**PROJECT REPORT**

**ON**

**DIABETES PATIENT CLASSIFICATION**

Submitted as a part of training and internship program

by

**DIGINIQUE TECHLABS**

(an IIT ROORKEE venture)

in

Data Analytics, Machine Learning & AI using Python



Submitted by:

Ansul Sah

B.Tech (Information Technology)

(2018-2022)

KIIT Deemed to be University, Bhubaneswar

Submitted to :

Mr. Bipul Shahi

(Instructor & project-mentor)

Diginique Techlabs

IIT Roorkee, Uttrakhand

Acknowledgement

I’m thankful to my instructor cum project mentor for assigning me the project on diabetic patient classification using Machine Learning in Python. This project led me to browse several forums, read docs, experimenting and testing various estimators(algorithms) and achieve the desired output. Thus, it helped me to enrich and nurture my knowledge in the field of Machine Learning.

I’m also thankful to Diginique Techlabs, IIT Roorkee for giving me an opportunity to complete my internship in their organisation. Without it’s existence, it would’nt have been possible.

Abstract

In this project, we were asked to work on a real world dataset, and to explore how machine learning algorithms can be used to find the patterns in data. We were expected to gain experience using a common data-mining and machine learning library, skcikit-learn, and were asked to submit a report about the dataset and the algorithms used. After performing the required tasks on a dataset of my choice, herein lies my final report.

**Keywords**: ML, diabetes, estimators, Classification, Supervised learning, AI, matplotlib, scikit-learn, pandas, numpy, seaborn,

The source code of this project can be found on my Github repository:

Introduction

**Machine learning** is a sub-domain of computer science which evolved from the study of pattern recognition in data, and also from the computational learning theory in artificial intelligence. It is the first-class ticket to most interesting careers in data analytics today. As data sources proliferate along with the computing power to process them, going straight to the data is one of the most straightforward ways to quickly gain insights and make predictions. **Machine Learning** can be thought of as the study of a list of sub-problems, viz: decision making, clustering, classification, forecasting, deep-learning, inductive logic programming, support vector machines, reinforcement learning, similarity and metric learning, genetic algorithms, sparse dictionary learning, etc. Supervised learning, or classification is the machine learning task of inferring a function from a labelled data. In Supervised learning, we have a training set, and a test set. The training and test set consists of a set of examples consisting of input and output vectors, and the goal of the supervised learning algorithm is to infer a function that maps the input vector to the output vector with minimal error. In an optimal scenario, a model trained on a set of examples will classify an unseen example in a correct fashion, which requires the model to generalize from the training set in a reasonable way. In layman’s terms, supervised learning can be termed as the process of concept learning, where a brain is exposed to a set of inputs and result vectors and the brain learns the concept that relates said inputs to outputs. A wide array of supervised machine learning algorithms are available to the machine learning enthusiast, for example Neural Networks, Decision Trees, Support Vector Machines, Random Forest, Naïve Bayes Classifier, Bayes Net, Majority Classifier, etc., and they each have their own merits and demerits. There is no single algorithm that works for all cases, as quoted by the No free lunch theorem. In this project, we try and find patterns in a dataset, which is a sample of 768 patients, with 9 parameters, the last one being the target (weather the patient has diabetes).

Problems and Issues in Supervised learning

Before we get started, we must know about how to pick a good machine learning

algorithm for the given dataset. To intelligently pick an algorithm to use for a supervised learning task, we must consider the following factors:

1. Heterogeneity of Data:

Many algorithms like neural networks and support vector machines like their feature vectors to be homogeneous numeric and normalized. The algorithms that employ distance metrics are very sensitive to this, and hence if the data is

heterogeneous, these methods should be the afterthought. Decision Trees can handle heterogeneous data very easily.

2. Redundancy of Data:

If the data contains redundant information, i.e. contain highly correlated values,

then it’s useless to use distance-based methods because of numerical instability. In this case, some sort of Regularization can be employed to the data to prevent this situation.

3. Dependent Features:

If there is some dependence between the feature vectors, then algorithms that monitor complex interactions like Neural Networks and Decision Trees fare better than other algorithms.

4. Bias-Variance Trade-off:

A learning algorithm is biased for a particular input x if, when trained on each of

these data sets, it is systematically incorrect when predicting the correct output for x, whereas a learning algorithm has high variance for a particular input x if it predicts different output values when trained on different training sets. The prediction error of a learned classifier can be related to the sum of bias and variance of the learning algorithm, and neither can be high as they will make the prediction error to be high. A key feature of machine learning algorithms is that they are able to tune the balance between bias and variance automatically, or by manual tuning using bias parameters, and using such algorithms will resolve this situation.

5. Curse of Dimensionality:

If the problem has an input space that has a large number of dimensions, and the problem only depends on a subspace of the input space with small dimensions, the machine learning algorithm can be confused by the huge number of dimensions and hence the variance of the algorithm can be high. In practice, if the data scientist can manually remove irrelevant features from the input data, this is likely to improve the accuracy of the learned function. In addition, there are many algorithms for feature selection that seek to identify the relevant features and discard the irrelevant ones, for instance Principle Component Analysis for unsupervised learning. This reduces the dimensionality.

