Middlesex University Dubai

MSc Data Science

Modelling, Regression and Machine-Learning

CST 4050

IRENE TETSOMA EREKU

SUMMATIVE ASSESSMENT-COURSEWORK 2

TOPIC: DIABETES PREDICTION USING MACHINE LEARNING

DESCRIPTION-THIS MACHINE LEARNING COURSEWORK WAS CARRIED OUT BY ONE INDIVIDUAL, FROM DATA SELECTION, PREPROCESSING, MACHINE BASELINES AND ALL OTHER PROCESSES

ABSTRACT

Classification techniques are widely employed in the medical sector for classifying data into different classes based on specified constraints. Diabetes is a disease that affects the body's ability to produce the hormone insulin, which causes carbohydrate metabolism to become abnormal and blood glucose levels to rise. High blood sugar is the most common symptom of diabetes.

Many researchers are performing trials to diagnose diseases using various classification algorithms of machine learning approaches such as SVM, Naive Bayes, Decision Tree, Decision Table, and others, since studies have shown that machine-learning algorithms are effective in detecting diseases.

Machine learning has been applied to many facets of medical health because of its rapid progress. In this project decision tree, random forest, logistic regression, support vector machine, kneighbors classifier, amongst others were used to predict the diabetes level.

INTRODUCTION

Machine learning has proven to be a great tool for training and predicting a certain system. It is the process of extracting structures from data. In recent years, machine learning has emerged as a dependable and supportive tool in the medical field. Automatic learning has sparked a lot of interest in the medical field since it takes less time to recognize and interact with patients, which means more time for patient care. Carbohydrate been one of the nutrients that gives energy, it is normally digested into a sugar called glucose by the stomach and intestines. Glucose is the body's primary energy source. After digestion, glucose enters the bloodstream and provides energy to the body (What causes diabetes?, 2022). Diabetes is divided into three types: type 1, type 2, and gestational diabetes. Diabetes 1 is caused by the body producing insufficient or no insulin. Young children, teenagers, and young adults are the most common victims of this kind of diabetes. They are mostly caused by a deficiency of insulin I in their bodies. This type of diabetes necessitates the injection of insulin into the patient's bloodstream. There is currently no known etiology for this form of diabetes. (Method for the analysing of blood glucose dynamics in diabetes mellitus patients, 2022).

Patients with this kind of diabetes cannot be properly treated with oral drugs alone, and insulin therapy is essential. Obesity, hypertension, dyslipidaemia, arteriosclerosis, and other disorders are typically connected with type 2 diabetes, which is more common in middle-aged and elderly adults (Robertson, Lehmann, Sandham and Hamilton, 2022).

Diabetes type 2 is caused by a lack of insulin resistance. Type 2 diabetes affects primarily adults, although it is now affecting children and teenagers. This group of patients has insufficient insulin in their bodies. Obesity is a key cause of type 2 diabetes, with a higher percentage of people at risk when their BMI is greater than 25. Patients who are non-insulin dependent fall within this category (NIDDM). (2022)
Gestations Diabetes is a condition that develops during pregnancy. After the birth of a child, this kind of diabetes can be healed. If this kind of diabetes is not treated properly, it has a high risk of developing.

Diabetes is becoming more widespread in people's daily lives as their living standards rise. As a result, learning how to diagnose and analyze diabetes fast and accurately is a topic worth researching. Diabetes is diagnosed using fasting blood glucose, glucose tolerance, and random blood glucose readings in medicine. (Cox and Edelman, 2022).

Blurry vision, fatigue, hunger, urinating often excessive thirst, and weight loss or gain are all symptoms of diabetes. Diabetes can be caused by high and low blood pressure, smoking, and having a high or low BMI. These are the most significant pre-diabetic alterations. By building insulin resistance, pre-diabetes increases the chances of curing diabetes with healthy diet and exercise.

Type 2 diabetes has the greatest number of symptoms. Prolonged diabetes has been linked to issues, according to a large study. Cardiovascular illness, macrovascular
The pancreas produces insulin to transport glucose from the bloodstream to the body's cells. In diabetes, the body either doesn't produce enough insulin or the cells can't utilise it properly. Instead, glucose accumulates in the in the blood, causing diabetes, otherwise known as high blood sugar. This has the potential to induce a variety of serious illnesses, including stroke, kidney failure, and heart attacks. In 2014, diabetes affected around 422 million people globally. In 2040, the population will reach 642 million (Carbohydrates and Diabetes (for Teens) - Nemours Kids Health, 2022).
The study's major goal in relation to the dataset is to create a machine learning (ML)-based system that can predict factors that affect diabetes, and how the insulin level can increase the chances of diabetes. The earlier we get a diagnosis, the easier it will be to control. Machine learning can assist patients in making a preliminary diagnosis of diabetes mellitus based on their daily physical examination data, and it can also be used by professionals as a reference.

The focus of this study is on diabetes-affected pregnant women and identify the prediction of diabetes in a patient, Naive Bayes, SVM,Random forest classifier, logistic regression,linear discrimant analysis, and decision Tree machine learning classification methods are employed and assessed on the dataset. On several criteria, the experimental performance of the three algorithms is compared, and they all attain good accuracy. This predictive analysis will help individuals and medical expects to know how certain factors such as age pregnancies, blood pressure, and especially insulin increases the chances of diabetes and with this many health safety precautions can be taken.

BACKGROUND

Machine Learning is concerned with the creation of algorithms and procedures that enable computers to learn and gain intelligence based on prior experience. It is a sub-discipline of Artificial Intelligence (AI) that is strongly linked to statistics. Learning implies that the system can recognize and comprehend the incoming data to make decisions and predictions based on it.
Early detection of diabetes disease lowers medical expenditures and lowers the likelihood of people developing more serious health problems. (Balkau et al., 2022).
Framingham Diabetes Risk Scoring Model (FDRSM) was used to predict the risk for developing diabetes Miletus in America between the aged group of 45-65 years using Logistic Regression. The Parental history of diabetes, obesity, high blood pressure, low levels of high-density lipoprotein cholesterol, raised triglyceride levels, and impaired fasting glucose are among the risk variables considered in this simple clinical model. The sample size was 3140 people, and the area under the receiver operating characteristic curve (AROC) was reported to be 85.0 percent. (Lai et al., 2022).

