Colab: https://colab.research.google.com/drive/1QlmcDSR8_NwZgmHPkevLUhWyz9f6Fjiq?usp=sharing

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

```
!wget "https://drive.google.com/uc?export=download&id=1GVhrh2rH6hUunV4Tf7lQoSFsqov9
```

```
    --2022-12-11 08:38:58--  https://drive.google.com/uc?export=download&id=1GVhrh
    Resolving drive.google.com (drive.google.com)... 74.125.128.101, 74.125.128.10
    Connecting to drive.google.com (drive.google.com)|74.125.128.101|:443... conne
    HTTP request sent, awaiting response... 303 See Other
    Location: https://doc-0s-50-docs.googleusercontent.com/docs/securesc/ha0ro937g
    Warning: wildcards not supported in HTTP.
    --2022-12-11 08:39:00--  https://doc-0s-50-docs.googleusercontent.com/docs/sec
    Resolving doc-0s-50-docs.googleusercontent.com (doc-0s-50-docs.googleuserconte
    Connecting to doc-0s-50-docs.googleusercontent.com (doc-0s-50-docs.googleuserc
    HTTP request sent, awaiting response... 200 OK
    Length: 761835 (744K) [text/csv]
    Saving to: ‘healthyfime.csv’

    healthyfime.csv     100%[===================>] 743.98K  --.-KB/s    in 0.009s

    2022-12-11 08:39:00 (78.3 MB/s) - ‘healthyfime.csv’ saved [761835/761835]
```

Saving...

```
")
```

	age	gender	height_cm	weight_kg	body fat_%	diastolic	systolic	gripForce	fo
0	27.0	M	172.3	75.24	21.3	80.0	130.0	54.9	
1	25.0	M	165.0	55.80	15.7	77.0	126.0	36.4	
2	31.0	M	179.6	78.00	20.1	92.0	152.0	44.8	
3	32.0	M	174.5	71.10	18.4	76.0	147.0	41.4	
4	28.0	M	173.8	67.70	17.1	70.0	127.0	43.5	

```
df.info()
```

```
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 13393 entries, 0 to 13392
    Data columns (total 12 columns):
     #   Column                   Non-Null Count  Dtype  
    ---  ------                   --------------  -----  
     0   age                      13393 non-null  float64
```

```
 1   gender                   13393 non-null  object
 2   height_cm                13393 non-null  float64
 3   weight_kg                13393 non-null  float64
 4   body fat_%               13393 non-null  float64
 5   diastolic                13393 non-null  float64
 6   systolic                 13393 non-null  float64
 7   gripForce                13393 non-null  float64
 8   sit and bend forward_cm  13393 non-null  float64
 9   sit-ups counts           13393 non-null  float64
 10  broad jump_cm            13393 non-null  float64
 11  class                    13393 non-null  object
dtypes: float64(10), object(2)
memory usage: 1.2+ MB
```

```
df.replace({"M":0, "F":1} , inplace = True)
df.head()
```

	age	gender	height_cm	weight_kg	body fat_%	diastolic	systolic	gripForce	fo
0	27.0	0	172.3	75.24	21.3	80.0	130.0	54.9	
1	25.0	0	165.0	55.80	15.7	77.0	126.0	36.4	
2	31.0	0	179.6	78.00	20.1	92.0	152.0	44.8	
3	32.0	0	174.5	71.10	18.4	76.0	147.0	41.4	
4	28.0	0	173.8	67.70	17.1	70.0	127.0	43.5	

```
classes = list(df['class'].unique())
mapping_dict = { ch : i for i, ch in enumerate(sorted(classes, reverse=True)) }
Saving…
                                                        , inplace = True)
df.head()
```

```
{'D': 0, 'C': 1, 'B': 2, 'A': 3}
```

	age	gender	height_cm	weight_kg	body fat_%	diastolic	systolic	gripForce	fo
0	27.0	0	172.3	75.24	21.3	80.0	130.0	54.9	
1	25.0	0	165.0	55.80	15.7	77.0	126.0	36.4	
2	31.0	0	179.6	78.00	20.1	92.0	152.0	44.8	
3	32.0	0	174.5	71.10	18.4	76.0	147.0	41.4	
4	28.0	0	173.8	67.70	17.1	70.0	127.0	43.5	

```
df["class"].unique()
```

