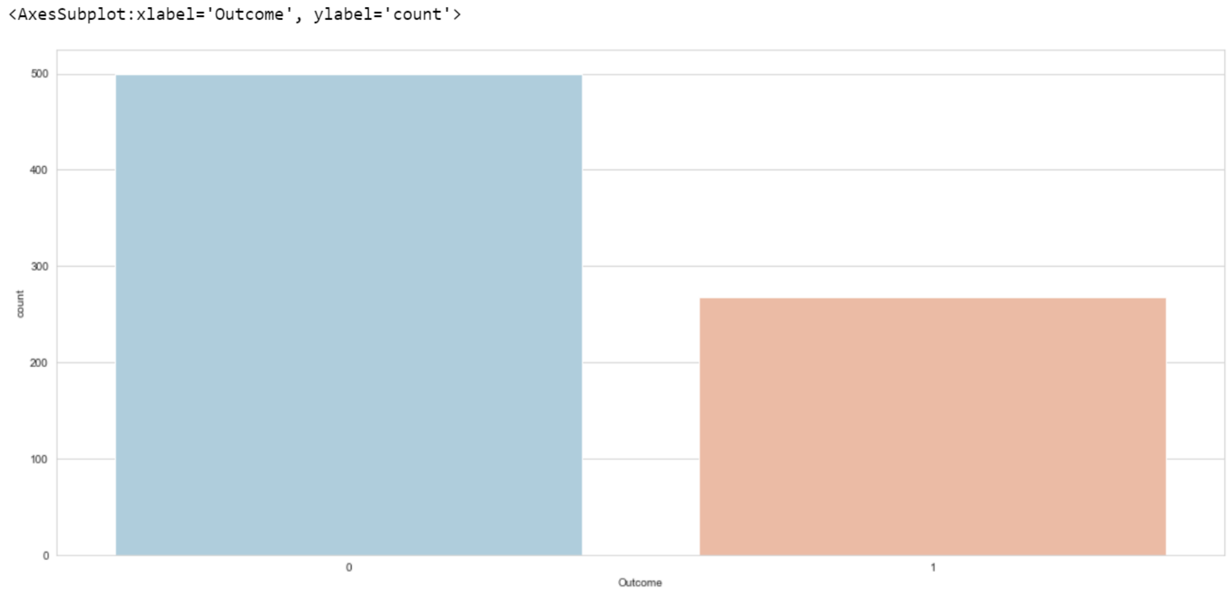


```
plt.rcParams['figure.figsize'] = (19, 8)
sns.heatmap(data[data['Outcome']== 1 ].corr(), annot = True, fmt = '0.2f')
plt.show()
```

