Kingdom of Saudi Arabia
Ministry of Education
King Faisal University
College of Computer Sciences & Information Technology

Diabetes Prediction

A project submitted as a requirement for the course "Artificial Intelligence-CS-411"

By:

Group 3- Section 83

Rawan AlSultan	216021870
Dalal AlBdah	216020137
Ghaliah AlMulhim	216010489
Norah AlShaikhMubarak	216017861

To Laboratory Instructor:

Ms. Muneera Alhajri

Course Instructors

Dr. Noor E Hafsa

Dr. Alaa Sagheer

Submission Date: 16-Dec-19

Project Marking Scheme

Criteria		**Mark contribution**
Document	Justification of Problem	1/
	Justification of methods and solution	2/
	Document organization and grammar	1/
Presentation	Clarity	1/
	Q/A	1/
Application	Code organization and readability	1/
	Solution Implementation	2/
	System report	1/
Total		10/

Table of Contents

Table of Figures

1. Introduction

1.1 Background of the Problem

This report proposes methods on how to predict diabetes. Diabetes is a disease also called blood sugar that happens when a person's blood glucose gets too high. Blood glucose is the main source of energy within a person's body which comes from the food that we eat. Insulin, which is a hormone made by the pancreas, helps in the process of making energy by making the glucose from food get into the cells to be used for energy. Furthermore, there are two types of diabetes. Type 1, which is usually diagnosed in children, in this type, the body stops making insulin because the immune system attacks and destroys the cells in the pancreas that make insulin and for survival, the patient needs to take insulin every day to stay alive. Type 2, which is the most common type of diabetes and usually diagnosed in middle-aged and older people, in this type, the body does not make or use insulin very well. Perdition of diabetes is considered very important because diabetes can be effectively managed when caught early. Otherwise, if it is left untreated within the early stages, it could lead to potential consequences that include heart disease, eye problems, stroke, and kidney damage, etc. In this project we have focused on diabetes predation of patients diagnosed with type 2.

1.2 Brief Literature Review

Many types of research were made concerning the prediction of diabetes. For instance, A model for early prediction of diabetes [1] this research used Artificial neural networks (ANN), random forest (RF) and K-means clustering to implement the prediction of diabetes based on medical attributes. Their results showed that the ANN technique provided the highest accuracy of 75.7%. Diabetes disease prediction using data mining [2] this research used KNN and the Naïve Bayes algorithms to implement the prediction of diabetes. They implemented as a smart system that can be used by the patient to determine if the patient is diabetic or not. Impact of Different Data Types on Classifier Performance [3] used the k-nearest neighbor (KNN), random forest and Naïve Bayesian algorithms to predict diabetes and for evaluating the models they have used K-fold cross-validation technique.

1.3 Brief Description about the Proposed Solution

To tackle this problem, we have applied machine learning techniques as a solution to this problem. The dataset was retrieved from the National Institute of Diabetes and Digestive and Kidney Diseases. We started by visualizing the data using a bar plot, histogram, heatmap correlation. After that, we preprocessed the data which included data normalization, and ended up splitting the data into training and testing sets. We applied more than one machine learning algorithms which are: Logistic Regression, Decision Tree Classifier, Random Forest Classifier, K-Nearest Neighbors (KNN) Classifier, Artificial Neural Network (ANN) classifier, and Support Vector Machine (SVM) classifier that used for classification problems. Furthermore, we applied the hold-out cross validation on training and testing data, and model evaluation to test the model's accuracy using a confusion matrix and finally, we compared the predicted results with the actual ones.

1.4 Description of Report's Content

This report is divided into five sections. Following up with the second section which includes the problem statement where we explain the problem in detail. Third section is the proposed methodology where we explain the process of work in detail. Forth section is the results and discussion of the work and ending with the conclusion to summarize the work.

2. Problem Statement

Diabetes has become a common disease to mankind for young and old people. The growth of the diabetic is increasing day-by-day because of some of the various causes, viral infection, chemical contents mix with the food, autoimmune reaction, bad diet, change in lifestyles, etc. Hence, prediction of diabetes is very important to save human life from this disease and let them know at the early stages to take the required medicines. The problem that hospitals face is that it's difficult for doctors to predict whether the patient is diabetic or not.

We focused on this project to study all the diagnoses in order to classify the output whether the patient has diabetes or not by applying the following procedure:

- Train several models on training data
- Applying hold out cross validation
- Test the models on testing data
- Choose a model with a highest accuracy to test testing data and new data

