PROJECT RAVI

A Machine Learning Framework to Predict Onset of Diabetes

MIGUL JAIN

Class XI

Vels Vidyashram High School, Chennai

Under the guidance of:

Manish Sharma, DIGITSUTRA

NOVEMBER 1, 2022

TABLE OF CONTENT:

ABSTRACT:

Diabetes is a common chronic disease that many people suffer from every year. Anything that leads to high sugar level in the body – such as unhealthy lifestyle, consumption of fast foods, or genetic differences – can be a cause for diabetes (Both Type 1 and 2). The consequences for ignoring the symptoms can be very extreme.

However, with the help of Machine Learning, we can now create an algorithm to predict the onset of diabetes using parameters whose relation we could not understand

The aim of this project is to find out and apply the most accurate method for the prediction of diabetes. The methods used in the code are:

1. Logistic Regression
2. Random Forest Classifier
3. SVM
4. Naïve Bayes' Classifier

INSPIRATION:

Three years ago, my grandfather passed away. Diabetes played a major role in his death. It was a wake-up call for me. I realized how chronic diseases like diabetes are initially ignored as some small problem, and then becomes too big for us to handle. It gave me the idea to come up with some sort of model so that using it, people can accurately predict if they will have diabetes or not. Knowing this can save many lives just like my grandfathers

CHAPTER 1:

INTRODUCTION:

Machine learning is a subset of artificial intelligence that empowers systems to learn from data without explicit programming. Using algorithms and statistical models, it helps computers to recognize patterns, make predictions, and improve their performance over time

The term Machine Learning was first coined in 1959 by an IBM employee, Arthur Samuel. Its first use was to recognize and filter sonar signals from the invalid ones. However, it has advanced quite a bit since then. Now, we use Machine Learning to create self-driving cars.

There are three major types of Machine Learning. They are as follows:

a. Supervised Machine Learning
b. Unsupervised Machine Learning
c. Reinforced Machine Learning

In this project, we use the concept of Supervised Machine Learning for our purposes.

OBJECTIVE:

In this project we have tried to predict diabetes using machine learning. We have used Random Forest Classifier, Logistic Regression, Support Vector Machine and Naïve Bayes' Classifier to predict the same.

The errors encountered in this project are:

1. False Negative: When a patient is diabetic, but the algorithm predicts that it is not.
2. False Positive: When a patient is not diabetic, but the algorithm predicts otherwise.

A main goal in this project is also to try out different methods of classification to improve upon the accuracy of our model.

SCOPE OF STUDY:

In this project various classification methods have been used. They are as follows:

- Logistic Regression: It is a statistical model used for classification and prediction analysis. Often, it is used for binary classification: True/False, 0/1.

- Random Forest Classifier: It contains several decision trees on various subsets of the given dataset. Then it takes the average in order to improve the predictive accuracy on that dataset.

- Support Vector Machine: Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes.

Confusion matrix: It does the performance of a machine learning model on a set of test data. It is generally used to measure the performance of classification model

CHAPTER 2:

LITERATURE SURVEY:

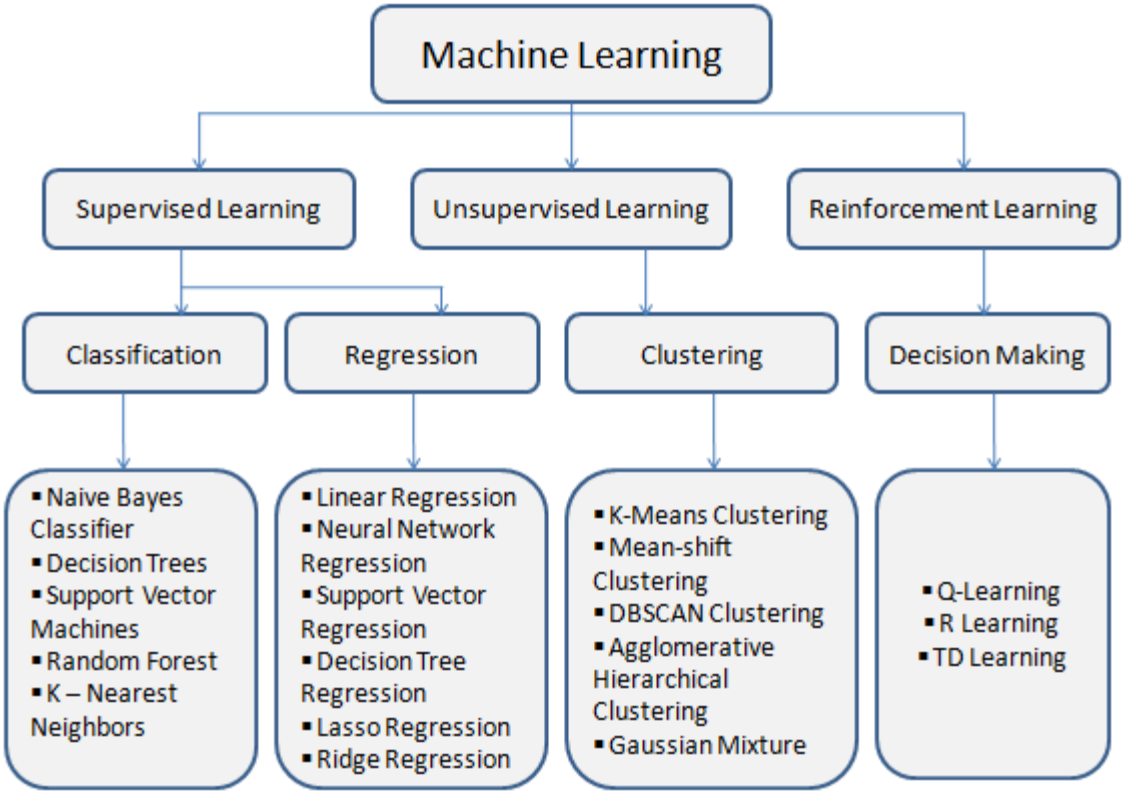
Machine Learning is broadly classified into three categories:

1. Supervised Learning: It is defined by its use of labeled datasets to train algorithms that to classify data or predict outcomes accurately. Simply put, it uses the labelled data received and creates a relation from it in the form of an equation using both the dependent and independent variables. This equation is then used to predict outputs for new data. Supervised learning helps organizations solve for a variety of real-world problems at scale, such as classifying spam in a separate folder from your inbox.

2. Unsupervised Learning: Unsupervised learning uses machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human

intervention. Its ability to discover similarities and differences in information make it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition.

3. Reinforced Learning: It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation. Reinforcement learning differs from supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it is bound to learn from its experience.



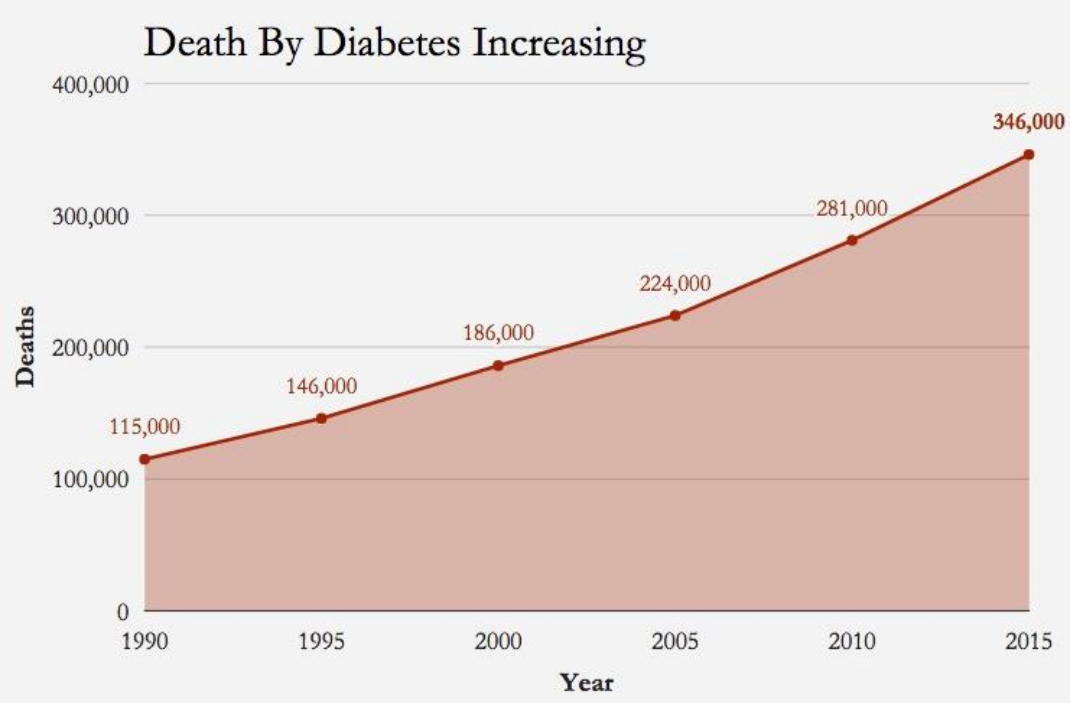
Classification of Machine Learning

CHAPTER 3:

PROBLEM STATEMENT:

Diabetes can depend on a lot of factors. Due to this, it is extremely difficult to find a relation between the input and output parameters for a human. This is where Machine Learning shines

Thanks to Machine Learning, we have been able to apply supervised learning concepts to predict diabetes using 8 common factors.



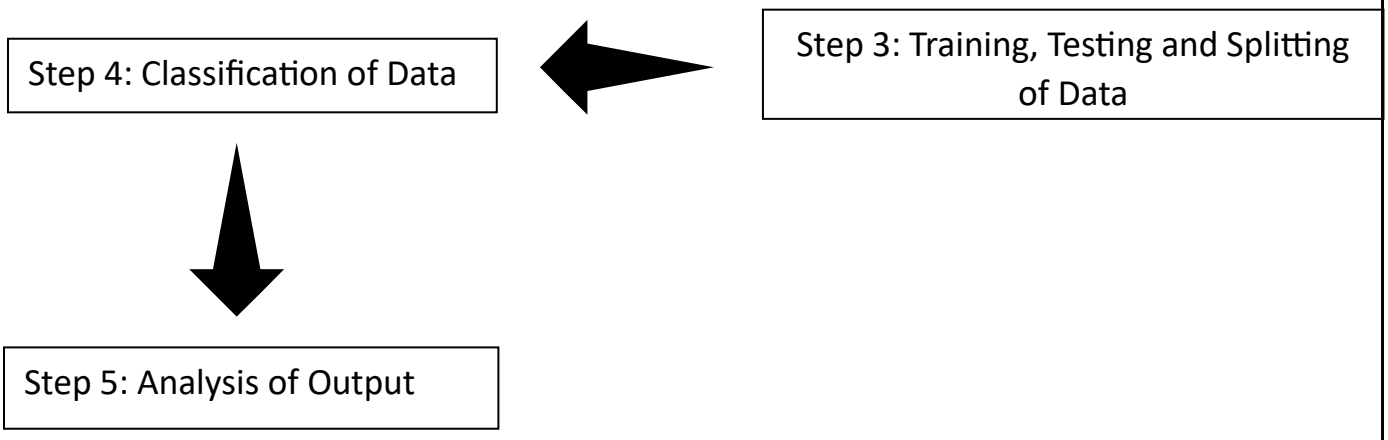
CHAPTER 4:

MACHINE LEARNING PROCESS FLOW:

The model will be created in the following manner:

Step 1:Extraction of PIMA

Step 2: Data Analysis and Refining



ABOUT OUR DATASET:

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 a patient has diabetes, based on certain diagnostic measurements included in the dataset.

From the data set in the (.csv) File We can find several variables, some of them are independent (several medical predictor variables) and only one target

dependent variable (Outcome).

- The dataset consists of 769 data points, with 9 features each.

- For the outcome attribute: 0 means No Diabetes, and 1 means Diabetes

CHAPTER 5:

EXPERIMENTAL SETUP

Step 1: Extraction of PIMA Dataset

```
import pandas as pd
import numpy as np
import sklearn as sk
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
df1=pd.read_csv('diabetes.csv')
```

Step 2: Data Analysis and Refining

```
df1.head(10)
```

(Gives the first 10 rows of the data used)

	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
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

```
df1.shape
```

(Gives the dimensions of the data)

```
(768, 9)
```

In [5]:

df1.columns

(Gives the column labels of the data)

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness','Insulin
','BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