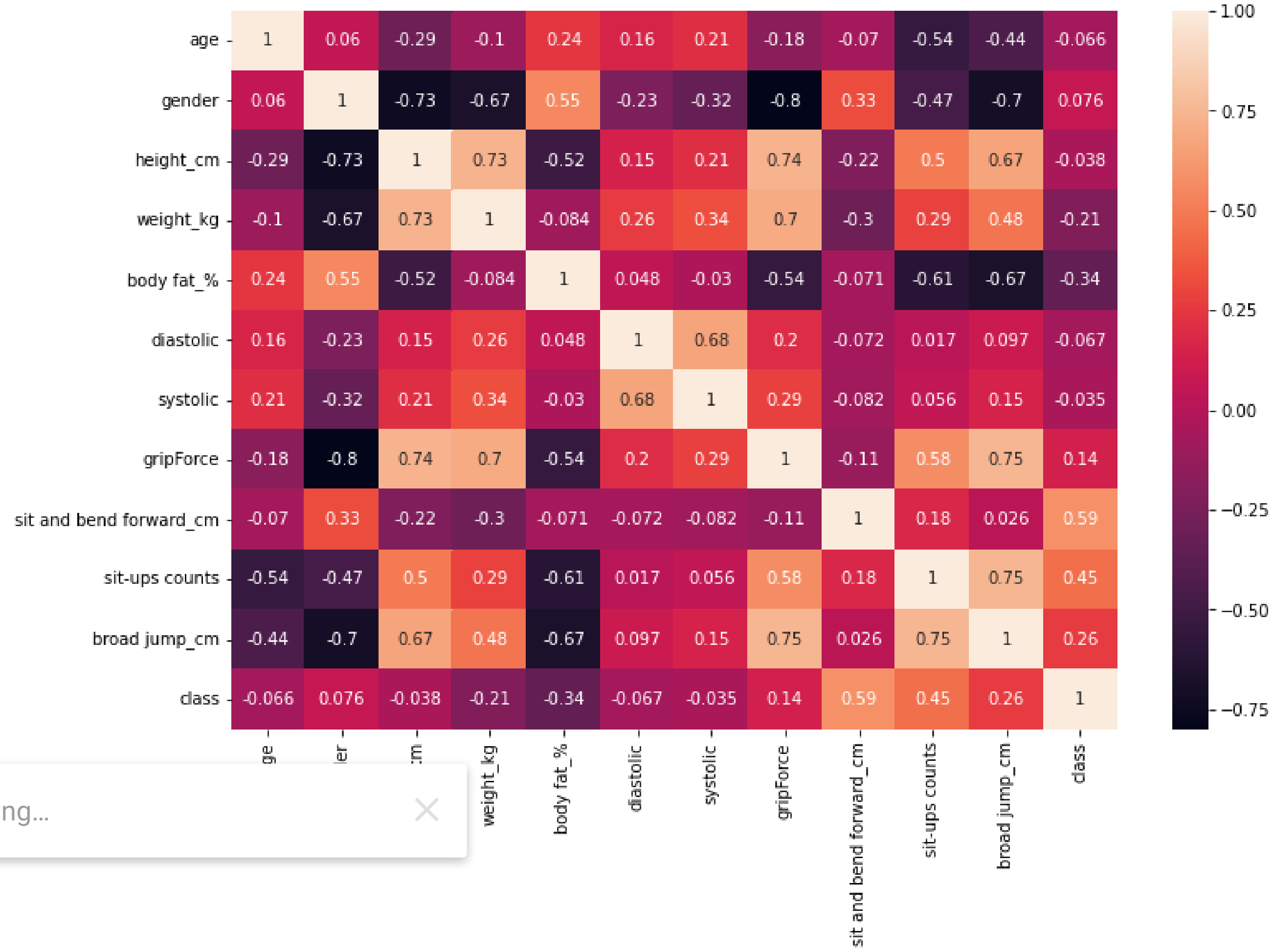
```
array([1, 3, 2, 0])
```

```
df["class"].value_counts()
```

```
1    3349
0    3349
3    3348
2    3347
Name: class, dtype: int64
```

```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fee6130d730>
```



