EXPOSYS DATA LABS

(Analysis On Diabetes Data Set)

Data Science Intern

BY

KOLLURU SAI VENKATA SAILESH

Abstract

This Project is about Diabetes patients. The abstract of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

The Final target from the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Table Of Content

INTRODUCTION

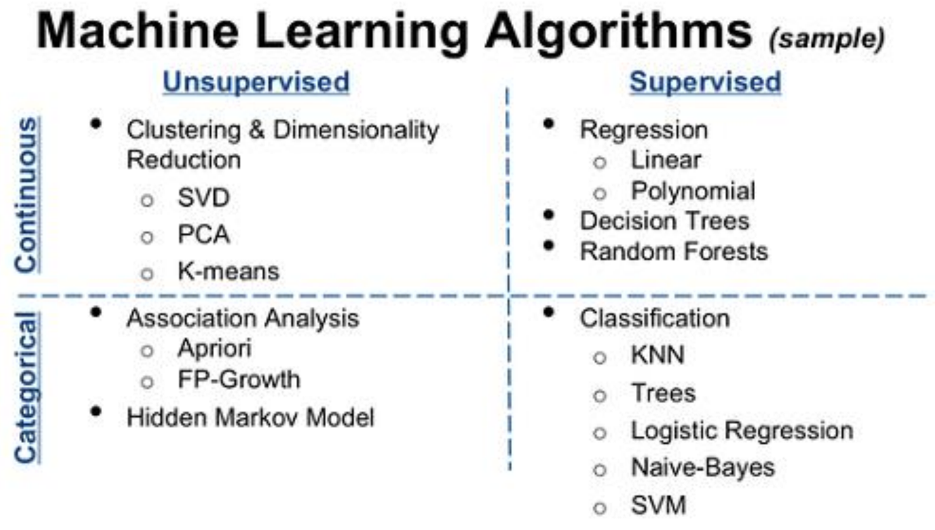
Diabetes is a disease that occurs when your blood glucose, also called blood sugar, is too high. Blood glucose is your main source of energy and comes from the food you eat. Insulin, a hormone made by the pancreas, helps glucose from food get into your cells to be used for energy. Sometimes your body doesn’t make enough — or any — insulin or doesn’t use insulin well. Glucose then stays in your blood and doesn’t reach your cells.

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.

At Present according to WHO about 422 million people worldwide have diabetes. Diabetes affects a large population across the globe and the collection of these datasets is a continuous process and it comprises various patient related attributes such as age, gender, symptoms, insulin levels, blood pressure, blood glucose levels, weight etc. We are working on the Pima Indians Diabetes Dataset (PIDD), extracted from the University of California, Irvine (UCI) machine learning repository.

EXISTING METHODS

In Machine Learning there are numerous algorithms which are classified into three categories: Supervised learning, Unsupervised learning, Semi-supervised learning.



The Supervised Learning/Predictive Models

Supervised learning algorithms are used to construct predictive models. A predictive model predicts missing values using other values present in the dataset. Supervised learning algorithm has a set of input data and also a set of output, and builds a model to make realistic predictions for the response to the new dataset. Supervised learning includes Decision Tree, Bayesian Method, Artificial Neural Network, Instance based learning, Ensemble Method. These are booming techniques in Machine learning.

Unsupervised Learning / Descriptive Models

Descriptive models are developed using unsupervised learning methods. In this model we have a known set of inputs but output is unknown. Unsupervised learning is mostly used on transactional data. This method includes clustering algorithms like k-Means clustering and k-Medians clustering.

Semi-supervised Learning

Semi Supervised learning method uses both labeled and unlabeled data on training dataset. Classification, Regression techniques come under Semi Supervised

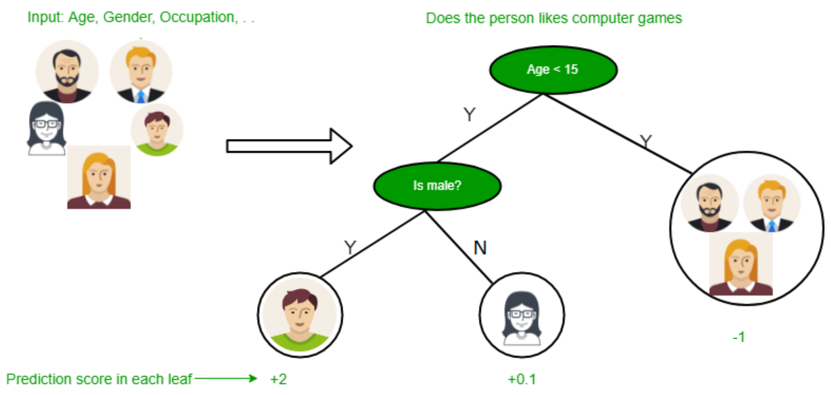
Learning. Logistic Regression, Linear Regression are examples of regression techniques.

PROPOSED METHOD WITH ARCHITECTURE

For this data set we need to focus on Supervised Learning that too Classification techniques because the data set has a target column which is Outcome that is discrete in nature. So we need to consider Classification techniques.

So, I will propose Classification Techniques like Logistic Regression, Decision Tree, Random Forest, Naive Bayes Classification, Ada Boosting, Gradient Boosting Classifier.

For Clear understanding of Classification understand the below fig.


