**Report on Mini Project**

**Title**: Comparing the Performance of Different Classifier Algorithms using Diabetic dataset.

**Introduction:** This mini project deals with studying the performance of different Classifier Algorithms namely:

Logistic Regression Decision Tree

Ada-boost KNN

**Requirements:**

For the execution of Program following is required:

* Python Interpreter
* Sklearn library
* Pandas
* Matplotlib
* Diabetic Dataset

## Theory:

**Data Pre-processing:**

1. Zero Removal
2. Outliers Removal

## Classification:

Classification is a data mining technique that assigns categories to a collection of data in order to aid in more accurate prediction. The goal is to create a set of classification rules that will answer a question, make a decision, or predict behavior. To start, a set of training data is developed that contains a certain set of attributes as well as the likely outcome.

The job of the classification algorithm has two-step process such as:

* 1. **Learning Step** : Construction of Classification Model Different Algorithms are used to build a classifier by making the model learn using the training set available. Model has to be trained for prediction of accurate results.
  2. **Classification Step**: Model used to predict class labels and testing the constructed model on test data and hence estimate the accuracy of the classification rules.

## Different Classifiers used in Machine Learning:

1. Logistic Regression
2. Decision Trees
3. Random Forest
4. K-Nearest Neighbour
5. Naïve Bayes Classifiers
6. Support Vector Machines

## Techniques Used: Decision Trees :

Decision Trees are a type of Supervised Machine Learning where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split.

In Decision Tree the major challenge is to identification of the attribute for the root node in each level. This process is known as attribute selection. We have two popular attribute selection measures:

1)Information Gain 2)Gini Index

1. Information Gain

When we use a node in a decision tree to partition the training instances into smaller subsets the entropy changes. Information gain is a measure of this change in entropy.

* + Entropy

Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy more the information content.

1. Gini Index

Gini index measures the degree or probability of a particular variable being wrongly classified when it is randomly chosen. If all the elements belong to a single class, then it can be called pure. The degree of Gini index varies between 0 and 1, where 0 denotes that all elements belong to a certain class or if there exists only one class, and 1 denotes that the elements are randomly distributed across various classes. A Gini Index of 0.5 denotes equally distributed elements into some classes.

## K-nearest neighbor classification

A k-nearest-neighbor algorithm, often abbreviated k-nn, is an approach to data classification that estimates how likely a data point is to be a member of one group or the other depending on what group the data points nearest to it are in.

The k-nearest-neighbor is an example of a "lazy learner" algorithm, meaning that it does not build a model using the training set until a query of the data set is performed.

A k-nearest-neighbor is a data classification algorithm that attempts to determine what group a data point is in by looking at the data points around it.

An algorithm, looking at one point on a grid, trying to determine if a point is in group A or B, looks at the states of the points that are near it. The range is arbitrarily determined, but the point is to take a sample of the data. If the majority of the points are in group A, then it is likely that the data point in question will be A rather than B, and vice versa.

## Adabooster classifier

AdaBoost is one of the first boosting algorithms to be adapted in solving practices. Adaboost helps you **combine multiple “weak classifiers” into a single “strong classifier”**. Here are some (fun) facts about Adaboost!

The weak learners in AdaBoost are decision trees with a single split, called decision stumps. AdaBoost works by putting more weight on difficult to classify instances and less on those already handled well. AdaBoost algorithms can be used for both classification and regression problem.

There are some steps for classification :

1. Understanding Adaboost using decision stumps
2. Visualizing ensembles of decision stumps
3. Comparing linear regression with adaboosting

**About Diabetic dataset :**

Diabetic dataset has 769 entries consisting of various attributes like Pregnancies, glucose, blood pressure, skin thickness, insulin, BMI, age & diabetic pedigree function.

Description about attributes:

1. Pregnancies : What is rate of diabetes in pregnancy
2. glucose : amount of glucose
3. blood pressure : blood pressure of person
4. skin thickness : thickness of skin
5. insulin : amount of insulin
6. BMI : body mass index
7. Age : age of person affected by diabetes

**Source Code:**

Here we have implemented three classification algorithm like ada-booster classifier, knn classifier , decision tree , naïve algorithm.

import numpy as np import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

from sklearn.model\_selection import train\_test\_split from sklearn.naive\_bayes import GaussianNB

from sklearn.tree import DecisionTreeClassifier from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import precision\_recall\_fscore\_support

data=pd.read\_csv('C:/Users/Admin/Downloads/diabetes.csv') data.head()

data.describe() g=sns.PairGrid(data,vars=['Glucose','Insulin','BMI'],hue="Outcome",size=2.4) g.map\_diag(plt.hist)

g.map\_upper(plt.scatter) g.map\_lower(sns.kdeplot,cmap="Blues\_d")

plt.show()

