Santander Private Banking UK requires a case study and model

1)To Identify and visualize which factors contribute to customer churn

2)Build a prediction model that will perform the following:

Classify if a customer is going to churn or not.

Preferably and based on model performance, choose a model that will attach a probability to the churn to make it easier for customer service to target low hanging fruits in their efforts to prevent churn.

```
cd /content/drive/MyDrive/Colab_Notebooks/DL/Krish_Naik/Churn_Modelling
```

```
/content/drive/MyDrive/Colab_Notebooks/DL/Krish_Naik/Churn_Modelling
```

```
#! pip list

```

```
!pip install tensorflow
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.12.0)
Requirement already satisfied: tensorboard<2.13,>=2.12 in /usr/local/lib/python3.10/dist-packages (from tensorflo
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0
Requirement already satisfied: keras<2.13,>=2.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.
Requirement already satisfied: jax>=0.3.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.8)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.3
Requirement already satisfied: tensorflow-estimator<2.13,>=2.12.0 in /usr/local/lib/python3.10/dist-packages (fro
Requirement already satisfied: numpy<1.24,>=1.22 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.8.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: flatbuffers>=2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorfl
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 i
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (16.
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.
Requirement already satisfied: scipy>=1.7 in /usr/local/lib/python3.10/dist-packages (from jax>=0.3.15->tensorflo
Requirement already satisfied: ml-dtypes>=0.0.3 in /usr/local/lib/python3.10/dist-packages (from jax>=0.3.15->ten
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in /usr/local/lib/python3.10/dist-packages (from te
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from ten
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-aut
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-a
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from request
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modul
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib
```

```
#!pip install --upgrade tensorflow

# Tensorflow version should be greater than 2.0.0 , so that Keras will be supported in version higher than 2.0
```

```
import tensorflow as tf
tf.__version__
```

```
'2.12.0'
```

```
#import some basic libraries

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

```

```
dataset=pd.read_csv('Churn_Modelling.csv')
dataset.head()
```

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | E |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 15634602 | Hargrave | 619 | France | Female | 42 | 2 | |
| **1** | 2 | 15647311 | Hill | 608 | Spain | Female | 41 | 1 | 8 |
| **2** | 3 | 15619304 | Onio | 502 | France | Female | 42 | 8 | 15 |
| **3** | 4 | 15701354 | Boni | 699 | France | Female | 39 | 1 | |
| **4** | 5 | 15737888 | Mitchell | 850 | Spain | Female | 43 | 2 | 12 |

```
# Divide the Dataset into Independent and Dependent features

# Row number , Customer Id & Surname is not required, so we are removing

X = dataset.iloc[:,3:13] # from 3rd column to 13th column
y = dataset.iloc[:,13]

```

```
X.head()
```

| | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard |
|---|---|---|---|---|---|---|---|---|
| **0** | 619 | France | Female | 42 | 2 | 0.00 | 1 | 1 |
| **1** | 608 | Spain | Female | 41 | 1 | 83807.86 | 1 | 0 |
| **2** | 502 | France | Female | 42 | 8 | 159660.80 | 3 | 1 |
| **3** | 699 | France | Female | 39 | 1 | 0.00 | 2 | 0 |
| **4** | 850 | Spain | Female | 43 | 2 | 125510.82 | 1 | 1 |

```
y.head()
```

```
0    1
1    0
2    1
3    0
4    0
Name: Exited, dtype: int64
```

## Feature Engineering

```
pd.get_dummies(X['Geography'])
```

| | France | Germany | Spain |
|---|---|---|---|
| **0** | 1 | 0 | 0 |
| **1** | 0 | 0 | 1 |
| **2** | 1 | 0 | 0 |
| **3** | 1 | 0 | 0 |
| **4** | 0 | 0 | 1 |
| **...** | ... | ... | ... |
| **9995** | 1 | 0 | 0 |
| **9996** | 1 | 0 | 0 |
| **9997** | 1 | 0 | 0 |
| **9998** | 0 | 1 | 0 |
| **9999** | 1 | 0 | 0 |