Input: Age, Gender, Occupation, . .
Does the person likes computer games
Age < 15
Y
Y
Is male?
Y
N
-1
Prediction score in each leaf
+2
+0.1

In this figure Output is in the form of yes or no. If we use Classification algorithms we will get the best accuracy compared to regression. So, If there is a discrete column as the target column then I will propose Classification techniques is the best option. As our data set is about predicting whether a person has Diabetes or not so it is a discrete column then we can use Classification algorithms to predict whether a person has diabetes or not. By using classification techniques we can predict outcomes with highest accuracy compared to regression.

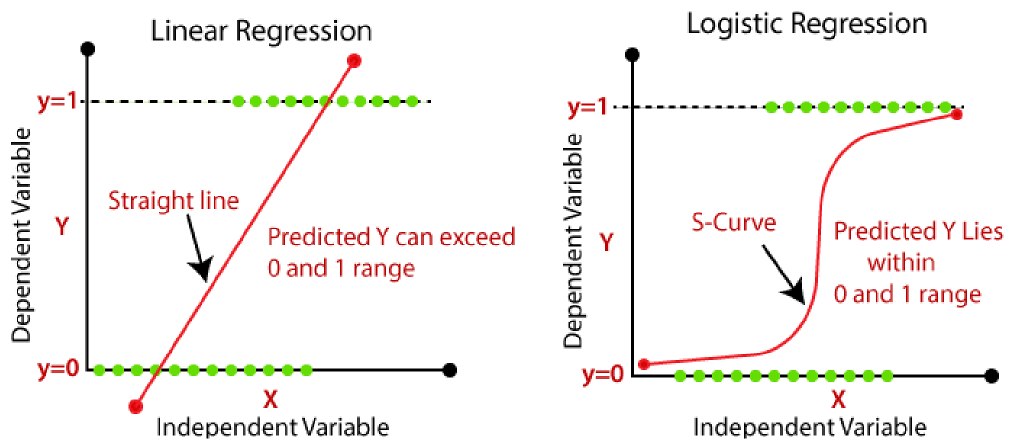
METHODOLOGY

Logistic Regression:

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes. In simple words, the dependent variable is binary in nature having data coded as either 1 (success/yes) or 0 (failure/no).

Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

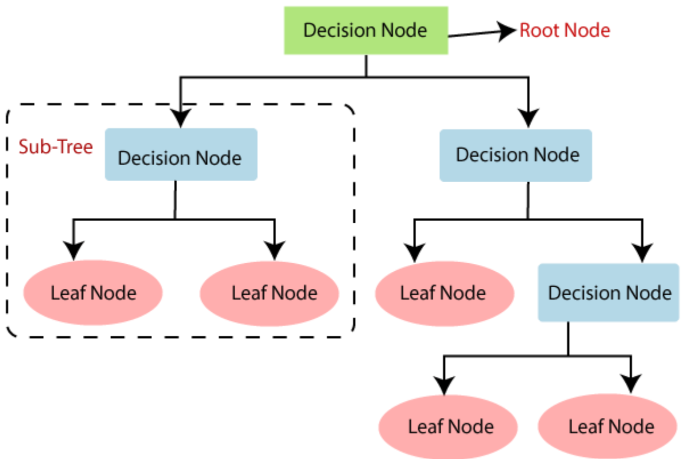
Difference between Linear Regression and Logistic Regression is as below:



Decision Tree:

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

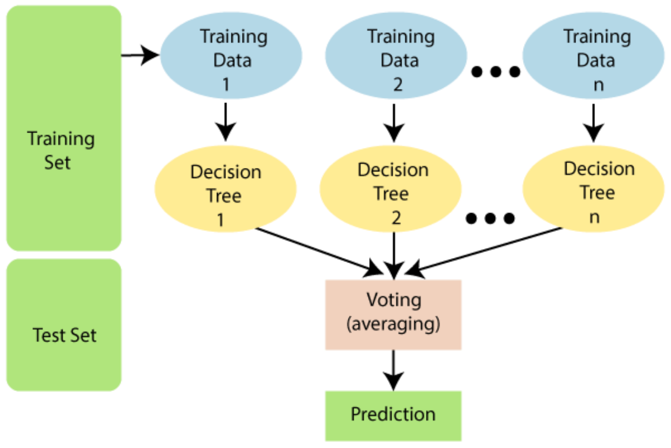


Random Forest:

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

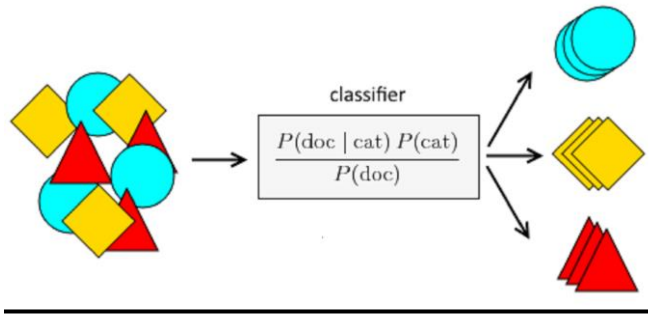
As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.



Naive Bayesian:

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.


classifier
$\frac{P(\text{doc} \mid \text{cat})\, P(\text{cat})}{P(\text{doc})}$

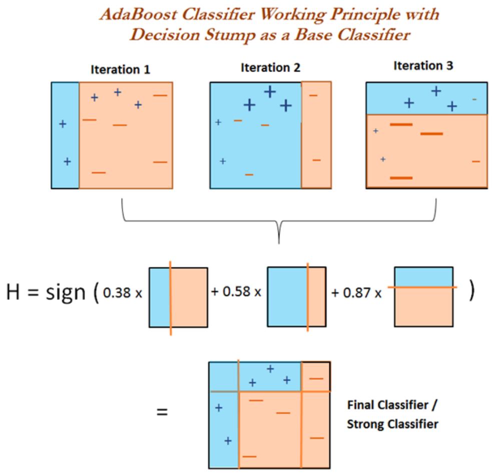
Above figure shows how Naive Bayes works.