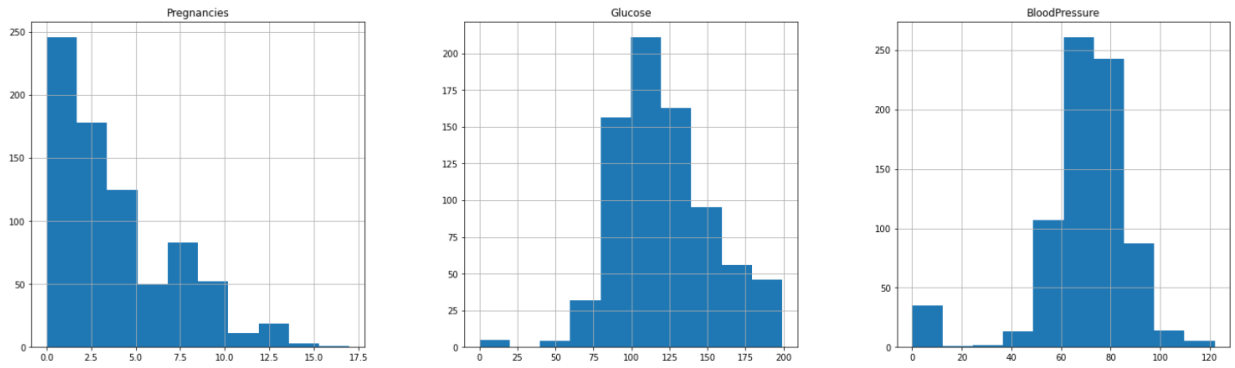
df1.describe()

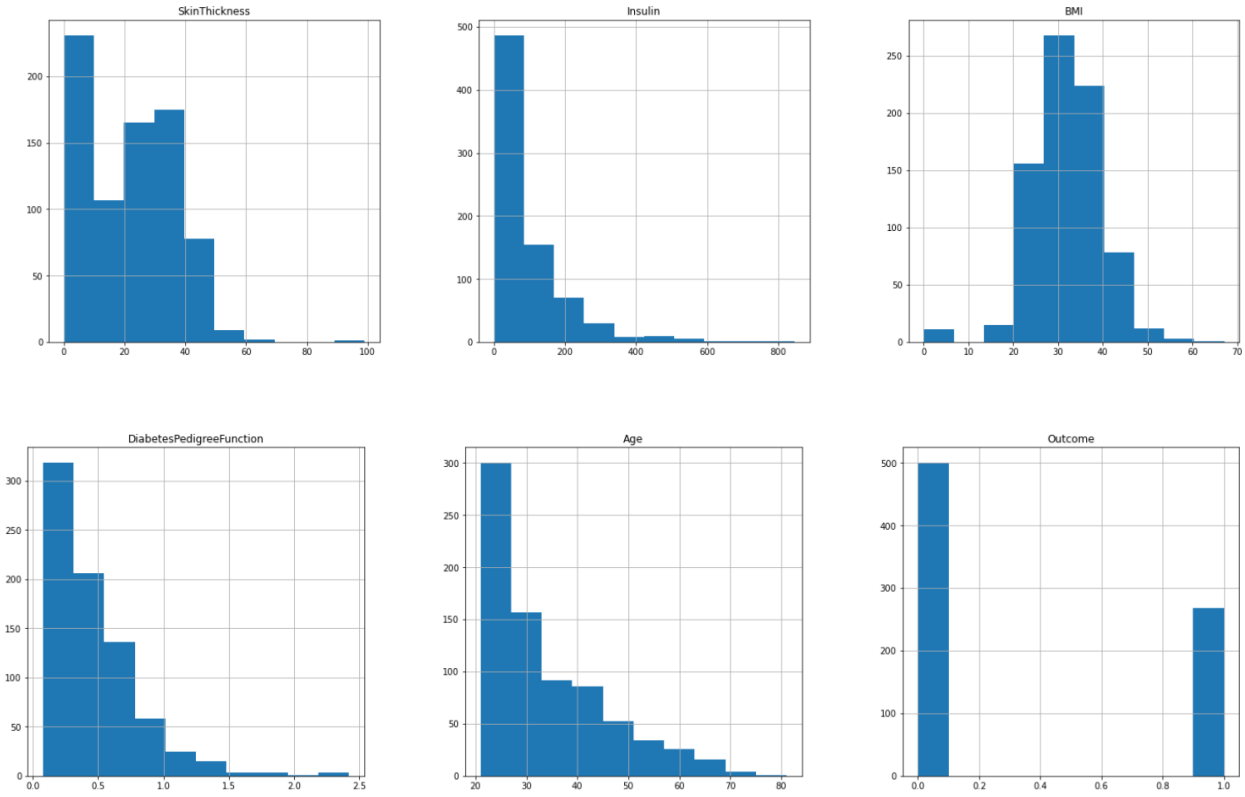
(Gives standard information regarding the overall values for each column)

	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

p=df1.hist(figsize=(25,25))

(Creates a histogram for each column's data)





