# Neural Network

이원형 교수
Dr. Lee, WonHyong
whlee@handong.edu

## ❖ Tools

- ✓ Python (Numpy, Matplotlib, Pandas, etc.)
- ✓ Jupyter Notebook
- ✓ Colab
- ✓ Keras

Keras is most suitable for:

- Rapid Prototyping
- Small Dataset
- Multiple back-end support

TensorFlow is most suitable for:

- Large Dataset
- High Performance
- Functionality
- Object Detection

PyTorch is most suitable for:

- Flexibility
- Short Training Duration
- Debugging capabilities

- **Keras**
  - Official website
    - https://keras.io/
  - Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.
    - In our study, it is assumed that we are using Tensorflow (python version).

- **To test if Keras is successfully installed,**
  - Run python
  - Type
    - import tensorflow as tf
    - import keras
    - tf.__version__
    - keras.__version__

# INTRODUCTION TO KERAS

- **There are two ways to create a network model.**
  - ✓ Study
    - – https://keras.io/guides/functional_api/
    - – https://machinelearningmastery.com/keras-functional-api-deep-learning/
  1. Using Sequential API
     - – Intuitive, Easy, and it covers most of the existing deep learning models.
     - – But, for each layer, it is not available to have multiple previous/next layers.
  2. Using Functional API
     - – Still easy
     - – More flexible: Multiple different input sources, multiple output destinations, and re-using layers are available.

- **But! We are just going to follow example codes fist.**

## ❖ Basic Neural Network Implementation

✓ A question

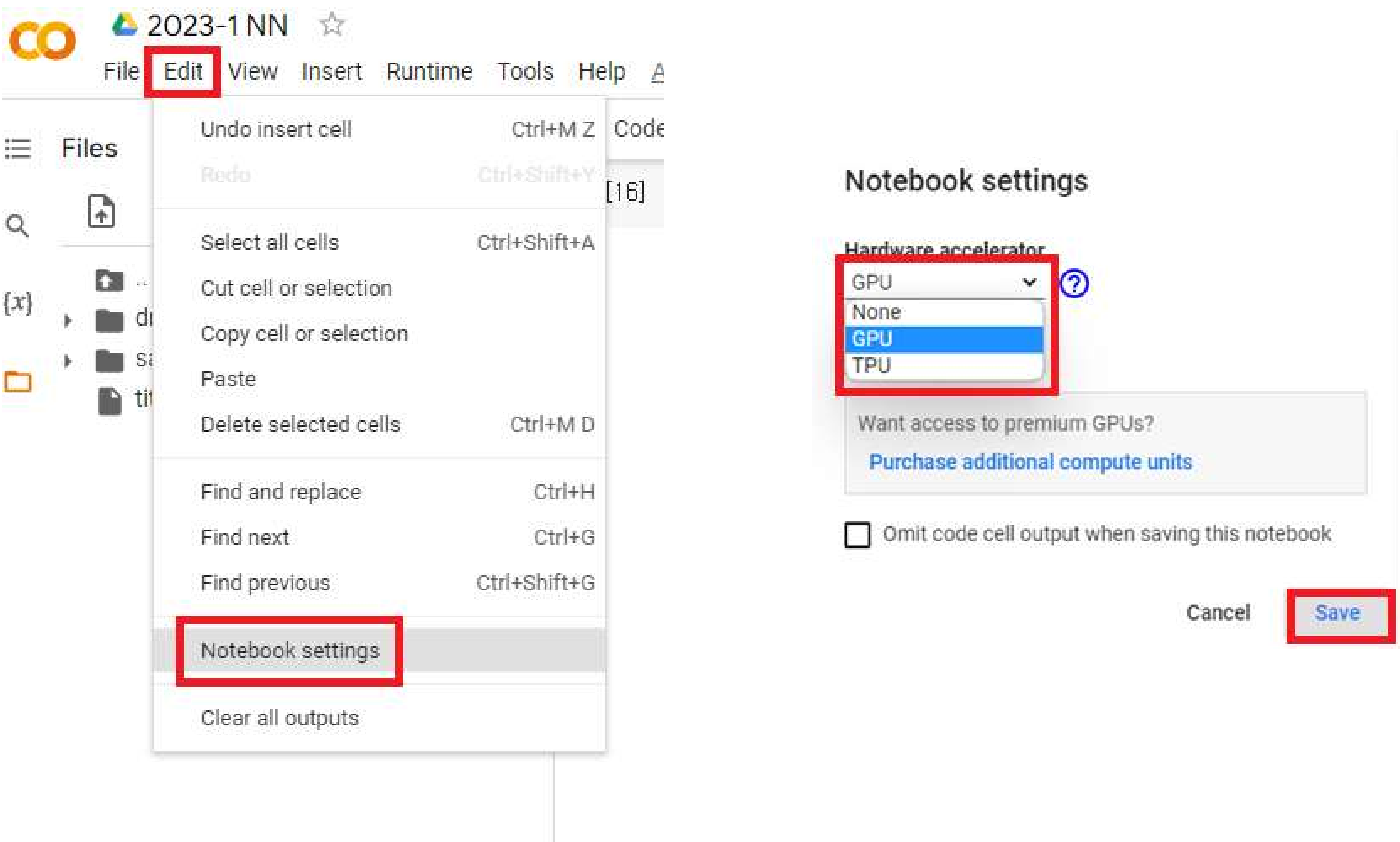
– Could Jack(Leonardo DiCaprio) have survived from the Titanic cruise ship?

