experiments

April 7, 2025

1 Feature Transformation Using Sklearn with ANN

[1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
import pickle
```

[2]:
```
# Load the dataset
data = pd.read_csv('../Datasets/Churn_Modelling.csv')
data.head()
```

[2]:
```
   RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age  \
0          1    15634602  Hargrave          619    France  Female   42
1          2    15647311      Hill          608     Spain  Female   41
2          3    15619304      Onio          502    France  Female   42
3          4    15701354      Boni          699    France  Female   39
4          5    15737888  Mitchell          850     Spain  Female   43

   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              1          1               1
1       1   83807.86              1          0               1
2       8  159660.80              3          1               0
3       1       0.00              2          0               0
4       2  125510.82              1          1               1

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0
```

[3]:
```
# Preprocess the data
## Drop irrelevant columns

data = data.drop(['RowNumber', 'CustomerId', 'Surname'], axis=1)
data
```

[3]:
```
      CreditScore Geography  Gender  Age  Tenure    Balance  NumOfProducts  \
0             619    France  Female   42       2       0.00              1
1             608     Spain  Female   41       1   83807.86              1
2             502    France  Female   42       8  159660.80              3
3             699    France  Female   39       1       0.00              2
4             850     Spain  Female   43       2  125510.82              1
…             …       …    …   …    …       …            …
9995          771    France    Male   39       5       0.00              2
9996          516    France    Male   35      10   57369.61              1
9997          709    France  Female   36       7       0.00              1
9998          772   Germany    Male   42       3   75075.31              2
9999          792    France  Female   28       4  130142.79              1

      HasCrCard  IsActiveMember  EstimatedSalary  Exited
0             1               1        101348.88       1
1             0               1        112542.58       0
2             1               0        113931.57       1
3             0               0         93826.63       0
4             1               1         79084.10       0
…           …             …              …     …
9995          1               0         96270.64       0
9996          1               1        101699.77       0
9997          0               1         42085.58       1
9998          1               0         92888.52       1
9999          1               0         38190.78       0

[10000 rows x 11 columns]
```

[4]:
```python
##  Encode categorical variables
label_encoder_gender = LabelEncoder() # we use this for converting the char␣
 ↪data to int data
data['Gender']= label_encoder_gender.fit_transform(data['Gender'])
data
```

[4]:
```
      CreditScore Geography  Gender  Age  Tenure    Balance  NumOfProducts  \
0             619    France       0   42       2       0.00              1
1             608     Spain       0   41       1   83807.86              1
2             502    France       0   42       8  159660.80              3
3             699    France       0   39       1       0.00              2
4             850     Spain       0   43       2  125510.82              1
…             …       …    …   …    …       …            …
9995          771    France       1   39       5       0.00              2
9996          516    France       1   35      10   57369.61              1
9997          709    France       0   36       7       0.00              1
9998          772   Germany       1   42       3   75075.31              2
9999          792    France       0   28       4  130142.79              1
```

```
      HasCrCard  IsActiveMember  EstimatedSalary  Exited
0             1               1        101348.88       1
1             0               1        112542.58       0
2             1               0        113931.57       1
3             0               0         93826.63       0
4             1               1         79084.10       0
…           …             …               …      …
9995          1               0         96270.64       0
9996          1               1        101699.77       0
9997          0               1         42085.58       1
9998          1               0         92888.52       1
9999          1               0         38190.78       0

[10000 rows x 11 columns]
```

[5]:
```python
## Onehot encode "Geography"
from sklearn.preprocessing import OneHotEncoder
onehot_encoder_geo=OneHotEncoder()
geo_encoder= onehot_encoder_geo.fit_transform(data[['Geography']])
geo_encoder
```

[5]:
```
<Compressed Sparse Row sparse matrix of dtype 'float64'
        with 10000 stored elements and shape (10000, 3)>
```

[6]:
```python
geo_encoder.toarray()
```

[6]:
```
array([[1., 0., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       …,
       [1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.]])
```

[7]:
```python
onehot_encoder_geo.get_feature_names_out(['Geography'])
```

[7]:
```
array(['Geography_France', 'Geography_Germany', 'Geography_Spain'],
      dtype=object)
```

[8]:
```python
geo_encoder_df=pd.DataFrame(geo_encoder.toarray(), columns=onehot_encoder_geo.
 ↪get_feature_names_out(['Geography']))
geo_encoder_df
```

[8]:
```
      Geography_France  Geography_Germany  Geography_Spain
0                  1.0                0.0              0.0
1                  0.0                0.0              1.0
2                  1.0                0.0              0.0
3                  1.0                0.0              0.0
```