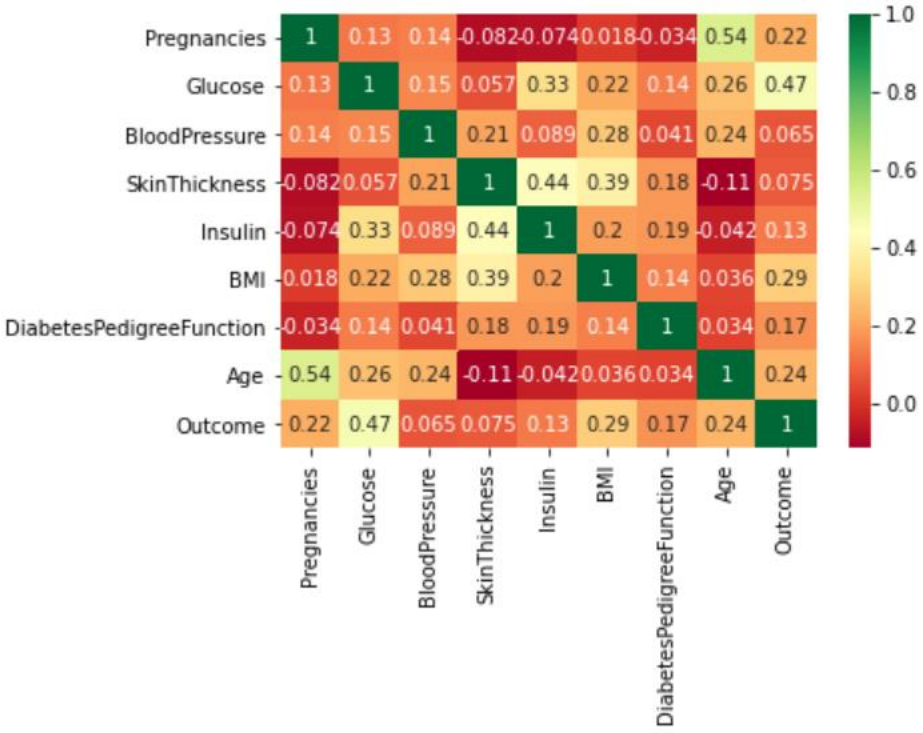
plt.figure(figsize=(15,15))

```
<Figure size 1080x1080 with 0 Axes>
<Figure size 1080x1080 with 0 Axes>
```

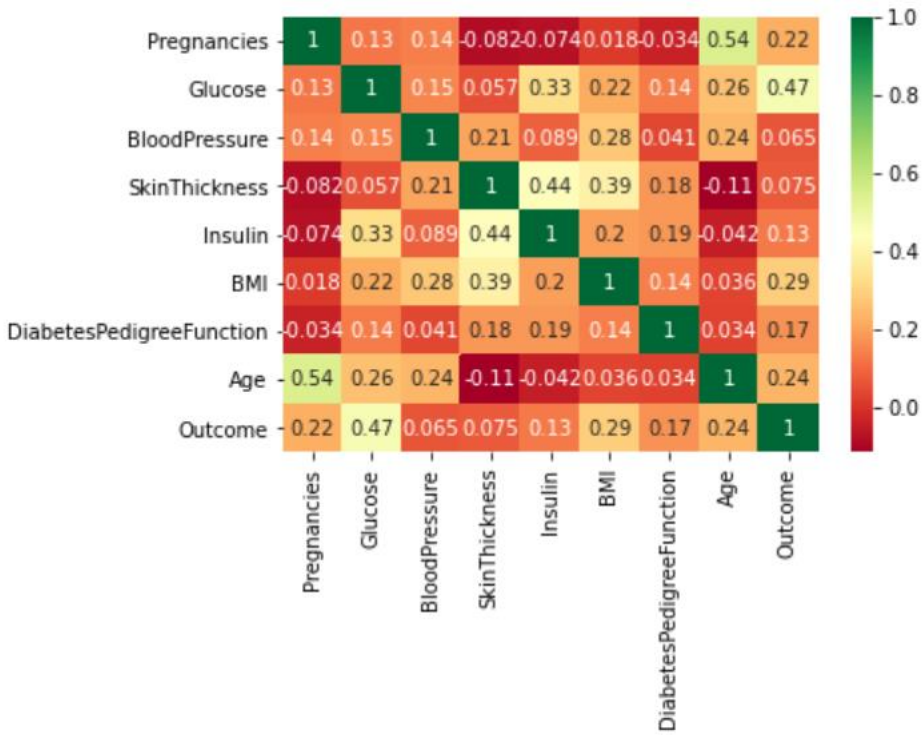
p = sns.heatmap(df1.corr(), annot=True,cmap ='RdYlGn')