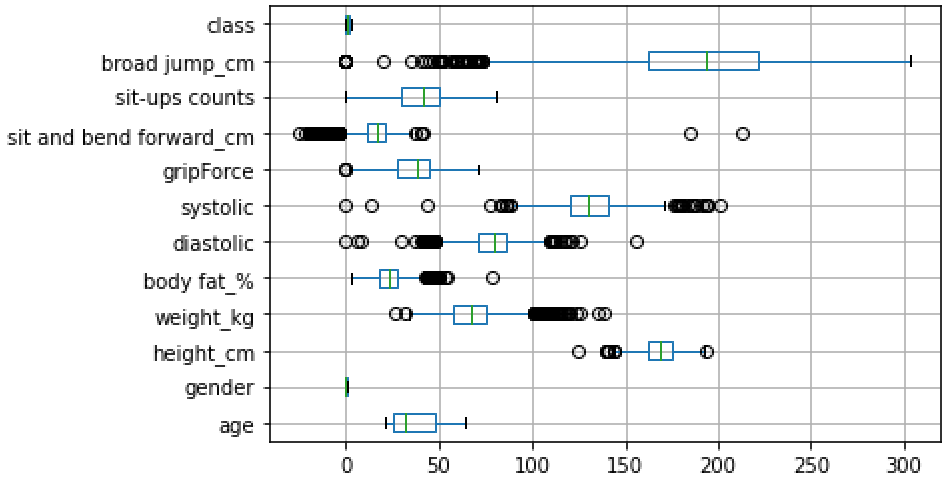
Saving... ×

```
df.describe()
```

	age	gender	height_cm	weight_kg	body fat_%	diastol
count	13393.000000	13393.000000	13393.000000	13393.000000	13393.000000	13393.0000
mean	36.775106	0.367804	168.559807	67.447316	23.240165	78.7968
std	13.625639	0.482226	8.426583	11.949666	7.256844	10.7420

```
df.boxplot(rot=0, vert=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fee5ec8b280>
```



```
X, y = df.iloc[:, :-1], df.iloc[:, -1]
print(X.shape, y.shape)
```

```
(13393, 11) (13393,)
```

Saving...

```
rt train_test_split
X_dev, X_test, y_dev, y_test = train_test_split(X, y, test_size=0.1, random_state=4
print('Train : ', X_dev.shape, y_dev.shape)
print('Test : ', X_test.shape, y_test.shape)
```

```
Train :  (12053, 11) (12053,)
Test :  (1340, 11) (1340,)
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_dev)
```

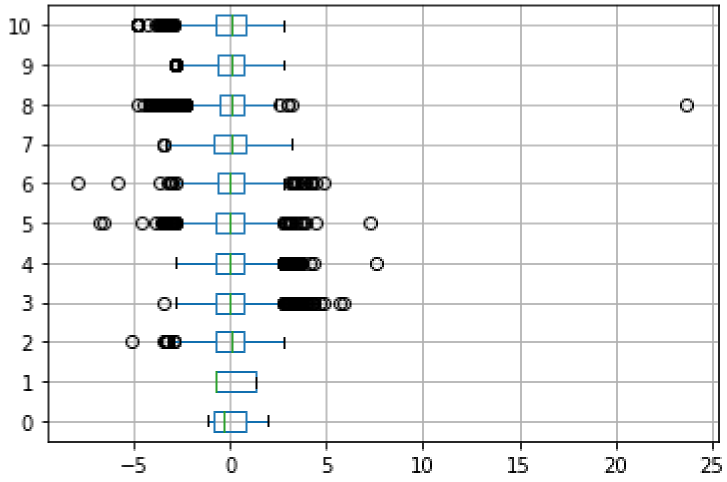
```
StandardScaler()
```

```
X_dev = scaler.transform(X_dev)
X_test = scaler.transform(X_test)
```

```
X_dev = pd.DataFrame(X_dev)
X_test = pd.DataFrame(X_test)
```

```
X_dev.boxplot(rot=0, vert=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fee5f57b910>
```



```
import tensorflow as tf
```

```
tf.__version__
```

```
'2.9.2'
```

```
dir(tf.keras)
```

```
['Input',
 'Model',
 'Sequential',
 '__builtins__',
 '__cached__',
```

Saving...

```
 '__loader__',
 '__name__',
 '__package__',
 '__path__',
 '__spec__',
 '__version__',
 '_sys',
 'activations',
 'applications',
 'backend',
 'callbacks',
 'constraints',
 'datasets',
 'dtensor',
 'estimator',
 'experimental',
 'initializers',
 'layers',
 'losses',
 'metrics',
 'mixed_precision',
 'models',
 'optimizers',
 'preprocessing',
 'regularizers',
 'utils',
 'wrappers']
```

```
dir(tf.keras.activations)
```

```
     ['__builtins__',
      '__cached__',
      '__doc__',
      '__file__',
      '__loader__',
      '__name__',
      '__package__',
      '__path__',
      '__spec__',
      '_sys',
      'deserialize',
      'elu',
      'exponential',
      'gelu',
      'get',
      'hard_sigmoid',
      'linear',
      'relu',
      'selu',
      'serialize',
      'sigmoid',
      'softmax',
      'softplus',
      'softsign',
      'swish',
      'tanh']
```

