1. Problem Description

Using TensorFlow to build a neural network to analyze data set a from heart UCI healthy study to classify and predict the presence or absence of heart disease.

2. Importing Classes and Functions

```
# data processing, CSV file I/O
import pandas as pd
# linear algebra
import numpy as np
# Deep Learning Libraries
import tensorflow as tf
# Misc. Libraries
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, StratifiedKFold
# Turn off TF warnings
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
```

3. Loading & Spliting The Dataset

```
# Load "heart.csv" data
heart_data_df = pd.read_csv("heart.csv")

# Splitting data into features (X) and label/target (Y)
X = heart_data_df.drop("target", axis=1)
Y = heart_data_df["target"]

# Split data set into 15% test and 85% training (which will be splitted during training)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.15)
```

I did not split clearly split data into 70% training, 15% validation, and 15% test in the beginning. I first split the data into 15% testing & 85% training. The training data set will then be split again into training and validation with tensorflow fit() function and K-fold cross validation technique.

4. Prepare the data: Encoding categorical data into one-hot & normalizing true number data.

```
# One-hot encoding for categorical data
X = pd.get_dummies(X, prefix_sep='_', drop_first=False,
                   columns=["sex", "cp", "fbs", "restecg", "exang", "slope", "ca"])
X = pd.get_dummies(X, prefix_sep='_', drop_first=True,
                   columns=['thal']) # thal is 1-3 so drop_first=True

# Standardising/Normalizing true number data
continuous_columns = ["age", "trestbps", "chol", "thalach", "oldpeak"]
sc = StandardScaler(copy=False)
X[continuous_columns] = sc.fit_transform(X[continuous_columns])
```

5. Our Neural Network Model

```
# Create and compile our model
model = tf.keras.Sequential(
    [
        tf.keras.layers.Dense(units=15, activation="relu"),
        tf.keras.layers.Dense(units=1, activation="softmax"),
    ]
)
model.compile(optimizer="adam", loss='binary_crossentropy' , metrics=["acc"])
```

- A "*softmax*" activation function is used in the output layer to ensure the output values are in the range of 0 and 1 and may be used as predicted probabilities.
- 1 hidden layer inside which is Rectified Linear.
- The network uses the efficient Adam gradient descent optimization algorithm with a binary_crossentropy loss function.

6. Train & Evaluate The Model with k-Fold Cross Validation

- I use the StratifiedKFold class from the scikit-learn Python machine learning library to split up the training dataset into 7 folds. The folds are stratified, meaning that the algorithm attempts to balance the number of instances of each class in each fold.

- Training the model for 50 epochs seems to be pretty fine in order to avoid overfitting. I already performed training with 200 and 1000 epochs and around 50 epochs was the reasonable number of epochs for early stopping.

```python
# load training data and split into 7 folds (close to 15% validation set) for cross validation
folds = list(StratifiedKFold(n_splits=7, shuffle=True, random_state=1).split(X_train, y_train))

# train our model using K-fold cross validation
for j, (train_idx, val_idx) in enumerate(folds):
    print('\nFold ', j)
    X_train_cv = X_train.iloc[train_idx]
    y_train_cv = y_train.iloc[train_idx]
    X_valid_cv = X_train.iloc[val_idx]
    y_valid_cv = y_train.iloc[val_idx]

    model.fit(np.array(X_train_cv), np.array(y_train_cv), epochs=200,
              validation_data=(np.array(X_valid_cv), np.array(y_valid_cv)))
    print(model.evaluate(np.array(X_valid_cv), np.array(y_valid_cv)))
```

