**Practical 3**

**Binary Classification Task**

**Aim:** Implementing deep neural network for performing binary classification task.

**Description:**

**Binary Classification**

* Goal: Classify data points into exactly two categories (classes).
* Applications: Spam filtering, sentiment analysis (positive/negative reviews), image recognition (cat/dog), fraud detection (fraudulent/legitimate transaction).
* Training Data: Labeled data examples with each point belonging to one of the two classes.
* Model Learning: A binary classification model learns to distinguish between the two classes based on the features (attributes) of the data points.
* Evaluation: Performance is often measured using metrics like accuracy (percentage of correctly classified examples), precision (proportion of true positives among predicted positives), recall (proportion of identified true positives), and F1-score (harmonic mean of precision and recall).

**TensorFlow and Keras**

TensorFlow and Keras are popular deep learning libraries frequently used in tandem for building and training neural networks. TensorFlow provides a robust framework for developing machine learning models, offering low-level control over model architecture and computation. Keras, on the other hand, offers a high-level API that simplifies the process of building neural networks, enabling rapid prototyping and experimentation. By integrating Keras within TensorFlow, users can leverage the simplicity and flexibility of Keras while benefiting from the scalability and performance optimizations of TensorFlow's backend. This combination empowers developers to efficiently create and train neural networks for a wide range of tasks, from image classification to natural language processing.

**Code:**

# pip install keras

from keras.models import Sequential

from keras.layers import Dense

import pandas as pd

names = [

"No. of pregnancies",

"Glucose level",

"Blood Pressure",

"skin thickness",

"Insulin",

"BMI",

"Diabetes pedigree",

"Age",

"Class",

]

#csv file with no column names expected

df = pd.read\_csv("/content/pima-indians-diabetes.data.csv", names=names)

df.head(3)

binaryc = Sequential()

from tensorflow.tools.docs.doc\_controls import doc\_in\_current\_and\_subclasses

binaryc.add(Dense(units=10, activation="relu", input\_dim=8))

binaryc.add(Dense(units=8, activation="relu"))

binaryc.add(Dense(units=1, activation="sigmoid"))

binaryc.compile(loss="binary\_crossentropy", optimizer="adam", metrics="accuracy")

X = df.iloc[:, :-1]

y = df.iloc[:, -1]

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

xtrain, xtest, ytrain, ytest = train\_test\_split(X, y, test\_size=0.25, random\_state=1)

xtrain.shape

ytrain.shape

binaryc.fit(xtrain, ytrain, epochs=200, batch\_size=20)

predictions = binaryc.predict(xtest)

predictions.shape

class\_labels = []

for i in predictions:

if i > 0.5:

class\_labels.append(1)

else:

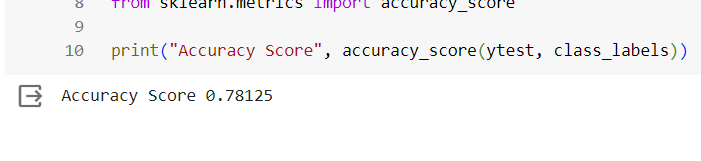
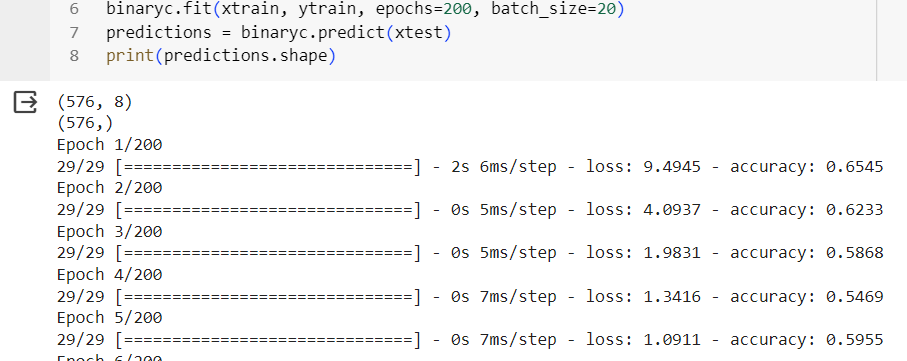
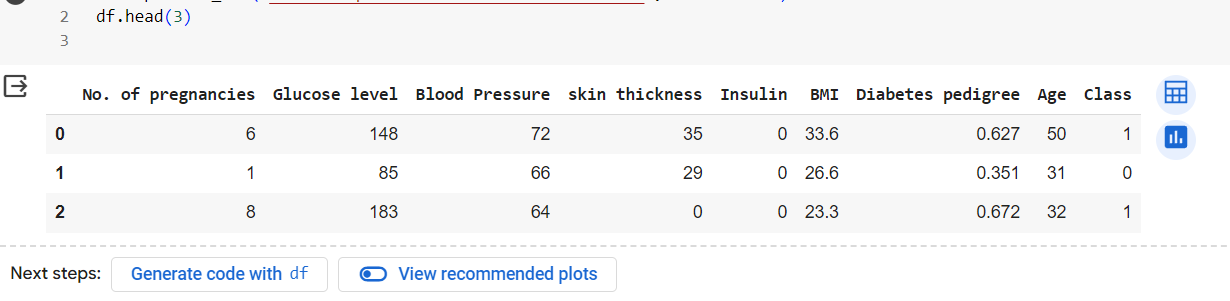
class\_labels.append(0)

class\_labels

from sklearn.metrics import accuracy\_score

print("Accuracy Score", accuracy\_score(ytest, class\_labels))

**Output:**



**Learning:**

This code builds and trains a neural network using Keras to predict diabetes from given health data. It starts by importing necessary libraries and defining the dataset's features and labels. Then, a neural network model is created with three layers. The model is compiled with appropriate loss function and optimizer. Next, the data is split into training and testing sets. The model is trained on the training data, and predictions are made on the test data. Finally, the accuracy of the predictions is evaluated using the test labels, and the accuracy score is printed.