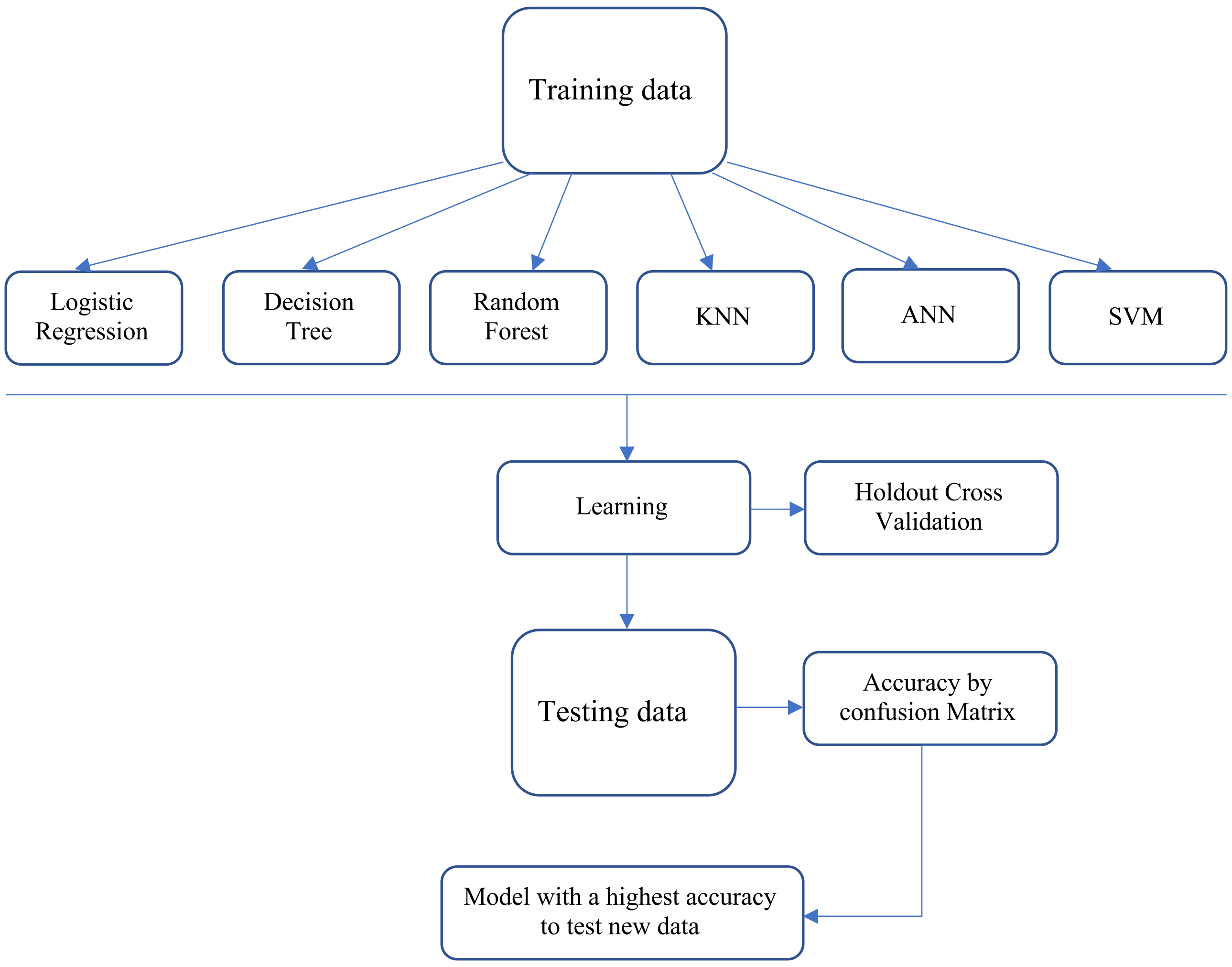


Figure 2.1 Problem Statement

3. Methodology

Python is used to do all the process, training, and testing for diabetes dataset. All the processes are done using Python 3 Google Colab.

3.1 Data Exploration

The dataset was retrieved from the National Institute of Diabetes and Digestive and Kidney Diseases.

The target attribute:

The target attribute is (Outcome) which indicates whether the patient has diabetes (1) or not (0).

The regular attributes:

There are 8 attributes which are

- Pregnancies: Number pregnancies
- Glucose: Plasma glucose concentration over 2 hours in an oral glucose tolerance test
- BloodPressure: Diastolic blood pressure (mm Hg)
- SkinThickness: Triceps skin fold thickness (mm)
- Insulin: 2-Hour serum insulin (mu U/ml)
- BMI: Body mass index (weight in kg/(height in m)2)
- DiabetesPedigreeFunction: Diabetes pedigree function (a function which scores the likelihood of diabetes based on family history)
- Age: Age (years)

Interesting in the descriptive:

- There are 768 rows and 9 columns.
- There are 500 patients do not have diabetes(0), and 268 patients have diabetes.
- The datatypes are int64 and float64.
- The most attributes that affect the result of the outcome is the Glucose, then BMI, then Age, then the number of pregnancies.

3.2 Data Visualization

Bar Plot:

In describing the dataset, we count the number of diabetic patients and the non-diabetic patients. Therefore, function "count" from seaborn library is used to count the numbers, and we visualized it with bar plot as in Figure 4.1, there are 500 patients that are non- diabetic, and 268 patients who are diabetic.