- Run **fit() function** train the neural network. Results: **55.56% accuracy**

```python
In [9]: # train our model using K-fold cross validation
        for j, (train_idx, val_idx) in enumerate(folds):
            print('\nFold ', j)
            X_train_cv = X_train.iloc[train_idx]
            y_train_cv = y_train.iloc[train_idx]
            X_valid_cv = X_train.iloc[val_idx]
            y_valid_cv = y_train.iloc[val_idx]

            model.fit(np.array(X_train_cv), np.array(y_train_cv), epochs=200,
                      validation_data=(np.array(X_valid_cv), np.array(y_valid_cv)))
            print(model.evaluate(np.array(X_valid_cv), np.array(y_valid_cv)))
```

```
c: 0.5430 - val_loss: 6.8148 - val_acc: 0.5556
Epoch 200/200
221/221 [==============================] - 0s 88us/sample - loss: 7.0075 - ac
c: 0.5430 - val_loss: 6.8148 - val_acc: 0.5556
36/1 [=====================================================================
=============================================================================
=============================================================================
=============================================================================
=============================================================================
=============================================================================
=============================================================================
=============================================================================
=============================================================================
=============================================================================
=============================================================================
=============================================================================
=============================================================================
=============================================================================
========] - 0s 108us/sample - loss: 5.3240 - acc: 0.5556
[6.814772764841716, 0.5555556]
```

- Run **evaluate() function** to test the final model against your earlier test data (15% test data)

```
# evaluate against test data
model.evaluate(np.array(X_test), np.array(y_test))
```

- Result: **54.34% accuracy**

```
In [10]: # evaluate against test data
         model.evaluate(np.array(X_test), np.array(y_test))

         46/1 [=========================================================================
         ===============================================================================
         ===============================================================================
         ===============================================================================
         ===============================================================================
         ===============================================================================
         ===============================================================================
         ===============================================================================
         ===============================================================================
         ===============================================================================
         ===============================================================================
         ===============================================================================
         ===============================================================================
         ===============================================================================
         ===============================================================================
         ===============================================================================
         ===============================================================================
         ===========================================] - 0s 42us/sample - loss: 7.3333 -
         acc: 0.5435

Out[10]: [6.999956835871157, 0.54347825]
```

7. Let's test a few more configurations:

#1 15 unit Relu & outer sigmoid - 50 epochs

```
# RELU & SIGMOID
model2 = tf.keras.Sequential(
    [
        tf.keras.layers.Dense(units=15 , activation="relu"),
        tf.keras.layers.Dense(units=1 , activation="sigmoid"),
    ]
)
model2.compile(optimizer="adam", loss='binary_crossentropy', metrics=["acc"])
model2.fit(np.array(X_train), np.array(y_train) , epochs=50, validation_split=0.2) # 0.2 validation split is close to 15% of the
model2.evaluate(np.array(X_test), np.array(y_test))
```

```
Epoch 48/50
205/205 [==============================] - 0s 248us/sample - loss: 0.3122 - acc: 0.8683 - val_loss: 0.3515 - val_acc: 0.8269
Epoch 49/50
205/205 [==============================] - 0s 314us/sample - loss: 0.3108 - acc: 0.8683 - val_loss: 0.3489 - val_acc: 0.8269
Epoch 50/50
205/205 [==============================] - 0s 376us/sample - loss: 0.3096 - acc: 0.8683 - val_loss: 0.3482 - val_acc: 0.8269
46/1 [========================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==========] - 0s 85us/sample - loss: 0.3724 - acc: 0.8913
```

```
[0.3281131905058156, 0.8913044]
```

#1 15 unit Relu & outer sigmoid - 200 epochs

```
# RELU & SIGMOID
model2 = tf.keras.Sequential(
    [
        tf.keras.layers.Dense(units=15 , activation="relu"),
        tf.keras.layers.Dense(units=1 , activation="sigmoid"),
    ]
)
model2.compile(optimizer="adam", loss='binary_crossentropy', metrics=["acc"])
model2.fit(np.array(X_train), np.array(y_train) , epochs=200, validation_split=0.2) # 0.2 validation split is close to 15% of th
model2.evaluate(np.array(X_test), np.array(y_test))
```

```
Epoch 198/200
205/205 [==============================] - 0s 129us/sample - loss: 0.1934 - acc: 0.9220 - val_loss: 0.3491 - val_acc: 0.8462
Epoch 199/200
205/205 [==============================] - 0s 243us/sample - loss: 0.1930 - acc: 0.9366 - val_loss: 0.3470 - val_acc: 0.8654
Epoch 200/200
205/205 [==============================] - 0s 181us/sample - loss: 0.1923 - acc: 0.9317 - val_loss: 0.3479 - val_acc: 0.8654
46/1 [========================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==========] - 0s 64us/sample - loss: 0.3417 - acc: 0.8696
```

```
[0.3213981454787047, 0.8695652]
```

⇒ less epochs is better

#2 15 unit Tanh & outer Sigmoid - 50 epoch

```
# TANH & SIGMOID
model3 = tf.keras.Sequential(
    [
        tf.keras.layers.Dense(units=15, activation="tanh"),
        tf.keras.layers.Dense(units=1, activation="sigmoid"),
    ]
)
model3.compile(optimizer="adam", loss='binary_crossentropy', metrics=["acc"])
model3.fit(np.array(X_train), np.array(y_train), epochs=50, validation_split=0.2) # 0.2 validation split is close to 15% of the
model3.evaluate(np.array(X_test), np.array(y_test))
```

```
Epoch 48/50
205/205 [==============================] - 0s 129us/sample - loss: 0.3235 - acc: 0.8683 - val_loss: 0.3752 - val_acc: 0.8269
Epoch 49/50
205/205 [==============================] - 0s 248us/sample - loss: 0.3221 - acc: 0.8683 - val_loss: 0.3718 - val_acc: 0.8462
Epoch 50/50
205/205 [==============================] - 0s 133us/sample - loss: 0.3210 - acc: 0.8683 - val_loss: 0.3691 - val_acc: 0.8462
46/1 [==========================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
===========] - 0s 85us/sample - loss: 0.3803 - acc: 0.8696
```

#2 15 unit Tanh & outer Sigmoid - 200 epochs

```
# TANH & SIGMOID
model3 = tf.keras.Sequential(
    [
        tf.keras.layers.Dense(units=15, activation="tanh"),
        tf.keras.layers.Dense(units=1, activation="sigmoid"),
    ]
)
model3.compile(optimizer="adam", loss='binary_crossentropy', metrics=["acc"])
model3.fit(np.array(X_train), np.array(y_train), epochs=200, validation_split=0.2) # 0.2 validation split is close to 15% of the
model3.evaluate(np.array(X_test), np.array(y_test))
```

```
Epoch 198/200
205/205 [==============================] - 0s 152us/sample - loss: 0.2019 - acc: 0.9317 - val_loss: 0.3732 - val_acc: 0.8269
Epoch 199/200
205/205 [==============================] - 0s 167us/sample - loss: 0.2012 - acc: 0.9317 - val_loss: 0.3708 - val_acc: 0.8269
Epoch 200/200
205/205 [==============================] - 0s 157us/sample - loss: 0.2006 - acc: 0.9317 - val_loss: 0.3702 - val_acc: 0.8269
46/1 [==========================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
==============================================================================================================================
===========] - 0s 212us/sample - loss: 0.3810 - acc: 0.8478

[0.35523187595865, 0.84782606]
```

⇒ <u>Less epoch is better</u>

#3 50 units Relu & outer layer Sigmoid - 200 epochs

```
In [32]: # RELU & SIGMOID with more units
model4 = tf.keras.Sequential(
    [
        tf.keras.layers.Dense(units=50, activation="relu"),
        tf.keras.layers.Dense(units=1, activation="sigmoid"),
    ]
)
model4.compile(optimizer="adam", loss='binary_crossentropy', metrics=["acc"])
model4.fit(np.array(X_train), np.array(y_train), epochs=200, validation_split=0.2) # 0.2 validation split is close to 15% of the
model4.evaluate(np.array(X_test), np.array(y_test))
```

```
Epoch 198/200
205/205 [==============================] - 0s 267us/sample - loss: 0.0910 - acc: 0.9756 - val_loss: 0.3293 - val_acc: 0.8846
Epoch 199/200
205/205 [==============================] - 0s 190us/sample - loss: 0.0902 - acc: 0.9756 - val_loss: 0.3263 - val_acc: 0.8846
Epoch 200/200
205/205 [==============================] - 0s 124us/sample - loss: 0.0895 - acc: 0.9756 - val_loss: 0.3251 - val_acc: 0.8846
46/1 [=======================================================================================================================
=============================================================================================================================
=============================================================================================================================
=============================================================================================================================
=============================================================================================================================
=============================================================================================================================
=============================================================================================================================
=============================================================================================================================
=============================================================================================================================
=============================================================================================================================
=============================================================================================================================
==========] - 0s 106us/sample - loss: 0.4388 - acc: 0.8043
```

```
Out[32]: [0.408673002668049, 0.8043478]
```