In studies on Diabetes Mellitus, data mining techniques have been frequently employed to investigate the risk factors for the disease to predict Diabetes mellitus and pre-diabetes, researchers used machine learning approaches such as logistic regression, artificial neural networks, and decision trees (Meng et al., 2022) 735 participants with diabetes or pre-diabetes and 752 healthy patients from Guangzhou, China were involved in the study. The accuracy of a decision tree model was reported to be 77.87 percent, 76.13 percent for a logistic regression model, and 73.23 percent for an Artificial Neural Network (ANN) technique. Random Forest, Support Vector Machines (SVM), k-nearest Neighbors (KNN), and naive Bayes are some of the other machine learning approaches that have been used (Kandhasamy and Balamurali, 2022).

One of the machine learning techniques is SVM. SVM works by identifying linear plane methods for categorization model data, which is then utilized to construct a new model. The main goal of classification is to ensure consistency. When the system's efficiency and support are improved through simplicity of use, reliability rises. SVMs are a potential method for diabetes prediction, where an intelligible rule set has been generated and prediction accuracy of 94 percent has been achieved (Intelligible Support Vector Machines for Diagnosis of Diabetes Mellitus, 2022).
The Linear Discrimination Analysis and Morlet Wavelet Support Vector Machine classifier (LDA-MWAVM) system can be used to diagnose disease automatically.

Using LDA, there are three stages: feature extraction, feature reduction, and feature reduction. Morlet Wavelet Support Vector Machine is used to classify the data (MWSVM). This system has an efficiency of 89.74 percent. When compared to other methods, efficiency is lower (An automatic diabetes diagnosis system based on LDA-Wavelet Support Vector Machine Classifier | Expert Systems with Applications: An International Journal, 2022).

Adaboost and Bagging ensemble machine learning approaches have a role to play when diabetes prediction is talked about (Perveen, Shahbaz, Guergachi and Keshavjee, 2022). Based on diabetes risk variables, a decision tree was used to classify Diabetes Mellitus and patients as diabetic or non-diabetic. The results of the experiment show that Adaboost machine learning ensemble technique outperforms bagging and a similar technique as decision tree. (Sisodia and Sisodia, 2022). Building a diabetes prediction system with the primary goal of predicting whether or not a candidate will get diabetes at a given age. The proposed system is built on the notion of machine learning and uses a decision tree to implement it. The obtained findings were good since the suggested system performs well in forecasting diabetes events at a specific age with improved accuracy when utilizing a Decision Tree (Orabi, Kamal and Rabah, 2022).

Research showed certain algorithms were to classify the risk of diabetes mellitus. The author used four well-known machine learning classification approaches to achieve the goal: Decision Tree, Artificial Neural Networks, Logistic Regression, and Naive Bayes. Bagging and Boosting procedures are used to improve the robustness of created models. Experiments demonstrate that the Random Forest algorithm produces the best outcomes,considering all the algorithms that have been used(Nai-arun and Moungmai, 2022)

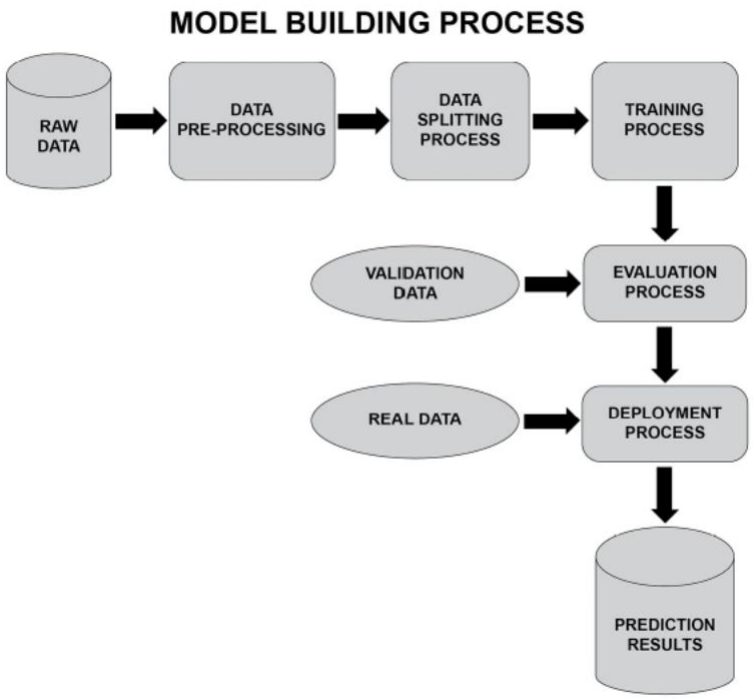
Three classification techniques were used: Nave Bayes, Decision Tree, and SVM To detect Diabetes, their findings revealed that the Nave Bayes method outperforms the other two algorithms. (Sisodia and Sisodia, 2022). Predictive models using Gradient Boosting Machine and Logistic Regression techniques to predict the probability of patients having Diabetes based on their demographic information and laboratory results from their visits to medical facilities. We also compare these methods with other widely used machine learning techniques such as Random Forest which gave an accuracy of 100 percent (Zou et al., 2022).

Machine learning approaches are commonly employed in diabetes prediction, and they produce better outcomes. In the medical industry, Decision trees are a prominent machine learning method with good classification capability. For this research linear regression, linear discriminant analysis will be used as the baseline algorithms also other algorithms such as random forest, decision tree classifier, support vector machine, naïve bayes will be applied as main algorithms.

METHOD AND RESULTS

The machine learning process is a set of operations that are carried out to deploy a successful prediction model. A few of the processes are iterative, meaning they can be repeated based on the results of the previous and subsequent steps. To train a successful model, we need to clean our data thoroughly. The model's performance is determined by the quality of the features. All libraries needed for the model building were imported, these libraries are pandas which is used for data cleaning ,NumPy which is used as multidimensional array, matplot and seaborn which are used for data visualization, confusion matrix which is used to measure the performance of the classification model, algorithm such as logistic regression, support vector machine, linear regression, decision tree classifier, random forest classifier etc were imported to see how different algorithms perform. Also, certain evaluating models such as confusion matrix, mean squared error, F1 score, mean absolute error were used to evaluate the performance of the algorithms

Several methods are applied to the data to achieve the right prediction in the model building process, these process ranges from collection of datasets, preparation of the data, training and tuning application of the different model and validation in other to achieve a predictive result. The diagram below shows steps in sample model building process.