10000 rows × 3 columns

```
geography = pd.get_dummies(X['Geography'], drop_first=True)
geography.head(5)
```

| | Germany | Spain |
|---|---|---|
| **0** | 0 | 0 |
| **1** | 0 | 1 |
| **2** | 0 | 0 |
| **3** | 0 | 0 |
| **4** | 0 | 1 |

```
gender = pd.get_dummies(X['Gender'],drop_first=True)
gender.head(5)
```

| | Male |
|---|---|
| **0** | 0 |
| **1** | 0 |
| **2** | 0 |
| **3** | 0 |
| **4** | 0 |

```
## Concatenate these variales wirh dataframe

X = X.drop(['Geography','Gender'], axis=1) # axis= 1 will remove only the column.
```

```
X = pd.concat([X, geography, gender], axis=1)
```

## Splitting the dataset into Training & Test set

```
from sklearn.model_selection import train_test_split

```

```
3 X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=0)
```

## Feature Scaling:

```
1 from sklearn.preprocessing import StandardScaler
2 sc = StandardScaler()
3
4 X_train = sc.fit_transform(X_train)
5 X_test = sc.transform(X_test)
```

```
1 X_train
```

```
array([[ 0.16958176, -0.46460796,  0.00666099, ..., -0.5698444 ,
         1.74309049, -1.09168714],
       [-2.30455945,  0.30102557, -1.37744033, ...,  1.75486502,
        -0.57369368,  0.91601335],
       [-1.19119591, -0.94312892, -1.031415  , ..., -0.5698444 ,
        -0.57369368, -1.09168714],
       ...,
       [ 0.9015152 , -0.36890377,  0.00666099, ..., -0.5698444 ,
        -0.57369368,  0.91601335],
       [-0.62420521, -0.08179119,  1.39076231, ..., -0.5698444 ,
         1.74309049, -1.09168714],
       [-0.28401079,  0.87525072, -1.37744033, ...,  1.75486502,
        -0.57369368, -1.09168714]])
```

```
1 X_test
```

```
array([[-0.55204276, -0.36890377,  1.04473698, ...,  1.75486502,
        -0.57369368, -1.09168714],
       [-1.31490297,  0.10961719, -1.031415  , ..., -0.5698444 ,
        -0.57369368, -1.09168714],
       [ 0.57162971,  0.30102557,  1.04473698, ..., -0.5698444 ,
         1.74309049, -1.09168714],
       ...,
       [-0.74791227, -0.27319958, -1.37744033, ..., -0.5698444 ,
         1.74309049,  0.91601335],
       [-0.00566991, -0.46460796, -0.33936434, ...,  1.75486502,
        -0.57369368,  0.91601335],
       [-0.79945688, -0.84742473,  1.04473698, ...,  1.75486502,
        -0.57369368,  0.91601335]])
```

```
1 X_train.shape
```

```
(8000, 11)
```

## Lets create ANN

```
1 from tensorflow.keras.models import Sequential
2 from tensorflow.keras.layers import Dense
3 from tensorflow.keras.layers import LeakyReLU, PReLU, ELU, ReLU  #Activation functions
4 from tensorflow.keras.layers import Dropout
5
```

## Initialize ANN

```
1 classifier = Sequential()
```

```
# Adding input layer

classifier.add(Dense(units=11, activation='relu'))   # Dense is used for adding input layer, Hidden layer & O/P laye
                                                     # units = 11 , number of input columns
                                                     # Activation will be Relu. This will be added to the next layer
```

```
# Adding  1st Hidden  layer with 7 Neurons

classifier.add(Dense(units=7, activation='relu'))   # Dense is used for adding input layer, Hidden layer & O/P layer
classifier.add(Dropout(0.2))                         # units = 7 , number of neurons in 1st hidden layer
                                                     # Activation will be Relu.
                                                     # Dropout will deactivate the neuron once the value of 0.2 is r
```

```
# Adding  2nd Hidden  layer with 6 Neurons

classifier.add(Dense(units=6, activation='relu'))   # Dense is used for adding input layer, Hidden layer & O/P layer
                                                     # units = 6 , number of neurons in 2nd hidden layer
                                                     # Activation will be Relu. This will be added to the next layer
```

```
# Adding the O/P layer

classifier.add(Dense(units=1, activation='sigmoid'))   # # Dense is used for adding input layer, Hidden layer & O/P
                                                       # units = 1 , number of neurons in O/P layer
                                                       # Activation will be Sigmoid. Since its a Binary Classificati
```