Saving… ×

```
dir(tf.keras.optimizers)
```

```
     ['Adadelta',
      'Adagrad',
      'Adam',
      'Adamax',
      'Ftrl',
      'Nadam',
      'Optimizer',
      'RMSprop',
      'SGD',
      '__builtins__',
      '__cached__',
      '__doc__',
      '__file__',
      '__loader__',
      '__name__',
      '__package__',
      '__path__',
      '__spec__',
      '_sys',
      'deserialize',
      'experimental',
      'get',
      'legacy',
      'schedules',
      'serialize']
```

```
# Sequential API
# Functional API --> Complex non-sequential networks - CNNs, ResNet, post-read
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
model = Sequential([
          Dense(100, activation="relu", input_shape=(11,)),
          Dense(4, activation="softmax")
        ])
```

```
type(model.weights)
```

```
    list
```

```
for param in model.weights:
  print(param.shape)
```

```
    (11, 100)
    (100,)
    (100, 4)
    (4,)
```

```
model = Sequential()
                                        elu", input_shape=(11,), name="hidden_1"))
                                        ftmax", name="output"))
```

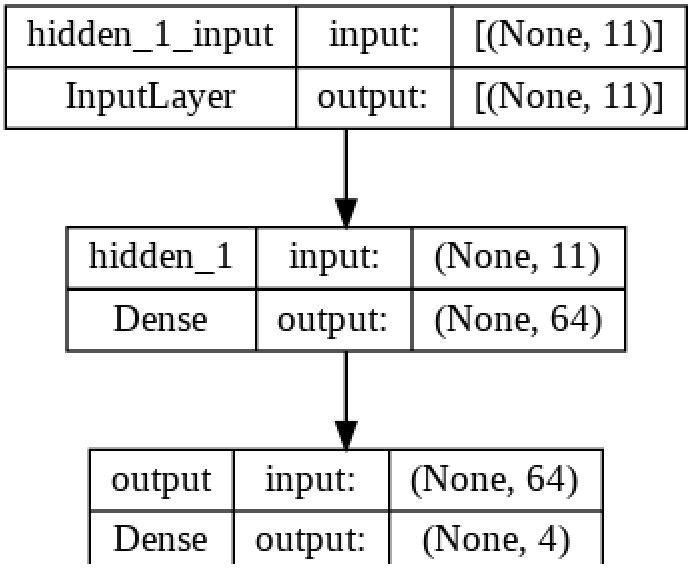
Saving…

```
model.summary()
```

```
    Model: "sequential_4"
    _________________________________________________________________
     Layer (type)                Output Shape              Param #
    =================================================================
     hidden_1 (Dense)            (None, 64)                768

     output (Dense)              (None, 4)                 260

    =================================================================
    Total params: 1,028
    Trainable params: 1,028
    Non-trainable params: 0
    _________________________________________________________________
```

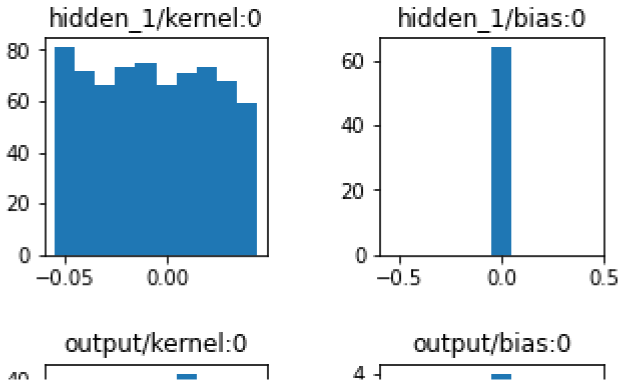
```
from tensorflow.keras.utils import plot_model
```

```
plot_model(model,
    to_file='model.png',
    show_shapes=True, show_layer_names=True)
```



```
# Weigjts and bias
# Weghts - randomly,multiple ways --> Glorot Normal, Glorot Uniform, HE Normal, HE
# bias - zeros


model = Sequential()
model.add(Dense(64,
                activation="relu",
                input_shape=(11,),
                name="hidden_1",
                kernel_initializer = "random_uniform",
                bias_initializer = "zeros"))
model.add(Dense(4, activation="softmax", name="output",
                                  r = "he_normal",
                                  = "ones"))


# Plot histograms of weight and bias values
import matplotlib.pyplot as plt
fig, axes = plt.subplots(2, 2, figsize=(5,5))
fig.subplots_adjust(hspace=0.5, wspace=0.5)

# get the weights from the layers
weight_layers = [layer for layer in model.layers]

for i, layer in enumerate(weight_layers):
    for j in [0, 1]:
        axes[i, j].hist(layer.weights[j].numpy().flatten(), align='left')
        axes[i, j].set_title(layer.weights[j].name)
```