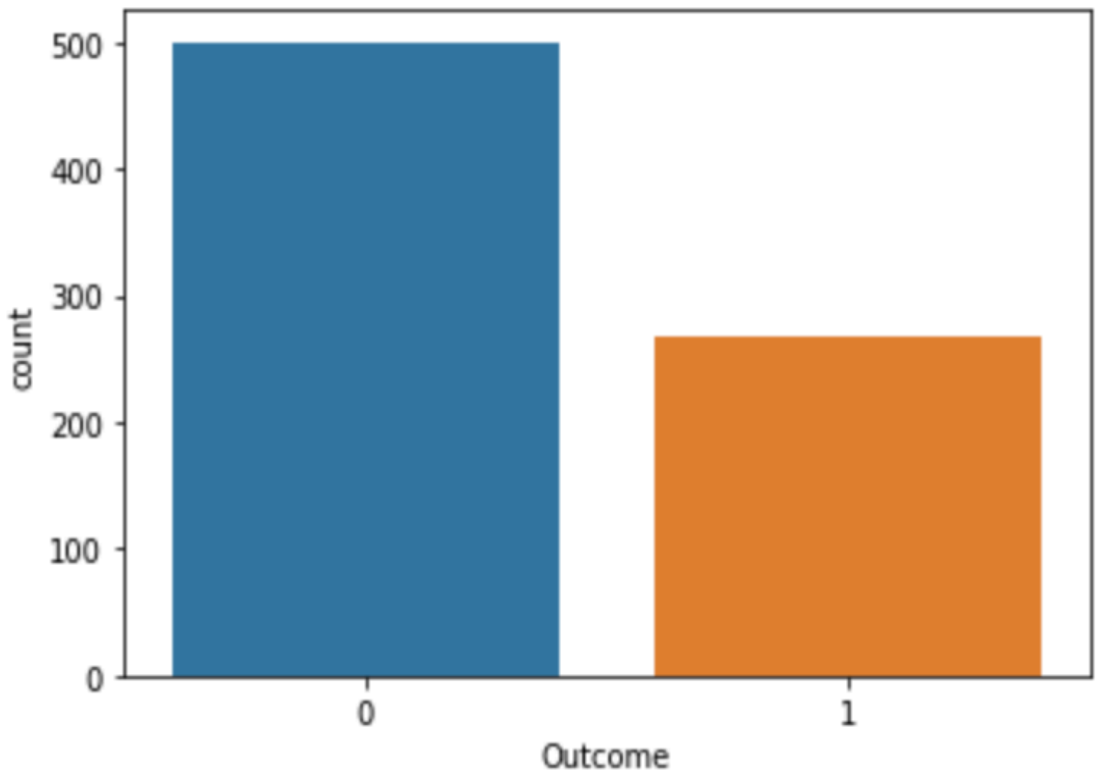


Figure 3.2 Visualization with Bar Plot

Histogram:

Histogram is used to understand and find the data distribution of the features as shown in Figure 3.3 for (0) outcome that the patient does not have diabetes, and Figure 3.4 for (1) outcome that the patient has diabetes.

The result as following:

For the Age attribute, the highest range of age for patients who do not have diabetes is between 20 – 30 years old as shown in Figure 3.3, on the other hand, the highest range of age for patients who have diabetes is between 20 – 55 years old as in Figure 3.4.

For the Glucose attribute, the highest range of glucose for patients who do not have diabetes is between 90 – 120 as in Figure 3.3, on the other hand, the highest range of glucose for patients who have diabetes are between 100 – 200 as in Figure 3.4.