## Train my ANN

```
classifier.compile(optimizer='Adam', loss='binary_crossentropy', metrics=['accuracy'])  # Adam the best Optimizer. A
                                                                                        # Loss Function will be 'Bir
                                                                                        # Metrics will be 'Accurcay'
```

```
# For information purpose
#In case if you want to add your own learning rate, you can import it from the below library

#import tensorflow
#opt = tensorflow.keras.optimizers.Adam(learning_rate=0.01)

# Then add the optimizer values like below
# classifier.compile(optimizer=opt, loss='binary_crossentropy', metrics=['acccuracy'])
```

## Training our Neural Network

```
#Early stopping: when the Training Accuracy is not increasing for a long time, at that time, we can stop(Early Stop)

#Google 'Early Stopping in Keras'.

#tf.keras.callbacks.EarlyStopping(
#   monitor="val_loss",
#    min_delta=0,
#    patience=0,
#    verbose=0,
#    mode="auto",
#    baseline=None,
#    restore_best_weights=False,
#    start_from_epoch=0,
#)


#Early Stoping; You can play around with the values
```

```

import tensorflow as tf
early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor="val_loss",
    min_delta=0.0001,
    patience=20,
    verbose=1,      # Verbose should be 1 as I need to see all the details
    mode="auto",
    baseline=None,
    restore_best_weights=False,
    start_from_epoch=0,
)
```

```
model_history = classifier.fit(X_train, y_train, validation_split=0.33, batch_size=10, epochs=1000, callbacks=early_

# Validation_split =.33 ==> out of 100% of training data, we take (100 - 33 = 66) 66% of data for validation.

# 536/536 is the iteration based on the epoch and Batch size

# loss => training loss
# val_loss = validation loss

# After some point of time my accurcay will be stagnant, at that point of time,

#Since in the above iteration, most of the time, its going around 88%. so I am going to stop . Stop the google colab
 #introduce Early stopping

# early_stopping ==> though we gave epochs = 10000 , it will not run the epoch until 1000 as its a time consuming. c
# and if the Accuracy value is not getting increased after a particular time, the call_back will be executed and the
# If the Val_loss is not improving much , it will be stopped.

```

```
536/536 [==============================] - 2s 4ms/step - loss: 0.3269 - accuracy: 0.8666 - val_loss: 0.3595 - v
Epoch 101/1000
536/536 [==============================] - 3s 6ms/step - loss: 0.3297 - accuracy: 0.8638 - val_loss: 0.3593 - v
Epoch 102/1000
536/536 [==============================] - 3s 6ms/step - loss: 0.3280 - accuracy: 0.8645 - val_loss: 0.3591 - v
Epoch 103/1000
536/536 [==============================] - 2s 4ms/step - loss: 0.3279 - accuracy: 0.8642 - val_loss: 0.3598 - v
Epoch 104/1000
536/536 [==============================] - 2s 4ms/step - loss: 0.3257 - accuracy: 0.8660 - val_loss: 0.3596 - v
Epoch 105/1000
536/536 [==============================] - 3s 5ms/step - loss: 0.3256 - accuracy: 0.8628 - val_loss: 0.3585 - v
Epoch 106/1000
536/536 [==============================] - 3s 6ms/step - loss: 0.3274 - accuracy: 0.8653 - val_loss: 0.3591 - v
Epoch 107/1000
536/536 [==============================] - 3s 5ms/step - loss: 0.3250 - accuracy: 0.8668 - val_loss: 0.3616 - v
Epoch 108/1000
536/536 [==============================] - 2s 4ms/step - loss: 0.3260 - accuracy: 0.8675 - val_loss: 0.3615 - v
Epoch 109/1000
```