6. Overfitting:

The programmer should know that there is a possibility that the output values may constitute of an inherent noise which is the result of human or sensor errors. In this case, the algorithm must not attempt to infer the function that exactly matches all the data. Being too careful in fitting the data can cause overfitting, after which the model will answer perfectly for all training examples but will have a very high error for unseen samples. A practical way of preventing this is stopping the learning process prematurely, as well as applying filters to the data in the pre-learning phase to remove noises.

Only after considering all these factors can we pick a supervised learning algorithm that works for the dataset we are working on. For example, if we were working with a dataset consisting of heterogeneous data, then decision trees would fare better than other algorithms. If the input space of the dataset we were working on had 1000 dimensions, then it’s better to first perform PCA on the data before using a supervised learning algorithm on it.

**A brief description on Machine Learning model and the models used**:

* **What is a machine learning model?**
* Machine Learning algorithm is a hypothesis set that is taken at the beginning before the training starts with real-world data. When we say Linear Regression algorithm, it means a set of functions that define similar characteristics as defined by Linear Regression and from those set of functions we will choose one function that fits the most by the training data.

**Different types of machine learning:**

* **Supervised learning:**

[Supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) describes a class of problem that involves using a model to learn a mapping between examples and the target variable.

The model that’s been worked on here is an example of supervised learning.

* **Unsupervised learning**

[Unsupervised learning](https://en.wikipedia.org/wiki/Unsupervised_learning) describes a class of problems that involves using a model to describe or extract relationships in data.

Compared to supervised learning, unsupervised learning operates upon only the input data without outputs or target variables. As such, unsupervised learning does not have a teacher correcting the model, as in the case of supervised learning.

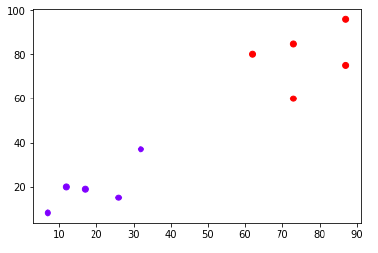
* **Reinforcement learning**

Reinforcement learning is the training of machine learning models to make a sequence of decisions. The agent learns to achieve a goal in an uncertain, potentially complex environment.

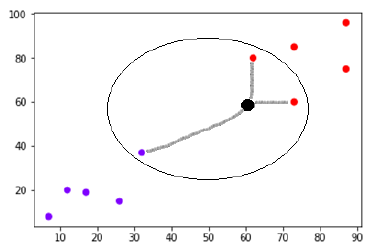
Now, we’ll provide a short description on each of the algorithms used later:

1. **K-neighbors classifier**

Suppose we have a dataset which can be plotted as follows –



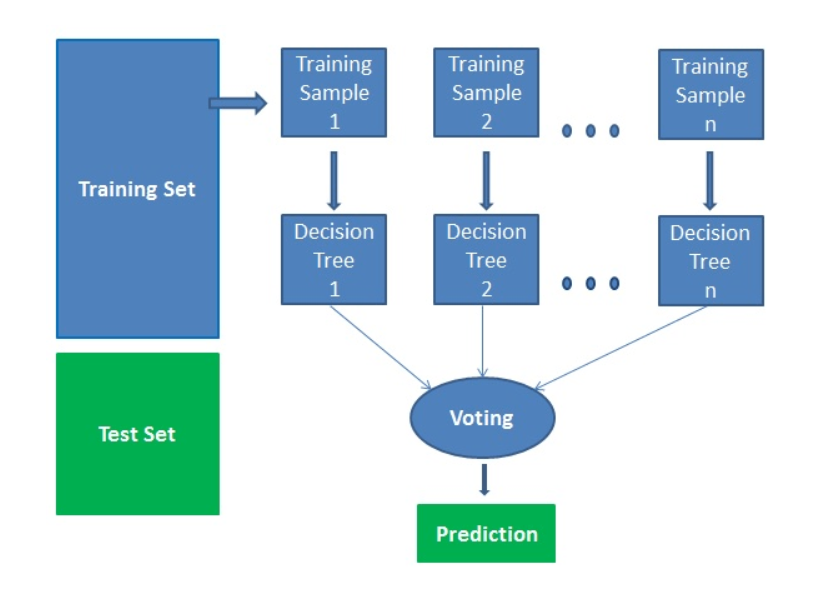
Now, we need to classify new data point with black dot (at point 60,60) into blue or red class. We are assuming K = 3 i.e. it would find three nearest data points. It is shown in the next diagram −



We can see in the above diagram the three nearest neighbours of the data point with black dot. Among those three, two of them lies in Red class hence the black dot will also be assigned in red class.