```
4                    0.0                 0.0                1.0
…                  …                 …               …
9995                 1.0                 0.0                0.0
9996                 1.0                 0.0                0.0
9997                 1.0                 0.0                0.0
9998                 0.0                 1.0                0.0
9999                 1.0                 0.0                0.0

[10000 rows x 3 columns]
```

[9]:
```
## Combine one hot encoder columns with the original data
data = pd.concat([data.drop('Geography', axis=1), geo_encoder_df], axis=1)
data.head()
```

[9]:
```
   CreditScore  Gender  Age  Tenure    Balance  NumOfProducts  HasCrCard  \
0          619       0   42       2       0.00              1          1
1          608       0   41       1   83807.86              1          0
2          502       0   42       8  159660.80              3          1
3          699       0   39       1       0.00              2          0
4          850       0   43       2  125510.82              1          1

   IsActiveMember  EstimatedSalary  Exited  Geography_France  \
0               1        101348.88       1               1.0
1               1        112542.58       0               0.0
2               0        113931.57       1               1.0
3               0         93826.63       0               1.0
4               1         79084.10       0               0.0

   Geography_Germany  Geography_Spain
0                0.0              0.0
1                0.0              1.0
2                0.0              0.0
3                0.0              0.0
4                0.0              1.0
```

[10]:
```
## Save the encoders and standeredScaler
with open('label_encoder_gender.pkl', 'wb') as file:
        pickle.dump(label_encoder_gender, file)

with open('onehot_encoder_geo.pkl', 'wb') as file:
        pickle.dump(onehot_encoder_geo, file)
```

[11]:
```
data.head()
```

[11]:
```
   CreditScore  Gender  Age  Tenure    Balance  NumOfProducts  HasCrCard  \
0          619       0   42       2       0.00              1          1
1          608       0   41       1   83807.86              1          0
```

```
2          502       0   42       8  159660.80              3            1
3          699       0   39       1       0.00              2            0
4          850       0   43       2  125510.82              1            1

   IsActiveMember  EstimatedSalary  Exited  Geography_France  \
0               1        101348.88       1               1.0
1               1        112542.58       0               0.0
2               0        113931.57       1               1.0
3               0         93826.63       0               1.0
4               1         79084.10       0               0.0

   Geography_Germany  Geography_Spain
0                0.0              0.0
1                0.0              1.0
2                0.0              0.0
3                0.0              0.0
4                0.0              1.0
```

[12]:
```
## Divide the dataset into independent and dependent features
X = data.drop('Exited', axis=1)
y = data['Exited']

