Lavina Golani

16

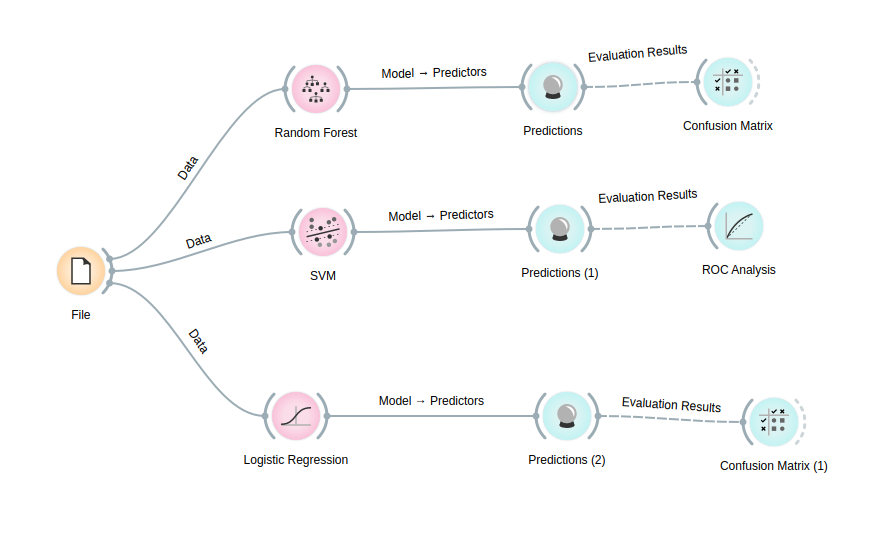
EXPERIMENT -6

**Aim:** Given a case study with data set. You are expected to perform Classification using Python/R/Java.

Quote your observations after the Classification.

**Observations:**

**Using Orange Tool O/p:**



**Using Python/R/Java:**

from sklearn.linear\_model import LogisticRegression

import pandas as pd

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

from sklearn.metrics import classification\_report,confusion\_matrix

from sklearn import tree

import graphviz

file = '/home/lavina/Desktop/pima-indians-diabetes-database/diabetes.csv'

# load data set into frame

dataFrame = pd.read\_csv(file, names=['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'])

dataFrame = dataFrame.dropna()

print(dataFrame.describe())

print(dataFrame.columns)

# logistic regression

# seperating the data into training and testing

boundary =int (dataFrame.shape[0] \*0.6)

features = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']

# trainingData = dataFrame.loc[0:boundary, features]

# trainingTarget = dataFrame.loc[:boundary, ['Outcome']]

# print(trainingData.shape)

# testingData = dataFrame.loc[boundary:, features]

# testingTarget = dataFrame.loc[boundary:, ['Outcome']]

X = dataFrame.loc[:, ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']]

y = dataFrame.loc[:, ['Outcome']]

trainingData, testingData, trainingTarget, testingTarget = train\_test\_split(X, y)

print(testingData.shape)

logisticRegression = LogisticRegression()

#training the model

logisticRegression.fit(trainingData, trainingTarget)

#testing model

predict = logisticRegression.predict(testingData)

# Use score method to get accuracy of model

score = logisticRegression.score(testingData,testingTarget)

print('logistic regression', score)

#analysing output

print(confusion\_matrix(testingTarget, predict))

print(classification\_report(testingTarget, predict))

#making model and fitting the testing data

decisionTree = tree.DecisionTreeClassifier()

decisionTree.fit(trainingData , trainingTarget)

# predicting with test data

predicted = decisionTree.predict(testingData)

score = decisionTree.score(testingData,testingTarget)

print('decision tree',score)

print(confusion\_matrix(testingTarget,predicted))

print(classification\_report(testingTarget,predicted))

treeGraph = tree.export\_graphviz(decisionTree , feature\_names=features,

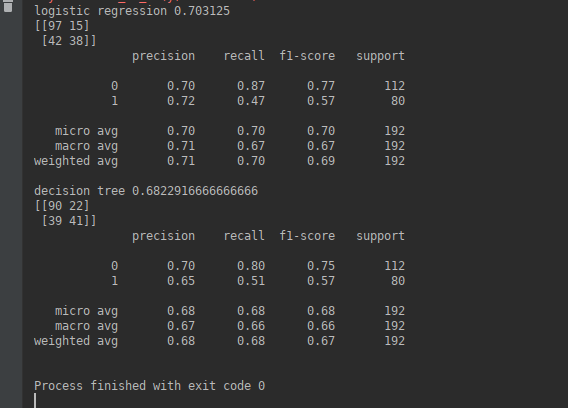
class\_names=["Diabetic", 'Non-Diabetic'],

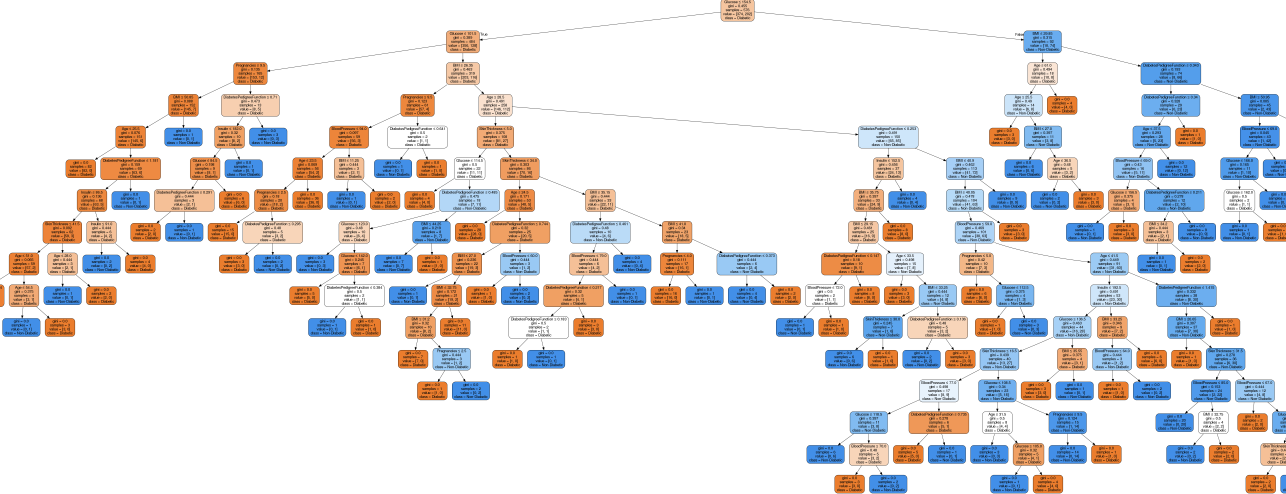
filled=True , rounded=True,

special\_characters= True,

out\_file='tree.dot')

graph = graphviz.Source(treeGraph)

****



**Conclusion: Hence we have successfully implemented classification in R and Python.**

**We observed that decision tree is prone to over fitting .It is better to do it with categorical variables to prevent over fitting and restrict height of the tree. Random forests can also be used to prevent overfitting.**

**Logistic regression is also used for classification. SVM can be used and it also takes care of the edge cases.**