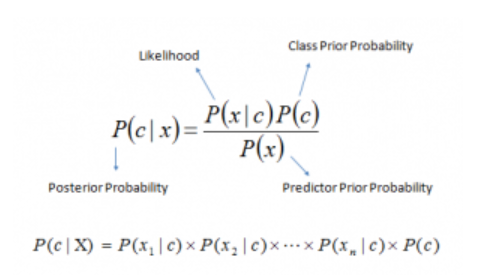
1. **Random forest classifier**

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the ‘max\_samples’ parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

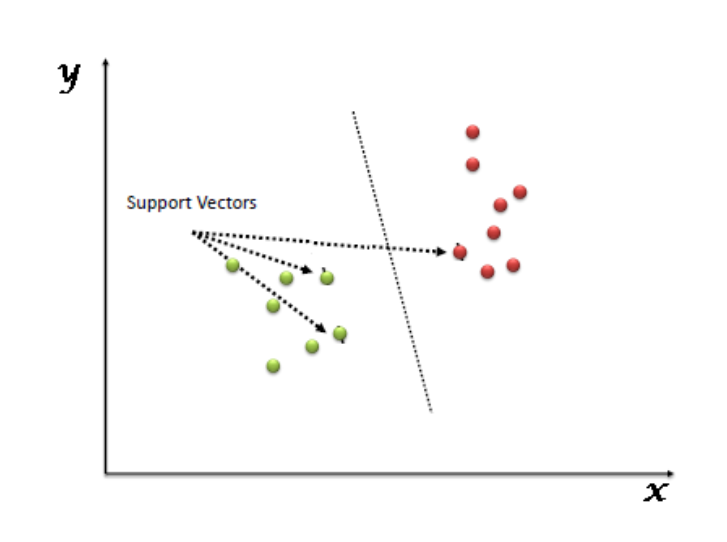
****

1. **Naïve bayes(Gaussian, Bernoulli, Multinomial)**

Naïve Bayes classifiers are a family of simple "[probabilistic classifiers](https://en.wikipedia.org/wiki/Probabilistic_classification)" based on applying [Bayes' theorem](https://en.wikipedia.org/wiki/Bayes%27_theorem) with strong (naïve) [independence](https://en.wikipedia.org/wiki/Statistical_independence) assumptions between the features. They are among the simplest [Bayesian network](https://en.wikipedia.org/wiki/Bayesian_network) models.[[1]](https://en.wikipedia.org/wiki/Naive_Bayes_classifier#cite_note-1) But they could be coupled with [Kernel density estimation](https://en.wikipedia.org/wiki/Kernel_density_estimation) and achieve higher accuracy levels. It works on bayes’ probability theorem:

1. **Support Vector Classifier(SVM)**

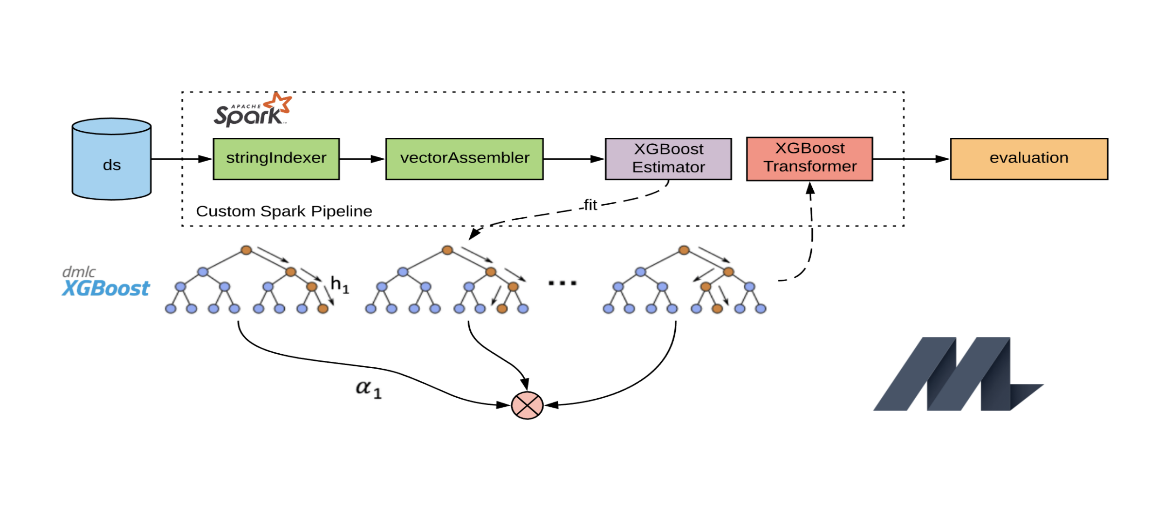
In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well. Support Vectors are simply the co-ordinates of individual observation. The SVM classifier is a frontier which best segregates the two classes (hyper-plane/ line).



1. **Extreme Gradient Boost(XGB)**

XGBoost (Extreme Gradient Booster) 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.

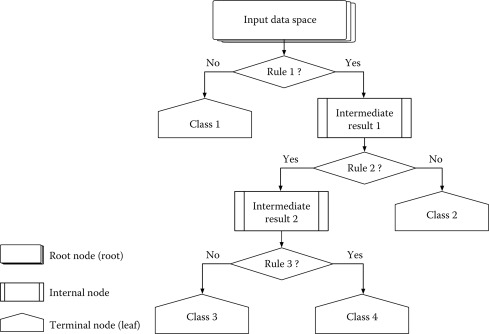
It works on the following principle:

****

1. **Decision tree classifier**

A **decision tree classifier** is a **tree** in which internal nodes are labelled by features. The **classifier** categorizes an object xi by recursively testing for the weights that the features labelling the internal nodes have in vector xi, until a leaf node is reached. The label of this node is then assigned to xi.