#3 50 units Relu & outer layer Sigmoid - 50 epochs

```
]: # RELU & SIGMOID with more units
model4 = tf.keras.Sequential(
    [
        tf.keras.layers.Dense(units=50, activation="relu"),
        tf.keras.layers.Dense(units=1, activation="sigmoid"),
    ]
)
model4.compile(optimizer="adam", loss='binary_crossentropy', metrics=["acc"])
model4.fit(np.array(X_train), np.array(y_train), epochs=50, validation_split=0.2) # 0.2 validation split is close to 15% of the (
model4.evaluate(np.array(X_test), np.array(y_test))
```

```
Epoch 45/50
205/205 [==============================] - 0s 143us/sample - loss: 0.2683 - acc: 0.8878 - val_loss: 0.3580 - val_acc: 0.8462
Epoch 46/50
205/205 [==============================] - 0s 457us/sample - loss: 0.2665 - acc: 0.8976 - val_loss: 0.3564 - val_acc: 0.8462
Epoch 47/50
205/205 [==============================] - 0s 357us/sample - loss: 0.2649 - acc: 0.8976 - val_loss: 0.3556 - val_acc: 0.8462
Epoch 48/50
205/205 [==============================] - 0s 238us/sample - loss: 0.2644 - acc: 0.8976 - val_loss: 0.3493 - val_acc: 0.8462
Epoch 49/50
205/205 [==============================] - 0s 409us/sample - loss: 0.2615 - acc: 0.8927 - val_loss: 0.3536 - val_acc: 0.8462
Epoch 50/50
205/205 [==============================] - 0s 271us/sample - loss: 0.2598 - acc: 0.8976 - val_loss: 0.3579 - val_acc: 0.8269
46/1 [=======================================================================================================================
=============================================================================================================================
=============================================================================================================================
=============================================================================================================================
=============================================================================================================================
=============================================================================================================================
=============================================================================================================================
=============================================================================================================================
```

#4 15 units Relu & outer layer Softmax - 50 epochs - use the default validation split instead of k-fold

```
In [34]: # RELU & SOFTMAX no K-fold CV
model5 = tf.keras.Sequential(
    [
        tf.keras.layers.Dense(units=15 , activation="relu"),
        tf.keras.layers.Dense(units=1 , activation="softmax"),
    ]
)
model5.compile(optimizer="adam", loss='binary_crossentropy', metrics=["acc"])
model5.fit(np.array(X_train), np.array(y_train) , epochs=50, validation_split=0.2) # 0.2 validation split is close to 15% of the
model5.evaluate(np.array(X_test), np.array(y_test))
```

```
Epoch 48/50
205/205 [==============================] - 0s 633us/sample - loss: 6.3577 - acc: 0.5854 - val_loss: 8.5512 - val_acc: 0.4423
Epoch 49/50
205/205 [==============================] - 0s 700us/sample - loss: 6.3577 - acc: 0.5854 - val_loss: 8.5512 - val_acc: 0.4423
Epoch 50/50
205/205 [==============================] - 0s 295us/sample - loss: 6.3577 - acc: 0.5854 - val_loss: 8.5512 - val_acc: 0.4423
46/1 [=====================================================================================================================
==========================================================================================================================
==========================================================================================================================
==========================================================================================================================
==========================================================================================================================
==========================================================================================================================
==========================================================================================================================
==========================================================================================================================
==========================================================================================================================
==========================================================================================================================
==========================================================================================================================
==========] - 0s 149us/sample - loss: 7.2857 - acc: 0.4783
Out[34]: [7.99995057479195, 0.47826087]
```