# NEURAL NETWORK

❖ **What do we need?**

- ✓ Data: Titanic crew and passenger list with personal profile
  - You can easily get this information from Google search
  - or Download the file "titanic.xls" from HDLMS
- ✓ Pre-processing
  - To collect data only which have full information we want to see.
  - To divide data into a training set and a test(validation) set.
  - Reshape the data format
- ✓ Model selection
  - In this case, we will use a basic neural network which has one hidden layer and outputs one value in a range of 0-1.

# Colab Settings



## ❖ Data Importation

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

raw_data = pd.read_excel('titanic.xls')
raw_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 14 columns):
pclass       1309 non-null int64
survived     1309 non-null int64
name         1309 non-null object
sex          1309 non-null object
age          1046 non-null float64
sibsp        1309 non-null int64
parch        1309 non-null int64
ticket       1309 non-null object
fare         1308 non-null float64
cabin        295 non-null object
embarked     1307 non-null object
boat         486 non-null object
body         121 non-null float64
home.dest    745 non-null object
dtypes: float64(3), int64(4), object(7)
memory usage: 143.2+ KB
```

Number of siblings or spouse

Number of parant or children

## ❖ Data Importation

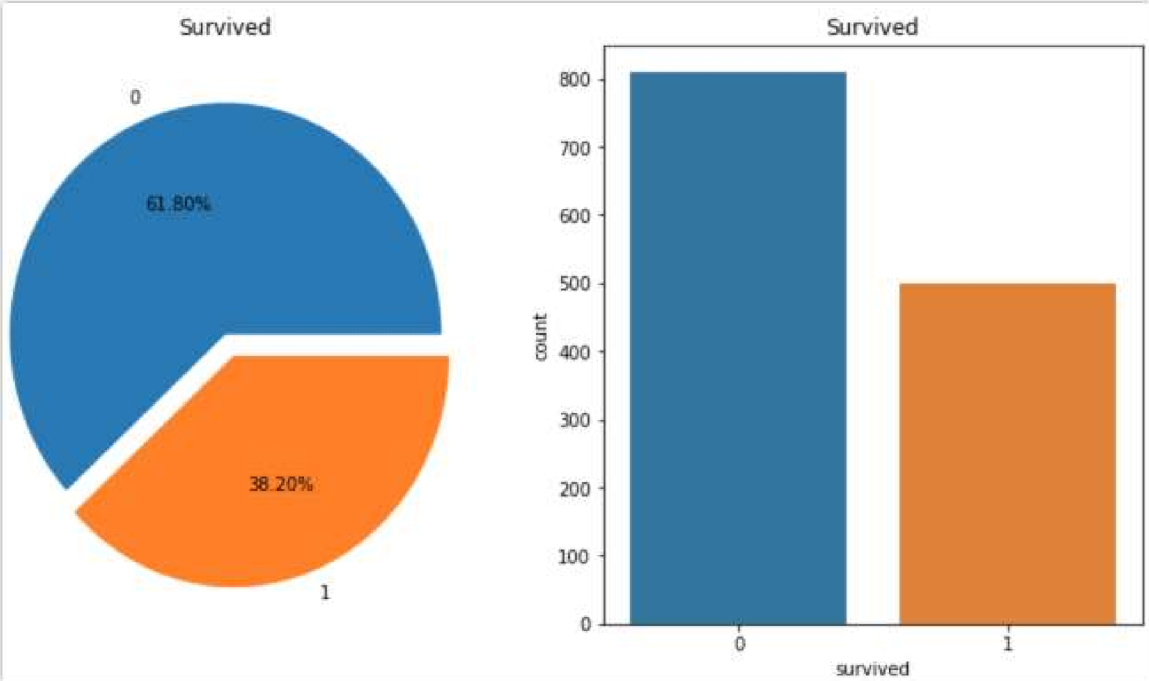
```
raw_data.describe()
```