Ada Boosting:

Ada Boost, short for "Adaptive Boosting". It focuses on classification problems and aims to convert a set of weak classifiers into a strong one. The final equation for classification can be represented as

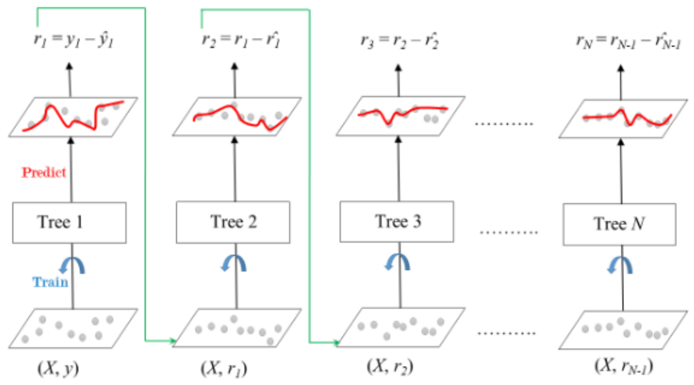
$$F(x) = sign(\sum_{m=1}^{M} \theta_m f_m(x)),$$

where f_m stands for the m_th weak classifier and theta_m is the corresponding weight. It is exactly the weighted combination of M weak classifiers. The whole procedure of the AdaBoost algorithm can be summarized as follows Fig.


AdaBoost Classifier Working Principle with Decision Stump as a Base Classifier
Iteration 1
Iteration 2
Iteration 3
H = sign (0.38 x
+ 0.58 x
+ 0.87 x
)
=
Final Classifier / Strong Classifier

Gradient Boosting:

Gradient boosting is a greedy algorithm and can overfit a training dataset quickly. It can benefit from regularization methods that penalize various parts of the algorithm and generally improve the performance of the algorithm by reducing overfitting. Gradient boosting is a type of machine learning boosting. It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error. If a small change in the prediction for a case causes a large drop in error, then the next target outcome of the case is a high value. Predictions from the new model that are close to its targets will reduce the error. If a small change in the prediction for a case causes no change in error, then the next target outcome of the case is zero. Changing this prediction does not decrease the error. Following fig will give complete clarity about Gradient Boosting.



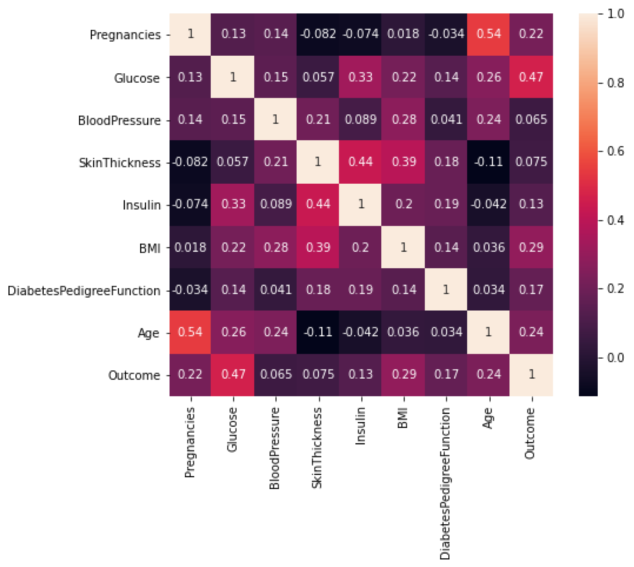
IMPLEMENTATION

As we applied different algorithms to the data set. But before applying those algorithms we need to understand the dataset and also need to Visualize the data set. Before doing that I need to do Preprocessing. For that I am checking whether there is null value or not. There are no null values in the dataset. Now according to attributes we need to check whether the 0 values make sense for the dataset or not. So this dataset has zero values for Glucose, BP, Skin Thickness, Insulin, BMI which makes no sense of having zero values for these attributes so I replaced it with the mean of that column.

Also, she needs to describe the dataset using .describe() method **DataFrame.describe()** method generates descriptive statistics that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding Nan values. This method tells us a lot of things about a dataset. One important thing is that the describe () method deals only with numeric values. It doesn't work with any categorical values. So if there are any categorical values in a column the describe () method will ignore it and display summary for the other columns unless parameter include="all" is passed.
Now, let's understand the statistics that are generated by the describe () method:

- count tells us the number of NoN-empty rows in a feature.
- mean tells us the mean value of that feature.
- std tells us the Standard Deviation Value of that feature.
- min tells us the minimum value of that feature.
- 25%, 50%, and 75% are the percentile/quartile of each feature. This quartile information helps us to detect Outliers.
- max tells us the maximum value of that feature.

After that we need to learn the correlation between every two variables. For that I represented it in visualization using the seaborn package and the method is heatmap.



A positive correlation is a relationship between two variables in which both variables move in the same direction. Negative correlation is a relationship between two variables in which one variable increases as the other decreases, and vice versa.