The following diagram explains the working of decision tree classifier:

****

*Now, as all the models have been explained briefly, let’s move to the dataset part, which is the core of this field….*

**The dataset**:

This dataset is downloaded from:

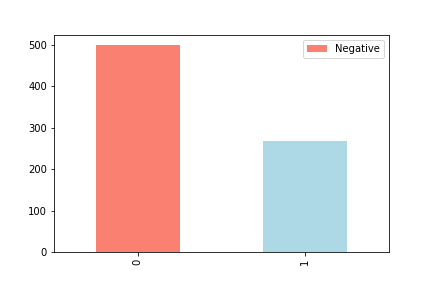
<https://www.kaggle.com/uciml/pima-indians-diabetes-database>

It consists of 9 columns, namely:

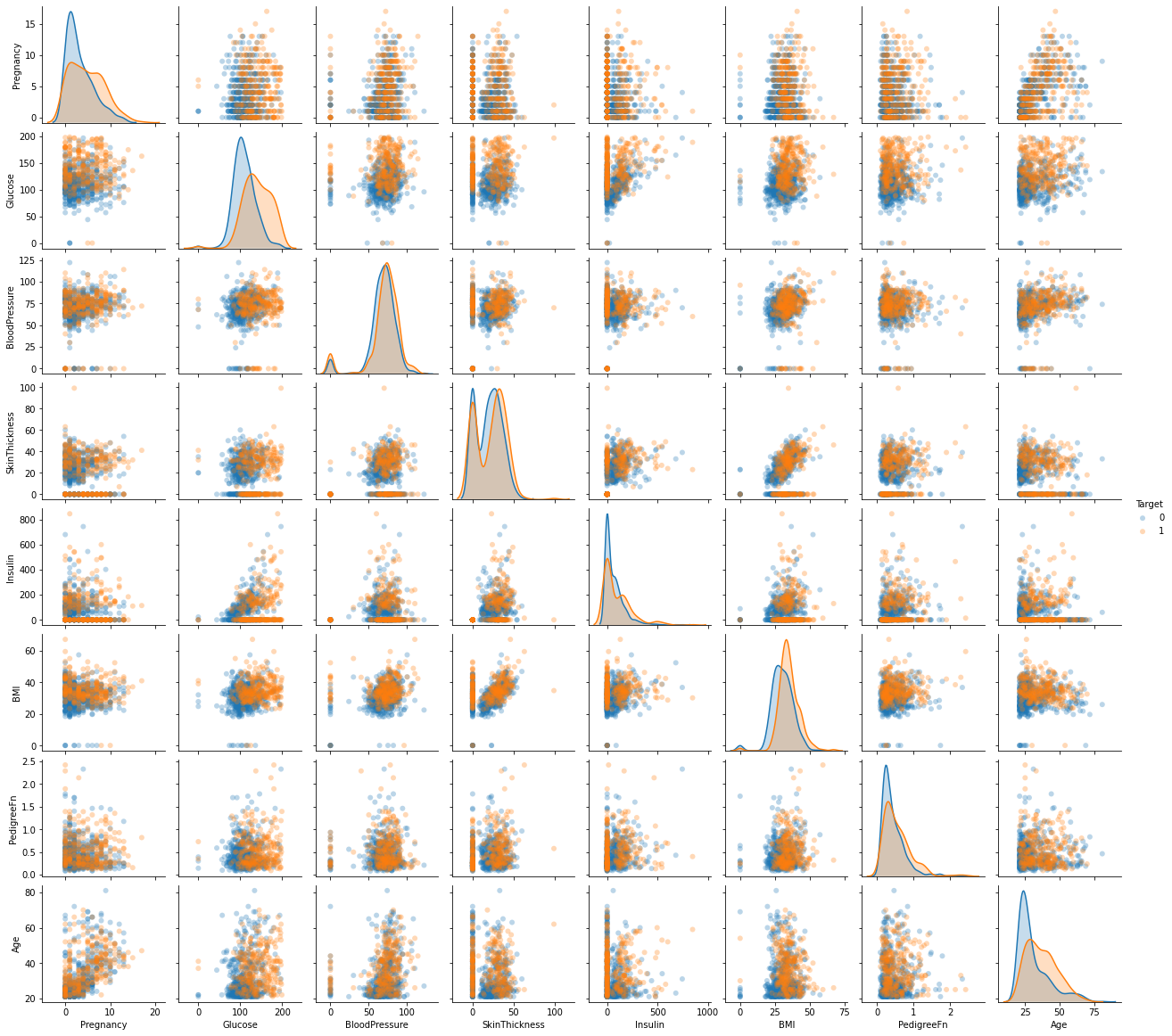
* **Pregnancies**: Number of times pregnant
* **Glucose**: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
* **Blood Pressure**: Diastolic blood pressure (mm Hg)
* **Skin Thickness**: Triceps skin fold thickness (mm)
* **Insulin**:2-Hour serum insulin (mu U/ml)
* **BMI**: Body mass index (weight in kg/(height in m)^2)
* **Age**: Age(years)
* **Diabetes Pedigree Function**: Diabetes pedigree function
* **Pregnancies**: Number of times pregnant
* **Outcome**: Class variable (0 or 1) 268 of 768 are 1, the others are 0

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective 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.