Saving…



```
# loss function - cce, sparse cce, mse, mae
# optimiser - adam, rmsprop, sgd, .....
# metrics - loss, accuracy
# losss function and optimiser with the model ---> model compilation

model_2C = Sequential([
          Dense(64, activation="relu", input_shape=(11,)),
          Dense(1, activation="sigmoid")])

model_2C.compile(optimizer = "sgd",
                 loss = "binary_crossentropy",
                 metrics = ["accuracy"])


model_2C.compile(optimizer = tf.keras.optimizers.SGD(learning_rate=0.001),
                 loss = tf.keras.losses.BinaryCrossentropy(),
                 metrics = ["accuracy"])


model.compile(optimizer = "adam",
                                    ategorical_crossentropy",
  Saving...                   ×     racy"])


history = model.fit(X_dev, y_dev, epochs=10, batch_size=256, validation_split=0.1,


# history (callbacks)
history = model.fit(X_dev, y_dev, epochs=500, batch_size=256, validation_split=0.1,
```

```
Epoch 1/500
43/43 [==============================] - 1s 9ms/step - loss: 1.2894 - accuracy
Epoch 2/500
43/43 [==============================] - 0s 10ms/step - loss: 1.0877 - accurac
Epoch 3/500
43/43 [==============================] - 0s 5ms/step - loss: 0.9662 - accuracy
Epoch 4/500
43/43 [==============================] - 0s 3ms/step - loss: 0.9076 - accuracy
Epoch 5/500
43/43 [==============================] - 0s 2ms/step - loss: 0.8790 - accuracy
Epoch 6/500
43/43 [==============================] - 0s 3ms/step - loss: 0.8631 - accuracy
Epoch 7/500
43/43 [==============================] - 0s 2ms/step - loss: 0.8534 - accuracy
Epoch 8/500
43/43 [==============================] - 0s 3ms/step - loss: 0.8472 - accuracy
```

```
    Epoch 9/500
    43/43 [==============================] - 0s 3ms/step - loss: 0.8413 - accuracy
    Epoch 10/500
    43/43 [==============================] - 0s 3ms/step - loss: 0.8360 - accuracy
    Epoch 11/500
    43/43 [==============================] - 0s 2ms/step - loss: 0.8314 - accuracy
    Epoch 12/500
    43/43 [==============================] - 0s 3ms/step - loss: 0.8278 - accuracy
    Epoch 13/500
    43/43 [==============================] - 0s 3ms/step - loss: 0.8219 - accuracy
    Epoch 14/500
    43/43 [==============================] - 0s 3ms/step - loss: 0.8180 - accuracy
    Epoch 15/500
    43/43 [==============================] - 0s 3ms/step - loss: 0.8123 - accuracy
    Epoch 16/500
    43/43 [==============================] - 0s 3ms/step - loss: 0.8066 - accuracy
    Epoch 17/500
    43/43 [==============================] - 0s 3ms/step - loss: 0.8008 - accuracy
    Epoch 18/500
    43/43 [==============================] - 0s 3ms/step - loss: 0.7947 - accuracy
    Epoch 19/500
    43/43 [==============================] - 0s 3ms/step - loss: 0.7877 - accuracy
    Epoch 20/500
    43/43 [==============================] - 0s 2ms/step - loss: 0.7810 - accuracy
    Epoch 21/500
    43/43 [==============================] - 0s 2ms/step - loss: 0.7726 - accuracy
    Epoch 22/500
    43/43 [==============================] - 0s 3ms/step - loss: 0.7656 - accuracy
    Epoch 23/500
    43/43 [==============================] - 0s 3ms/step - loss: 0.7574 - accuracy
    Epoch 24/500
    43/43 [==============================] - 0s 3ms/step - loss: 0.7511 - accuracy
    Epoch 25/500
    43/43 [==============================] - 0s 2ms/step - loss: 0.7443 - accuracy
  Saving…
                                ==========] - 0s 2ms/step - loss: 0.7390 - accuracy
    Epoch 27/500
    43/43 [==============================] - 0s 2ms/step - loss: 0.7329 - accuracy
    Epoch 28/500
    43/43 [==============================] - 0s 2ms/step - loss: 0.7280 - accuracy
    Epoch 29/500
    43/43 [==============================] - 0s 2ms/step - loss: 0.7234
```

```
history.__dict__.keys()
```