Data

The Dataset of female patients with minimum twenty-one-year age of Pima Indian population has been taken Kaggle. There are 768 instances in this dataset, divided into two groups: diabetic and non-diabetic, with eight risk factors: number of pregnancies, two-hour plasma glucose concentration in an oral glucose tolerance test, diastolic blood pressure, triceps skin fold thickness, two-hour serum insulin, body mass index, diabetes pedigree function, and age as in the original dataset.

Hypothesis and Research Questions

Question 1: Predict a patient having diabetes
Question 2: The insulin level could be a major factor that causes diabetes.
Question 3: The probability of a patient getting diabetes because of their body mass index (BMI).

Data Visualization and Exploratory Analysis

The data set's structural data were checked, using the scatter plot to have a visual view of the data at its raw form in other to get a clear form of our data and some outliers were present, Scatterplots are used to show the relationship between each, and every attribute or features taken pairwise. Looking at the scatterplots, as seen in the appendix we can say that no two attributes are able to clearly separate the two outcomes as seen in the appendix

Further exploratory analysis was carried out to have clearer visual of the data using the histogram**.** Also, a correlation matrix was plotted to show the correlation coefficients between the variables in the diabetes data The correlation between two distinct variables is shown in each cell of the table, the symmetry of a correlation matrix is complete. The top right cell has the same value as the bottom left cell also darker shades of the chart represent higher values than the lighter shade.

Data Pre-Processing

Data pre-processing is a phase in the data mining and data analysis process that turns raw data into a format that computers and machine learning can understand and evaluate. It may not only contain errors and inconsistencies, but it is frequently incomplete and lacks a consistent design The data processing process involves basically three steps which are.

Data Cleaning

Data refers to tasks such as filling in missing values in data and correcting errors.
In this case our data was collected, and null values were checked for in columns, and each row in the data set and there were no missing values or non-values in the data set. (Fig 1) in appendix shows how the null values were checked for and how the outliers were treated in the data set Outliers were checked for in each column and seven outliers were found in pregnancies, blood pressure, skin thickness, insulin, body mass index, diabetes pedigree function and age. Outliers were removed

Feature Engineering

Feature selection was applied, and two more attributes were created; New BMI and New insulin score which was used to further the predictions. Label encoder was also applied which is used to change categorical values to numerical values as machines cannot interpret the categorical data directly. Therefore, the categorical data must be converted into numerical data for further processing. These transformation processes are performed with Label Encoding. Details of how feature engineering and feature selection is seen in the appendix. Principal component analysis (PCA) which is used to reduce the dimensionality was not needed as the dataset is not large and all features were important.
New features were combined, which were New BMI and New insulin score were created to further the analysis, the data was standardised and been that the data set is not a large dataset, it was Splitted into a training size of 70% and test size of 30%, random state was set to zero (0).

Baseline Algorithms and Techniques

Studies has shown that prediction on diabetes from previous research using classification algorithms with the basis of all the parameters SVM-linear, random forest and KNN logistic regression were best models that can be used to find out if patient is diabetic or not. Linear Regression was used as the baseline as it did not show much great performance which was compared to other classification algorithms such as random forest Support vector machine (SVM), logistic regression, k-Nearest Neighbour, decision tree classifier. Other new machine learning algorithms such as Ada boaster classifier, were used for the purpose of this research.

Table 2: Below show the Comparison of baseline algorithm

```
df=pd.DataFrame(data_scores,columns=['Model','MSE','RMSE','MAE','R2'])
df
```

	Model	MSE	RMSE	MAE	R2
0	Logistic Regression	0.004329	0.065795	0.004329	0.982407
1	Linear Discriminant Analysis	0.004329	0.065795	0.004329	0.982407
2	Linear Regression	0.067921	0.260616	0.218104	0.723966

Three different algorithms were used to check which can be used as a baseline algorithm which gave an accuracy of 72%, in which linear regression gave the least compared to logistic regression and linear discriminant analysis.in this case it is expected that main algorithms can have an accuracy above 72%.

Table 2: Shows results of all different algorithms

]:

	Model	MSE	RMSE	MAE	R2
0	Logistic Regression	0.004329	0.065795	0.004329	0.982407
1	Linear Discriminant Analysis	0.004329	0.065795	0.004329	0.982407
2	Linear Regression	0.067921	0.260616	0.218104	0.723966
3	Support Vector Machine	0.004329	0.065795	0.004329	0.982407
4	KNeighbors Classifier	0.004329	0.065795	0.004329	0.982407
5	Support Vector Machine	0.004329	0.065795	0.004329	0.982407
6	KNeighbors Classifier	0.004329	0.065795	0.004329	0.982407
7	Decision Tree Classifier	0.004329	0.065795	0.004329	0.982407
8	NAIVE BAYES	0.021645	0.147122	0.021645	0.912034
9	GradientBoostingClassifier	0.004329	0.065795	0.004329	0.982407
10	AdaBoostClassifier	0.021645	0.147122	0.021645	0.912034
11	ExtraTreesClassifier	0.021645	0.147122	0.021645	0.912034
12	Support Vector Machine	0.004329	0.065795	0.004329	0.982407
13	KNeighbors Classifier	0.004329	0.065795	0.004329	0.982407
14	Decision Tree Classifier	0.004329	0.065795	0.004329	0.982407
15	NAIVE BAYES	0.021645	0.147122	0.021645	0.912034
16	GradientBoostingClassifier	0.004329	0.065795	0.004329	0.982407
17	AdaBoostClassifier	0.021645	0.147122	0.021645	0.912034
18	ExtraTreesClassifier	0.021645	0.147122	0.021645	0.912034
19	ExtraTreesClassifier	0.021645	0.147122	0.021645	0.912034
20	Random Forest Classifier	0.000000	0.000000	0.000000	1.000000

From the above result, Linear regression did not give a very good accuracy compared to logistic regression and linear discriminant analysis k-neighbour classifier, support vector machine gave an R-squared value of 98% and a root mean squared error(RSME) of 0.66 while random forest classifier gave a 100% with a root mean squared error(RSME) of 0.00.Random forest gave a good performance as compared to other main algorithm, although more parameters will be used to evaluate some of the models to see if there will be a better fit.