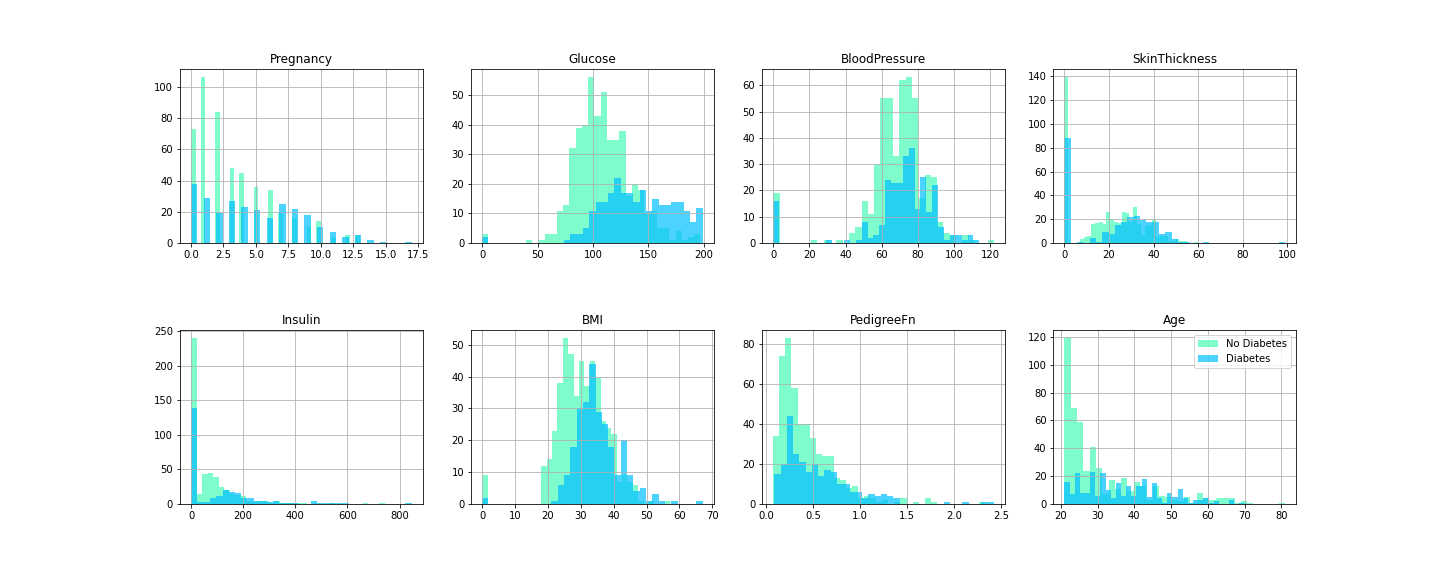
On visualizing the positive v/s negative patients, we infer that 34.83% of total patients are Positive.



Plotting a graph between the features with each other gives us the following results:

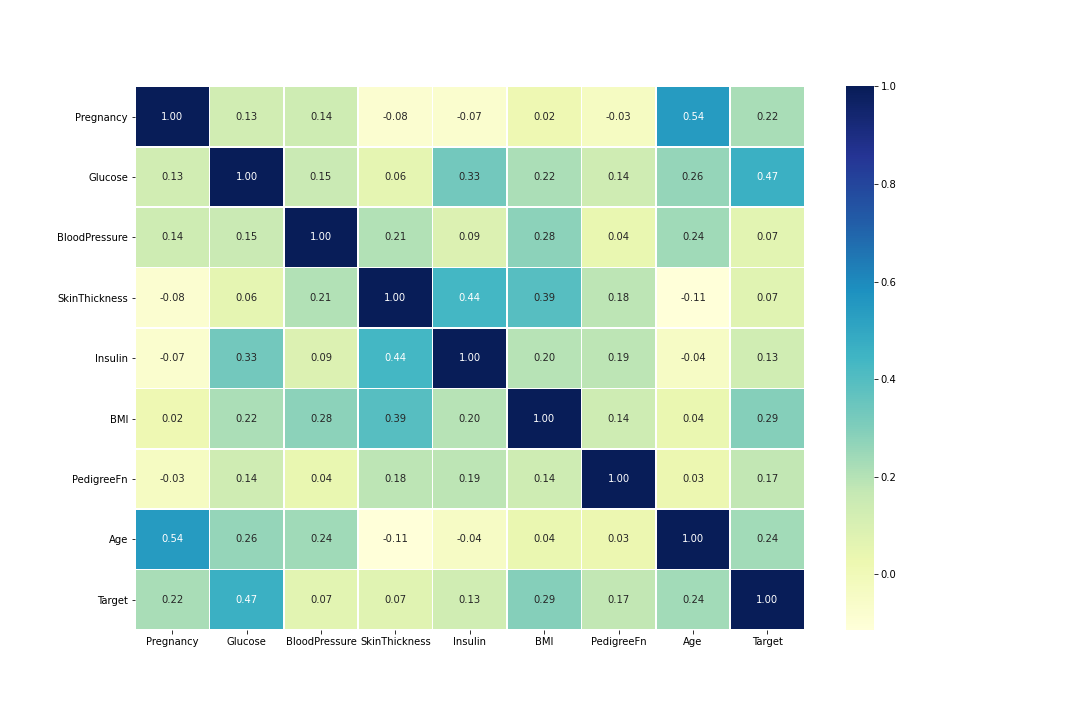


A plot between target and different features looks as below:



**Inference from the above graph**:

* People have 0-1 children are mostly found positive.
* People have Glucose level above 130 are mostly found positive.
* People have blood Pressure above 30 are mostly found positive.
* People have skin thickness of 2mm are mostly found positive.
* People have Insulin levels of 10 are mostly found positive.
* People have BMI above 40 are mostly found positive.
* People have Pedigree function above 40 are mostly found positive.
* People who are above 20 are mostly found positive.

