

▼ Artificial Neural Network

▼ Importing the libraries

```
import numpy as np
import pandas as pd
import tensorflow as tf
```

```
tf.__version__
```

```
↳ '2.2.0'
```

▼ Part 1 - Data Preprocessing

▼ Importing the dataset

```
dataset = pd.read_csv('Churn_Modelling.csv')
X = dataset.iloc[:, 3:-1].values
y = dataset.iloc[:, -1].values
```

```
print(X)
```

```
↳ [[619 'France' 'Female' ... 1 1 101348.88]
    [608 'Spain' 'Female' ... 0 1 112542.58]
    [502 'France' 'Female' ... 1 0 113931.57]
    ...
    [709 'France' 'Female' ... 0 1 42085.58]
    [772 'Germany' 'Male' ... 1 0 92888.52]
    [792 'France' 'Female' ... 1 0 38190.78]]
```

```
print(y)
```

```
↳ [1 0 1 ... 1 1 0]
```

▼ Encoding categorical data

Label Encoding the "Gender" column

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:, 2] = le.fit_transform(X[:, 2])
```

```
print(X)
```

```
[[619 'France' 0 ... 1 1 101348.88]
 [608 'Spain' 0 ... 0 1 112542.58]
 [502 'France' 0 ... 1 0 113931.57]
 ...
 [709 'France' 0 ... 0 1 42085.58]
 [772 'Germany' 1 ... 1 0 92888.52]
 [792 'France' 0 ... 1 0 38190.78]]
```

One Hot Encoding the "Geography" column

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])], remainder=
X = np.array(ct.fit_transform(X))
```

```
print(X)
```

```
[[1.0 0.0 0.0 ... 1 1 101348.88]
 [0.0 0.0 1.0 ... 0 1 112542.58]
 [1.0 0.0 0.0 ... 1 0 113931.57]
 ...
 [1.0 0.0 0.0 ... 0 1 42085.58]
 [0.0 1.0 0.0 ... 1 0 92888.52]
 [1.0 0.0 0.0 ... 1 0 38190.78]]
```

▼ Splitting the dataset into the Training set and Test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_s
```

▼ Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

▼ Part 2 - Building the ANN

▼ Initializing the ANN

```
ann=tf.keras.models.Sequential()
```

▼ Adding the input layer and the first hidden layer

```
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

▼ Adding the second hidden layer

```
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

▼ Adding the output layer

```
ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

▼ Part 3 - Training the ANN

▼ Compiling the ANN

```
ann.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

▼ Training the ANN on the Training set

```
ann.fit(X_train,y_train,batch_size=32,epochs=100)
```

▼ Part 4 - Making the predictions and evaluating the model

▼ Predicting the result of a single observation

Homework

Use our ANN model to predict if the customer with the following informations will leave the bank:

Geography: France

Credit Score: 600

Gender: Male

Age: 40 years old

Tenure: 3 years

Balance: \$ 60000

Number of Products: 2

Does this customer have a credit card? Yes

Is this customer an Active Member: Yes

Estimated Salary: \$ 50000

So, should we say goodbye to that customer?

Solution

```
print(ann.predict(sc.transform([[1,0,0,600,1,40,3,60000,2,1,1,50000]]))>0.5)
```

```
↳ [[False]]
```

Therefore, our ANN model predicts that this customer stays in the bank!

Important note 1: Notice that the values of the features were all input in a double pair of square brackets. That's because the "predict" method always expects a 2D array as the format of its inputs. And putting our values into a double pair of square brackets makes the input exactly a 2D array.

Important note 2: Notice also that the "France" country was not input as a string in the last column but as "1, 0, 0" in the first three columns. That's because of course the predict method expects the one-hot-encoded values of the state, and as we see in the first row of the matrix of features X, "France" was encoded as "1, 0, 0". And be careful to include these values in the first three columns, because the dummy variables are always created in the first columns.

▼ Predicting the Test set results

```
y_pred = ann.predict(X_test)
y_pred = (y_pred > 0.5)
```

```
y_pred = (y_pred > 0.5)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1))
```

```
↳ [[0 0]
    [0 1]
    [0 0]
    ...
    [0 0]
    [0 0]
    [0 0]]
```

▼ Making the Confusion Matrix

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

```
↳ [[1530  65]
    [ 207 198]]
0.864
```