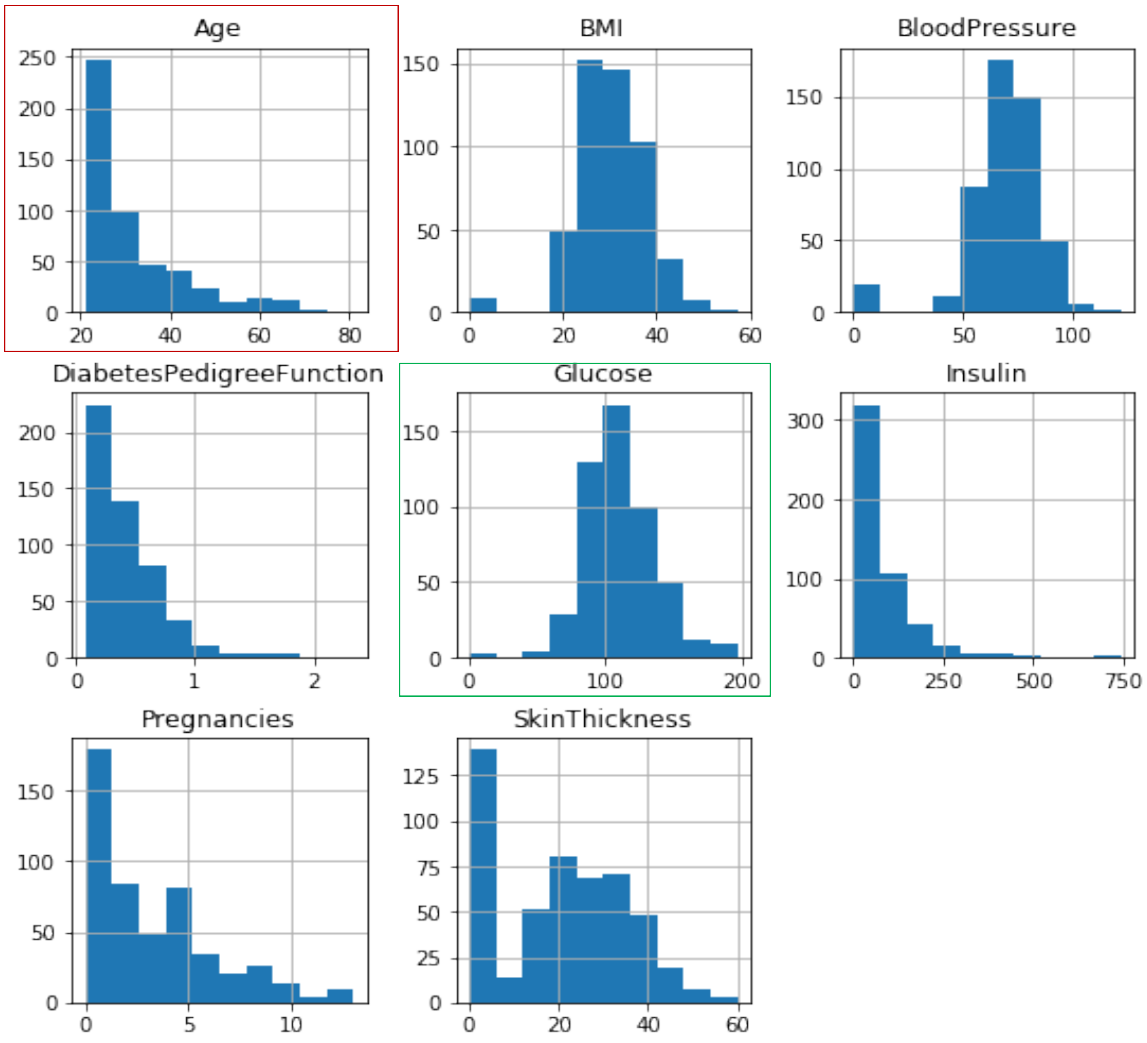


Figure 3.3 Visualization with Histogram (0) Outcome

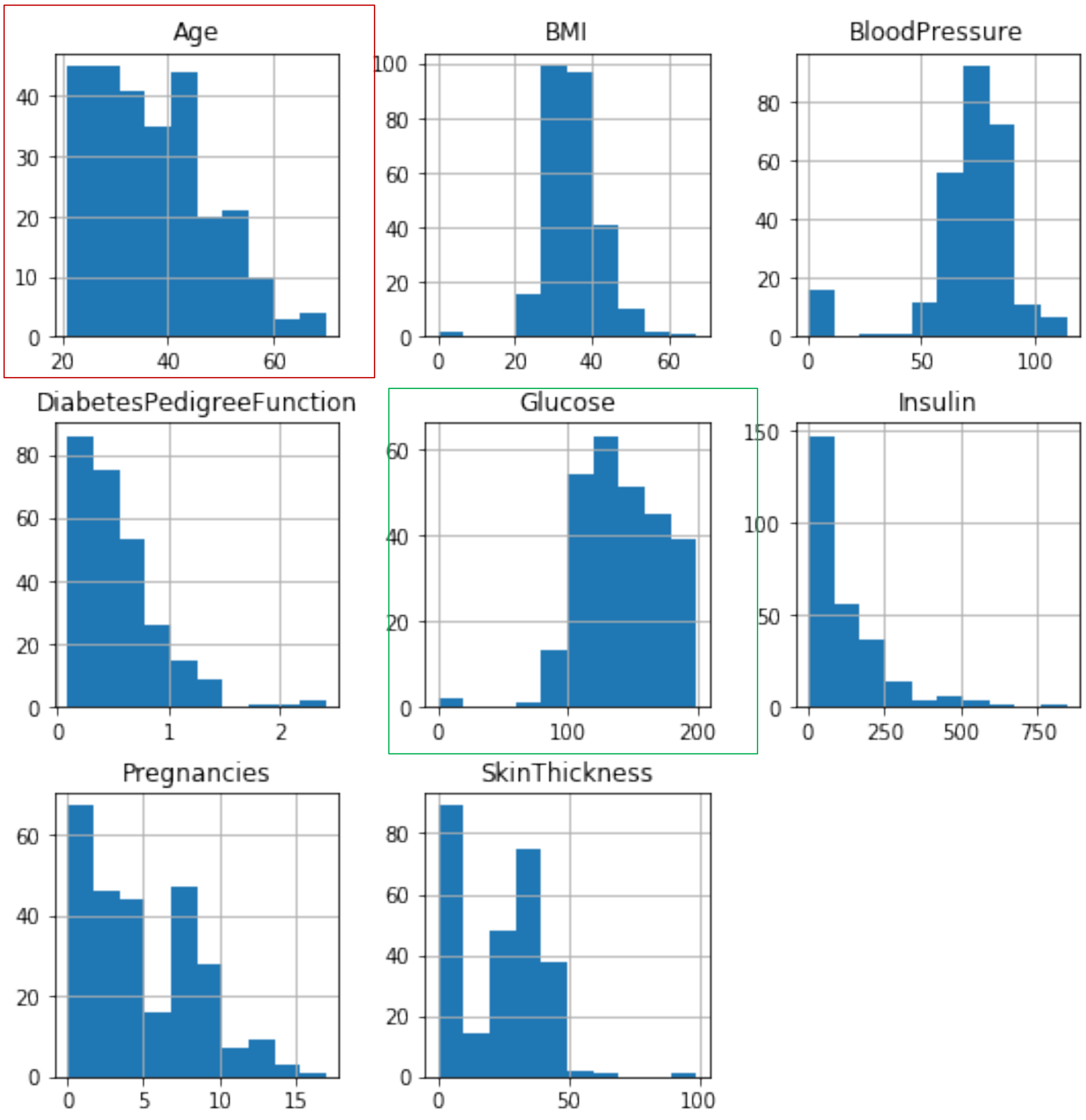
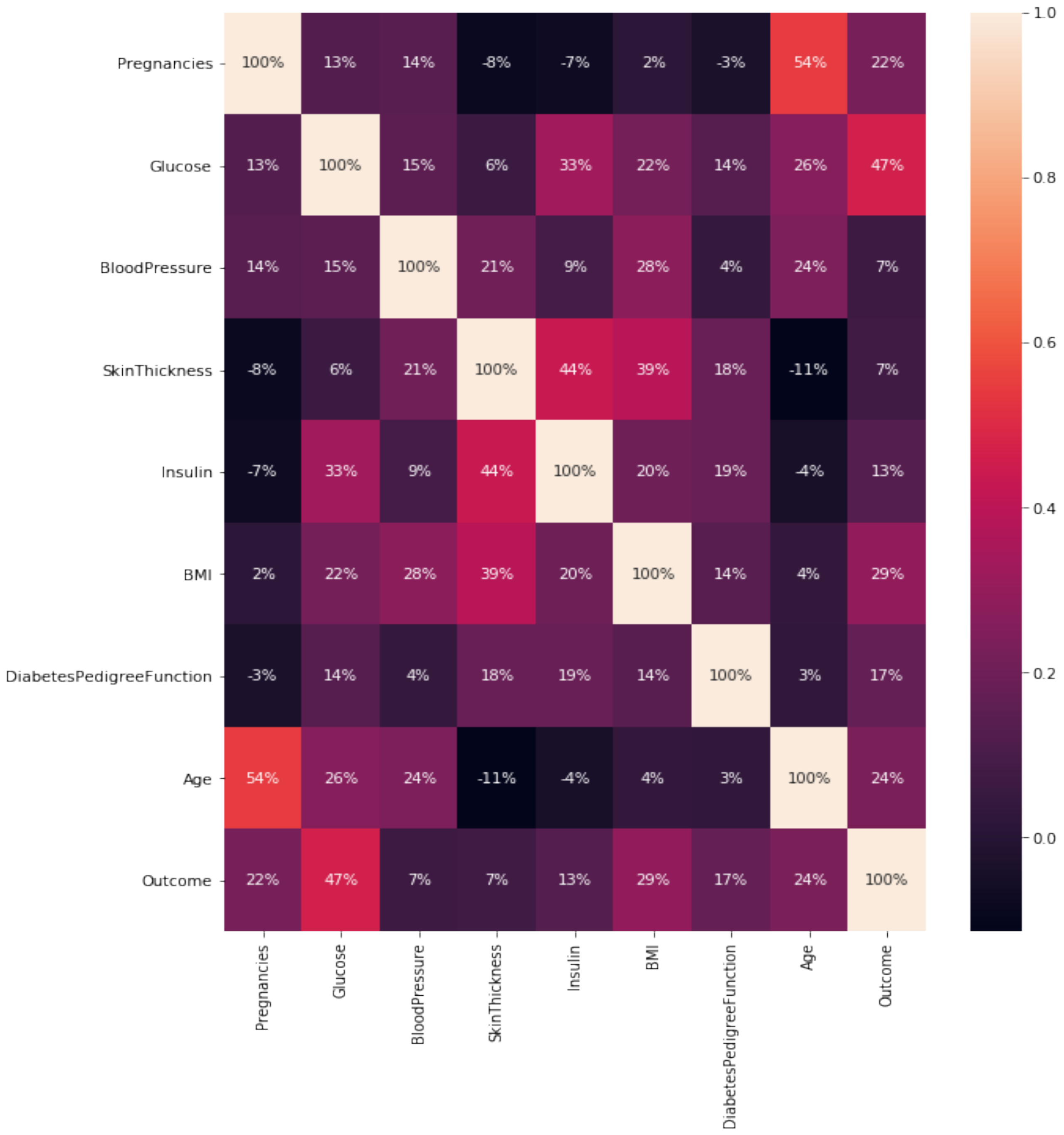