Table 3: Shows the results obtained after k-fold cross validation has been applied

Algorthim	Mean Result %	Standard Deviation Result %
K-Nearest Neighbour	95	0.022088
Support Vector Classifier	97	0.015014
Decision Tree Classifier	100	0.000000
Random Forest Classifier	100	0.000000
Gradient Boosting Classifier	100	0.000000

Table 4: Evaluation parameters of the predictive models

Models	Accuracy %	Recall %	Precision %	F1 Score %	Runtime
Random Forest	100	100	100	100	-0.003503999
Decision Tree Classifier	100	100	100	100	0.0033179998
K-Neighbor Classifier	0.91	0.93	0.91	0.92	-0.001876499
Gradient Boosting Classifier	100	100	100	100	-0.003811099

The table above shows the performance of the algorithm after the application validation, scaling, and standardization of the algorithms. Also, the time complexity/runtime of these algorithms were checked. The results showed random forest had an accuracy of 100% with a runtime of -0.003503999, While decision tree which also has an 100% accuracy with a time complexity of 0.003317999.Gradient Boosting Classifier also had an accuracy of 100% and the runtime was recorded as -0.00381109.K-neighbour had the least compared to three other algorithm, with these it can be concluded that random forest classifier, decision tree classifier, gradient boosting classifier had the best accuracy.

CONCLUSION

The early identification of diabetes is one of the most pressing real-world medical issues. In this research, systematic efforts are undertaken to build a system that may forecast diseases such as diabetes. Three machine learning classification methods are researched and assessed on several measures as part of this research.

Using the support vector machine (SVM), Random forest classifier, logistic regression, gradient boosting classifier, linear discriminant analysis, KNN, decision tree classifier were models created to diagnose diabetes, This also goes to answer the hypothesis question whether if body mass index(BMI) is linked to a higher chance of acquiring diabetes problems and also how insulin causes diabetes as studies has shown that a normal insulin level ranges from 2 to 20 mIU/mL

Accuracy, recall, precision, and F1 score, are all factors that are considered while evaluating the models. In comparison to the other models utilized, the experimental findings revealed that all the models produced good results; however, the random forest model delivers the best accuracy of 100% and precision of 100% for diabetes prediction. F1 score may provide better insight into the performance of our models because our dataset is an example of imbalanced class. The F1 score achieves a good mix of precision and recall. So, based on all the characteristics, it can be concluded that random forest classifier, decision tree classifier, gradient boosting classifier are models for determining if a patient is diabetic or not. Also an ROC curve was plotted to show accuracy of the models

The developed approach, together with the machine learning classification techniques used, could be used to predict, or detect other diseases in the future. The work can be expanded and enhanced for diabetes analysis automation, as well as some other machine learning techniques.

REFERENCES

https://www.nichd.nih.gov/. 2022. *What causes diabetes?*. [online] Available at: <https://www.nichd.nih.gov/health/topics/diabetes/conditioninfo/causes#f1> [Accessed 6 April 2022].

Ieeexplore.ieee.org. 2022. *Method for the analysing of blood glucose dynamics in diabetes mellitus patients*. [online] Available at: <https://ieeexplore.ieee.org/abstract/document/4588883> [Accessed 7 April 2022].

Robertson, G., Lehmann, E., Sandham, W. and Hamilton, D., 2022. *Blood Glucose Prediction Using Artificial Neural Networks Trained with the AIDA Diabetes Simulator: A Proof-of-Concept Pilot Study*.

Cox, M. and Edelman, D., 2022. *Tests for Screening and Diagnosis of Type 2 Diabetes*.

Enggjournals.com. 2022. [online] Available at: <http://www.enggjournals.com/ijet/docs/IJET13-05-03-292.pdf> [Accessed 15 April 2022].

Kidshealth.org. 2022. *Carbohydrates and Diabetes (for Teens) - Nemours KidsHealth*. [online] Available at: <https://kidshealth.org/en/teens/carbs-diabetes.html> [Accessed 6 April 2022].

Balkau, B., Lange, C., Fezeu, L., Tichet, J., de Lauzon-Guillain, B., Czernichow, S., Fumeron, F., Froguel, P., Vaxillaire, M., Cauchi, S., Ducimetière, P. and Eschwège, E., 2022. *Predicting Diabetes: Clinical, Biological, and Genetic Approaches*.

Lai, H., Huang, H., Keshavjee, K., Guergachi, A. and Gao, X., 2022. *Predictive models for diabetes mellitus using machine learning techniques*.

Meng, X., Huang, Y., Rao, D., Zhang, Q. and Liu, Q., 2022. *Comparison of three data mining models for predicting diabetes or prediabetes by risk factors*.

Ieeexplore.ieee.org. 2022. *Intelligible Support Vector Machines for Diagnosis of Diabetes Mellitus*. [online] Available at: <https://ieeexplore.ieee.org/document/5378519> [Accessed 15 April 2022].

Kandhasamy, J. and Balamurali, S., 2022. *Performance Analysis of Classifier Models to Predict Diabetes Mellitus*.

Expert Systems with Applications: An International Journal. 2022. *An automatic diabetes diagnosis system based on LDA-Wavelet Support Vector Machine Classifier | Expert Systems with Applications: An International Journal*. [online] Available at: <https://dl.acm.org/doi/abs/10.1016/j.eswa.2011.01.017> [Accessed 15 April 2022].

Sisodia, D. and Sisodia, D., 2022. *Prediction of Diabetes using Classification Algorithms*.

Perveen, S., Shahbaz, M., Guergachi, A. and Keshavjee, K., 2022. *Performance Analysis of Data Mining Classification Techniques to Predict Diabetes*.

Sisodia, D. and Sisodia, D., 2022. *Prediction of Diabetes using Classification Algorithms*.

Orabi, K., Kamal, Y. and Rabah, T., 2022. *Early Predictive System for Diabetes Mellitus Disease*.

Nai-arun, N. and Moungmai, R., 2022. *Comparison of Classifiers for the Risk of Diabetes Prediction*.

Zou, Q., Qu, K., Luo, Y., Yin, D., Ju, Y. and Tang, H., 2022. *Predicting Diabetes Mellitus With Machine Learning Techniques*.