Then we need to do Scaling to make sure that dataset is in the same range. For this we have options like Standard Scaling, Min-max Scaling etc I choose Standard Scaling for that we need to follow the below Formula.

$$z = \frac{x_i - \mu}{\sigma}$$

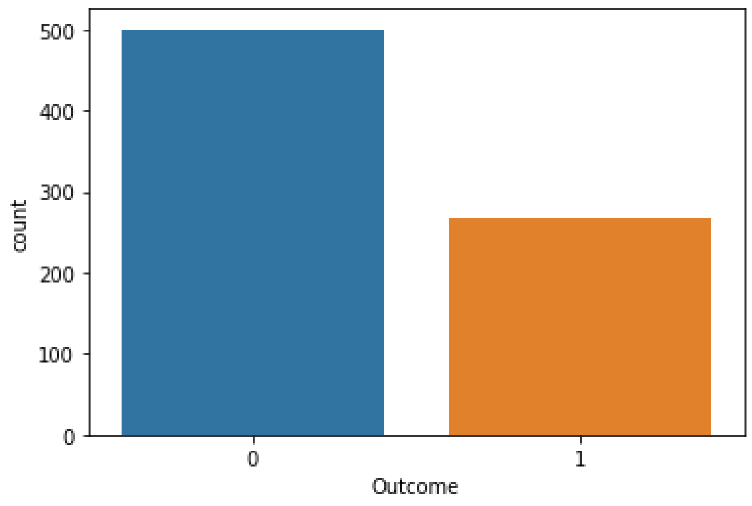
where:

r=Correlation coefficient
X =Average of observations of variable X
Y =Average of observations of variable Y

Data Z is rescaled such that μ = 0 and □ = 1, and is done through this formula.

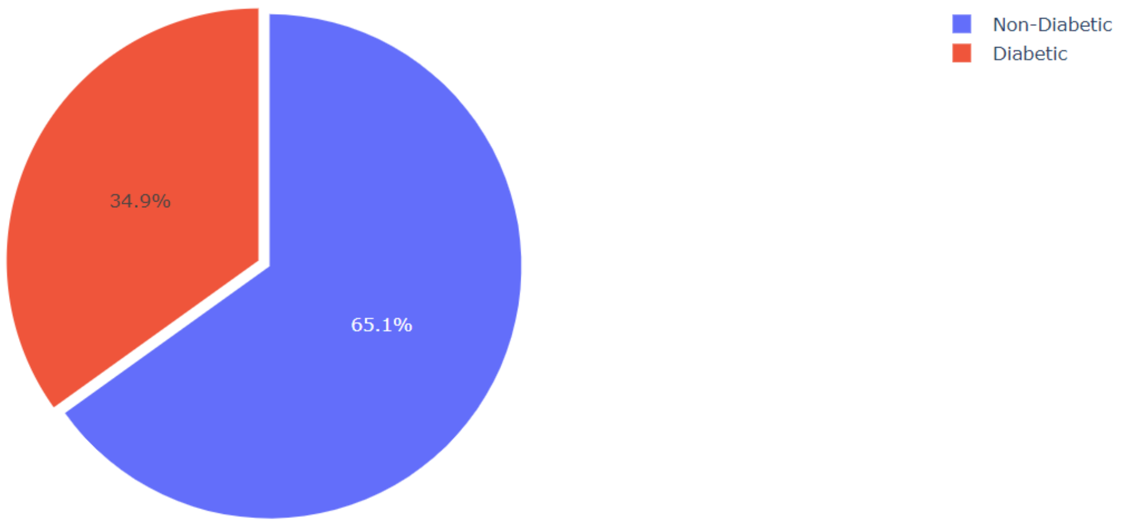
Then we need to do some Graphical Representation of positive and Negative outcomes from the dataset.

Histograms and pieChart of outcome attribute is as follows:



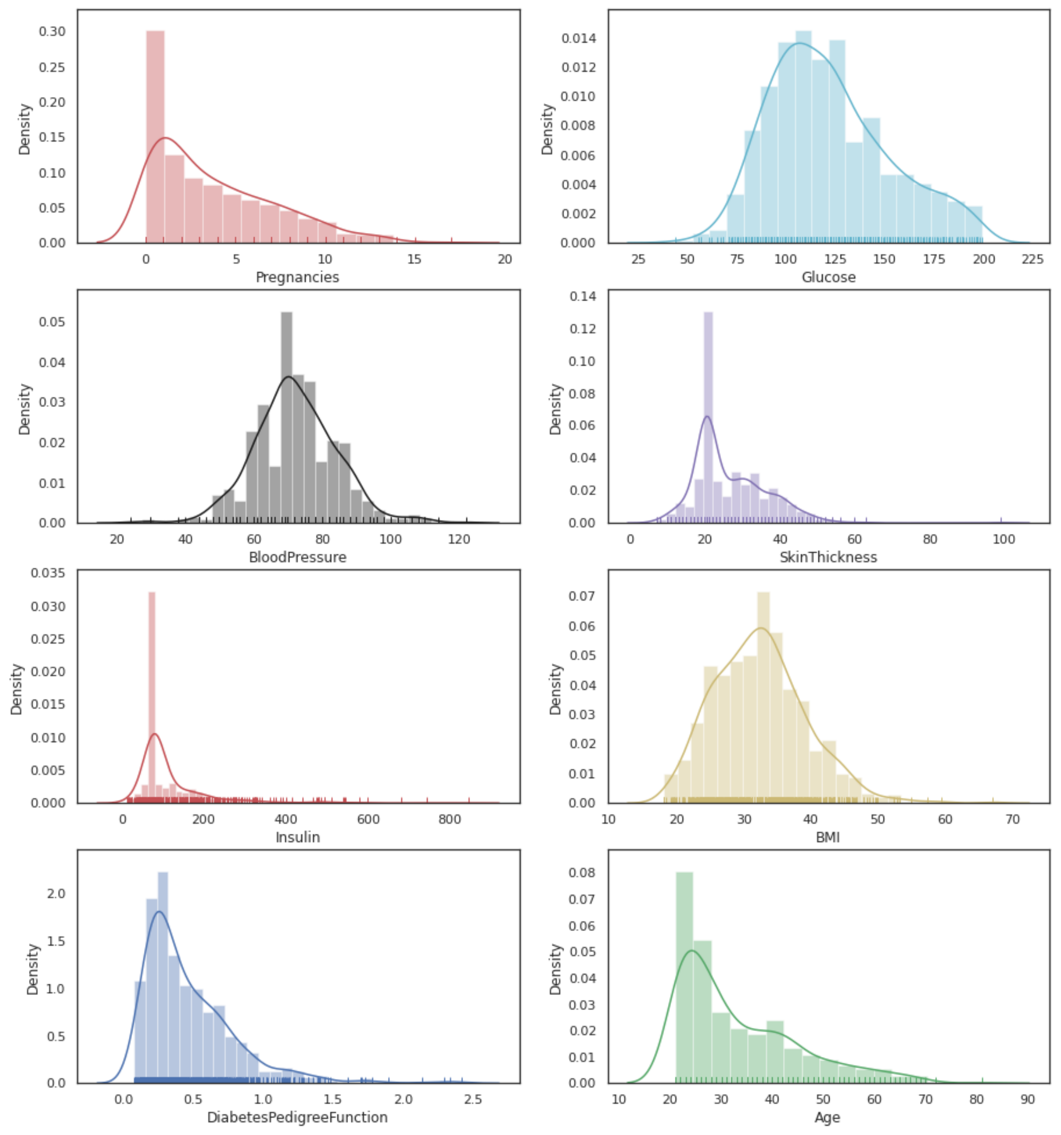
0→ Positive

1→ Negative

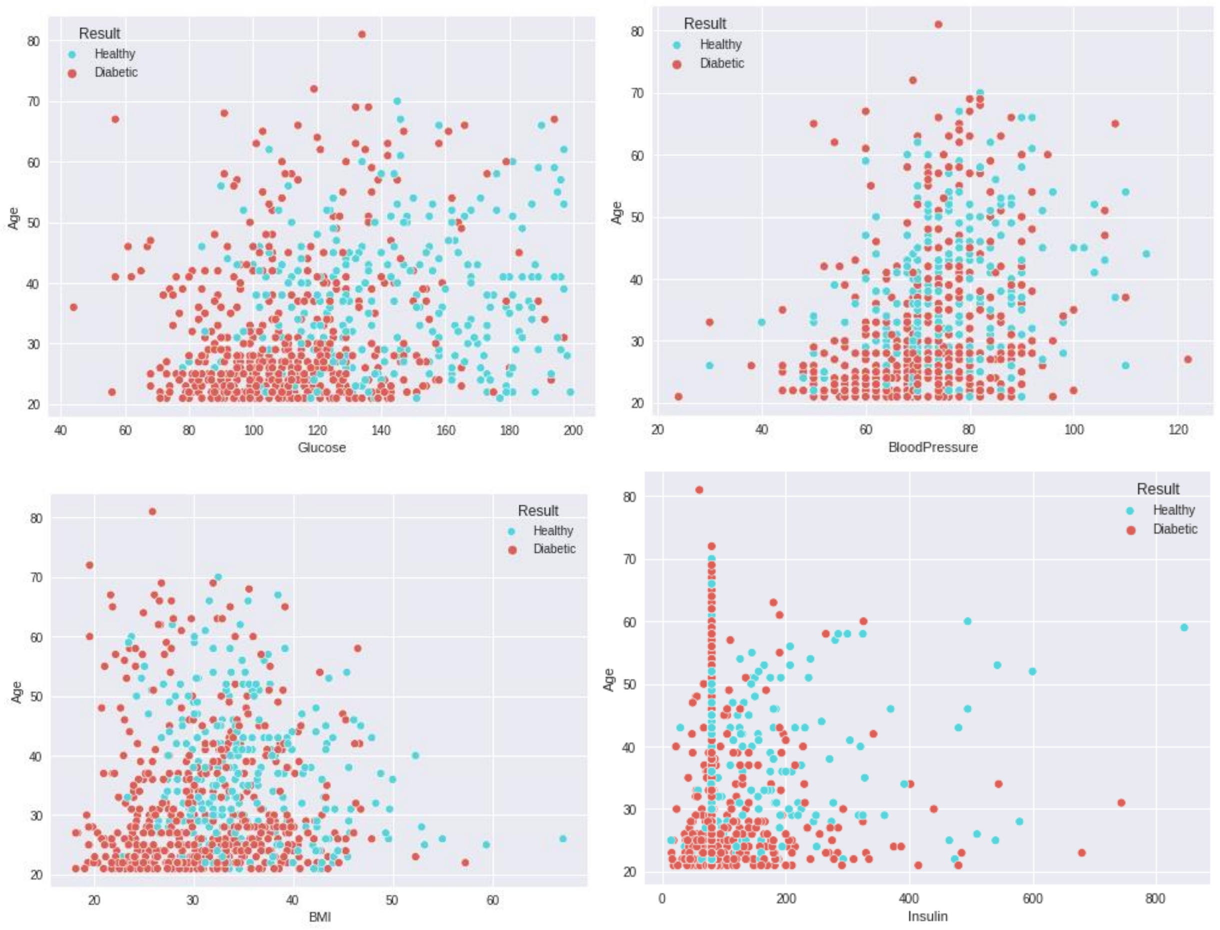


Following graph represents the bar chart of every attribute with respective

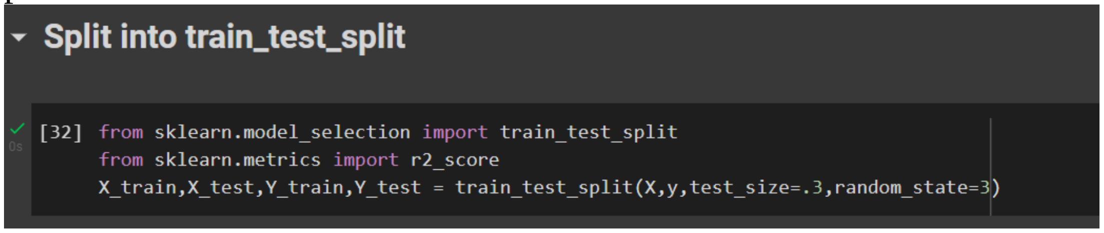
Density.



Following Graph represents Different types of scatter plots which make better understanding about the data set.



Train Test Split : To have unknown data points to test the data rather than testing with the same points with which the model was trained. This helps capture the model performance much better.

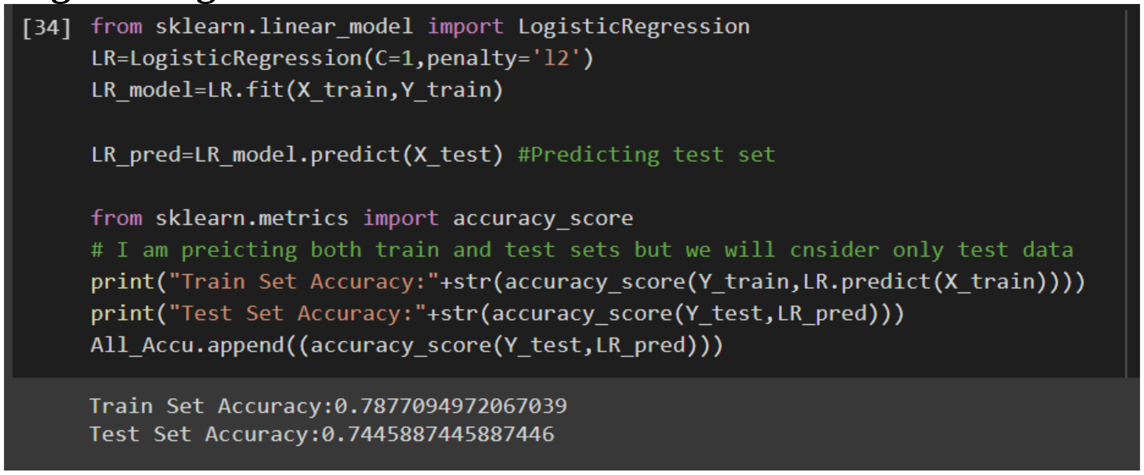


▾ Split into train_test_split

```
[32] from sklearn.model_selection import train_test_split
     from sklearn.metrics import r2_score
     X_train,X_test,Y_train,Y_test = train_test_split(X,y,test_size=.3,random_state=3)