Figure 3.4 Visualization with Histogram (1) Outcome

Heatmap Correlation

In order to know which attributes affect the outcome, correlation is used to visualize the data with correlation numbers between every two attributes as shown in figure 3.5.

The results are: the most attributes that affect the result of outcome are Glucose with 47%, then BMI with 29%, then Age with 24%, then the number of pregnancies with 22%.



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	100%	13%	14%	-8%	-7%	2%	-3%	54%	22%
Glucose	13%	100%	15%	6%	33%	22%	14%	26%	47%
BloodPressure	14%	15%	100%	21%	9%	28%	4%	24%	7%
SkinThickness	-8%	6%	21%	100%	44%	39%	18%	-11%	7%
Insulin	-7%	33%	9%	44%	100%	20%	19%	-4%	13%
BMI	2%	22%	28%	39%	20%	100%	14%	4%	29%
DiabetesPedigreeFunction	-3%	14%	4%	18%	19%	14%	100%	3%	17%
Age	54%	26%	24%	-11%	-4%	4%	3%	100%	24%
Outcome	22%	47%	7%	7%	13%	29%	17%	24%	100%

Figure 3.5 Visualization with Heatmap Correlation

3.3 Data Preparation

- There is no need for data reduction because the data set has all the required and important attributes that help to predict the outcome and the data is clean, there is no missing value or outliers, and there are no Null values.
- Normalization is performed by scaling the data by using StandardScaler() from sklearn.preprocessing library. StandardScaler is a function that transforms the data in the dataset such that each value will have the mean value of the column subtracted, and then divide it by the standard deviation. StandardScaler is very useful in making the data normally distributed within every feature and it scales them such that the distribution is centered around 0, and a standard deviation of 1.
- We split the data set into 75% training and 25% testing by using train_test_split from sklearn.model_selection

3.4 Model Selection

6 types of models are applied to classify the data which are:

Logistic Regression

We applied this model because it is a Machine Learning algorithm used for classification problems, which is a predictive analysis algorithm based on the concept of probability. We expect our model to give us a set of outputs based on probability when we pass the inputs and returns a probability score between 0 and 1.

Decision Tree

We applied this model because it is a type of supervised learning approach for classification. The main advantage of decision trees is that they can handle categorical YES/NO as in our project.

Random Forest

We applied this model because it is one of the most accurate learning algorithms and it runs efficiently on large databases.

K-Nearest neighbor (KNN)

We applied this model because it is very simple in implementation and classes do not have to be linearly separable.

Artificial Neural Network (ANN)

We applied this model because of its ability to detect complex nonlinear relationships between dependent and independent variables and the ability to detect all possible interactions between predictor variables.

Support Vector Machine (SVM)

We applied this model because it gives high accuracy in terms of binary classification.

Cross Validation:

Hold out validation is done for all models to know the evaluation of each model by finding the score. The following lines of code show the process of validation.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test
_size = 0.25, random_state = 0)
```

```
  #Logistic Regression
  from sklearn.linear_model import LogisticRegression
  log = LogisticRegression(random_state = 0)
  log.fit(x_train, y_train)


  #Decision Tree
  from sklearn.tree import DecisionTreeClassifier
  tree = DecisionTreeClassifier(criterion = 'entropy', random_sta
te = 0)
  tree.fit(x_train, y_train)


  #Random Forest
  from sklearn.ensemble import RandomForestClassifier
  forest = RandomForestClassifier(n_estimators = 10, criterion =
'entropy', random_state = 0)
  forest.fit(x_train, y_train)


  #KNN
  from sklearn.neighbors import KNeighborsClassifier
  knn = KNeighborsClassifier()
  knn.fit(x_train, y_train)
```

```
#SVM
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(x_train, y_train)


#ANN
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(10, 10, 10),
max_iter=1000)
mlp.fit(x_train, y_train)
```

In the following lines of code, each print displays the score of each training module.