#remove the outlier of skin thickness max\_skinthickness=data.SkinThickness.max() data=data[data.SkinThickness!=max\_skinthickness] def replace\_zero(df, field, target):

mean\_by\_target = df.loc[df[field] != 0, [field, target]].groupby(target).mean() data.loc[(df[field] == 0)&(df[target] == 0), field] = mean\_by\_target.iloc[0][0] data.loc[(df[field] == 0)&(df[target] == 1), field] = mean\_by\_target.iloc[1][0]

for col in ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']: replace\_zero(data, col, 'Outcome')

# split data

X = data.iloc[:,:-1]

y = data.iloc[:, -1]

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3, random\_state=100) print(X\_train.shape)

print(X\_test.shape) print(y\_train.size) print(y\_test.size)

Classifier=KNeighborsClassifier() Classifier.fit(X\_train,y\_train) y\_pred=Classifier.predict(X\_test) cm=confusion\_matrix(y\_test,y\_pred) print('Confusion Matrix of KNN is:',cm)

Classifier1=DecisionTreeClassifier() Classifier1.fit(X\_train,y\_train) y\_pred=Classifier1.predict(X\_test) cm=confusion\_matrix(y\_test,y\_pred) print('Confusion Matrix of DT is:',cm)

Classifier2=GaussianNB() Classifier2.fit(X\_train,y\_train) y\_pred=Classifier2.predict(X\_test) cm=confusion\_matrix(y\_test,y\_pred) print('Confusion Matrix of NB is:',cm)

Classifier3=AdaBoostClassifier() Classifier3.fit(X\_train,y\_train) y\_pred=Classifier3.predict(X\_test) cm=confusion\_matrix(y\_test,y\_pred) print('Confusion Matrix of ABC is:',cm)

y1=[81.38,83.98,84.41] x1=['KNN','DT','ADC']

plt.plot(x1, y1, color='green', linestyle='dashed', linewidth = 3, marker='o', markerfacecolor='blue', markersize=12)

plt.ylim(20,100) #plt.xlim(1,8)

# naming the x axis plt.xlabel('x - axis')

# naming the y axis plt.ylabel('y - axis')

# giving a title to my graph plt.title('Comparative study')

# function to show the plot plt.show()

# helper functions

def train\_clf(clf, X\_train, y\_train): return clf.fit(X\_train, y\_train)

def pred\_clf(clf, features, target): y\_pred = clf.predict(features)

return f1\_score(target.values, y\_pred, pos\_label = 1)

def train\_predict(clf, X\_train, y\_train, X\_test, y\_test): train\_clf(clf, X\_train, y\_train)

print("F1 score for training set is: {:.4f}".format(pred\_clf(clf, X\_train, y\_train))) print("F1 score for testing set is:

{:.4f}\n".format(pred\_clf(clf, X\_test, y\_test)))

#load algorithms nb = GaussianNB()

dtc = DecisionTreeClassifier(random\_state=0) knn = KNeighborsClassifier()

abc = AdaBoostClassifier(random\_state=0) algorithms = [nb,dtc,knn,abc]

for clf in algorithms: print("{}:".format(clf))

train\_predict(clf, X\_train, y\_train, X\_test, y\_test)

knn=KNeighborsClassifier(n\_neighbors=8) clf\_=knn.fit(X\_train,y\_train) y\_pred=clf\_.predict(X\_test)

print('Accuracy of KNN is {}'.format(accuracy\_score(y\_test,y\_pred)\*100))

dtc=DecisionTreeClassifier() clf1\_=dtc.fit(X\_train,y\_train) y\_pred=clf1\_.predict(X\_test)

print('Accuracy of DT is {}'.format(accuracy\_score(y\_test,y\_pred)\*100))