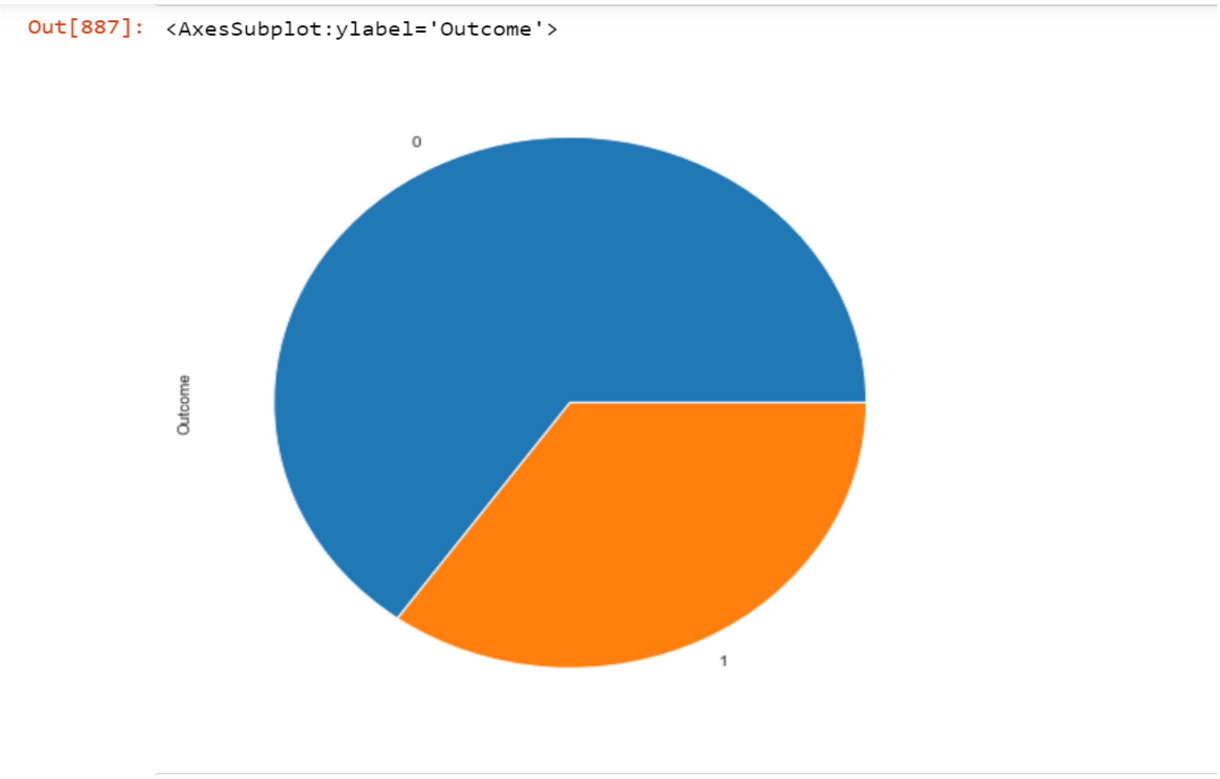
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.00	-0.05	0.14	-0.09	-0.04	-0.13	-0.07	0.44	
Glucose	-0.05	1.00	0.10	0.08	0.30	0.06	0.03	0.11	
BloodPressure	0.14	0.10	1.00	0.08	-0.08	0.23	-0.05	0.29	
SkinThickness	-0.09	0.08	0.08	1.00	0.05	0.43	0.13	-0.06	
Insulin	-0.04	0.30	-0.08	0.05	1.00	0.01	0.01	0.10	
BMI	-0.13	0.06	0.23	0.43	0.01	1.00	0.12	-0.19	
DiabetesPedigreeFunction	-0.07	0.03	-0.05	0.13	0.01	0.12	1.00	-0.09	
Age	0.44	0.11	0.29	-0.06	0.10	-0.19	-0.09	1.00	
Outcome									

2.2.3. Data Re- sampling

Data sampling refers to statistical methods for selecting observations from the domain with the objective of estimating a population parameter. Whereas data resampling refers to methods for economically using a collected dataset to improve the estimate of the population parameter and help to quantify the uncertainty of the estimate. It was found that the majority of the dataset was non-diabetic which is 500/768 records and the rest were diabetic. Which can be seen in the figures below. Therefore RandomOverSampling has been used just to cope up with the imbalance. Apparently, oversampling worked well among all the other sampling techniques.



```
<AxesSubplot:xlabel='Outcome', ylabel='count'>
```



```
Out[887]: <AxesSubplot:ylabel='Outcome'>
```

2.3) Modeling

Four models were used for early prediction of diabetes, following.

2.3.1. XGB Classifier(XGB)

XGBoost is one of the most popular machine learning algorithms these days. Regardless of the type of prediction task at hand; regression or classification.XGBoost is well known to provide better solutions than other machine learning algorithms. In fact, since its inception, it has become the "state-of-the-art" machine learning algorithm to deal with structured data.