```
  print('[0]Logistic Regression Training Accuracy: ', log.score(x
_train, y_train))
  print('[1]Decision Tree Classifier Training Accuracy: ', tree.s
core(x_train, y_train))
  print('[2]Random Forest Classifier Training Accuracy: ', forest
.score(x_train, y_train))
  print('[3]KNN Training Accuracy: ', knn.score(x_train, y_train)
)
  print('[4]SVM Training Accuracy: ', svclassifier.score(x_train,
 y_train))
  print('[5]ANN Training Accuracy: ', mlp.score(x_train, y_train)
)
```

The accuracy for each model in training phase are as following:

```
[0]Logistic Regression Training Accuracy:  0.765625
[1]Decision Tree Classifier Training Accuracy:  1.0
[2]Random Forest Classifier Training Accuracy:  0.984375
[3]KNN Training Accuracy:  0.8055555555555556
[4]SVM Training Accuracy:  0.7638888888888888
[5]ANN Training Accuracy:  0.8958333333333334
```

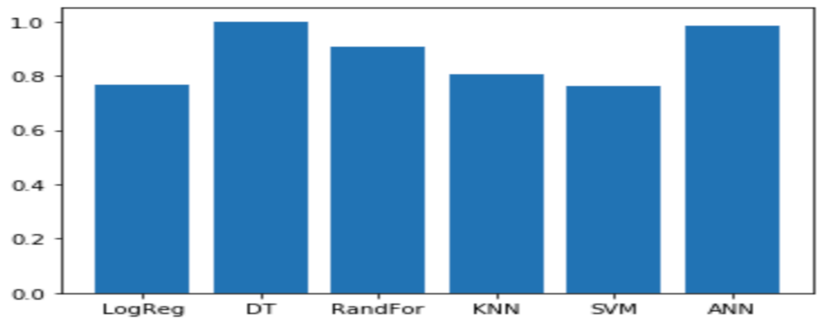


Figure 3.6 Training Accuracy

Decision Tree has the highest accuracy with 1.0

4. Results and Discussion

We test the model accuracy on test data by applying the confusion matrix as following:

'Testing Accuracy = ', (TP + TN)/ (TP + TN + FN + FP))

We applied the confusion matrix for all models and the accuracies are as following:

```
 Model  0 'Logistic Regression'
[[115  15]
 [ 25  37]]
Testing Accuracy =  0.7916666666666666

Model  1 'Decision Tree'
[[103  27]
 [ 19  43]]
Testing Accuracy =  0.7604166666666666

Model  2 'Random Forest'
[[109  21]
 [ 28  34]]
Testing Accuracy =  0.7447916666666666

Model  3 'K-Nearest neighbor (KNN)'
[[113  17]
 [ 22  40]]
Testing Accuracy =  0.796875

Model  4 'Support Vector Machine (SVM)'
[[117  13]
 [ 24  38]]
Testing Accuracy =  0.8072916666666666

Model  5 'Artificial Neural Network (ANN)'
[[107  23]
 [ 31  31]]
Testing Accuracy =  0.71875
```

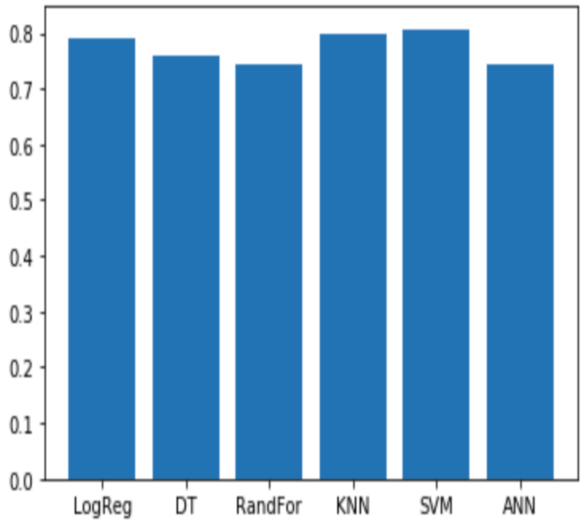


Figure 4.7 Testing Accuracy

Therefore, the model with the highest accuracy is Support Vector Machine (SVM) with 80% because SVM is used for the binary classification either 0 or 1 in our problem.

To ensure that the testing accuracy are all correct for all models, we used classification report and accuracy score libraries to calculate the test accuracy for each model as following, and finally, we conclude that **SVM** has the highest accuracy in testing data.

'Logistic Regression'

```
Model  0
              precision    recall  f1-score   support

           0       0.82      0.88      0.85       130
           1       0.71      0.60      0.65        62

    accuracy                           0.79       192
   macro avg       0.77      0.74      0.75       192
weighted avg       0.79      0.79      0.79       192

0.7916666666666666
```

'Decision Tree'

```
Model  1
              precision    recall  f1-score   support

           0       0.84      0.79      0.82       130
           1       0.61      0.69      0.65        62

    accuracy                           0.76       192
   macro avg       0.73      0.74      0.73       192
weighted avg       0.77      0.76      0.76       192

0.7604166666666666
```

'Random Forest'

```
Model  2
              precision    recall  f1-score   support

           0       0.80      0.84      0.82       130
           1       0.62      0.55      0.58        62

    accuracy                           0.74       192
   macro avg       0.71      0.69      0.70       192
weighted avg       0.74      0.74      0.74       192