```
raw_data.describe()
```

| | pclass | survived | age | sibsp | parch | fare | body |
|---|---|---|---|---|---|---|---|
| count | 1309.000000 | 1309.000000 | 1046.000000 | 1309.000000 | 1309.000000 | 1308.000000 | 121.000000 |
| mean | 2.294882 | 0.381971 | 29.881135 | 0.498854 | 0.385027 | 33.295479 | 160.809917 |
| std | 0.837836 | 0.486055 | 14.413500 | 1.041658 | 0.865560 | 51.758668 | 97.696922 |
| min | 1.000000 | 0.000000 | 0.166700 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| 25% | 2.000000 | 0.000000 | 21.000000 | 0.000000 | 0.000000 | 7.895800 | 72.000000 |
| 50% | 3.000000 | 0.000000 | 28.000000 | 0.000000 | 0.000000 | 14.454200 | 155.000000 |
| 75% | 3.000000 | 1.000000 | 39.000000 | 1.000000 | 0.000000 | 31.275000 | 256.000000 |
| max | 3.000000 | 1.000000 | 80.000000 | 8.000000 | 9.000000 | 512.329200 | 328.000000 |

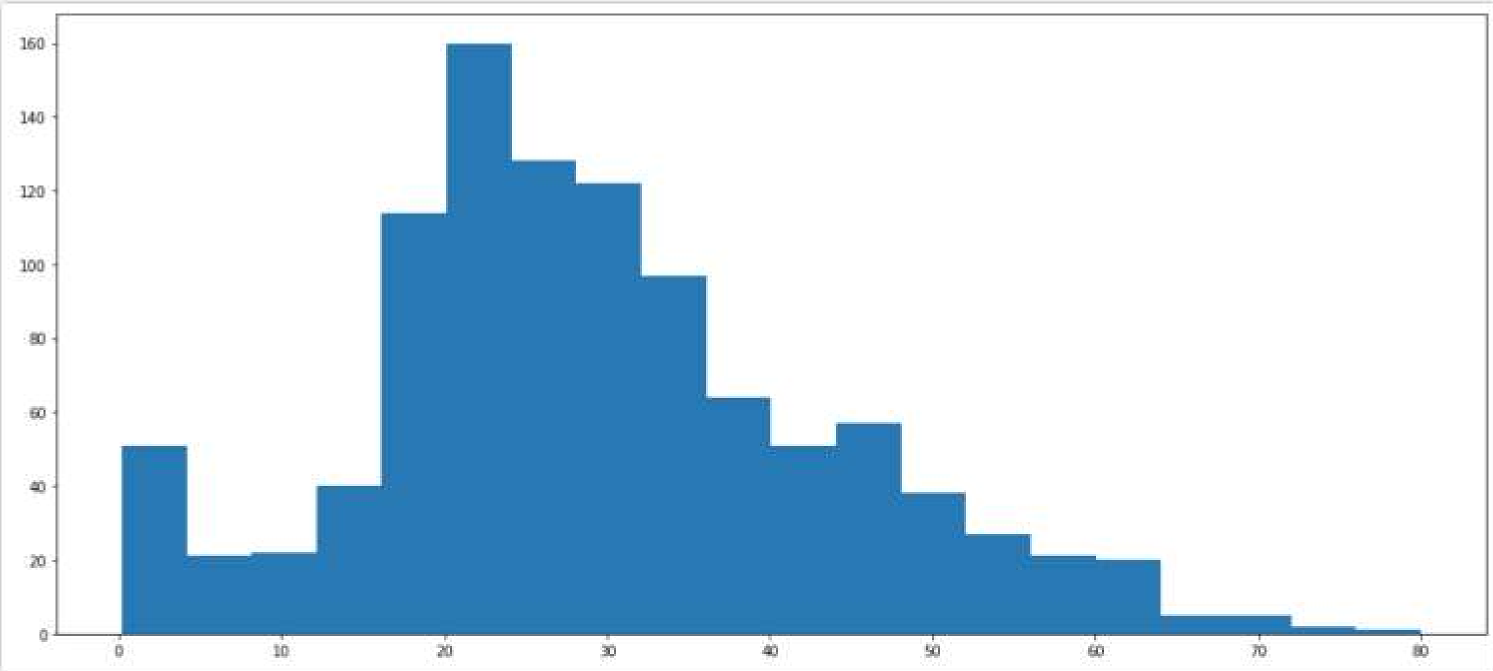
## ❖ Data Visualization