(Creates a heatmap to understand the relation of a parameter and the output)

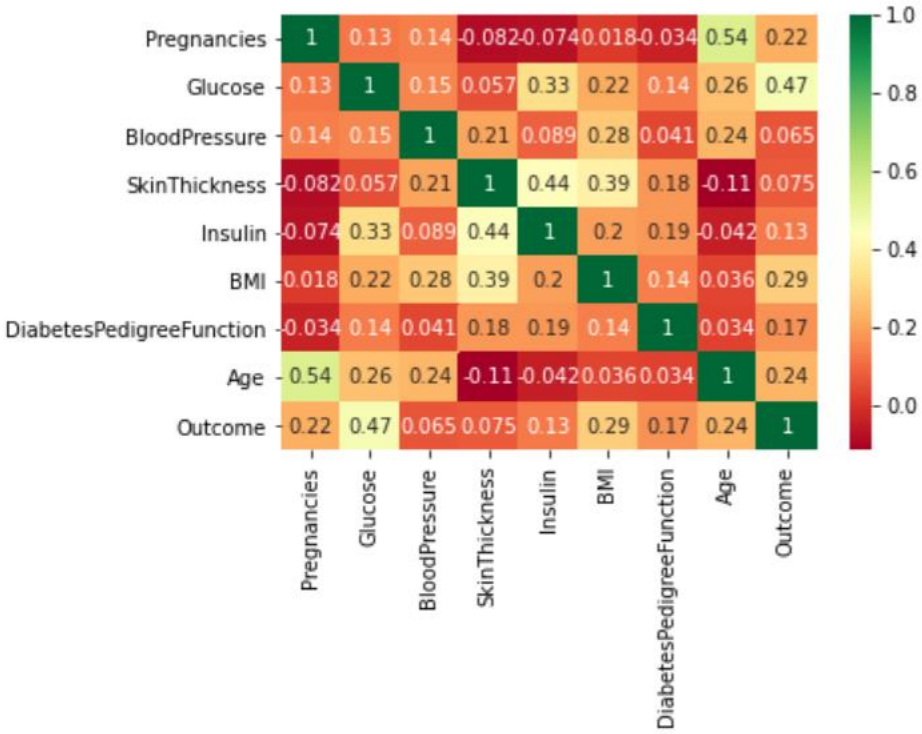
(a darker coloured slot means a higher correlation between the two attributes)




```
p = sns.heatmap(df1.corr(), annot=True,cmap ='RdYlGn')
```



p=sns.heatmap(df1.corr(),annot=True,cmap='RdYlGn')



df1['Outcome']= df1['Outcome'].astype('category')

df1.isnull()

df1.isnull().sum()

(Shows the total number of null values for each parameter)

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

df1.skew()

(Shows how unevenly the data for a parameter is distributed)

```
Pregnancies                 0.901674
Glucose                     0.173754
BloodPressure              -1.843608
SkinThickness               0.109372
Insulin                     2.272251
BMI                        -0.428982
DiabetesPedigreeFunction    1.919911
Age                         1.129597
dtype: float64
```

Step 3: Training, Testing and Splitting of Data

X=df1.drop('Outcome', axis=1)

(Creating a dataframe without the outcomes)

y=df1['Outcome']

(Creating a dataframe with only the outcomes)

X.shape

```
(768, 8)
```

In [17]:

y.shape

```
(768,)
```

In [53]:

```
from sklearn.model_selection import train_test_split

X_train,X_test, y_train, y_test=
train_test_split(X,y,test_size=0.1, random_state=40)
```

(Splitting the data into two parts- one for testing and one for training)

X_train.shape

```
(691, 8)
```

In [55]:

X_test.shape

```
(77, 8)
```

In [56]:

y_train.shape

```
(691,)
```

In [57]:

y_test.shape

```
(77,)
```

In [58]:

```
from sklearn.linear_model import LogisticRegression

lr1=LogisticRegression()

lr1.fit(X_train, y_train)
```

(Using the Logistic Regression concept to create a relation between parameters and the outcome)

```
LogisticRegression()
```

Step 4: Classification of Data

pred=lr1.predict(X_test)

pred2=lr1.predict(X_train)

print(pred)