0.7447916666666666
```

‘K-Nearest neighbor (KNN)’

```
Model  3
              precision    recall  f1-score   support

           0       0.84      0.87      0.85       130
           1       0.70      0.65      0.67        62

    accuracy                           0.80       192
   macro avg       0.77      0.76      0.76       192
weighted avg       0.79      0.80      0.79       192

0.796875
```

‘Support Vector Machine (SVM)’

```
Model  4
              precision    recall  f1-score   support

           0       0.83      0.90      0.86       130
           1       0.75      0.61      0.67        62

    accuracy                           0.81       192
   macro avg       0.79      0.76      0.77       192
weighted avg       0.80      0.81      0.80       192

0.8072916666666666
```

‘Artificial Neural Network (ANN)’

```
Model  5
              precision    recall  f1-score   support

           0       0.80      0.84      0.82       130
           1       0.62      0.55      0.58        62

    accuracy                           0.74       192
   macro avg       0.71      0.69      0.70       192
weighted avg       0.74      0.74      0.74       192

0.7447916666666666
```

Therefore, we applied the SVM model on the test data because it has the highest accuracy and the results are shown below:

The predicted result by SVM model

```
[1 0 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
 0 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 1 1 1 0 0 0 0 0 0 1
 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 0
 0 1 0 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0
 0 0 0 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1
 0 1 0 0 0 0 0]
```

The actual result

```
[1 0 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 1 0 0 0 0 1 1 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 0 0 1 0
 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 0 1 0 0 0
 0 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 1 1
 0 1 1 1 0 0 0]
```

The model could predict 170 out of 192 correctly as shown in Figure 4.8, 141 do not have diabetes and 51 have diabetes. Actually, the original test data contains 192 samples, 130 do not have diabetes and 62 have diabetes as shown in Figure 4.9.

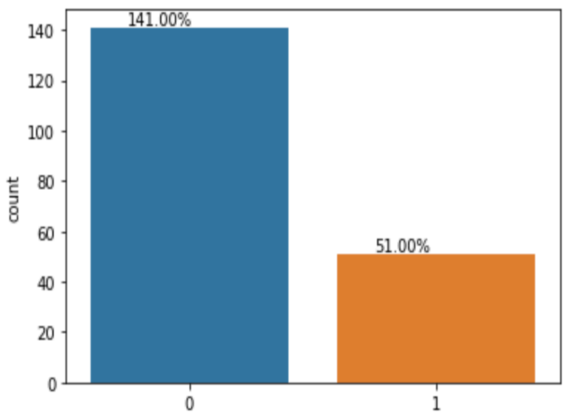


Figure 4.8 The Predicted Outcome

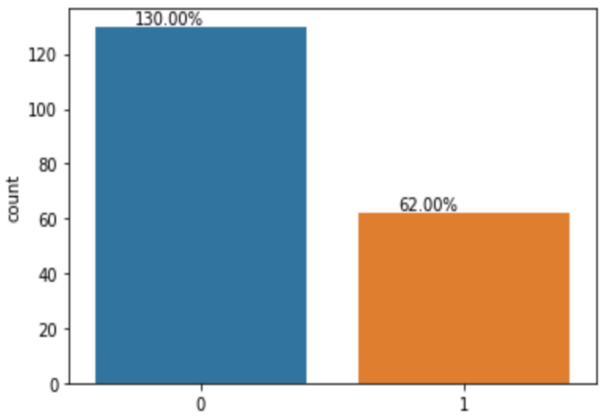


Figure 4.9 The Actual Outcome

After determining and testing the chosen model, we did a console to take new data as an input and predict it by using SVM model. As following, after taking the data, the predicted outcome is [1]

```
Enter the number of Pregnancies: 11
Enter the number of Glucose: 180
Enter the number of BloodPressure: 75
Enter the number of SkinThickness: 40
Enter the number of Insulin: 120
Enter the number of BMI: 35
Enter the number of DiabetesPedigreeFunction: 0.65
Enter the number of Age: 50
[1]
```

5. Conclusion

Diabetes is a disease that happens when a person's blood glucose gets too high. The problem that hospitals face is that it's difficult for doctors to predict whether the patient is diabetic or not. Therefore, the goal of the project is to help doctors to know whether the patient is diabetic or not. Prediction is based on certain diagnostic measurements. Therefore, the project focused on studying all these diagnoses in order to classify the output whether the patient has diabetes (1) or not (0). To find the highest accuracy model, 6 models were applied on the dataset, then we found the accuracy for all of them in order to choose the highest one to be used on the test data to predict new data. The highest model was Support Vector Machine (SVM) with 80% because SVM is used for the binary classification either 0 or 1 in our problem. The project states an important problem because the number of people who have diabetes is increasing gradually which may lead to other problems and diseases if patients do not know if they have diabetes or not. Therefore, predicting diabetes at early stages allows the patients to take the required medications and may prevent some patients from getting diabetes in the future. In addition, this project can be used as one of the doctor’s tools that are used to predict diabetes.

References

[1] Alam, T. M., Iqbal, M. A., Ali, Y., Wahab, A., Ijaz, S., Baig, T. I., … Abbas, Z. (2019, July 9). A model for early prediction of diabetes. Retrieved from https://www.sciencedirect.com/science/article/pii/S2352914819300176#bib18.

[2] D. Shetty, K. Rit, S. Shaikh, N. Patil. Diabetes disease prediction using data mining https://ieeexplore.ieee.org/abstract/document/8276012

[3] A. Singh, M.N. Halgamuge, R. Lakshmiganthan Impact of different data types on classifier performance of random forest, naive Bayes, and K-nearest neighbors algorithms https://minerva-access.unimelb.edu.au/bitstream/handle/11343/216910/2017_Asmita_Different_Data.pdf?sequence=1&isAllowed=y

[4] Google Colaboratory. (n.d.). Retrieved from https://colab.research.google.com/notebooks/welcome.ipynb#recent=true.

[5] Jagadish, K. (2019, May 7). Diabetics prediction using logistic regression. Retrieved from https://www.kaggle.com/kandij/diabetes-dataset#diabetes2.csv.

Appendix: Python Process Flows