```
f,ax=plt.subplots(1,2,figsize=(12,6))

raw_data['survived'].value_counts().plot.pie(explode=[0,0.1],autopct='%1.2f%%',ax=ax[0])
ax[0].set_title('Survived')
ax[0].set_ylabel('')

sns.countplot('survived',data=raw_data,ax=ax[1])
ax[1].set_title('Survived')
plt.show()
```



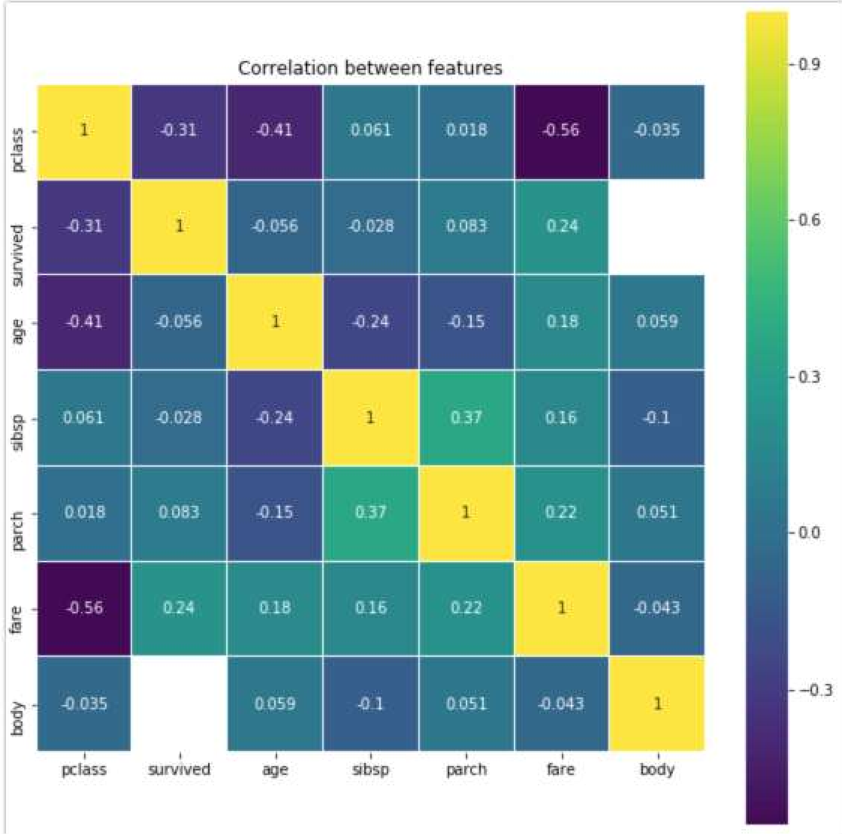
## ❖ Data Visualization

```
raw_data['age'].hist(bins=20,figsize=(18,8),grid=False);
```



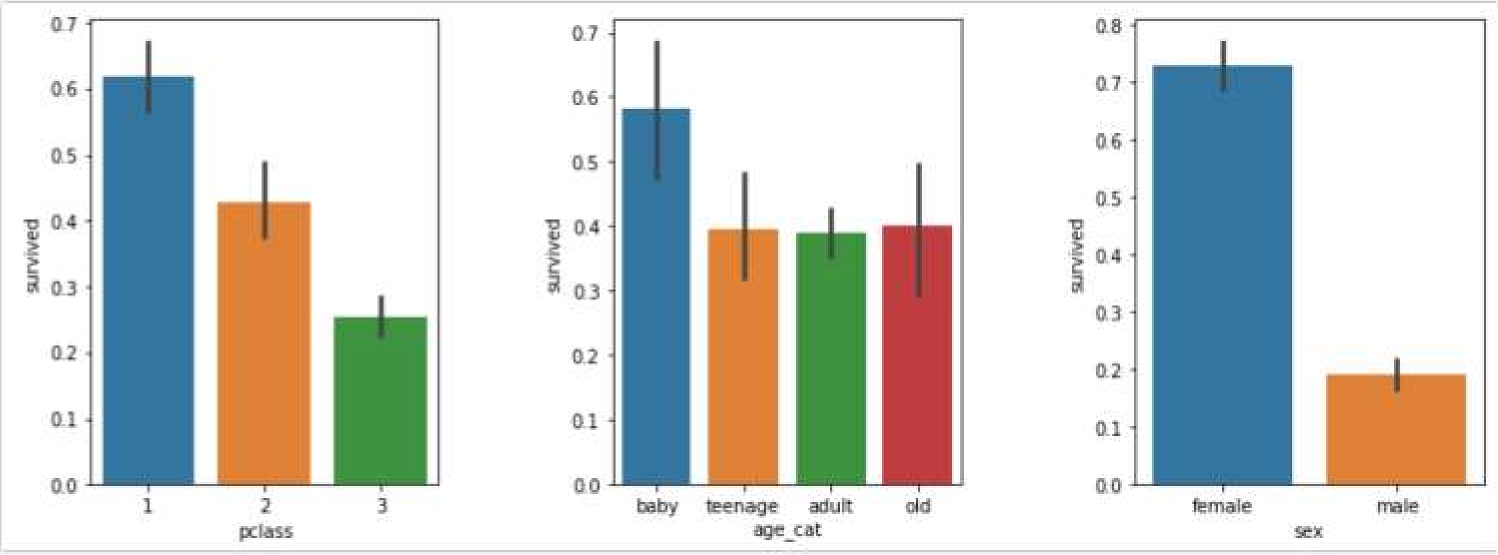
## ❖ Data Visualization