- **Speed and performance** : Originally written in C++, it is comparatively faster than other ensemble classifiers.
- **Core algorithm is parallelizable** : Because the core XGBoost algorithm is parallelizable it can harness the power of multi-core computers. It is also parallelizable onto GPU's and across networks of computers making it feasible to train on very large datasets as well.
- **Consistently outperforms other algorithm methods** : It has shown better performance on a variety of machine learning benchmark datasets.
- **Wide variety of tuning parameters** : XGBoost internally has parameters for cross-validation, regularization, user-defined objective functions, missing values, tree parameters, scikit-learn compatible API etc.

XGBoost (Extreme Gradient Boosting) belongs to a family of boosting algorithms and uses the gradient boosting (GBM) framework at its core. It is an optimized distributed gradient boosting library. In our case, this algorithm performs really well.

2.3.2. 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 a 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. This algorithm gives the best accuracy and recall score for our dataset.

2.3.3. AdaBoost Classifier

Ada-boost or Adaptive Boosting is an ensemble boosting classifier proposed by Yoav Freund and Robert Schapire in 1996. It combines multiple classifiers to increase the accuracy of classifiers. AdaBoost is an iterative ensemble method. AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get a high accuracy strong classifier. The basic concept behind Adaboost is to set the weights of classifiers and train the data sample in each iteration such that it ensures the accurate predictions of unusual observations. Any machine learning algorithm can be used as a base classifier if it accepts weights on the training set.

Adaboost should meet two conditions:

- The classifier should be trained interactively on various weighted training examples.
- In each iteration, it tries to provide an excellent fit for these examples by minimizing training error.

AdaBoost is easy to implement. It iteratively corrects the mistakes of the weak classifier and improves accuracy by combining weak learners. You can use many base classifiers with AdaBoost. AdaBoost is not prone to overfitting. This can be found out via experiment results, but there is no concrete reason available.

AdaBoost is sensitive to noise data. It is highly affected by outliers because it tries to fit each point perfectly. AdaBoost is slower compared to XGBoost. In our case, this algorithm performs pretty well but not the best.

2.3.4. Gradient Boost Classifier

In Gradient Boosting, each predictor tries to improve on its predecessor by reducing the errors. But the fascinating idea behind Gradient Boosting is that instead of fitting a predictor on the data at each iteration, it actually fits a new predictor to the residual errors made by the previous predictor. In order to make initial predictions on the data, the algorithm will get the log of the odds of the target feature. This is usually the number of True values(values equal to 1) divided by the number of False values(values equal to 0).Once it has the log(odds), we convert that value to a probability by using a logistic function in order to make predictions.For every instance in the training set, it calculates the residuals for that instance, or, in other words, the observed value minus the predicted value.Once it has done this, it build a new Decision Tree that actually tries to predict the residuals that was previously calculated. However, this is where it gets slightly tricky in comparison with Gradient Boosting Regression.When building a Decision Tree, there is a set number of leaves allowed. This can be set as a parameter by a user, and it is usually between 8 and 32. This leads to two of the possible outcomes:

- Multiple instances fall into the same leaf
- A single instance has its own leaf

to make new predictions, we do 2 things:

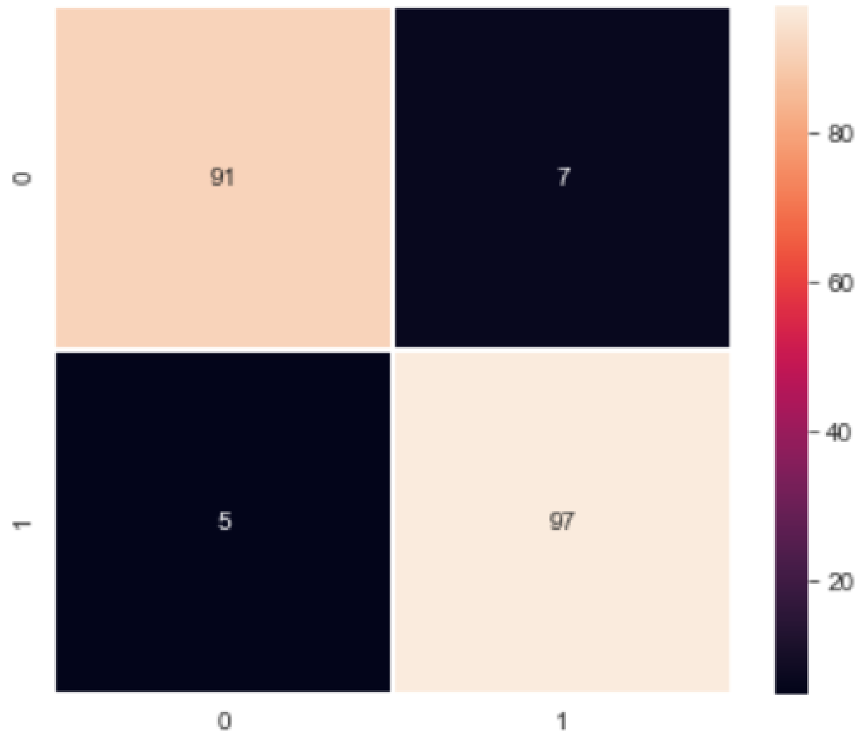
- get the log(odds) prediction for each instance in the training set
- convert that prediction into a probability
- The learning_rate is a hyperparameter that is used to scale each tree's contribution, sacrificing bias for better variance. In other words, we multiply this number by the predicted value so that we do not overfit the data
- Once we have calculated the log(odds) prediction, we now must convert it into a probability using the previous formula for converting log(odds) values into probabilities
- After we have done this process, we calculate the new residuals of the tree and create a new tree to fit the new residuals. Again, the process is repeated until a certain predefined threshold is reached, or the residuals are negligible
- This algorithm performs really well in our case

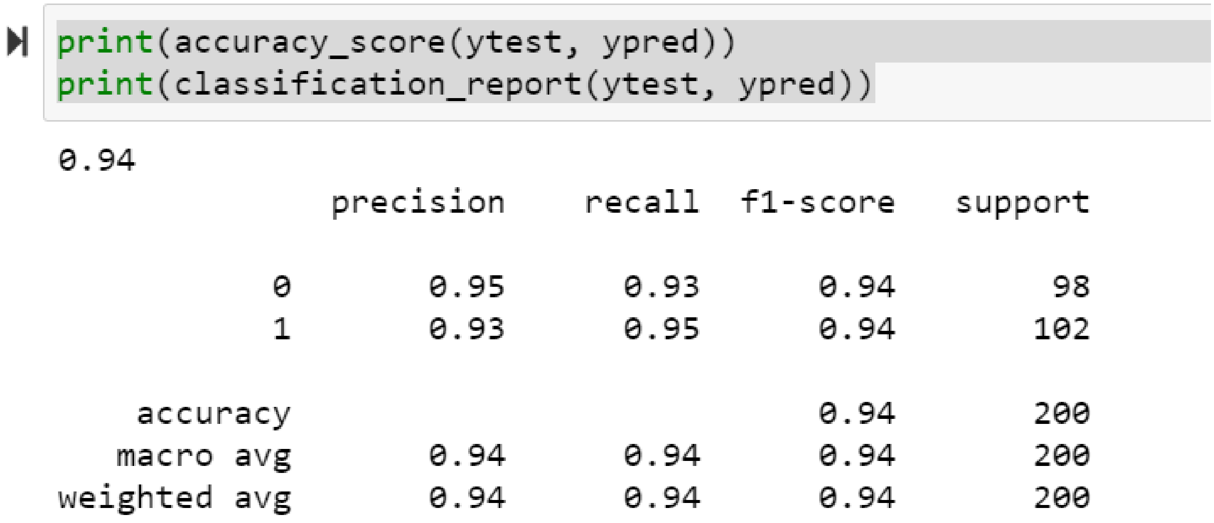
3) Results and discussion

Different classification algorithms were applied on our dataset, and results for all techniques were slightly different as the working criteria of each algorithm is different. The results were evaluated on the basis of accuracy and the recall score of each algorithm. The accuracy and recall of models was predicted with the help of a confusion matrix.