#4 15 units Relu & outer layer Softmax - 200 epochs - use the default validation split instead of k-fold

```
]: # RELU & SOFTMAX no K-fold CV
model5 = tf.keras.Sequential(
    [
        tf.keras.layers.Dense(units=15 , activation="relu"),
        tf.keras.layers.Dense(units=1 , activation="softmax"),
    ]
)
model5.compile(optimizer="adam", loss='binary_crossentropy', metrics=["acc"])
model5.fit(np.array(X_train), np.array(y_train) , epochs=200, validation_split=0.2) # 0.2 validation split is close to 15% of th
model5.evaluate(np.array(X_test), np.array(y_test))
```

```
Epoch 198/200
205/205 [==============================] - 0s 200us/sample - loss: 6.3577 - acc: 0.5854 - val_loss: 8.5512 - val_acc: 0.4423
Epoch 199/200
205/205 [==============================] - 0s 143us/sample - loss: 6.3577 - acc: 0.5854 - val_loss: 8.5512 - val_acc: 0.4423
Epoch 200/200
205/205 [==============================] - 0s 167us/sample - loss: 6.3577 - acc: 0.5854 - val_loss: 8.5512 - val_acc: 0.4423
46/1 [=====================================================================================================================
==========================================================================================================================
==========================================================================================================================
==========================================================================================================================
==========================================================================================================================
==========================================================================================================================
==========================================================================================================================
==========================================================================================================================
==========================================================================================================================
==========================================================================================================================
==========================================================================================================================
===========] - 0s 85us/sample - loss: 7.2857 - acc: 0.4783
]: [7.99995057479195, 0.47826087]
```

⇒ not much changes with epoch numbers when using softmax

Model Comparison Report - 200 epoch

Model description	Training Accuracy	Validation Accuracy	Test Accuracy
L1: 15 units - Relu **L2: 1 unit - Softmax** **StratifiedKFold CV**	0.5430	0.5556	0.5434
L1: 15 units - Relu **L2: 1 unit - Sigmoid**	0.9317	0.8654	0.8696
L1: 15 units - Tanh **L2: 1 unit - Sigmoid**	0.9317	0.8269	0.8478
L1: 50 units - Relu **L2: 1 unit - Sigmoid**	0.9756	0.8846	0.8043
L1: 15 units - Relu **L2: 1 unit - Softmax** **Default CV**	0.5854	0.4423	0.4783

Conclusion: According to models above, the best model is the 2nd one (15 relu + sigmoid). Even though our 4th model (50 relu + sigmoid) has very high accuracy (validation & training), it is overfitted as the test result is the lowest compared to the rest. So by reducing the number of units in the first layer we can increase the test accuracy and reduce overfitting. Reducing the number of epoch also has the potential to make our models fit better.

LEARNING:

- How to load data and make it available to Keras.
- How to prepare multi-class classification data for modeling using one hot encoding and normalizing.
- How to use Keras neural network models with scikit-learn.
- How to define a neural network using Keras for multi-class classification.
- How to evaluate a Keras neural network model using scikit-learn with k-fold cross validation

REFERENCES:

- *https://medium.com/@literallywords/stratified-k-fold-with-keras-e57c487b1416*
- *https://www.tensorflow.org/install/pip*
- *https://keras.io/getting-started/sequential-model-guide/*
- *https://medium.com/@contactsunny/how-to-split-your-dataset-to-train-and-test-datasets-using-scikit-learn-e7cf6eb5e0d*
- *https://stackoverflow.com/questions/47068709/your-cpu-supports-instructions-that-this-tensorflow-binary-was-not-compiled-to-u*
- *https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02*
- *https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6*
- *https://medium.com/datadriveninvestor/k-fold-cross-validation-6b8518070833*
- *https://stackoverflow.com/questions/40641640/suggest-using-kfold-splits-or-validation-split-kwarg-in-keras-training*
- *https://medium.com/@literallywords/stratified-k-fold-with-keras-e57c487b1416*