**A correlation matrix between different features**:

**Splitting the data into train and test datasets**:

For training and evaluation purpose, the data has been split into train and test datasets. The training dataset contains 90% of data, while the testing datasets contains 10%.

* Size of total dataset = 768\*8
* Size of Training dataset = 691\*8
* Size of testing dataset = 77\*8

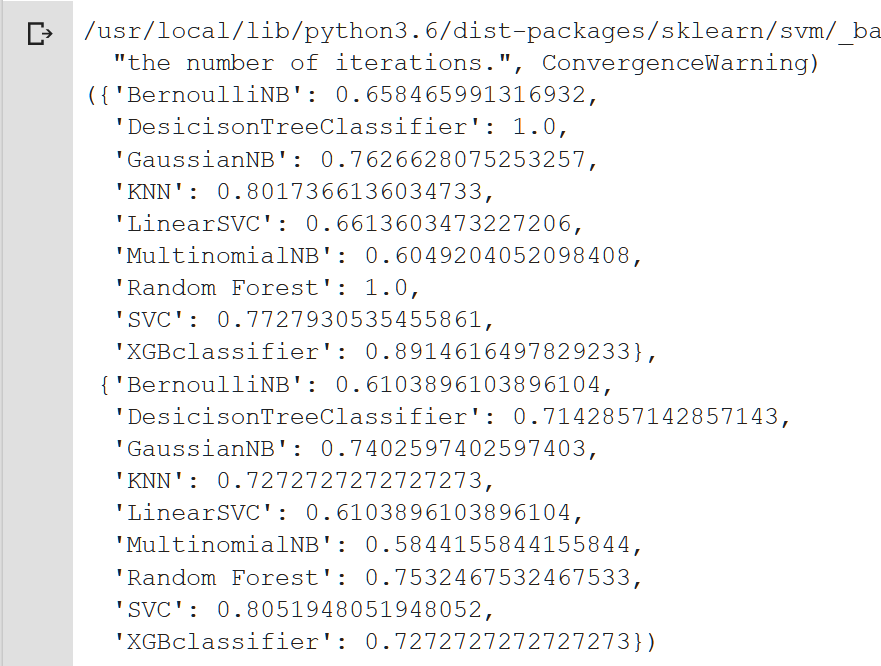
**∴** 768,691,77=rows; 8=columns

The training dataset will now be trained on 9 different estimators(without parameter tuning) to get a rough idea of which estimators is performing good.

The following estimators (ML algorithms) have been used:

* K-Neighbors Classifier
* Random Forest Classifier
* Gaussian naïve bayes
* Bernoulli naïve bayes
* Multinomial naïve bayes
* Linear SVC
* SVC
* XGB Classifier
* Decision Tree Classifier

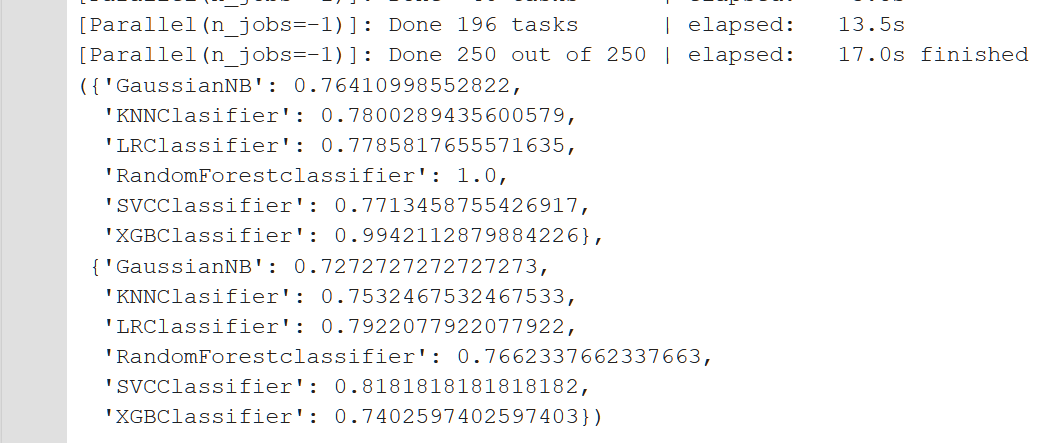
The train and test results obtained by each estimator is as follows:

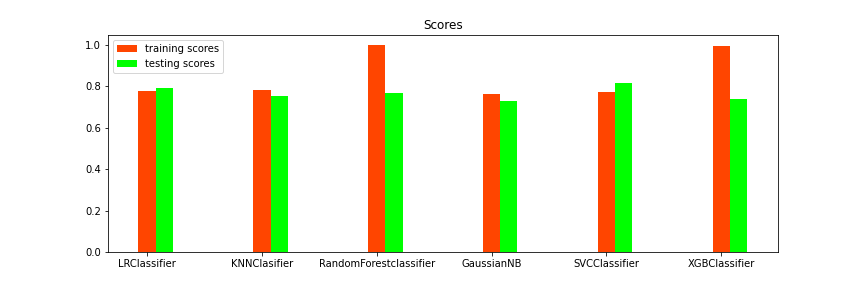


According to the results, the testing score is highest in SVC classifier.