3.1 Accuracy and Recall of XGB Classifier

From the figure below it's clear that this classifier works very well for our prediction. It wrongly classifies only 12 records and gives the f1-score of 94% which is really good.



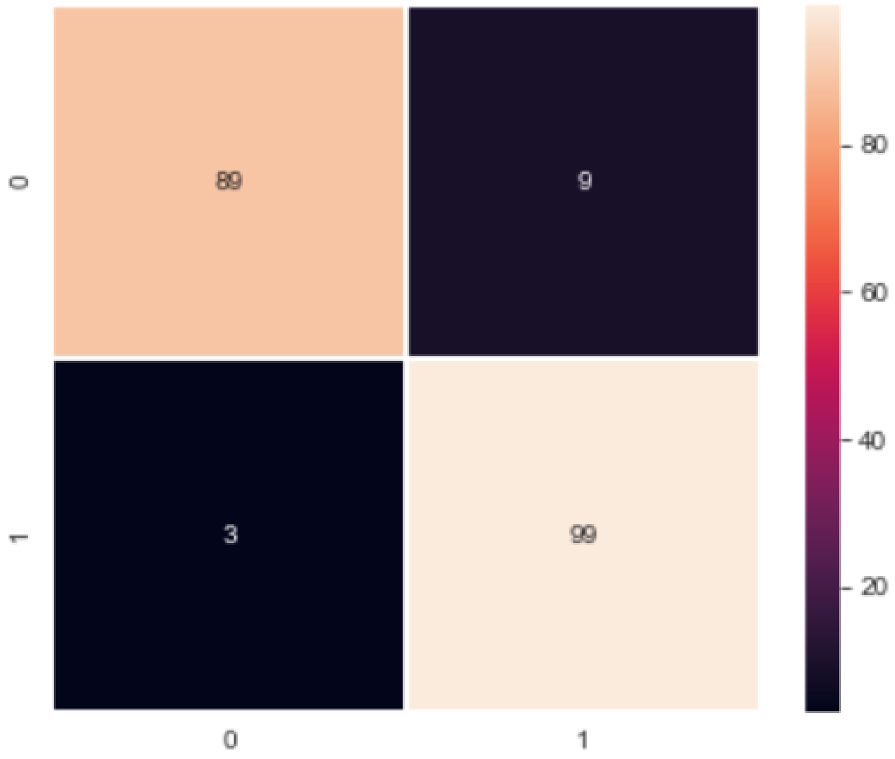


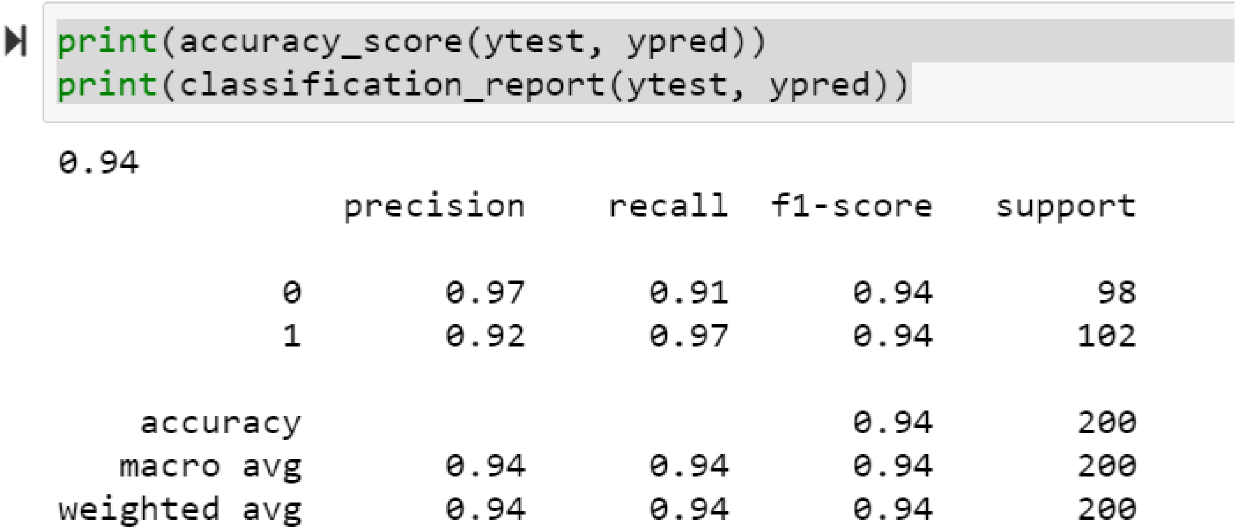
```
print(accuracy_score(ytest, ypred))
print(classification_report(ytest, ypred))
```

```
0.94
              precision    recall  f1-score   support

           0       0.95      0.93      0.94        98
           1       0.93      0.95      0.94       102

    accuracy                           0.94       200
   macro avg       0.94      0.94      0.94       200
weighted avg       0.94      0.94      0.94       200
```