In the above O/P, its "Epoch 57: early stopping" with Accuracy 0.8707

Double-click (or enter) to edit

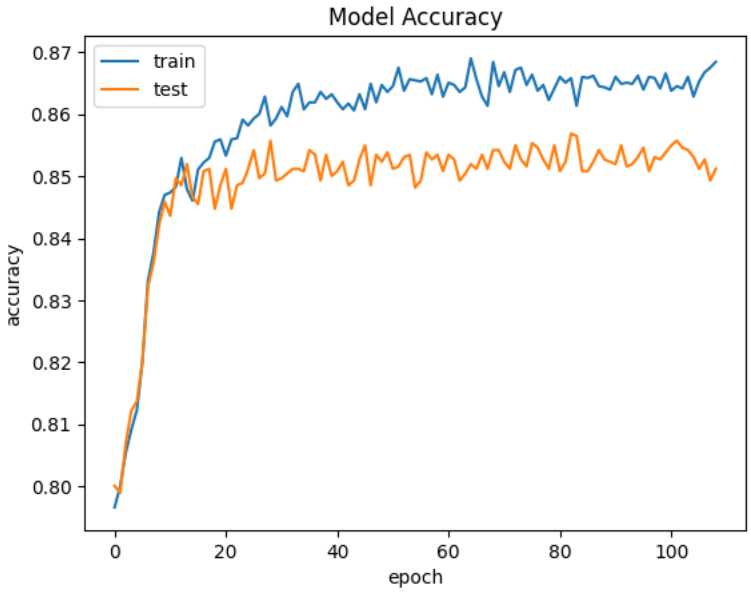
```
model_history.history.keys()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

## Summary History for Accuracy

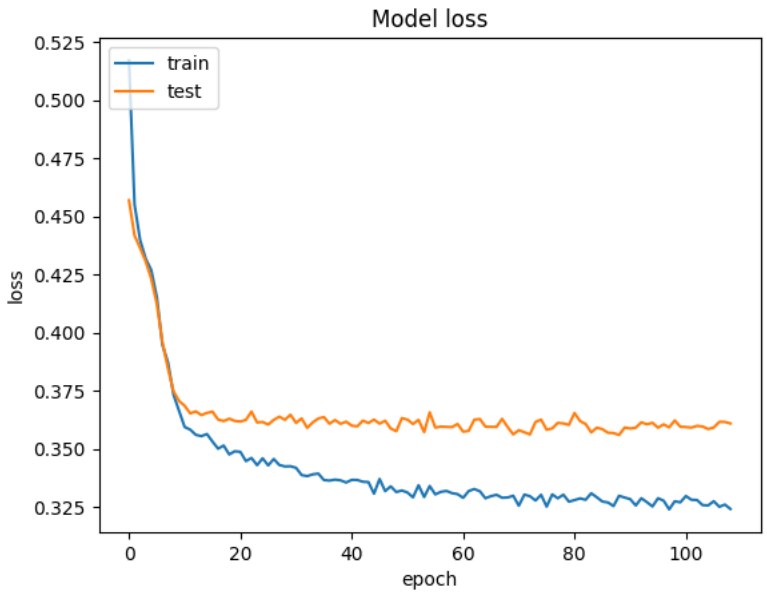
```
plt.plot(model_history.history['accuracy'])
plt.plot(model_history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()

# IN the below diagram, the size/gap is almost same or very less.
```



## Summary History for Loss

```
plt.plot(model_history.history['loss'])
plt.plot(model_history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()
```




## Making the Predictions and evaluating the Model

```
# Predicting Test set results
y_pred = classifier.predict(X_test)
y_pred = (y_pred >= 0.5)   # If its GEQ 0.5 I will take it as 1 else 0

```

```
63/63 [==============================] - 0s 2ms/step
```

## Create Confusion Matrix

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[1507,   88],
       [ 201,  204]])
```

## To calculate Accuracy

```
from sklearn.metrics import accuracy_score
score=accuracy_score(y_pred,y_test)
print('Accuracy :', score)
```

```
Accuracy : 0.8555
```