```
plt.figure(figsize=(10, 10))
sns.heatmap(raw_data.corr(), linewidths=0.01, square=True,
            annot=True, cmap=plt.cm.viridis, linecolor="white")
plt.title('Correlation between features')
plt.show()
```



## ❖ Data Visualization

```
raw_data['age_cat'] = pd.cut(raw_data['age'], bins=[0, 10, 20, 50, 100],
                             include_lowest=True, labels=['baby', 'teenage', 'adult', 'old'])
plt.figure(figsize=[12,4])
plt.subplot(131)
sns.barplot('pclass', 'survived', data=raw_data)
plt.subplot(132)
sns.barplot('age_cat', 'survived', data=raw_data)
plt.subplot(133)
sns.barplot('sex', 'survived', data=raw_data)
plt.subplots_adjust(top=1, bottom=0.1, left=0.10, right=1, hspace=0.5, wspace=0.5)
plt.show()
```



## Pre-processing

- We are going to use
  - input data: pclass, sex, age, sibsp, parch, fare
  - output data: survived

## ❖ Pre-processing

- ✓ Set to 1 for the 'female' data in the 'sex' column.
- ✓ Set to 0 for the 'male' data in the 'sex' column.

```python
tmp = []
for each in raw_data['sex']:
    if each == 'female':
        tmp.append(1)
    elif each == 'male':
        tmp.append(0)
    else:
        tmp.append(np.nan)

raw_data['sex'] = tmp
```

## ❖ Pre-processing

✓ Re-declare the format of some data to 'float' that we are going to use.

```
raw_data['survived'] = raw_data['survived'].astype('float')
raw_data['pclass'] = raw_data['pclass'].astype('float')
raw_data['sex'] = raw_data['sex'].astype('float')
raw_data['sibsp'] = raw_data['sibsp'].astype('float')
raw_data['parch'] = raw_data['parch'].astype('float')
raw_data['fare'] = raw_data['fare'].astype('float')
```

## ❖ Pre-processing

- ✓ Remove data which has no information in any of columns that we care.

```
raw_data = raw_data[raw_data['age'].notnull()]
raw_data = raw_data[raw_data['sibsp'].notnull()]
raw_data = raw_data[raw_data['parch'].notnull()]
raw_data = raw_data[raw_data['fare'].notnull()]
```

# Neural Network

## ❖ Pre-processing

```
raw_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 14 columns):
pclass       1309 non-null int64
survived     1309 non-null int64
name         1309 non-null object
sex          1309 non-null object
age          1046 non-null float64
sibsp        1309 non-null int64
parch        1309 non-null int64
ticket       1309 non-null object
fare         1308 non-null float64
cabin        295 non-null object
embarked     1307 non-null object
boat         486 non-null object
body         121 non-null float64
home.dest    745 non-null object
dtypes: float64(3), int64(4), object(7)
memory usage: 143.2+ KB
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1045 entries, 0 to 1308
Data columns (total 16 columns):
pclass       1045 non-null float64
survived     1045 non-null float64
name         1045 non-null object
sex          1045 non-null float64
age          1045 non-null float64
sibsp        1045 non-null float64
parch        1045 non-null float64
ticket       1045 non-null object
fare         1045 non-null float64
cabin        272 non-null object
embarked     1043 non-null object
boat         417 non-null object
body         119 non-null float64
home.dest    685 non-null object
age_cat      1045 non-null category
title        1045 non-null object
dtypes: category(1), float64(8), object(7)
memory usage: 131.8+ KB
```

## ❖ Pre-processing

- ✓ Select input data: pclass, sex, age, sibsp, parch, fare
- ✓ Select output data: survived
- ✓ Save 10% of data for validation (test set)

```
x_data = raw_data.values[:, [0,3,4,5,6,8]]
y_data = raw_data.values[:, [1]]

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_data, y_data,
                                                    test_size=0.1, random_state=7)
```

## ❖ Data Reshaping

```
X_train = np.asarray(X_train).astype(np.float32)
X_test = np.asarray(X_test).astype(np.float32)
y_train = np.asarray(y_train).astype(np.float32)
y_test = np.asarray(y_test).astype(np.float32)
```

## ❖ Tensorflow and Keras Importation