3.2 Accuracy and Recall of Random Forest Classifier

From the figure below it's clear that this classifier works very well for our prediction. It wrongly classifies only 12 records and gives the f1-score of 94% which is really good.



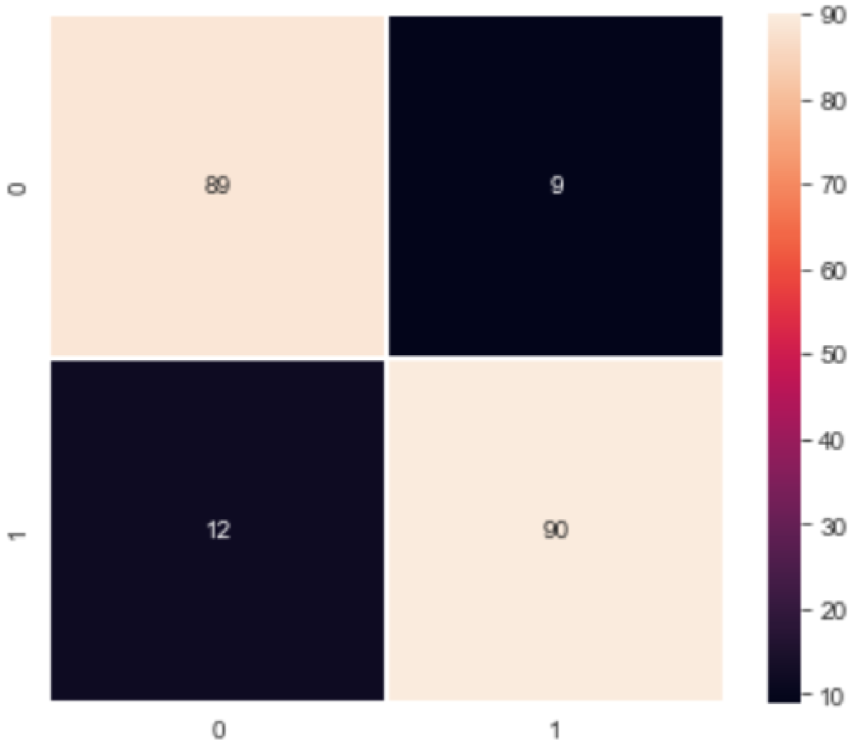


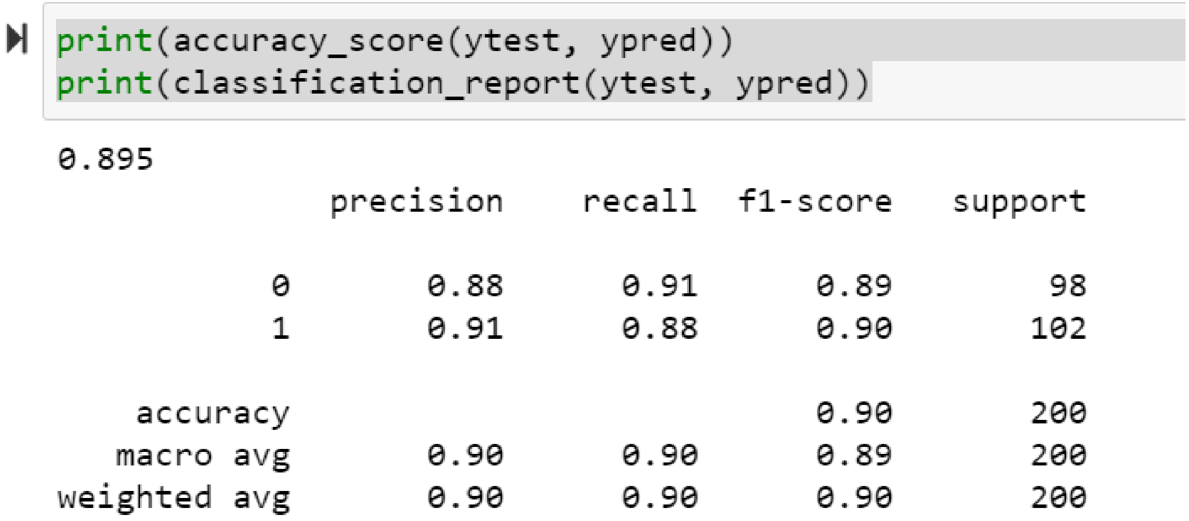
```
print(accuracy_score(ytest, ypred))
print(classification_report(ytest, ypred))
```

```
0.94
              precision    recall  f1-score   support

           0       0.97      0.91      0.94        98
           1       0.92      0.97      0.94       102

    accuracy                           0.94       200
   macro avg       0.94      0.94      0.94       200
weighted avg       0.94      0.94      0.94       200
```

3.3 Accuracy and Recall of AdaBoost Classifier

From the figure below it's clear that this classifier works very well for our prediction. It wrongly classifies 21 records and gives the f1-score of around 90% which is okay.