## Split the data in the training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

## Scale these features
scaler = StandardScaler()
X_train= scaler.fit_transform(X_train)
X_test= scaler.transform(X_test)
```

[13]:
```
X_train
```

[13]:
```
array([[ 0.35649971,  0.91324755, -0.6557859 , …,  1.00150113,
        -0.57946723, -0.57638802],
       [-0.20389777,  0.91324755,  0.29493847, …, -0.99850112,
         1.72572313, -0.57638802],
       [-0.96147213,  0.91324755, -1.41636539, …, -0.99850112,
        -0.57946723,  1.73494238],
       …,
       [ 0.86500853, -1.09499335, -0.08535128, …,  1.00150113,
        -0.57946723, -0.57638802],
       [ 0.15932282,  0.91324755,  0.3900109 , …,  1.00150113,
        -0.57946723, -0.57638802],
       [ 0.47065475,  0.91324755,  1.15059039, …, -0.99850112,
         1.72572313, -0.57638802]])
```

[14]:
```python
with open('scaler.pkl', 'wb') as file:
        pickle.dump(scaler, file)
```

[15]:
```python
data
```

[15]:
```
      CreditScore  Gender  Age  Tenure    Balance  NumOfProducts  HasCrCard  \
0             619       0   42       2       0.00              1          1
1             608       0   41       1   83807.86              1          0
2             502       0   42       8  159660.80              3          1
3             699       0   39       1       0.00              2          0
4             850       0   43       2  125510.82              1          1
…             …     …   …     …       …           …          …
9995          771       1   39       5       0.00              2          1
9996          516       1   35      10   57369.61              1          1
9997          709       0   36       7       0.00              1          0
9998          772       1   42       3   75075.31              2          1
9999          792       0   28       4  130142.79              1          1

      IsActiveMember  EstimatedSalary  Exited  Geography_France  \
0                  1        101348.88       1               1.0
1                  1        112542.58       0               0.0
2                  0        113931.57       1               1.0
3                  0         93826.63       0               1.0
4                  1         79084.10       0               0.0
…              …              …     …              …
9995               0         96270.64       0               1.0
9996               1        101699.77       0               1.0
9997               1         42085.58       1               1.0
9998               0         92888.52       1               0.0
9999               0         38190.78       0               1.0

      Geography_Germany  Geography_Spain
0                   0.0              0.0
1                   0.0              1.0
2                   0.0              0.0
3                   0.0              0.0
4                   0.0              1.0
…                 …              …
9995                0.0              0.0
9996                0.0              0.0
9997                0.0              0.0
9998                1.0              0.0
9999                0.0              0.0

[10000 rows x 13 columns]
```

2 Step by Step Training with ANN With Optimizer and Loss function

[16]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.callbacks import EarlyStopping, TensorBoard
import datetime
```

2.1 Build ANN Model

[17]:
```
(X_train.shape[1],)
```

[17]:
```
(12,)
```

So our ANN model is also called as Sequential model so in case of our project we have: - total 12 input data. - first hidden layer of 64 nodes or neurons. - second hidden layer of 32 nodes or neurons. - And the output layer have 1 neuron.

we use the Dense to create the layer which contain the neurons.

[18]:
```
model = Sequential([
        Dense(64, activation='relu', input_shape=(X_train.shape[1],)), ## HL-1␣
 ↪Connected with input layer
        Dense(32, activation='relu'), ## HL-2
        Dense(1, activation='sigmoid') ## Output layer
])
```

```
c:\Users\Uditya\Desktop\Deep Learning\Deep Learning for Beginner\venv\lib\site-
packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

[19]:
```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	832
dense_1 (Dense)	(None, 32)	2,080
dense_2 (Dense)	(None, 1)	33

Total params: 2,945 (11.50 KB)

Trainable params: 2,945 (11.50 KB)

Non-trainable params: 0 (0.00 B)

[20]:
```
opt = tf.keras.optimizers.Adam(learning_rate=0.01)
loss = tf.keras.losses.BinaryCrossentropy()
```

[21]:
```
# In order to forward and backward propagation we use to compile
# compile the model

model.compile(optimizer="adam", loss="binary_crossentropy",␣
 ↪metrics=['accuracy'])
```

[22]:
```
## Setup the Tensorboard
from tensorflow.keras.callbacks import EarlyStopping, TensorBoard

log_dir= "log/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorflow_callback=TensorBoard(log_dir=log_dir, histogram_freq=1)
```

Early Stopping: If we are train our model and let's say we are training it for 100 Epoch and after the training of 30 Epoch our loss value is not decreasing then no need to run the other 70 Epoch so for this we are using the EarlyStopping

[23]:
```
## setup Early Stopping
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=10,␣
 ↪restore_best_weights=True)
```