```
import tensorflow as tf
import keras
```

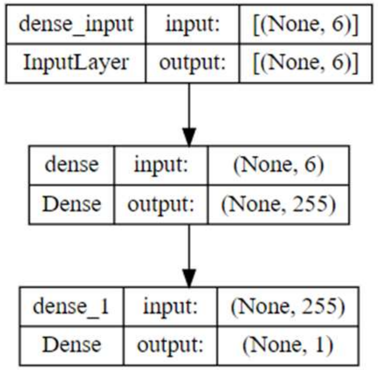
## ❖ Model Structure Creation

```python
from keras.models import Sequential
from keras.layers.core import Dense
np.random.seed(7)
```

```python
model = Sequential()
model.add(Dense(255, input_shape=(6,), activation='relu'))
model.add(Dense((1), activation='sigmoid'))
model.compile(loss='mse', optimizer='Adam', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 255)               1785

 dense_1 (Dense)             (None, 1)                 256

=================================================================
Total params: 2,041
Trainable params: 2,041
Non-trainable params: 0
_________________________________________________________________
```

```python
from IPython.display import SVG
from keras.utils.vis_utils import model_to_dot

SVG(model_to_dot(model, show_shapes=True).create(prog='dot', format='svg'))
```



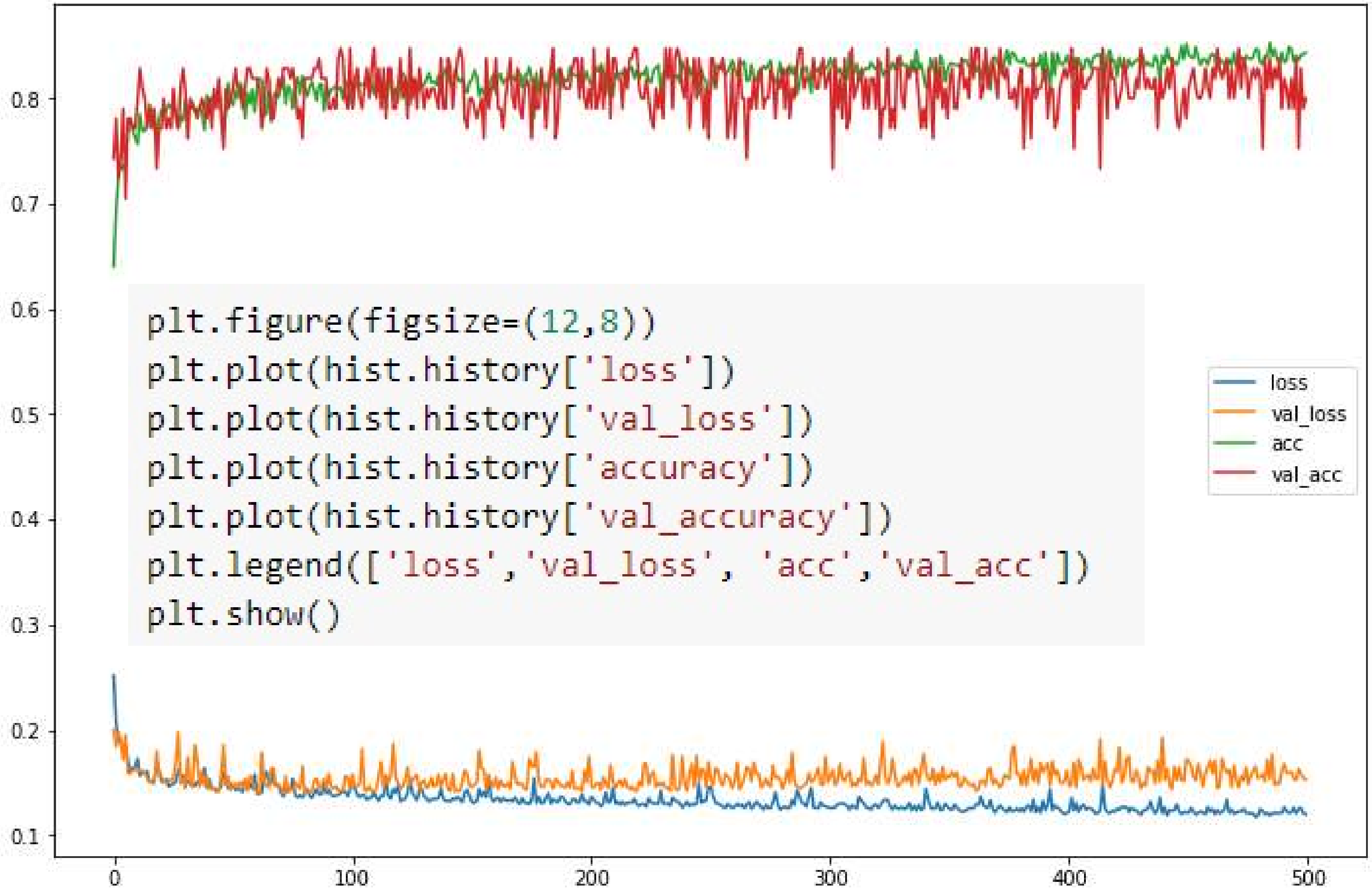
# NN PRACTICE

## ❖ Training

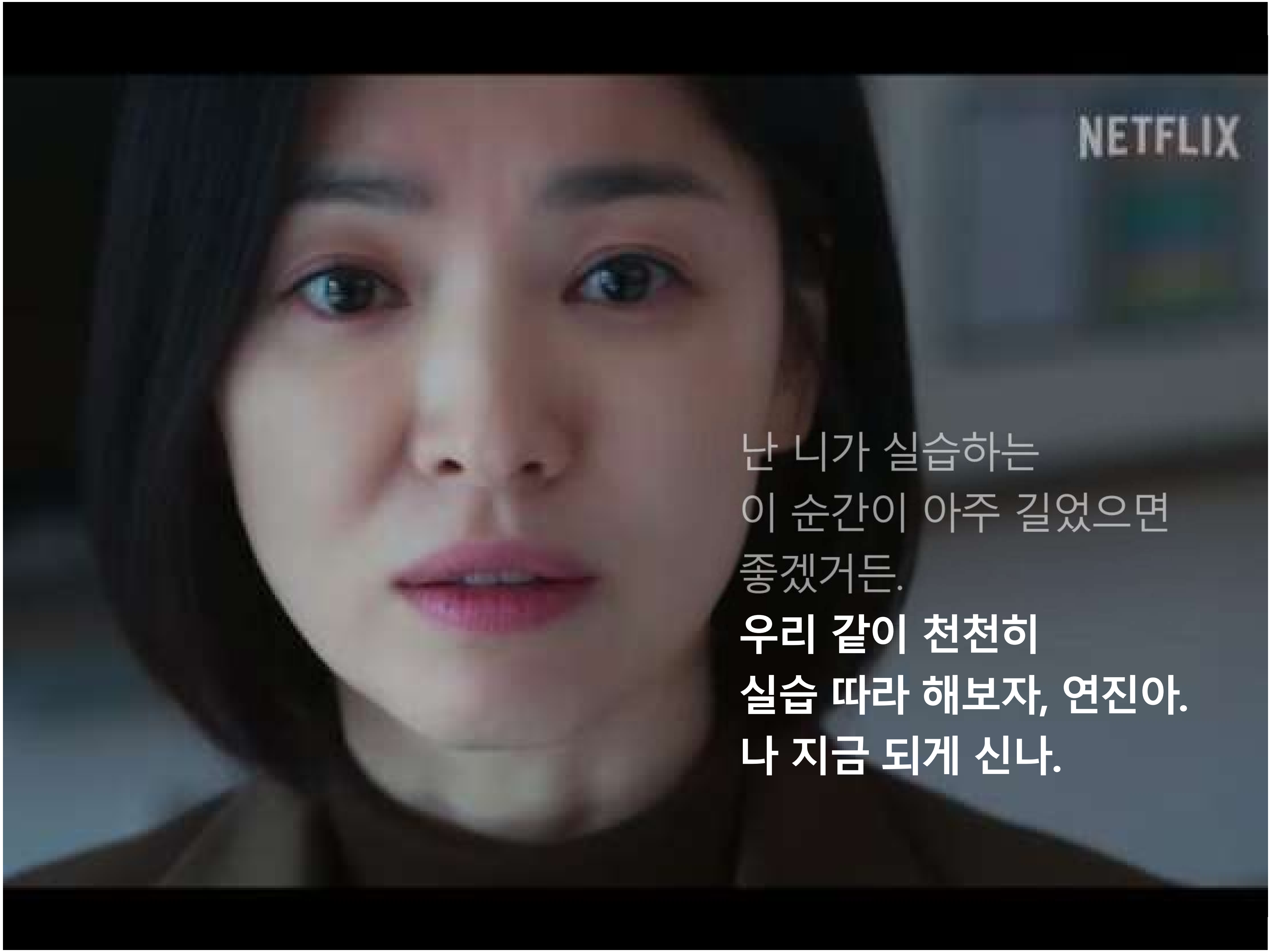
```
hist = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=500)
```