```

I done with six different Techniques with different accuracy values as mentioned below:

Logistic Regression:



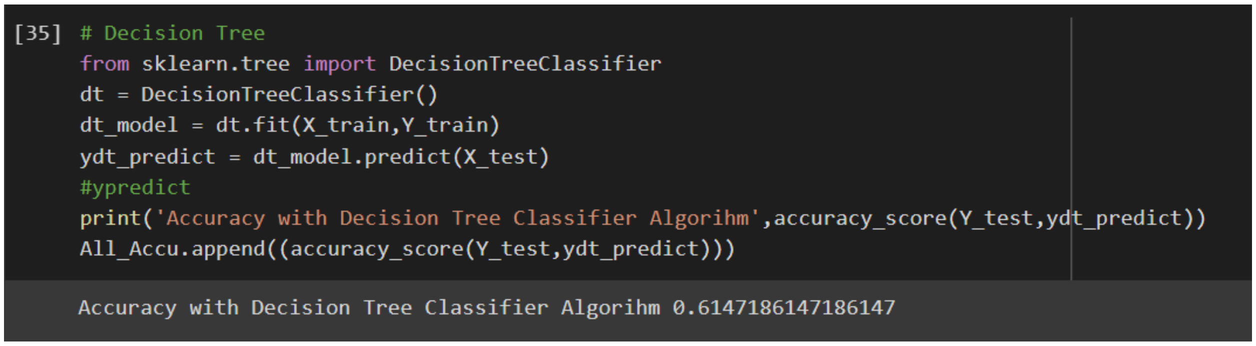
```
[34] from sklearn.linear_model import LogisticRegression
     LR=LogisticRegression(C=1,penalty='l2')
     LR_model=LR.fit(X_train,Y_train)

     LR_pred=LR_model.predict(X_test) #Predicting test set

     from sklearn.metrics import accuracy_score
     # I am preicting both train and test sets but we will cnsider only test data
     print("Train Set Accuracy:"+str(accuracy_score(Y_train,LR.predict(X_train))))
     print("Test Set Accuracy:"+str(accuracy_score(Y_test,LR_pred)))
     All_Accu.append((accuracy_score(Y_test,LR_pred)))
```

```
Train Set Accuracy:0.7877094972067039
Test Set Accuracy:0.7445887445887446
```