APPENDIX

```
#importing the liberies needed
import pandas as pd #used for data cleaning
import numpy as np
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt #used for data visualization
import seaborn as sns  #used for data visualization
from IPython.display import display
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
from sklearn.svm import SVC
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.metrics import classification_report
from sklearn.neural_network import MLPClassifier

import plotly.tools as tls
import plotly.offline as py
py.init_notebook_mode(connected=True)
from scipy import stats
from scipy.stats import norm

plt.style.use('ggplot')
%matplotlib inline
```

In [227]:
```
#Loading the dataset
df = pd.read_csv("diabetes.csv")
```

In [228]:
```
#Feature information,which gives details of each colum in the data set
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [229]:
```
# The size of the data set was examined. It consists of 768 observation units and 9 variables.
df.shape
```

Out[229]: (768, 9)

ANALYSING THE DATA

In [230]:
```
df.head()
```

Out[230]:

	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

In [231]:
```
# Descriptive statistics of the data set
df.describe()
```

Out[231]:

	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

[232]:
```
#checking for null values in the columns
df.isnull().sum()
```

t[232]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[233]:
```
#checking for null values in the dataset
df.isnull().any()
```

t[233]:
```
Pregnancies                 False
Glucose                     False
BloodPressure               False
SkinThickness               False
Insulin                     False
BMI                         False
DiabetesPedigreeFunction    False
Age                         False
Outcome                     False
dtype: bool
```

[234]:
```
#checked for null values in the dataset and no null value was present
df.isnull()
```

t[234]:

	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 [235]:
```
#Access to the correlation of the data set was provided. What kind of relationship is examined between the variables.
# If the correlation value is> 0, there is a positive correlation. While the value of one variable increases, the value of the ot
# Correlation = 0 means no correlation.
# If the correlation is <0, there is a negative correlation. While one variable increases, the other variable decreases.
# When the correlations are examined, there are 2 variables that act as a positive correlation to the Salary dependent variable.
# These variables are Glucose. As these increase, Outcome variable increases.
df.corr()
```

Out[235]:

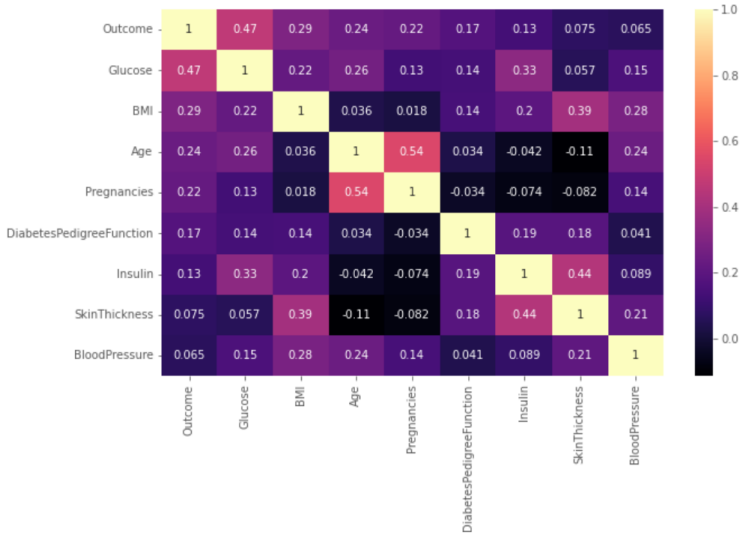
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

In [236]:
```
# Correlation matrix graph of the data set
k = 9
cols = df.corr().nlargest(k, 'Outcome')['Outcome'].index
cm = df[cols].corr()
plt.figure(figsize=(10,6))
sns.heatmap(cm, annot=True, cmap = 'magma')
```

Out[236]: <AxesSubplot:>

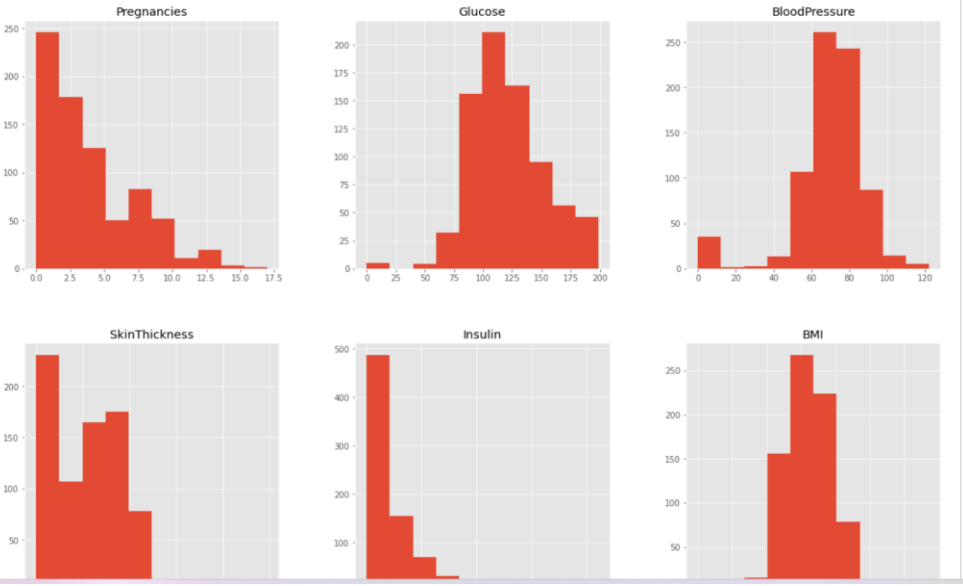

```
sns.heatmap(cm, annot=True, cmap = 'magma')
```

Out[236]: <AxesSubplot:>



In [237]:
```
#The diagonal shows the distribution of the the dataset with the kernel density plots,The scatter-plots shows the relationship be
g = sns.pairplot(df, hue="Outcome", palette="husl")
```

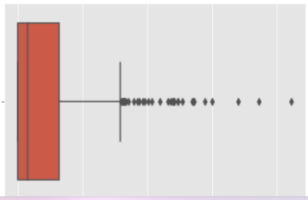
In [238]:
```
#histogram to have a clearer visual of the dataset and see where the outlier are opresent
p = df.hist(figsize = (20,20))
```



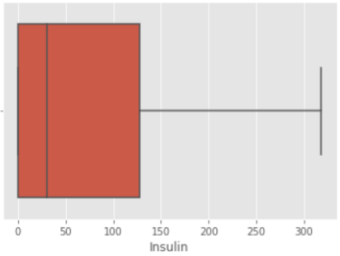
In [239]:
```
# In the data set, there were asked whether there were any outlier observations compared to the 25% and 75% quarters.
# It was found to be an outlier observation.
for feature in df:

    Q1 = df[feature].quantile(0.25)
    Q3 = df[feature].quantile(0.75)
    IQR = Q3-Q1
    lower = Q1- 1.5*IQR
    upper = Q3 + 1.5*IQR

    if df[(df[feature] > upper)].any(axis=None):
        print(feature,"yes")
    else:
        print(feature, "no")
```