(Based on the above relation, predicting the values of the training data)

```
array([1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0], dtype=int64)
```

y_test

```
370    1
388    1
611    1
550    0
232    0
      ..
337    1
688    0
415    1
345    0
737    0
Name: Outcome, Length: 77, dtype: category
Categories (2, int64): [0, 1]
```

Chapter 5: Analysis of Output

LOGISTIC REGRESSION:

```
from sklearn import metrics

accu=metrics.accuracy_score(y_test, pred)

accu2=metrics.accuracy_score(y_train,pred2)

print("Train Data Accuracy: ",accu2)

print("Test Data Accuracy :",accu)
```

(Shows overall accuracy in prediction)

```
Train Data Accuracy:  0.7800289435600579
Test Data Accuracy : 0.7922077922077922
```

```
precision=metrics.precision_score(y_test, pred)

print(precision)
```

```
0.8181818181818182
```

In [64]:

```
recall=metrics.recall_score(y_test, pred)

print(recall)
```

```
0.6
```

```
metrics.confusion_matrix(y_test, pred)
```

```
array([[43,  4],
       [12, 18]], dtype=int64)
```

NAÏVE BAYES MODEL:

```
from sklearn.naive_bayes import GaussianNB
```

```
gnb = GaussianNB()

gnb.fit(X_train, y_train)

Y_pred = gnb.predict(X_test)
```

(Doing the similar prediction process using a different relation – the Naïve Bayes model relation)

```
from sklearn import metrics

print("Gaussian Naive Bayes model accuracy(in %):",
metrics.accuracy_score(y_test, Y_pred)*100)
```

```
Gaussian Naive Bayes model accuracy(in %): 70.12987012987013
```

DECISION TREE:

```
from sklearn.tree import DecisionTreeClassifier


tree = DecisionTreeClassifier(random_state=0)
```

(Creation of the decision tree)

```
tree.fit(X_train, y_train)

print("Accuracy on training set:
{:.3f}".format(tree.score(X_train, y_train)))

print("Accuracy on test set: {:.3f}".format(tree.score(X_test,
y_test)))
```

```
Accuracy on training set: 1.000
Accuracy on test set: 0.675
```

```
tree = DecisionTreeClassifier(max_depth=3, random_state=0)

tree.fit(X_train, y_train)

print("Accuracy on training set:
{:.3f}".format(tree.score(X_train, y_train)))

print("Accuracy on test set: {:.3f}".format(tree.score(X_test,
y_test)))
```

```
Accuracy on training set: 0.771
Accuracy on test set: 0.818
```

RANDOM FOREST CLASSIFIER:

```
from sklearn.ensemble import RandomForestClassifier


rf = RandomForestClassifier(n_estimators=100,
random_state=0)

(Creation of multiple decision trees)

rf.fit(X_train, y_train)

print("Accuracy on training set: {:.3f}".format(rf.score(X_train,
y_train)))

print("Accuracy on test set: {:.3f}".format(rf.score(X_test,
y_test)))
```

```
Accuracy on training set: 1.000
Accuracy on test set: 0.792
```

```
rf1 = RandomForestClassifier(max_depth=3, n_estimators=100,
random_state=0)

rf1.fit(X_train, y_train)

print("Accuracy on training set: {:.3f}".format(rf1.score(X_train,
y_train)))

print("Accuracy on test set: {:.3f}".format(rf1.score(X_test,
y_test)))
```

```
Accuracy on training set: 0.796
Accuracy on test set: 0.740
```

SUPPORT VECTOR MACHINE (SVM):

```
from sklearn.svm import SVC


svc = SVC()

svc.fit(X_train, y_train)
```

(Using Support Vector Machine to make a relation)

```
print("Accuracy on training set: {:.2f}".format(svc.score(X_train,
y_train)))
```

```
print("Accuracy on test set: {:.2f}".format(svc.score(X_test,
y_test)))
```

```
Accuracy on training set: 0.77
Accuracy on test set: 0.79
```

```
from sklearn.preprocessing import MinMaxScaler


scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)


svc = SVC()
svc.fit(X_train_scaled, y_train)


print("Accuracy on training set:
{:.2f}".format(svc.score(X_train_scaled, y_train)))
print("Accuracy on test set:
{:.2f}".format(svc.score(X_test_scaled, y_test)))
```

```
Accuracy on training set: 0.80
Accuracy on test set: 0.64
```