2.1.1 Training the model

[24]:
```
history = model.fit(
        X_train, y_train, validation_data=(X_test, y_test), epochs = 100,
        callbacks=[tensorflow_callback, early_stopping_callback]
)
```

```
Epoch 1/100
250/250              4s 8ms/step -
accuracy: 0.8016 - loss: 0.4754 - val_accuracy: 0.8285 - val_loss: 0.3965
Epoch 2/100
250/250              2s 6ms/step -
accuracy: 0.8377 - loss: 0.3950 - val_accuracy: 0.8450 - val_loss: 0.3668
Epoch 3/100
250/250              3s 12ms/step -
accuracy: 0.8490 - loss: 0.3696 - val_accuracy: 0.8535 - val_loss: 0.3582
Epoch 4/100
250/250              5s 17ms/step -
```

```
accuracy: 0.8584 - loss: 0.3478 - val_accuracy: 0.8565 - val_loss: 0.3481
Epoch 5/100
250/250            2s 6ms/step -
accuracy: 0.8576 - loss: 0.3460 - val_accuracy: 0.8580 - val_loss: 0.3496
Epoch 6/100
250/250            2s 6ms/step -
accuracy: 0.8588 - loss: 0.3388 - val_accuracy: 0.8555 - val_loss: 0.3475
Epoch 7/100
250/250            2s 6ms/step -
accuracy: 0.8619 - loss: 0.3401 - val_accuracy: 0.8630 - val_loss: 0.3423
Epoch 8/100
250/250            3s 9ms/step -
accuracy: 0.8625 - loss: 0.3266 - val_accuracy: 0.8575 - val_loss: 0.3499
Epoch 9/100
250/250            1s 5ms/step -
accuracy: 0.8563 - loss: 0.3330 - val_accuracy: 0.8610 - val_loss: 0.3403
Epoch 10/100
250/250            2s 8ms/step -
accuracy: 0.8695 - loss: 0.3220 - val_accuracy: 0.8605 - val_loss: 0.3419
Epoch 11/100
250/250            4s 14ms/step -
accuracy: 0.8675 - loss: 0.3190 - val_accuracy: 0.8535 - val_loss: 0.3488
Epoch 12/100
250/250            3s 7ms/step -
accuracy: 0.8668 - loss: 0.3123 - val_accuracy: 0.8580 - val_loss: 0.3441
Epoch 13/100
250/250            2s 8ms/step -
accuracy: 0.8643 - loss: 0.3242 - val_accuracy: 0.8630 - val_loss: 0.3420
Epoch 14/100
250/250            2s 6ms/step -
accuracy: 0.8644 - loss: 0.3190 - val_accuracy: 0.8600 - val_loss: 0.3436
Epoch 15/100
250/250            2s 7ms/step -
accuracy: 0.8724 - loss: 0.3119 - val_accuracy: 0.8595 - val_loss: 0.3411
Epoch 16/100
250/250            2s 8ms/step -
accuracy: 0.8740 - loss: 0.3109 - val_accuracy: 0.8595 - val_loss: 0.3466
Epoch 17/100
250/250            2s 8ms/step -
accuracy: 0.8741 - loss: 0.3101 - val_accuracy: 0.8585 - val_loss: 0.3427
Epoch 18/100
250/250            2s 7ms/step -
accuracy: 0.8689 - loss: 0.3155 - val_accuracy: 0.8595 - val_loss: 0.3440
Epoch 19/100
250/250            2s 7ms/step -
accuracy: 0.8719 - loss: 0.3048 - val_accuracy: 0.8590 - val_loss: 0.3459
```

[25]:
```
model.save('model.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or
`keras.saving.save_model(model)`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')` or `keras.saving.save_model(model,
'my_model.keras')`.
```

2.1.2 Loading Tensorflow Extension

[26]:
```
%load_ext tensorboard
```

[28]:
```
%tensorboard --logdir log/fit/20250406-205147
```

```
Reusing TensorBoard on port 6008 (pid 14308), started 0:00:16 ago. (Use '!kill␣
↪14308' to kill it.)
```

```
<IPython.core.display.HTML object>
```