Now, each classifier will be tuned on the basis of it’s parameters to get the best-possible improvements for that model…

(**P.S. : I’ve used RandomizedSearchCV and not GridSearchCV, since GridSearch takes an invariably long amount of time and still does’nt complete).**

****Visualizing the training and testing results on the hyperparameter-tuned models:



* As visible, the SVC classifier has returned the highest accuracy on testing data (**81.81% which is our requirement**).

**Evaluating the model**:

The following evaluation metrics have been used for evaluating the model:

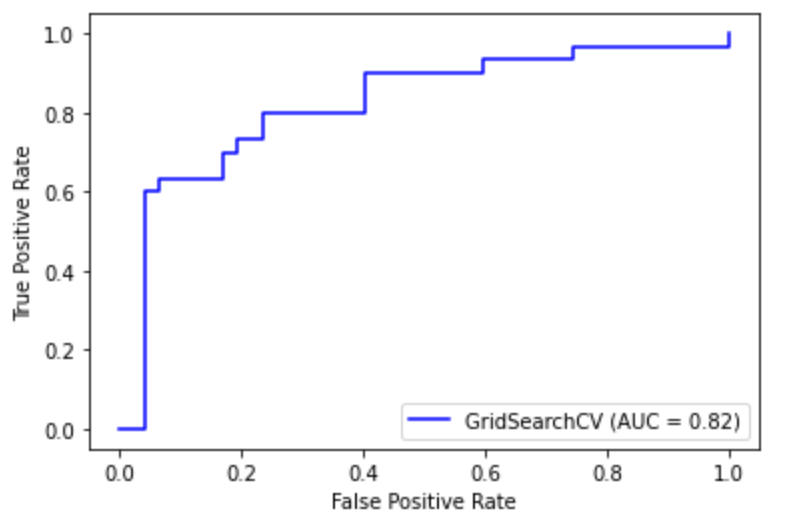
* ROC curve & AUC score
* Confusion matrix
* Classification report
* Cross-validated scores(Accuracy, Precision, Recall, f1)

1. **ROC curve & AUC score:**

A **receiver operating characteristic** curve, or **ROC** curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The **ROC** curve is created by plotting the true positive rate against the false positive rate at various threshold settings.

**AUC** represents the probability that a random positive (green) example is positioned to the right of a random negative (red) example. **AUC** ranges in value from 0 to 1. A model whose predictions are 100% wrong has an **AUC** of 0.0; one whose predictions are 100% correct has an **AUC** of 1.0.

The ROC curve for our metric is as follows:



**The AUC score achieved by the model is 82%.**

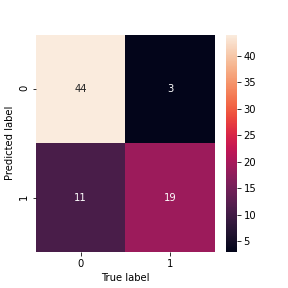
**Confusion matrix:**

A **confusion matrix** is a performance measurement for machine learning classification problem where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values.



The above is a diagram of the confusion matrix.

* True Positive (TP) : Observation is positive, and is predicted to be positive.
* False Negative (FN) : Observation is positive, but is predicted negative.
* True Negative (TN) : Observation is negative, and is predicted to be negative.
* False Positive (FP) : Observation is negative, but is predicted positive.

The confusion matrix for our model is as follows:

**Classification report**:

A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report as shown below.

The report shows the main classification metrics precision, recall and f1-score on a per-class basis. The metrics are calculated by using true and false positives, true and false negatives. Positive and negative in this case are generic names for the predicted classes. There are four ways to check if the predictions are right or wrong :

1. **TN / True Negative:** when a case was negative and predicted negative
2. **TP / True Positive:** when a case was positive and predicted positive
3. **FN / False Negative:** when a case was positive but predicted negative
4. **FP / False Positive:** when a case was negative but predicted positive

**Precision – What percent of your predictions were correct?**

Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class it is defined as the ratio of true positives to the sum of true and false positives.

**TP – True Positives**  
 **FP – False Positives**

Precision – Accuracy of positive predictions.  
*Precision = TP/(TP + FP)*

### ****Recall – What percent of the positive cases did you catch?****

Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives.

**FN – False Negatives**

Recall: Fraction of positives that were correctly identified.  
*Recall = TP/(TP+FN)*

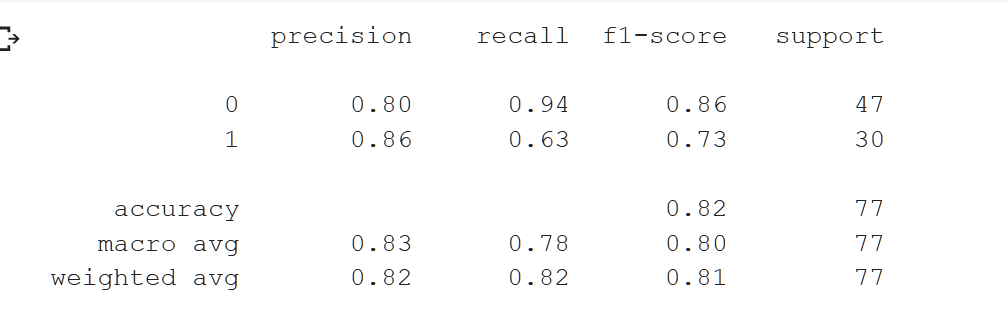
### ****F1 score – What percent of positive predictions were correct?****

The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. Generally speaking, F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy.

