:

Branch/Sec : IT-A
Lab Experiment 6

In [2]:
```
# Import the necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
```

In [3]:
```
# Step 1: Load the data
bank_data = pd.read_csv("BankMarketing.csv", delimiter=";")
print(bank_data.head())
```

```
                          marital
0
   age           job            education default  balance housing loan  \
    58    management  married   tertiary      no     2143     yes   no
1   44    technician   single  secondary      no       29     yes   no
2   33  entrepreneur  married  secondary      no        2     yes  yes
4   33       unknown   single    unknown      no        1      no   no
3   47   blue-collar  married    unknown      no     1506     yes   no
   contact  day month  duration                  pdays  previous          y
0
1  unknown    5   may       151  campaign 1     -1         0 unknown        no
2  unknown    5   may        76          1     -1         0 poutcome       no
   unknown    5   may       261          1     -1         0 unknown   no
3  unknown    5   may        92          1     -1         0 unknown        no
4  unknown    5   may       198          1     -1         0 unknown        no
```

In [4]:
```
# Step 2: Preprocess the data
# Convert categorical variables to numerical variables using one-hot encoding
bank_data = pd.get_dummies(bank_data, columns=["job", "marital", "education", "default'
# Convert target variable from yes/no to 1/0
bank_data["y"] = bank_data["y"].map({"yes": 1, "no": 0})
```

In [5]:
```
# Step 3: Split the data into training and testing sets
X = bank_data.drop("y", axis=1)
y = bank_data["y"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In [6]:

```
# Step 4: Build the feedforward neural network model
model = Sequential()
model.add(Dense(64, input_dim=X_train.shape[1], activation="relu"))
model.add(Dense(32, activation="relu"))
model.add(Dense(1, activation="sigmoid"))
model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
```

In [7]:

```
# Step 5: Train the model on the training data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
model.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test)
```

```
Epoch 45/50
1131/1131 [==============================] - 2s 2ms/step - loss: 0.1298 - accurac
y: 0.9446 - val_loss: 0.2913 - val_accuracy: 0.8946
Epoch 46/50
1131/1131 [==============================] - 2s 2ms/step - loss: 0.1287 - accurac
y: 0.9454 - val_loss: 0.3008 - val_accuracy: 0.8948
Epoch 47/50
1131/1131 [==============================] - 2s 2ms/step - loss: 0.1278 - accurac
y: 0.9450 - val_loss: 0.2979 - val_accuracy: 0.8943
Epoch 48/50
1131/1131 [==============================] - 2s 2ms/step - loss: 0.1263 - accurac
y: 0.9463 - val_loss: 0.3062 - val_accuracy: 0.8966
Epoch 49/50
1131/1131 [==============================] - 2s 2ms/step - loss: 0.1264 - accurac
y: 0.9474 - val_loss: 0.2984 - val_accuracy: 0.8928
Epoch 50/50
1131/1131 [==============================] - 2s 2ms/step - loss: 0.1254 - accurac
y: 0.9465 - val_loss: 0.3029 - val_accuracy: 0.8955
```

Out[7]:

```
<keras.callbacks.History at 0x202d25a7310>
```

In [8]:

```
# Step 6: Evaluate the model on the testing data
loss, accuracy = model.evaluate(X_test, y_test)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)
```

```
283/283 [==============================] - 0s 1ms/step - loss: 0.3029 - accuracy: 0.8
955
Test Loss: 0.30293911695480347
Test Accuracy: 0.8954992890357971
```

In [ ]: