DIABETES DETECTION USING MACHINE LEARNING

Presented by-

VIRAL PARIKH

DIVYANSH SHARMA

YASH PUROHIT

INTRODUCTION

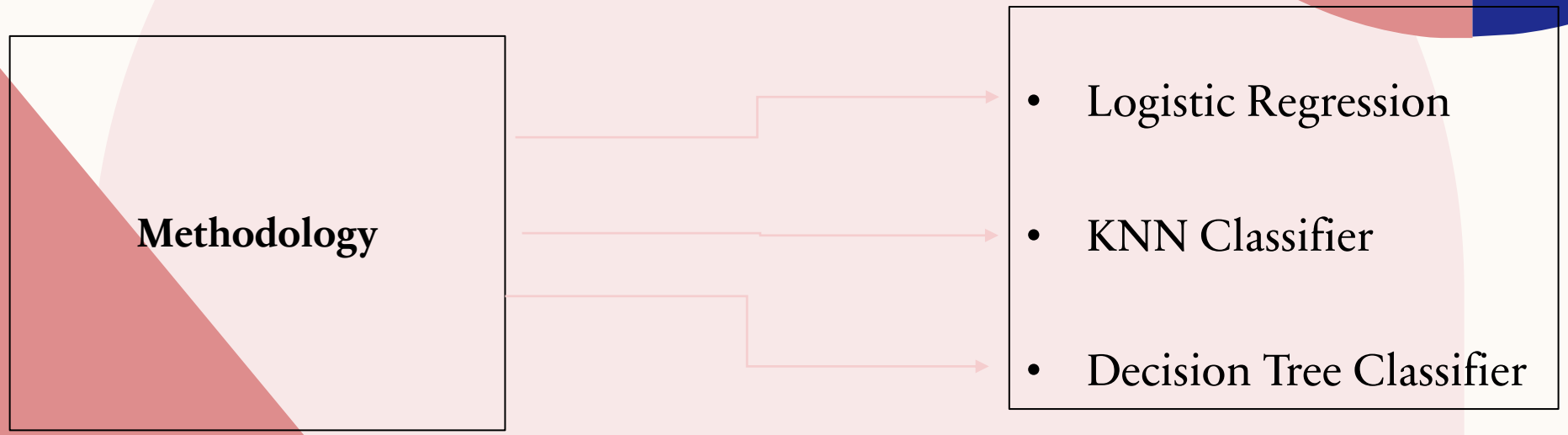
- Diabetes is one of deadliest diseases in the world.
- There were studies handled in predicting diabetes mellitus through physical and chemical tests, are available for diagnosing diabetes.
- Health condition diagnosis is an essential and critical aspect for healthcare professionals. Classification of a diabetes type is one of the most complex phenomena for healthcare professionals and comprises several tests.
- Data Science methods have the potential to benefit other scientific fields by shedding new light on common questions.
- Numerous techniques have been presented in the literature for diabetes classification.
- The symptoms of DM include polyuria, polydipsia, and significant weight loss among others.
- Diagnosis depends on blood glucose levels

OBJECTIVES

- The objective of this study is classified Indian PIMA dataset for diabetes.
- This is proposed to achieve through machine learning models.

METHODOLOGY

We cannot differentiate which of the algorithms are superior or not.
Classification algorithms used are:



LOGISTIC REGRESSION

- It is a supervised learning classification technique that forecasts the likelihood of a target variable.
- There will only be a choice between two classes.
- When the forecast is categorical, such as true or false, yes or no, or a 0 or 1, you can use it.
- A logistic regression technique can be used to determine whether or not an email is a spam.
- Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

LOGISTIC REGRESSION

In [70]:
```
from sklearn.linear_model import LogisticRegression
logisticRegr = LogisticRegression()
logisticRegr.fit(x_train, y_train)
logisticRegr.predict(x_test[0].reshape(1,-1))
predictions = logisticRegr.predict(x_test)
```

In [71]:
```
score = logisticRegr.score(x_test, y_test)
print(score)
```

```
0.7748917748917749
```

In [73]:
```
cf2 = confusion_matrix(predictions,y_test)
print(cf2)
print(classification_report(predictions, y_test))
print( "accuracy score: ", accuracy_score(predictions, y_test))
```

```
[[139  34]
 [ 18  40]]
              precision    recall  f1-score   support

           0       0.89      0.80      0.84       173
           1       0.54      0.69      0.61        58

    accuracy                           0.77       231
   macro avg       0.71      0.75      0.72       231
weighted avg       0.80      0.77      0.78       231

accuracy score:  0.7748917748917749
```

KNN ALOGRITHM

- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.This algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

KNN

In [27]:
```
knn = KNeighborsClassifier(10)
knn . fit(x_train , y_train)
y_pred = knn . predict(x_test)
print('acc:',metrics . accuracy_score(y_test , y_pred))
```

```
acc: 0.7532467532467533
```

DECISION TREE CLASSIFIER

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules, and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed based on features of the given dataset.