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

**We’ll use 2 forms of classification here:**

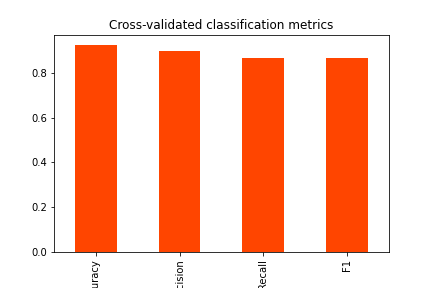
* 1. **Normal evaluation**
  2. **Cross-validated evaluation**

The general classification report for our model is:

The **cross-validated classification report** uses K-fold cross-validation and evaluates the model.

* Accuracy metric score is 92.5%.
* Precision metric score is 90.0%.
* Recall metric score is 86.66666666666666%.
* f1 metric score is 86.66666666666666%.

Visualizing the cross-validated accuracy score:



**ANALYZING THE FEATURE IMPORTANCE:**

**Feature Importance is one of the core concepts in machine learning which hugely impacts the performance of your model.** The data features that you use to train your machine learning models have a huge influence on the performance you can achieve.

Irrelevant or partially relevant features can negatively impact model performance.

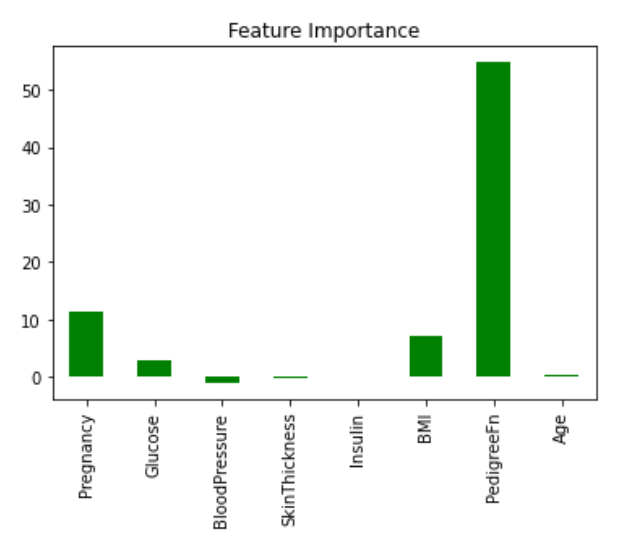
**Feature Importance** tells us how much has each column has contributed the most in achieving the accuracy score. This helps the researchers to work on the parameter that has mostly-likely caused diabetes to a person.

The feature importance has been done using clf.coef\_ where ‘clf’ is the model and ‘.coef\_’ is the coefficient of each feature.

This returns an array of different coefficients.



**VISUALIZATION OF FEATURE IMPORTANCE:**

****

* From the above plot, it can be concluded that PedigreeFunction(52.383250806853674%) is the most important feature in deciding if a person has diabetes, followed by Pregnancy, BMI, and so on …

**PROBLEMS WITH THE MODEL:**

* The model is underfitted (training score < testing data). I’m obliged to use SVC since the minimum required accuracy score is >80%, and multiple attempts of tuning hyperparameters failed to increase the training accuracy (& keeping testing score constant), so this needs to be put up with.

**CONCLUSION:**

The model learned from various parameters from the training data and succeeded in achieving a 81.5% accuracy on testing data. Even on testing the data with random datasets (derived from the main dataset), the model achieved an accuracy of above 70%. This might be because of underfitting of the model (other trained models which had higher training scores, but I didn’t used them because of accuracy requirement).

However, keeping the shortcomings aside, the model successfully predicts with an accuracy of 70% on any random dataset. This is just an initial attempt, and the model will be improved in the future and be made to achieve higher prediction score, as I continue learning and diving deep into the world of Data Science.

**Bibliography and references**:

* <https://www.kaggle.com/uciml/pima-indians-diabetes-database(source> of the dataset)
* <https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/>
* <https://github.com/mrdbourke/zero-to-mastery-ml/blob/master/section-3-structured-data-projects/end-to-end-heart-disease-classification-video.ipynb>
* <https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e>
* <https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>
* <https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.html>
* <https://en.wikipedia.org/wiki/Naive_Bayes_classifier>
* <https://www.sciencedirect.com/topics/computer-science/decision-tree-classifier>
* <https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/>
* <https://muthu.co/understanding-the-classification-report-in-sklearn/>
* <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html>
* <https://matplotlib.org/3.2.1/tutorials/intermediate/legend_guide.html>
* <https://seaborn.pydata.org/generated/seaborn.heatmap.html>
* <https://stackoverflow.com/questions/25009284/how-to-plot-roc-curve-in-python>
* <https://matplotlib.org/2.0.2/api/colors_api.html>
* https://github.com/adam-p/markdown-here/wiki/Markdown-Here-Cheatsheet

-----THANK YOU------