## To get the weights assigned to Neural Network

```
classifier.get_weights()   # These are all the weights value
```

```
              0.07202806,  0.5394073 ,  0.33068386, -0.23740686, -0.49479774,
              0.34787634],
            [-0.3795275 ,  0.547824  , -0.26878467,  0.24962221, -0.6195596 ,
             -0.68806213,  0.31104556,  0.00565256,  0.0132001 ,  0.16133149,
             -0.13530122],
            [ 0.3740936 ,  0.33844286, -0.03629395,  0.15874322,  0.23995042,
              0.31639075,  0.01470504,  0.10974654,  0.04500822, -0.03579355,
             -0.04594004]], dtype=float32),
     array([ 0.50589293, -0.08963937,  0.91366386,  0.09575628, -0.36327595,
            -0.13635701, -0.03612503, -0.10989795, -0.95137244, -0.47099164,
            -0.02579247], dtype=float32),
     array([[ 0.0166559 , -0.17833902,  0.06943022,  0.5816181 , -0.5013056 ,
              0.5250197 ,  0.0736379 ],
            [-0.02585014, -0.01116616,  0.4164095 ,  0.5307345 , -0.0553316 ,
              0.4187833 ,  0.30748808],
            [-1.2726372 , -0.6833254 ,  0.08398324,  0.40602022, -0.48742688,
              0.6639144 ,  0.26270002],
            [-0.23310532, -0.4241048 ,  0.08924681,  0.29747796, -0.57601947,
              0.18386602,  0.31719837],
            [ 0.46666396,  0.45733413,  0.36872295, -0.5179982 ,  0.38955614,
             -0.5727541 , -0.11012888],
            [ 0.13256729, -0.7387074 ,  0.24270202,  0.30282652, -0.39864323,
              0.40435734,  0.16977046],
            [ 0.52163255,  0.17678253, -0.05061479, -0.20306112,  0.2721399 ,
             -0.10630787, -0.8413602 ],
            [-0.43188822,  0.188136  , -1.368699  , -0.6481819 , -0.02907252,
              0.06124942,  0.3365523 ],
            [ 1.2982361 ,  0.9704765 , -0.41823176, -1.051541  ,  0.9643132 ,
             -1.0170395 , -0.4345287 ],
            [-0.60650176, -0.51276344, -0.5057549 , -0.0485401 , -0.02069807,
              0.15953368, -0.21401075],
            [ 0.53626853,  0.21261051, -0.26390353, -0.8310787 ,  0.24866629,
             -0.7560894 , -0.8292469 ]], dtype=float32),
     array([-0.04860691, -0.06202723,  0.3498477 ,  0.8805351 ,  0.09045316,
             0.16145904, -0.15595785], dtype=float32),
     array([[ 0.03264572,  0.9465647 ,  0.10119472, -0.3636538 , -0.7818476 ,
             -0.7780716 ],
            [-0.15745986,  0.6498983 , -0.41560486, -1.4087316 , -1.1607507 ,
             -1.036178  ],
            [ 0.81111526,  0.1849275 ,  0.16152415,  0.54406846,  0.15789898,
              0.5996481 ],
            [-0.9223942 , -0.04461425,  0.85996616,  0.49070287, -0.86165726,
             -0.4968312 ],
            [-0.02504773,  0.80555785,  0.06402837, -0.470465  , -0.3331471 ,
             -0.29270843],
            [ 0.6926857 ,  0.05040722,  0.60584533, -0.39555728,  0.74151   ,
             -0.81154794],
            [ 0.37341484,  0.36859223,  0.8924846 ,  0.32440257,  0.13066098,
              1.0155251 ]], dtype=float32),
     array([0.04574655, 0.1139995 , 0.3558023 , 0.21981609, 0.12850721,
            0.15664038], dtype=float32),
     array([[-0.7458111 ],
            [ 1.0752013 ],
            [-0.6146915 ],
            [-0.9058826 ],
            [-0.56294906],
            [-1.8829371 ]], dtype=float32),
     array([0.1041213], dtype=float32)]
```

```
1
1
1
1
1
1
1
1
1
1
1
```

✓ 0s completed at 8:19 PM