```
Epoch 1/500
30/30 [==============================] - 0s 5ms/step - loss: 0.1192 - accuracy: 0.8426 - val_loss: 0.1739 - val_accuracy: 0.8190
Epoch 2/500
30/30 [==============================] - 0s 4ms/step - loss: 0.1204 - accuracy: 0.8436 - val_loss: 0.1546 - val_accuracy: 0.8095
Epoch 3/500
30/30 [==============================] - 0s 3ms/step - loss: 0.1225 - accuracy: 0.8436 - val_loss: 0.1491 - val_accuracy: 0.8381
Epoch 4/500
30/30 [==============================] - 0s 3ms/step - loss: 0.1194 - accuracy: 0.8447 - val_loss: 0.1564 - val_accuracy: 0.8095
Epoch 5/500
30/30 [==============================] - 0s 2ms/step - loss: 0.1186 - accuracy: 0.8426 - val_loss: 0.1534 - val_accuracy: 0.8095
Epoch 6/500
30/30 [==============================] - 0s 2ms/step - loss: 0.1207 - accuracy: 0.8415 - val_loss: 0.1661 - val_accuracy: 0.8190
Epoch 7/500
30/30 [==============================] - 0s 4ms/step - loss: 0.1242 - accuracy: 0.8362 - val_loss: 0.1517 - val_accuracy: 0.8095
Epoch 8/500
30/30 [==============================] - 0s 3ms/step - loss: 0.1308 - accuracy: 0.8255 - val_loss: 0.1505 - val_accuracy: 0.8286
Epoch 9/500
30/30 [==============================] - 0s 3ms/step - loss: 0.1218 - accuracy: 0.8415 - val_loss: 0.1592 - val_accuracy: 0.7714
Epoch 10/500
30/30 [==============================] - 0s 3ms/step - loss: 0.1223 - accuracy: 0.8351 - val_loss: 0.1548 - val_accuracy: 0.8190
Epoch 11/500
30/30 [==============================] - 0s 2ms/step - loss: 0.1244 - accuracy: 0.8447 - val_loss: 0.1510 - val_accuracy: 0.8190
```

## ❖ Result Visualization




```python
plt.figure(figsize=(12,8))
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.legend(['loss','val_loss', 'acc','val_acc'])
plt.show()
```


NETFLIX
난 니가 실습하는
이 순간이 아주 길었으면
좋겠거든.
우리 같이 천천히
실습 따라 해보자, 연진아.
나 지금 되게 신나.

❖ **Prediction!**

```
dicaprio = np.array([3., 0., 19., 0., 0., 5.]).reshape(1,6)
winslet = np.array([1., 1., 17., 1., 2., 100.]).reshape(1,6)
```

```
model.predict(dicaprio)
```

```
array([[0.14074111]], dtype=float32)
```

```
model.predict(winslet)
```

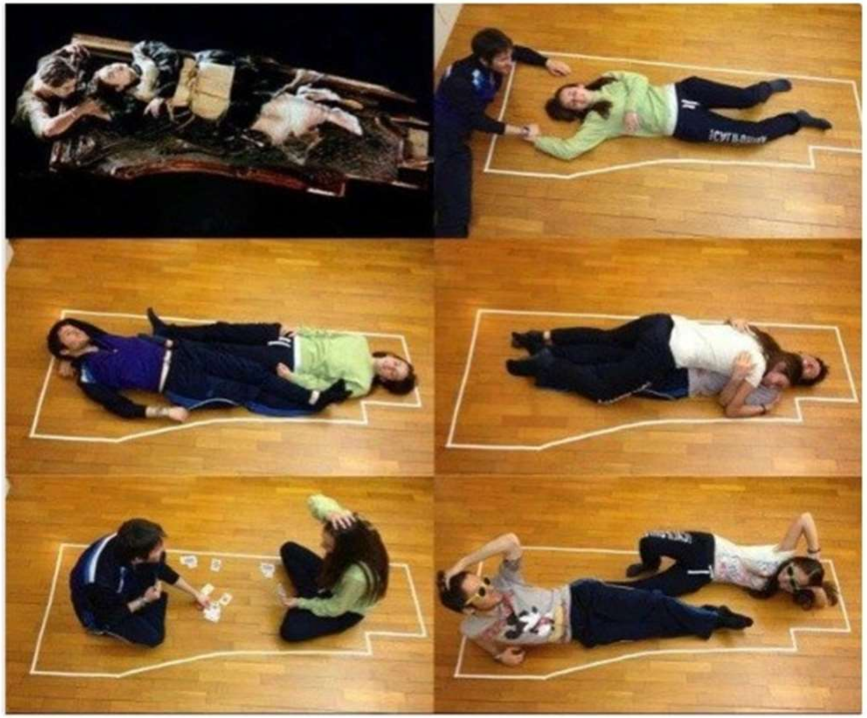
```
array([[0.9997488]], dtype=float32)
```


미국 탐색구조특수부대(US SAR TF)에 따르면, 0°C 이하 물에서 생존 기대시간은 15분~45분, 10°C 미만에서는 최대 3시간으로 잡고 있다.

Apr 28, 2014

hellodd.com
https://www.hellodd.com › 뉴스

차가운 바다에서 익사보다 무서운 것은? < 뉴스 ... - 헬로디디



Jack and Rose could have both fit on that wooden plank...quite comfortably

Thank you!