```
import numpy as np import matplotlib.pyplot as plt import pandas as pd import tensorflow as tf
from sklearn.model_selection import train_test_split from sklearn.preprocessing import
StandardScaler from sklearn.metrics import confusion_matrix  # Importing the dataset dataset =
pd.read_csv('D:\diabetes.csv') X = dataset.iloc[:, :-1].values  # Assuming the last column is the
target variable y = dataset.iloc[:, -1].values  # Splitting the dataset into the Training set and Test
set X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)  #
Feature Scaling sc = StandardScaler() X_train = sc.fit_transform(X_train) X_test =
sc.transform(X_test)  # Part 2 - Now let's make the ANN! # Importing the Keras libraries and
packages from tensorflow.keras.models import Sequential from tensorflow.keras.layers import
Dense  # Initialising the ANN classifier = Sequential()  # Adding the input layer and the first
hidden layer classifier.add(Dense(units=6, activation='relu', kernel_initializer='uniform',
input_dim=X_train.shape[1]))  # Adding the second hidden layer classifier.add(Dense(units=6,
kernel_initializer='uniform', activation='relu'))  # Adding the output layer
classifier.add(Dense(units=1, kernel_initializer='uniform', activation='sigmoid'))  # Compiling the
ANN classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])  #
Fitting the ANN to the Training set classifier.fit(X_train, y_train, batch_size=10, epochs=100)  #
Part 3 - Making the predictions and evaluating the model # Predicting the Test set results y_pred
= classifier.predict(X_test) y_pred = (y_pred > 0.5)  # Making the Confusion Matrix cm =
confusion_matrix(y_test, y_pred) print("Confusion Matrix:") print(cm) from
tensorflow.keras.layers import Dense  import seaborn as sns
sns.heatmap(cm,fmt=".0f",xticklabels=['Diabeties_yes','Diabeties_no'],yticklabels=['Diabeties_yes','Diabeties_no'],annot=True) #sns.heatmap(cm,fmt=".0f",annot=True) #Visualizing the neural
network from ann_visualizer.visualize import ann_viz from graphviz import Source
ann_viz(classifier,title='Neural Network') graph_source=Source.from_file('network.gv')
print(graph_source.source) classifier.get_weights() #Predicting the input record #create an
empty data frame that we have to predict variety=pd.DataFrame() variety['Pregnancies']=[6]
variety['Glucose']=[148] variety['BloodPressure']=[72] variety['SkinThickness']=[35]
variety['Insulin']=[0] variety['BMI']=[33.6] variety['DiabetesPedigreeFunction']=[0.627]
variety['Age']=[50] print(variety) y_pred1=classifier.predict(variety) print("the Outcome of the
Patient is:") print(y_pred1) #Calculating Performance Metrics for Training Set FP =
cm.sum(axis=0) - np.diag(cm)  FN = cm.sum(axis=1) - np.diag(cm) 62 TP = np.diag(cm) TN =
cm.sum() - (FP + FN + TP) FP = FP.astype(float) FN = FN.astype(float) TP = TP.astype(float)
TN = TN.astype(float) # Sensitivity, hit rate, recall, or true positive rate TPR = TP/(TP+FN)
print("Recall",TPR) # Specificity or true negative rate TNR = TN/(TN+FP)
print("Specificity",TNR) # Precision or positive predictive value PPV = TP/(TP+FP)
print("Precision",PPV) # Negative predictive value NPV = TN/(TN+FN) print("Negative Predictive
Value",NPV) # Fall out or false positive rate FPR = FP/(FP+TN) print("False Positive Rate",FPR)
# False negative rate FNR = FN/(TP+FN) print("False Negative Rate",FNR) # False discovery
rate FDR = FP/(TP+FP) print("False Discovery Rate",FDR) # Overall accuracy for each class
ACC = (TP+TN)/(TP+FP+FN+TN) print("Accuracry",ACC)
```