K-NEAREST NEIGHBOURSE:

```
from sklearn.neighbors import KNeighborsClassifier


training_accuracy = []
test_accuracy = []
neighbors_settings = range(1, 11)


for n_neighbors in neighbors_settings:
    knn = KNeighborsClassifier(n_neighbors=n_neighbors)
```

(Using K-Nearest Neighbourse concept to create a relation)

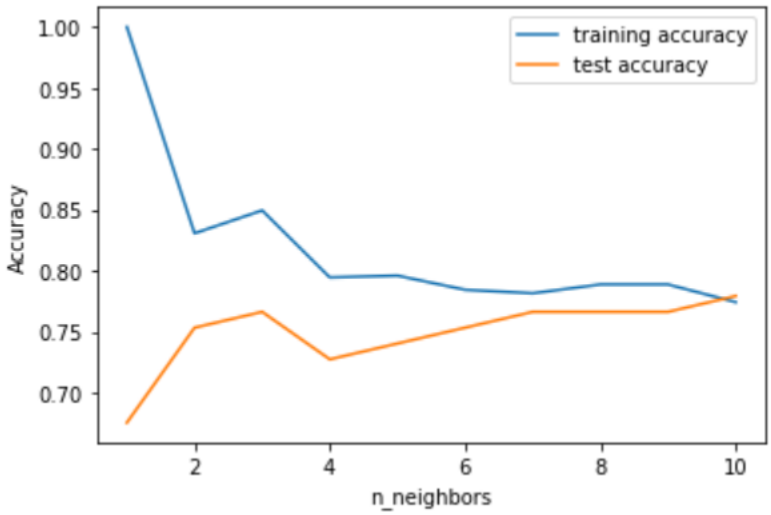
```
(
    knn.fit(X_train, y_train)
    training_accuracy.append(knn.score(X_train, y_train))
    test_accuracy.append(knn.score(X_test, y_test))


plt.plot(neighbors_settings, training_accuracy, label="training
accuracy")
```

(Creating a line graph from the output received)

```
plt.plot(neighbors_settings, test_accuracy, label="test
accuracy")
plt.ylabel("Accuracy")
```

```
plt.xlabel("n_neighbors")
plt.legend()
plt.savefig('knn_compare_model')
```




```
knn = KNeighborsClassifier(n_neighbors=9)
knn.fit(X_train, y_train)

print('Accuracy of K-NN classifier on training set:
{:.2f}'.format(knn.score(X_train, y_train)))
print('Accuracy of K-NN classifier on test set:
{:.2f}'.format(knn.score(X_test, y_test)))
```

```
Accuracy of K-NN classifier on training set: 0.79
Accuracy of K-NN classifier on test set: 0.77
```

RESULT ANALYSIS:

ACCURACY COMPARISION TABLE:

Accuracy= (Number of True Predictions) / (Total Number of Predictions)

Algorithm	Training Accuracy	Test Accuracy
Logistic Regression	78	79.2
Random Forest Classifier	79.6	74
Naïve Bayes Model	70.1	70.1
Decision Tree	77.1	81.8
SVM	77	79
K- Nearest Neighbors	79	77

Thus, we can conclude that for this project and this dataset, Decision Tree (from Testing Accuracy), gives us the most efficiency for the model.

CHAPTER 6:

CONCLUSION:

In this project we learn not only about multiple applicable algorithms, but also how to implement them in any purpose. The prediction of diabetes can be done by various methods such as: Logistic Regression, Decision Tree Classifier, Random

Forest Classifier, Support Vector Machine, etc. Out of all the methods, Decision Tree has given the most accuracy

FUTURE SCOPE:

Now that the model has been successfully created, the future scope of this project would be to convert in into a user-friendly app that anyone can use. The other path the can be taken is to give the application to doctors, since they would know more about the medical condition. The aim here is to help people with the diabetes prediction so that they don't have to suffer its consequences.

BIBLIOGRAPHY:

1. Tejas N. Joshi*, Prof. Pramila M. Chawan**, *M. Tech. student (Department of Computer Engg. and Info. Tech., V.J.T.I., Mumbai, Maharashtra, India. **Associate Professor (Department of Computer Engg. and Info. Tech., V.J.T.I.,

Mumbai, Maharashtra, India. Corresponding Author: Tejas N. Joshi

2. Aishwarya Mujumdar a , V Vaidehi Dr. Vellore Institute of Technology, Chennai, India and Mother Teresa Women's University, Kodaikanal, India