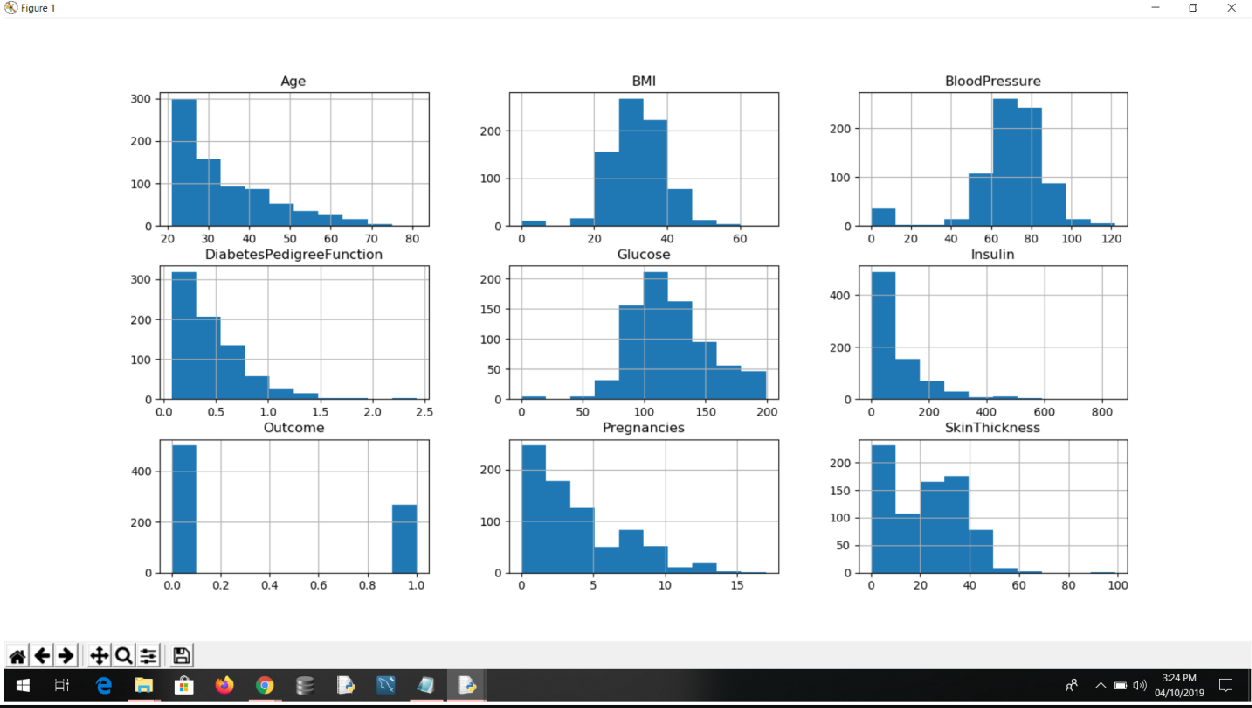
nb=GaussianNB() clf2\_=nb.fit(X\_train,y\_train) y\_pred=clf2\_.predict(X\_test)

print('Accuracy of NB is {}'.format(accuracy\_score(y\_test,y\_pred)\*100))

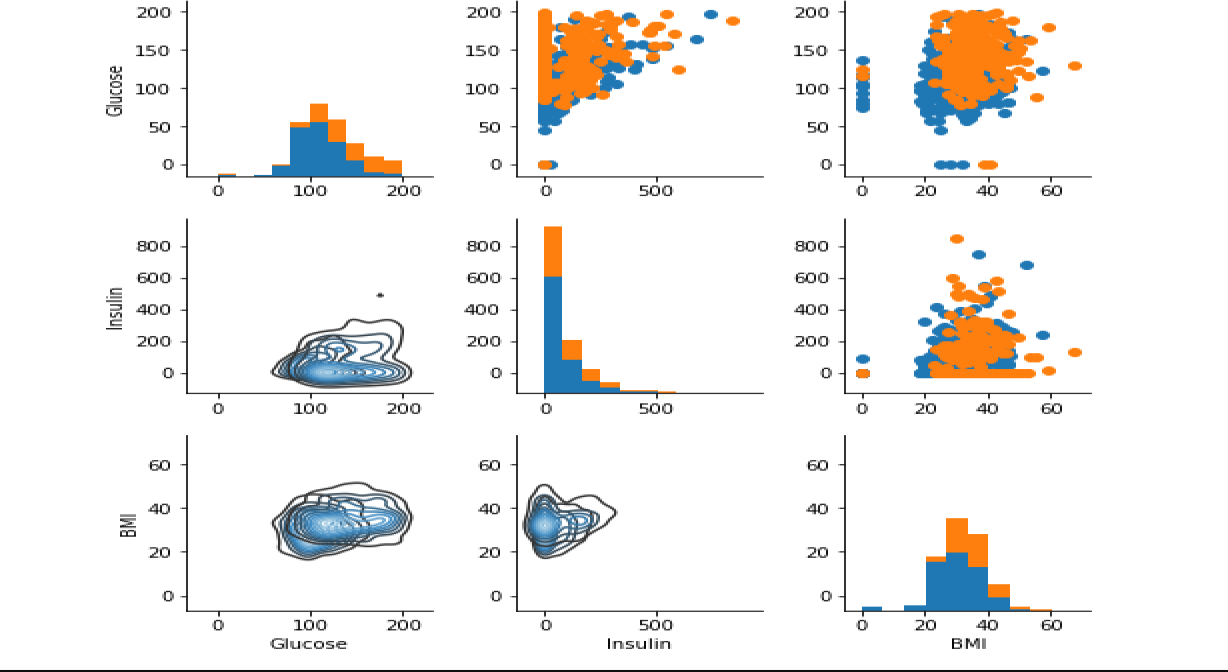
abc=AdaBoostClassifier() clf3\_=dtc.fit(X\_train,y\_train) y\_pred=clf3\_.predict(X\_test)

print('Accuracy of ABC is {}'.format(accuracy\_score(y\_test,y\_pred)\*100))