```
Pregnancies yes
Glucose no
BloodPressure yes
SkinThickness yes
Insulin yes
BMI yes
DiabetesPedigreeFunction yes
Age yes
Outcome no
```

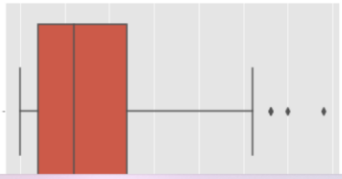
In [240]:
```
# The process of visualizing the Insulin variable with boxplot method was done. We find the outlier observations on the chart.
sns.boxplot(x = df["Insulin"]);
```



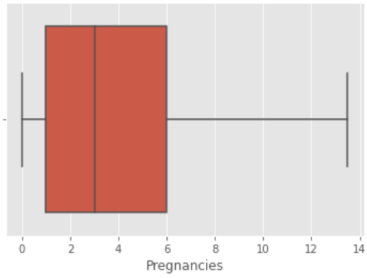
n [241]:
```
#We conduct a stand alone observation review for the Insulin variable
#We suppress contradictory values
Q1 = df.Insulin.quantile(0.25)
Q3 = df.Insulin.quantile(0.75)
IQR = Q3-Q1
lower = Q1 - 1.5*IQR
upper = Q3 + 1.5*IQR
df.loc[df["Insulin"] > upper,"Insulin"] = upper
sns.boxplot(x = df["Insulin"]);
```



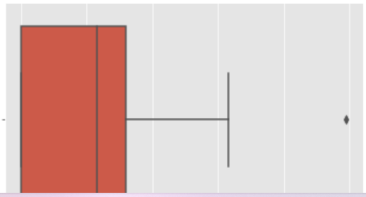
n [242]:
```
The process of visualizing the pregnancies variable with boxplot method was done. We find the outlier observations on the chart.
ns.boxplot(x = df["Pregnancies"]);
```



In [243]:
```
#We conduct a stand alone observation review for the pregnancies variable
#We suppress contradictory values
Q1 = df.Pregnancies.quantile(0.25)
Q3 = df.Pregnancies.quantile(0.75)
IQR = Q3-Q1
lower = Q1 - 1.5*IQR
upper = Q3 + 1.5*IQR
df.loc[df["Pregnancies"] > upper,"Pregnancies"] = upper
sns.boxplot(x = df["Pregnancies"]);
```



In [244]:
```
# Visualizing the Skinthickness variable with boxplot method was done. We find the outlier observations on the chart.
sns.boxplot(x = df["SkinThickness"]);
```



```
for feature in df:

    Q1 = df[feature].quantile(0.25)
    Q3 = df[feature].quantile(0.75)
    IQR = Q3-Q1
    lower = Q1- 1.5*IQR
    upper = Q3 + 1.5*IQR

    if df[(df[feature] > upper)].any(axis=None):
        print(feature,"yes")
    else:
        print(feature, "no")
```

```
Pregnancies no
Glucose no
BloodPressure no
SkinThickness no
Insulin no
BMI no
DiabetesPedigreeFunction no
Age no
Outcome no
```

Feature Engineering and Selection ¶

In [265]:
```
# According to BMI, some ranges were determined and categorical variables were assigned.
NewBMI = pd.Series(["Underweight", "Normal", "Overweight", "Obesity 1", "Obesity 2", "Obesity 3"], dtype = "category")
df["NewBMI"] = NewBMI
df.loc[df["BMI"] < 18.5, "NewBMI"] = NewBMI[0]
df.loc[(df["BMI"] > 18.5) & (df["BMI"] <= 24.9), "NewBMI"] = NewBMI[1]
df.loc[(df["BMI"] > 24.9) & (df["BMI"] <= 29.9), "NewBMI"] = NewBMI[2]
df.loc[(df["BMI"] > 29.9) & (df["BMI"] <= 34.9), "NewBMI"] = NewBMI[3]
df.loc[(df["BMI"] > 34.9) & (df["BMI"] <= 39.9), "NewBMI"] = NewBMI[4]
df.loc[df["BMI"] > 39.9 ,"NewBMI"] = NewBMI[5]
```

In [266]:
```
# A categorical variable creation process is performed according to the insulin value.
def set_insulin(row):
    if row["Insulin"] >= 16 and row["Insulin"] <= 166:
        return "Normal"
    else:
        return "Abnormal"
```

In [267]:
```
# The operation performed was added to the dataframe.
df = df.assign(NewInsulinScore=df.apply(set_insulin, axis=1))

df.head()
```

Out[267]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	NewBMI	NewInsulinScore
0	6	86	22	28	0	123	350	29	1	Obesity 3	Abnormal
1	1	23	19	22	0	62	196	10	0	Obesity 3	Abnormal
2	8	121	17	0	0	30	368	11	1	Obesity 1	Abnormal
3	1	27	19	16	62	77	53	0	0	Obesity 3	Normal
4	0	75	4	28	102	209	489	12	1	Obesity 3	Normal

LABEL ENCODING

n [268]:
```
from sklearn.preprocessing import LabelEncoder

NewInsulinScore_encoder = LabelEncoder()
df = df.copy()
df.NewInsulinScore = NewInsulinScore_encoder.fit_transform(df.NewInsulinScore)
df

NewBMI_encoder = LabelEncoder()
df = df.copy()
df.NewBMI = NewBMI_encoder.fit_transform(df.NewBMI)
df
```

ut[268]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	NewBMI	NewInsulinScore
0	6	86	22	28	0	123	350	29	1	3	0
1	1	23	19	22	0	62	196	10	0	3	0
2	8	121	17	0	0	30	368	11	1	1	0
3	1	27	19	16	62	77	53	0	0	3	1
4	0	75	4	28	102	209	489	12	1	3	1
...	...	...	...	...	...	...	...	...	...	...	...
763	10	39	25	41	108	118	55	42	0	3	1
764	2	60	21	20	0	155	187	6	0	3	0
765	5	59	22	16	71	58	115	9	0	3	1
766	1	64	14	0	0	95	195	26	1	3	0
767	1	31	21	24	0	98	169	2	0	3	0

768 rows × 11 columns

n [269]:
```
df.shape #showing new arritributes
```

ut[269]: (768, 11)

.269]: (768, 11)

.270]:
```
df[cols] = df[cols].apply(LabelEncoder().fit_transform)
```

.271]:
```
#train_test_splitting of the dataset