DECISION TREE CLASSIFIER

In [61]:
```
x = pd.DataFrame(df , columns=['Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']).value
y = df . Outcome . values . reshape(-1 , 1)
x_train , x_test , y_train , y_test = train_test_split(x , y ,test_size=0.3 , random_state=0)
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(x_train,y_train)
y_predd = dtc.predict(x_test)
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
cf1 = confusion_matrix(y_predd,y_test)
print(cf1)
print(classification_report(y_predd, y_test))
print( "accuracy score: ", accuracy_score(y_predd, y_test))
```

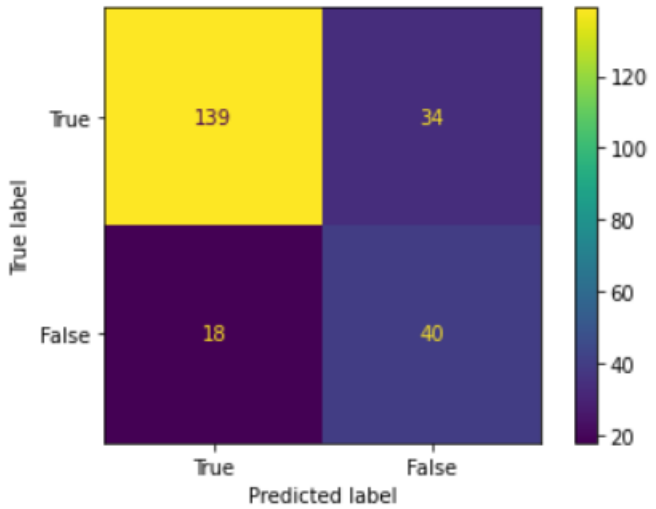
```
[[124  34]
 [ 33  40]]
              precision    recall  f1-score   support

           0       0.79      0.78      0.79       158
           1       0.54      0.55      0.54        73

    accuracy                           0.71       231
   macro avg       0.67      0.67      0.67       231
weighted avg       0.71      0.71      0.71       231

accuracy score:  0.70995670995671
```

STEPS TO BUILD A ML MODEL

- Import library
- Read dataset
- Data analysis
- Label encoding of the categorical columns
- Defining x and y
- Splitting x and y into train and test data.
- Import the model
- Train the model with x_train and y_train
- Predict with x_test and got predicted y
- Evaluation with confusion matrix, classification report , accuracy score.

Confusion Matrix- Logistic Regression

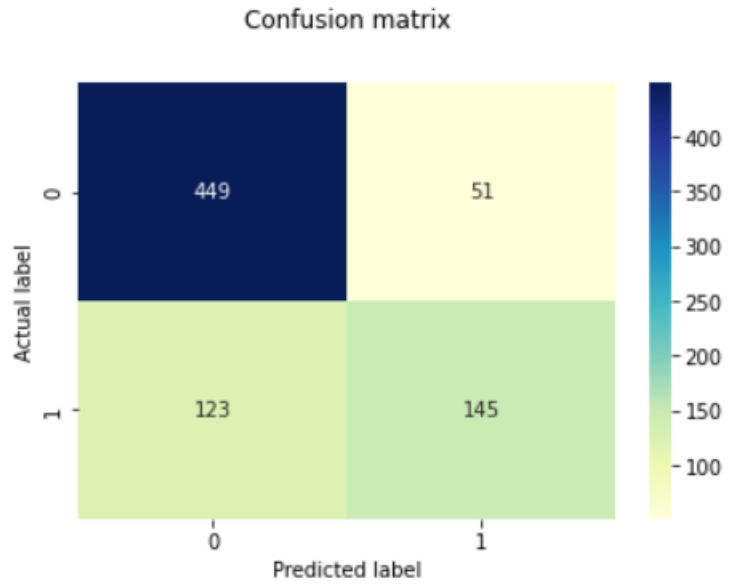
```
In [74]: cm_display1 = metrics.ConfusionMatrixDisplay(confusion_matrix = cf2, display_labels = [True, False])
         cm_display1.plot()
         plt.show()
```



Confusion Matrix- KNN

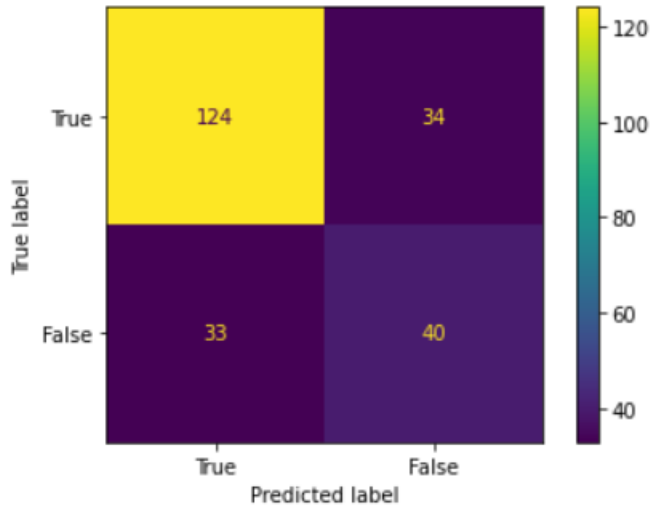
In [29]:
```
y_pred = knn.predict(x_test)
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y , knn . predict(x))
p = sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Out[29]:
```
Text(0.5, 15.0, 'Predicted label')
```



Confusion Matrix- Decision tree

```
In [65]: cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cf1, display_labels = [True, False])
         cm_display.plot()
         plt.show()
```



CONCLUSION

- Machine learning can help doctors identify and cure diabetes.
- We will conclude that improving the accuracy of the classification will help the machine learning models perform better.
- The performance analysis is in terms of accuracy rate among all the classification techniques such as logistic regression, K-nearest neighbors, SVM, random forest.
- By performing, we found accuracy for following:-

Algorithms	Accuracy Score
Logistic Regression	0.77
KNN	0.75
Decision Tree	0.70

- By the above observation, we conclude that Logistic Regression performs best on the given dataset with accuracy score – 0.77