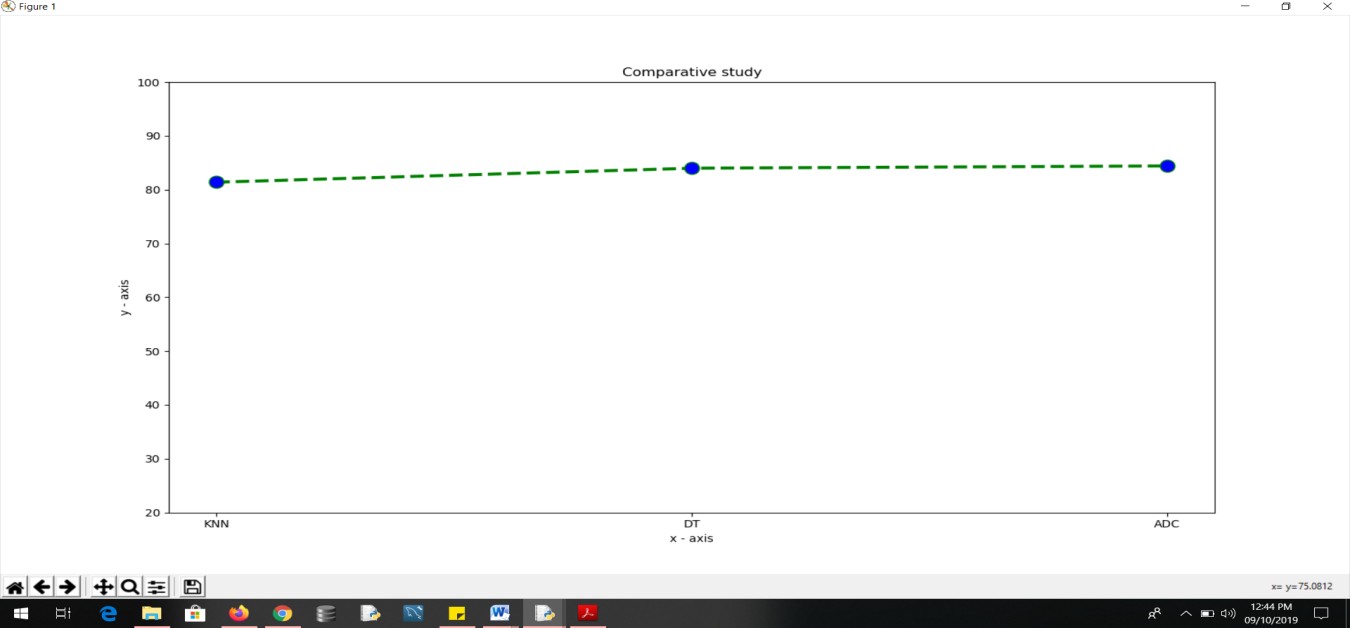
Output:



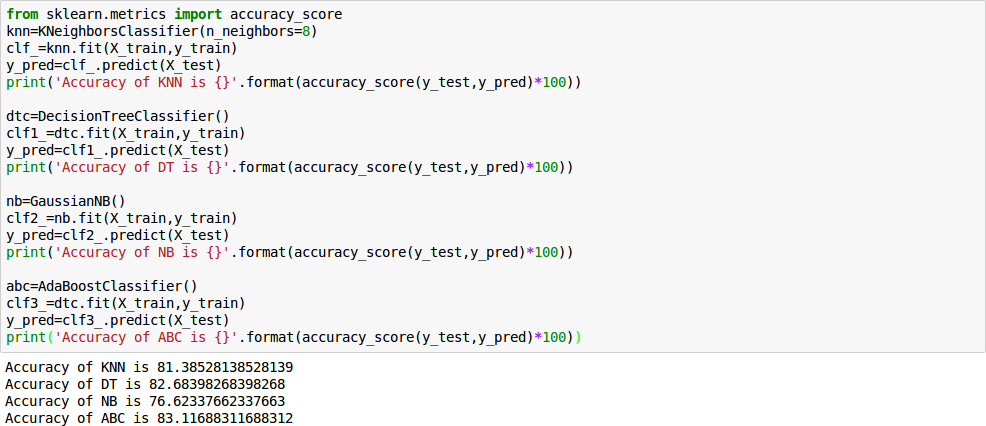
**Data Visualization:**



Comparative Accuracy Study:



**Classifier Output:**



**Conclusion:**

Thus, we studied different Classifier Algorithms and implemented those using python library sklearn on Diabetic dataset and compared their performances.