X=df.drop('NewInsulinScore',1)
Y=df['NewInsulinScore']
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.30,random_state=0)
```

.272]:
```
#defining evaluating models
data_scores=[]
def EvaluatingModels(true_value,predicted_value,model):
    MSE=mean_squared_error(true_value,predicted_value,squared=True)
    RMSE=mean_squared_error(true_value,predicted_value,squared=False)
    MAE=mean_absolute_error(true_value,predicted_value)
    R_Squared=r2_score(true_value,predicted_value)
    data_scores.append([model,MSE,RMSE,MAE,R_Squared])
    print("MAE_:", MAE)
    print("RMSE_:", RMSE)
    print("MSE_:", MSE)
    print("R_Squared_:", R_Squared)
```

.273]:
```
#Logistic regression
regress_dtree=LogisticRegression()
dtree_model=regress_dtree.fit(X_train,Y_train)
y_predtree=regress_dtree.predict(X_test)
EvaluatingModels(Y_test,y_predtree, 'Logistic Regression')
```

```
MAE_: 0.004329004329004329
RMSE_: 0.0657951694959769
MSE_: 0.004329004329004329
R_Squared_: 0.9824067022086824
```

[275]:
```
#Linear regression
linear_regressor=LinearRegression(normalize=True)
linear_regressor.fit(X_train,Y_train)
lin_prediction=linear_regressor.predict(X_test)
EvaluatingModels(Y_test,lin_prediction, 'Linear Regression')
```

```
MAE_: 0.2181041163659208
RMSE_: 0.26061612601703177
MSE_: 0.06792076514012539
R_Squared_: 0.7239664928680707
```

[276]:
```
df=pd.DataFrame(data_scores,columns=['Model','MSE','RMSE','MAE','R2'])
df
```

[276]:

	Model	MSE	RMSE	MAE	R2
0	Logistic Regression	0.004329	0.065795	0.004329	0.982407
1	Linear Discriminant Analysis	0.004329	0.065795	0.004329	0.982407
2	Linear Regression	0.067921	0.260616	0.218104	0.723966

```
Linear regression can be used as baseline model as it has the least value
```

In [287]:
```
regress_dtree=KNeighborsClassifier()
dtree_model =regress_dtree.fit(X_train,Y_train)
y_pretree = regress_dtree.predict(X_test)
EvaluatingModels(Y_test,y_predtree, 'KNeighbors Classifier')
```

```
MAE_: 0.004329004329004329
RMSE_: 0.0657951694959769
MSE_: 0.004329004329004329
R_Squared_: 0.9824067022086824
```

In [288]:
```
regress_dtree=DecisionTreeClassifier(criterion='entropy',random_state=0)
dtree_model =regress_dtree.fit(X_train,Y_train)
y_pretree = regress_dtree.predict(X_test)
EvaluatingModels(Y_test,y_predtree, 'Decision Tree Classifier')
```

```
MAE_: 0.004329004329004329
RMSE_: 0.0657951694959769
MSE_: 0.004329004329004329
R_Squared_: 0.9824067022086824
```

In [289]:
```
#Naive Bayes
rf_regressor=GaussianNB()
rf_model=rf_regressor.fit(X_train,Y_train)
y_rfpred=rf_model.predict(X_test)

EvaluatingModels(Y_test,y_rfpred, 'NAIVE BAYES')
```

```
MAE_: 0.021645021645021644
RMSE_: 0.14712247158412492
MSE_: 0.021645021645021644
R_Squared_: 0.912033511043412
```

In [290]:
```
# Gradient Boosting Classifier
gbc = GradientBoostingClassifier()
gb = GradientBoostingClassifier(random_state=0,learning_rate=0.01)
gbc.fit(X_train, Y_train)
gbc_acc=accuracy_score(Y_test,gbc.predict(X_test))
EvaluatingModels(Y_test,y_predtree, 'GradientBoostingClassifier')
```

```
MAE_: 0.004329004329004329
RMSE_: 0.0657951694959769
MSE_: 0.004329004329004329
R_Squared_: 0.9824067022086824
```

```
#Extra Trees Classifier
etc = ExtraTreesClassifier(n_estimators=100, random_state=1)
etc.fit(X_train,Y_train)

EvaluatingModels(Y_test,y_rfpred, 'ExtraTreesClassifier')
```

```
MAE_: 0.021645021645021644
RMSE_: 0.14712247158412492
MSE_: 0.021645021645021644
R_Squared_: 0.912033511043412
```

```
rf_regressor=RandomForestClassifier(random_state=1)
rf_model=rf_regressor.fit(X_train,Y_train)
y_rfpred = rf_model.predict(X_test)
EvaluatingModels(Y_test,y_rfpred, 'Random Forest Classifier')
```

```
MAE_: 0.0
RMSE_: 0.0
MSE_: 0.0
R_Squared_: 1.0
```

```
Random forest had 100% performance,while gradient boosting classifier,Decision tree classifier,linear
regression,Kneighbors,logistic regression had 98% performance
```

```
df=pd.DataFrame(data_scores,columns=['Model','MSE','RMSE','MAE','R2'])
df
```

	Model	MSE	RMSE	MAE	R2
0	Logistic Regression	0.004329	0.065795	0.004329	0.982407
1	Linear Discriminant Analysis	0.004329	0.065795	0.004329	0.982407
2	Linear Regression	0.067921	0.260616	0.218104	0.723966
3	Support Vector Machine	0.004329	0.065795	0.004329	0.982407
4	KNeighbors Classifier	0.004329	0.065795	0.004329	0.982407
5	Support Vector Machine	0.004329	0.065795	0.004329	0.982407
4	KNeighbors Classifier	0.004329	0.065795	0.004329	0.982407
5	Support Vector Machine	0.004329	0.065795	0.004329	0.982407
6	KNeighbors Classifier	0.004329	0.065795	0.004329	0.982407
7	Decision Tree Classifier	0.004329	0.065795	0.004329	0.982407
8	NAIVE BAYES	0.021645	0.147122	0.021645	0.912034
9	GradientBoostingClassifier	0.004329	0.065795	0.004329	0.982407
10	AdaBoostClassifier	0.021645	0.147122	0.021645	0.912034
11	ExtraTreesClassifier	0.021645	0.147122	0.021645	0.912034
12	Support Vector Machine	0.004329	0.065795	0.004329	0.982407
13	KNeighbors Classifier	0.004329	0.065795	0.004329	0.982407
14	Decision Tree Classifier	0.004329	0.065795	0.004329	0.982407
15	NAIVE BAYES	0.021645	0.147122	0.021645	0.912034
16	GradientBoostingClassifier	0.004329	0.065795	0.004329	0.982407
17	AdaBoostClassifier	0.021645	0.147122	0.021645	0.912034
18	ExtraTreesClassifier	0.021645	0.147122	0.021645	0.912034
19	ExtraTreesClassifier	0.021645	0.147122	0.021645	0.912034
20	Random Forest Classifier	0.000000	0.000000	0.000000	1.000000

In [298]:
```
#scaled and applied kfold to algorithms to reavaluate them
from sklearn.preprocessing import MinMaxScaler
scaler =MinMaxScaler(feature_range=(0, 1))
scaledX =scaler.fit_transform(X_train)
```

In [300]:
```
from sklearn.model_selection import RepeatedStratifiedKFold
models=[] #to initialze the models
models.append(('KNN',KNeighborsClassifier()))
models.append(('SVM',SVC()))
models.append(('DTR',DecisionTreeClassifier(criterion='entropy',random_state=0)))
models.append(('RFR',RandomForestClassifier()))
models.append(('GBC',GradientBoostingClassifier()))
#models are evaluated
results =[]
names =[]
for name,model in models:
    kfold = RepeatedStratifiedKFold(n_splits=10,n_repeats=3,random_state=1) #defining kfolds to use
    cv_results = cross_val_score(model,scaledX,Y_train,cv=kfold,scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    d_score= print('%s:%f(%f)'% (name,cv_results.mean(),cv_results.std()))
```