```
#Description: This program predicts diabetes based on data.
#import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#Load the data
from google.colab import files
uploaded = files.upload()
df = pd.read_csv('diabetes2.csv')
df.head(7)

#Count the number of rows and columns in the data set
df.shape

#Count the number of empty (NaN, NAN, na) values in each column
df.isna().sum()

#Get a count of the number of False(0) or True(1) cells
df['Outcome'].value_counts()

#Visualize the count
sns.countplot(df['Outcome'], label='count')
#Look at the types to see which columns need to be encoded
df.dtypes

#Create a histogram
df.groupby('Outcome').hist(figsize=(9, 9))

#Get the correlation of the columns
df.iloc[:,0:9].corr()

#Visualize the correlation
plt.figure(figsize=(10,12))
sns.heatmap(df.iloc[:,0:9].corr(), annot=True, fmt='.0%')

#Split the data set into independent (x) and dependent (y) data sets
x = df.iloc[:,0:8].values
y = df.iloc[:,8].values


#Split the data set into 75% training and 25% testing
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0
.25, random_state = 0)
```

```
#Scale the data (Feature Scaling)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)

#Create a function for models
def models(x_train, y_train):

  #Logistic Regression
  from sklearn.linear_model import LogisticRegression
  log = LogisticRegression(random_state = 0)
  log.fit(x_train, y_train)

  #Decision Tree
  from sklearn.tree import DecisionTreeClassifier
  tree = DecisionTreeClassifier(criterion = 'entropy', random_state = 0
)
  tree.fit(x_train, y_train)

  #Random Forest
  from sklearn.ensemble import RandomForestClassifier
  forest = RandomForestClassifier(n_estimators = 10, criterion = 'entro
py', random_state = 0)
  forest.fit(x_train, y_train)

  #KNN
  from sklearn.neighbors import KNeighborsClassifier
  knn = KNeighborsClassifier()
  knn.fit(x_train, y_train)

  #SVM
  from sklearn.svm import SVC
  svclassifier = SVC(kernel='linear')
  svclassifier.fit(x_train, y_train)

  #ANN
  from sklearn.neural_network import MLPClassifier
  mlp = MLPClassifier(hidden_layer_sizes=(10, 10, 10), max_iter=1000)
  mlp.fit(x_train, y_train)
```

```
print('[0]Logistic Regression Training Accuracy: ',
log.score(x_train, y_train))
print('[1]Decision Tree Classifier Training Accuracy: ',
tree.score(x_train, y_train))
print('[2]Random Forest Classifier Training Accuracy: ',
forest.score(x_train, y_train))
print('[3]KNN Training Accuracy: ',
knn.score(x_train, y_train))
print('[4]SVM Training Accuracy: ',
svclassifier.score(x_train, y_train))
print('[5]ANN Training Accuracy: ',
mlp.score(x_train, y_train))

return log, tree, forest, knn, svclassifier, mlp


#Get all the models
model = models(x_train, y_train)

#Visualization of training accuracy
d1= ['LogReg', 'DT', 'RandFor', 'KNN', 'SVM', 'ANN']
d2 = [0.766, 1.0, 0.909, 0.806, 0.764, 0.984]
plt.bar(d1, d2)
plt.show

#Test model accuracy on test data on confusion matrix
from sklearn.metrics import confusion_matrix
for i in range (len(model)):
  print('Model ', i)
  cm = confusion_matrix(y_test, model[i].predict(x_test))
  TP = cm[0][0]
  TN = cm[1][1]
  FN = cm[0][1]
  FP = cm[1][0]
  print(cm)
  print('Testing Accuracy = ', (TP + TN)/ (TP + TN + FN + FP))
  print()

#Visualization of testing accuracy
d1= ['LogReg', 'DT', 'RandFor', 'KNN', 'SVM', 'ANN']
d2 = [0.792, 0.760, 0.745, 0.797, 0.807, 0.745]
plt.bar(d1, d2)
plt.show
```

```
#Show another way to get metrics of the models
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
for i in range (len(model)):
  print('Model ', i)
  print(classification_report(y_test, model[i].predict(x_test)))
  print(accuracy_score(y_test, model[i].predict(x_test)))
  print()

#Print the prediction of SVM Classifier Model
pred = model[4].predict(x_test)
print(pred)
print()
print(y_test)

#The predicted result
p = sns.countplot(pred)
for px in p.patches:
        p.annotate('{:.2f}%'.format(px.get_height()), (px.get_x()+0.15,
 px.get_height()+1))

#The actual result
y = sns.countplot(y_test)
for yt in y.patches:
        y.annotate('{:.2f}%'.format(yt.get_height()), (yt.get_x()+0.15,
 yt.get_height()+1))

#console
v1 = input("Enter the number of Pregnancies: ")
v2 = input("Enter the number of Glucose: ")
v3 = input("Enter the number of BloodPressure: ")
v4 = input("Enter the number of SkinThickness: ")
v5 = input("Enter the number of Insulin: ")
v6 = input("Enter the number of BMI: ")
v7 = input("Enter the number of DiabetesPedigreeFunction: ")
v8 = input("Enter the number of Age: ")

test = [v1, v2, v3, v4, v5, v6, v7, v8]
testArr = np.reshape(test, (1, -1))
testpred = model[4].predict(testArr)
print(testpred)
```