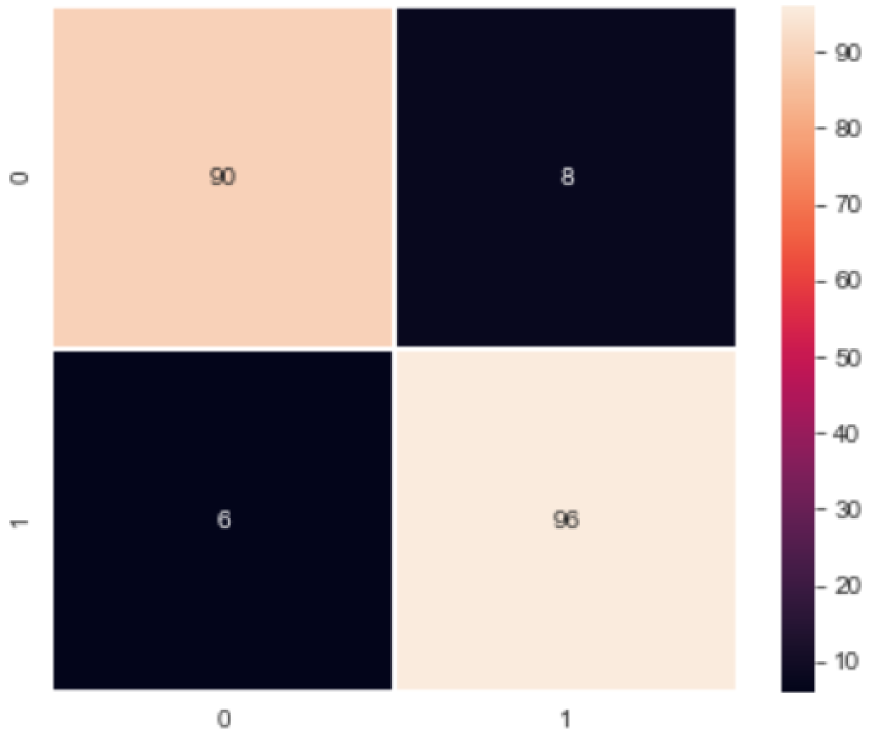


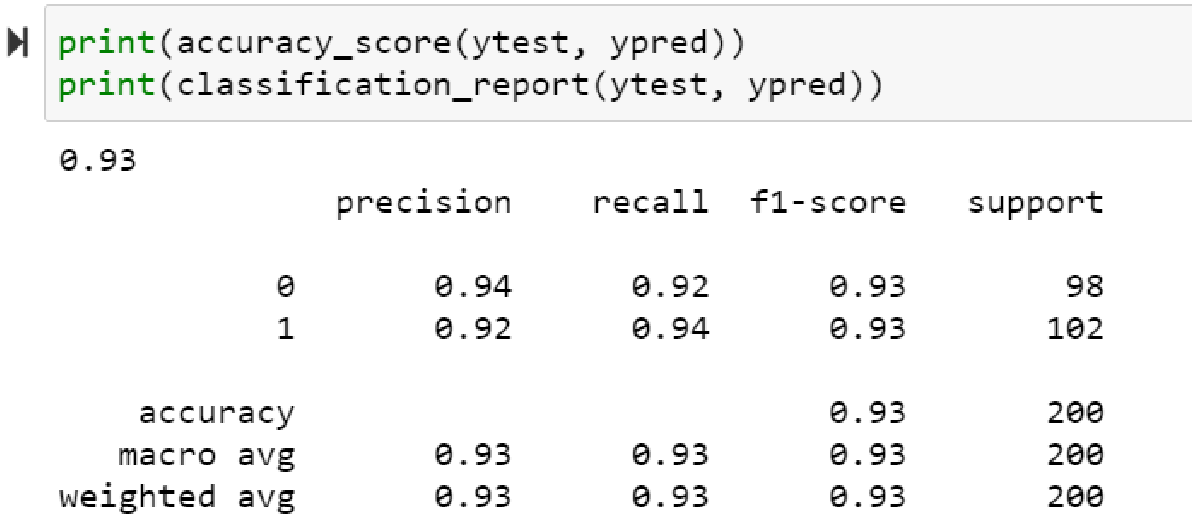
```
print(accuracy_score(ytest, ypred))
print(classification_report(ytest, ypred))
```

```
0.895
              precision    recall  f1-score   support

           0       0.88      0.91      0.89        98
           1       0.91      0.88      0.90       102

    accuracy                           0.90       200
   macro avg       0.90      0.90      0.89       200
weighted avg       0.90      0.90      0.90       200
```

3.1 Accuracy and Recall of Gradient Boost Classifier

From the figure below it's clear that this classifier works very well for our prediction. It wrongly classifies only 14 records and gives the f1-score of 93% which is good enough.





```
print(accuracy_score(ytest, ypred))
print(classification_report(ytest, ypred))
```

```
0.93
              precision    recall  f1-score   support

           0       0.94      0.92      0.93        98
           1       0.92      0.94      0.93       102

    accuracy                           0.93       200
   macro avg       0.93      0.93      0.93       200
weighted avg       0.93      0.93      0.93       200
```

4) Conclusion

Machine learning and data mining techniques are valuable in disease diagnosis. The capability to predict diabetes early, assumes a vital role for the patient's appropriate treatment procedure. In this report, a few existing classification methods for medical diagnosis of diabetes patients have been discussed on the basis of accuracy and recall(f1 score). Four machine learning techniques were applied on the Pima Indians diabetes dataset, as well as trained and validated against a test dataset. The results of our model implementations have shown that XGB and Random Forest classifiers outperforms the other two models. The limitation is that a structured dataset has been selected but in the future, unstructured data will also be considered, and these methods will be applied to other medical domains for prediction, such as for different types of cancer, psoriasis, and Parkinson's disease. Other attributes including physical inactivity, family history of diabetes, and smoking habit, are also planned to be considered in the future for the diagnosis of diabetes.