```
KNN:0.950408(0.022088)
SVM:0.978884(0.015014)
DTR:1.000000(0.000000)
RFR:1.000000(0.000000)
GBC:1.000000(0.000000)
```

302]:
```
import timeit #to check the time complexity needed to run the algorthim
start=timeit.timeit()
model = RandomForestClassifier()
model.fit(scaledX,Y_train)
#estimated the accuracy on validation dataset
rescaledValidationX = scaler.transform(X_test)
predictions=model.predict(rescaledValidationX)
print(accuracy_score(Y_test, predictions))
print(confusion_matrix(Y_test, predictions))#used tp check confusion matrix
print(classification_report(Y_test, predictions))
end = timeit.timeit()
print(end - start )
```

```
1.0
[[130   0]
 [  0 101]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       130
           1       1.00      1.00      1.00       101

    accuracy                           1.00       231
   macro avg       1.00      1.00      1.00       231
weighted avg       1.00      1.00      1.00       231

-0.0035039999056607485
```

[305]:
```
model = DecisionTreeClassifier(criterion='entropy',random_state=0)
model.fit(scaledX,Y_train)
#estimated the accuracy on validation dataset
rescaledValidationX = scaler.transform(X_test) #scaled data
predictions=model.predict(rescaledValidationX)
print(accuracy_score(Y_test, predictions))
print(confusion_matrix(Y_test, predictions))
print(classification_report(Y_test, predictions))
end = timeit.timeit()
print(end - start )#print time complexity
```

```
1.0
[[130   0]
 [  0 101]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       130
           1       1.00      1.00      1.00       101

    accuracy                           1.00       231
   macro avg       1.00      1.00      1.00       231
weighted avg       1.00      1.00      1.00       231

-0.0033179998863488436
```

:
```
model =KNeighborsClassifier()
model.fit(scaledX,Y_train)
#estimated the accuracy on validation dataset
rescaledValidationX = scaler.transform(X_test)
predictions=model.predict(rescaledValidationX)
print(accuracy_score(Y_test, predictions))
print(confusion_matrix(Y_test, predictions))
print(classification_report(Y_test, predictions))
end = timeit.timeit()
print(end - start )
```

```
0.9090909090909091
[[121   9]
 [ 12  89]]
              precision    recall  f1-score   support

           0       0.91      0.93      0.92       130
           1       0.91      0.88      0.89       101

    accuracy                           0.91       231
   macro avg       0.91      0.91      0.91       231
weighted avg       0.91      0.91      0.91       231

-0.0018764999695122242
```

In [307]:
```
model=GradientBoostingClassifier()
model.fit(scaledX,Y_train)
#estimated the accuracy on validation dataset
rescaledValidationX = scaler.transform(X_test)
predictions=model.predict(rescaledValidationX)
print(accuracy_score(Y_test, predictions))
print(confusion_matrix(Y_test, predictions))
print(classification_report(Y_test, predictions))
end = timeit.timeit()
print(end - start )
```

```
1.0
[[130   0]
 [  0 101]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       130
           1       1.00      1.00      1.00       101

    accuracy                           1.00       231
   macro avg       1.00      1.00      1.00       231
weighted avg       1.00      1.00      1.00       231

-0.003811099799349904
```

In [ ]:

```
from sklearn import metrics
#set up plotting area
plt.figure(0).clf()

#fit logistic regression model and plot ROC curve
model = LogisticRegression()
model.fit(X_train, Y_train)
y_pred = model.predict_proba(X_test)[:, 1]
fpr, tpr, _ = metrics.roc_curve(Y_test, y_pred)
auc = round(metrics.roc_auc_score(Y_test, y_pred), 4)
plt.plot(fpr,tpr,label="Logistic Regression, AUC="+str(auc))

#fit gradient boosted model and plot ROC curve
model = GradientBoostingClassifier()
model.fit(X_train, Y_train)
y_pred = model.predict_proba(X_test)[:, 1]
fpr, tpr, _ = metrics.roc_curve(Y_test, y_pred)
auc = round(metrics.roc_auc_score(Y_test, y_pred), 4)
plt.plot(fpr,tpr,label="Gradient Boosting, AUC="+str(auc))

#fit random forest classifier model and plot ROC curve
model = RandomForestClassifier()
model.fit(X_train, Y_train)
y_pred = model.predict_proba(X_test)[:, 1]
fpr, tpr, _ = metrics.roc_curve(Y_test, y_pred)
auc = round(metrics.roc_auc_score(Y_test, y_pred), 4)
plt.plot(fpr,tpr,label="RandomForestClassifier, AUC="+str(auc))

#fit random forest classifier model and plot ROC curve
model = DecisionTreeClassifier()
model.fit(X_train, Y_train)
y_pred = model.predict_proba(X_test)[:, 1]
fpr, tpr, _ = metrics.roc_curve(Y_test, y_pred)
auc = round(metrics.roc_auc_score(Y_test, y_pred), 4)
plt.plot(fpr,tpr,label="DecisionTreeClassifier, AUC="+str(auc))


#add legend
plt.legend()
```

```
plt.legend()
```

Out[313]: <matplotlib.legend.Legend at 0x1eec8a86260>