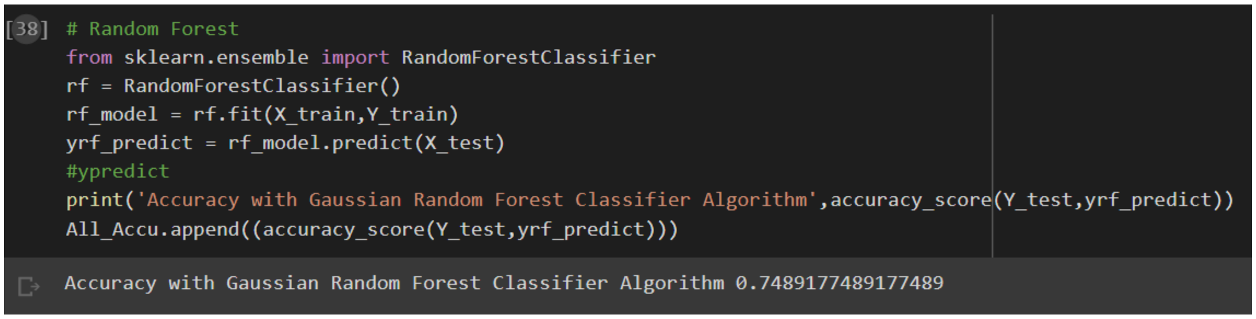
Decision Tree:



```
[35] # Decision Tree
     from sklearn.tree import DecisionTreeClassifier
     dt = DecisionTreeClassifier()
     dt_model = dt.fit(X_train,Y_train)
     ydt_predict = dt_model.predict(X_test)
     #ypredict
     print('Accuracy with Decision Tree Classifier Algorihm',accuracy_score(Y_test,ydt_predict))
     All_Accu.append((accuracy_score(Y_test,ydt_predict)))
```

```
Accuracy with Decision Tree Classifier Algorihm 0.6147186147186147
```

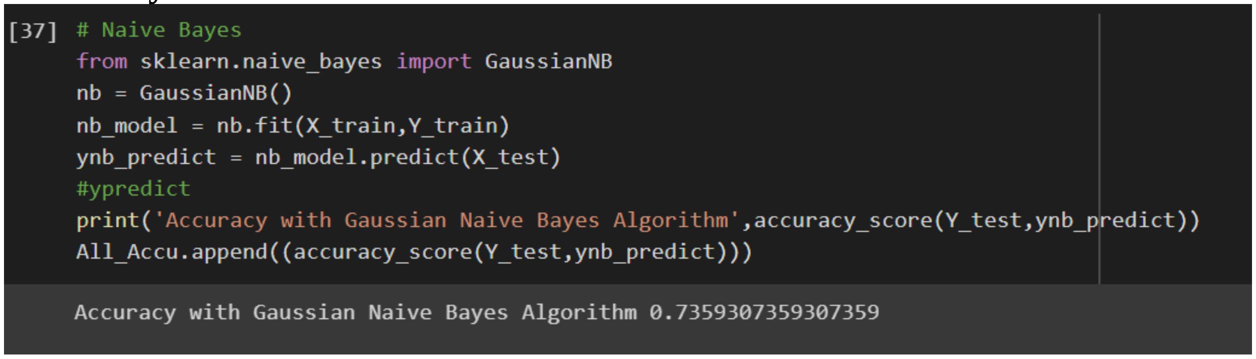
Random Forest:



```
[38] # Random Forest
     from sklearn.ensemble import RandomForestClassifier
     rf = RandomForestClassifier()
     rf_model = rf.fit(X_train,Y_train)
     yrf_predict = rf_model.predict(X_test)
     #ypredict
     print('Accuracy with Gaussian Random Forest Classifier Algorithm',accuracy_score(Y_test,yrf_predict))
     All_Accu.append((accuracy_score(Y_test,yrf_predict)))
```

```
Accuracy with Gaussian Random Forest Classifier Algorithm 0.7489177489177489
```

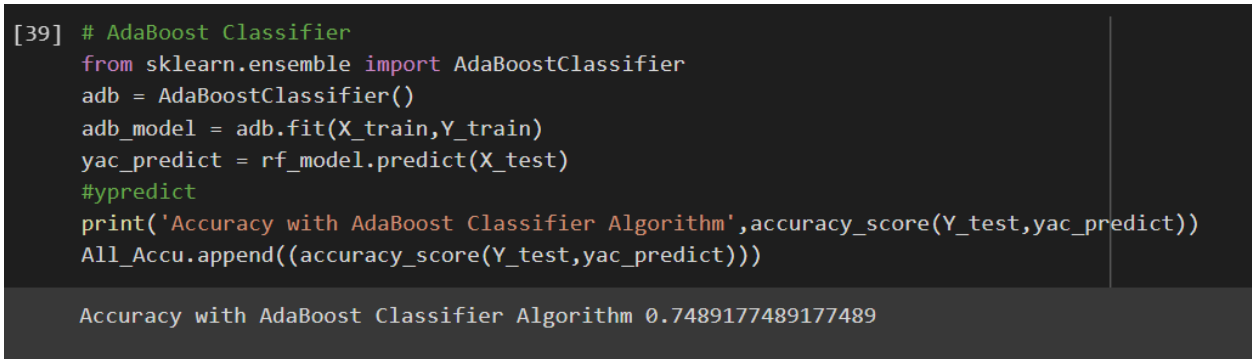
Naive Bayesian:



```
[37] # Naive Bayes
     from sklearn.naive_bayes import GaussianNB
     nb = GaussianNB()
     nb_model = nb.fit(X_train,Y_train)
     ynb_predict = nb_model.predict(X_test)
     #ypredict
     print('Accuracy with Gaussian Naive Bayes Algorithm',accuracy_score(Y_test,ynb_predict))
     All_Accu.append((accuracy_score(Y_test,ynb_predict)))
```

```
Accuracy with Gaussian Naive Bayes Algorithm 0.7359307359307359
```

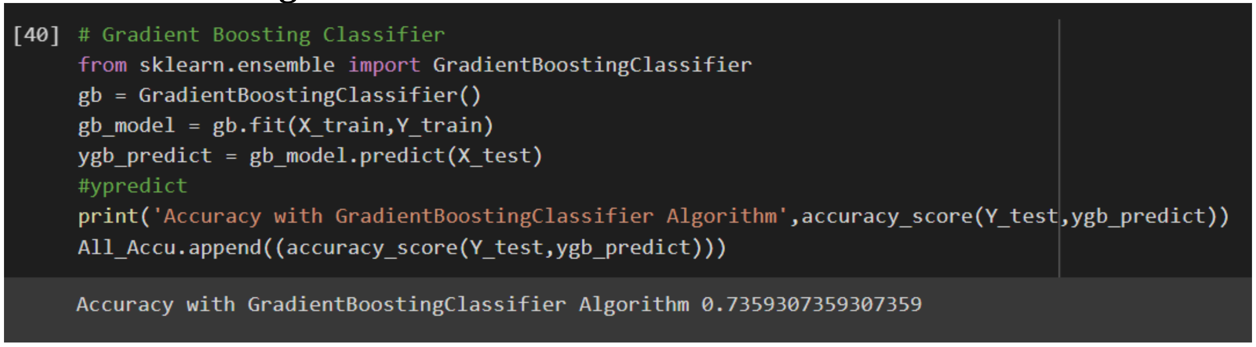
AdaBoosting Classifier:



```
[39] # AdaBoost Classifier
     from sklearn.ensemble import AdaBoostClassifier
     adb = AdaBoostClassifier()
     adb_model = adb.fit(X_train,Y_train)
     yac_predict = rf_model.predict(X_test)
     #ypredict
     print('Accuracy with AdaBoost Classifier Algorithm',accuracy_score(Y_test,yac_predict))
     All_Accu.append((accuracy_score(Y_test,yac_predict)))

     Accuracy with AdaBoost Classifier Algorithm 0.7489177489177489
```

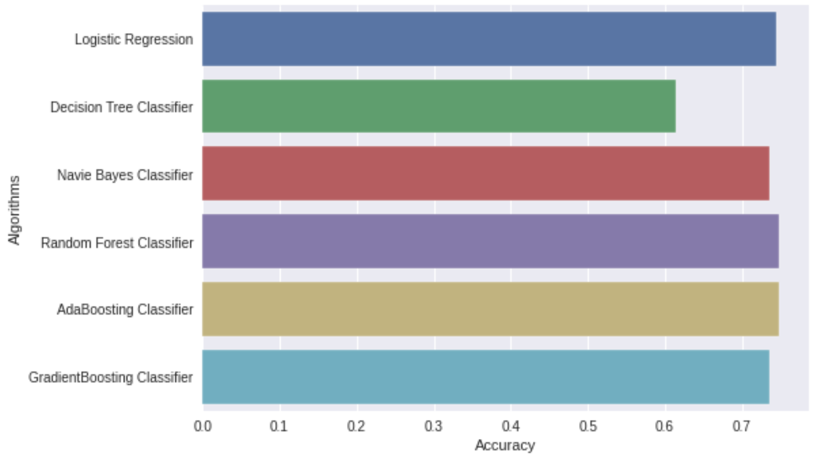
Gradient Boosting Classifier:



```
[40] # Gradient Boosting Classifier
     from sklearn.ensemble import GradientBoostingClassifier
     gb = GradientBoostingClassifier()
     gb_model = gb.fit(X_train,Y_train)
     ygb_predict = gb_model.predict(X_test)
     #ypredict
     print('Accuracy with GradientBoostingClassifier Algorithm',accuracy_score(Y_test,ygb_predict))
     All_Accu.append((accuracy_score(Y_test,ygb_predict)))

     Accuracy with GradientBoostingClassifier Algorithm 0.7359307359307359
```

Graphical representation of all techniques in a single bar plot as follows:



<u>**CONCLUSION**</u>

We can conclude that Random Forest and Logistic Regression has the highest accuracy because it has the highest Accuracy of 74%. As we run multiple times there is a change in accuracy because of getting different values each time so most of the time Logistic Regression and Random Forest comes as highest accuracy.

One reason why Random Forest Classifier showed an improved performance was because of the presence of outliers. As mentioned before, since Random Forest is not a distance based algorithm, it is not much affected by outliers, whereas distance based algorithms such as Decision tree, Naive Bayes, Logistic Regression showed a lower performance.

Based on the feature importance:

- Glucose is the most important factor in determining the onset of diabetes followed by BMI and Age.
- Other factors such as Diabetes Pedigree Function, Pregnancies, Blood Pressure, Skin Thickness and Insulin also contribute to the prediction.

As we can see, the results derived from Feature Importance makes sense as one of the first things that actually is monitored in high-risk patients is the Glucose level. An increased BMI might also indicate a risk of developing Type II Diabetes. Normally, especially in case of Type II Diabetes, there is a high risk of developing as the age of a person increases (given other factors).