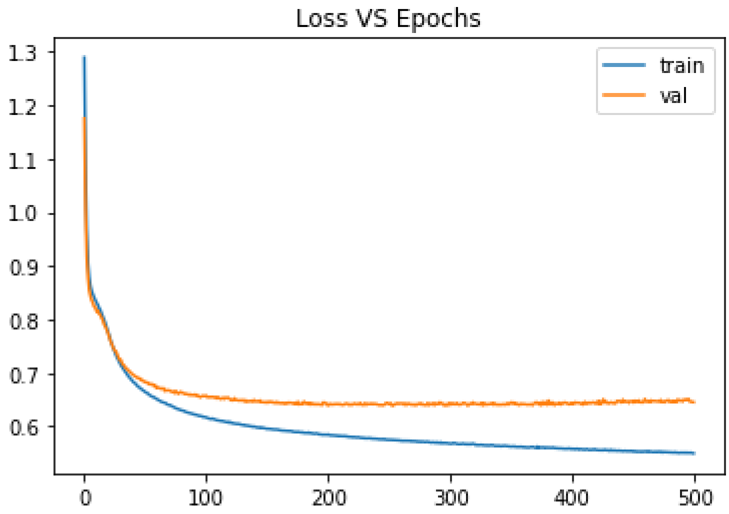
```
    dict_keys(['validation_data', 'model', '_chief_worker_only',
    '_supports_tf_logs', 'history', 'params', 'epoch'])
```

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

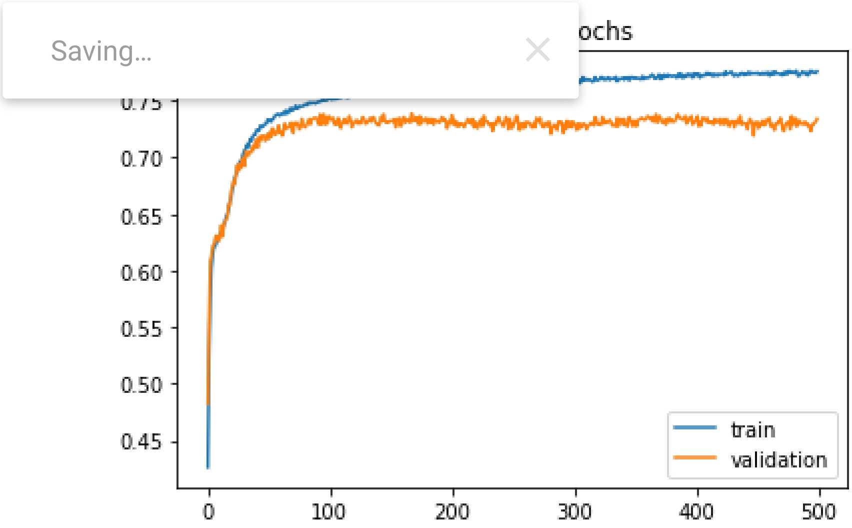
```
    dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
epochs = history.epoch
loss = history.history["loss"]
accuracy = history.history["accuracy"]
val_loss = history.history["val_loss"]
val_accuracy = history.history["val_accuracy"]
```

```
plt.figure()
plt.plot(epochs, loss, label="train")
plt.plot(epochs, val_loss, label="val")
plt.legend()
plt.title("Loss VS Epochs")
plt.show()
```



```
plt.figure()
plt.plot(epochs, accuracy, label="train")
plt.plot(epochs, val_accuracy, label="validation")
plt.legend()
plt.title("Accuracy VS Epochs")
plt.show()
```



```
model.evaluate(X_test, y_test)
```

```
42/42 [==============================] - 0s 1ms/step - loss: 0.5862 - accuracy
[0.5861693620681763, 0.7552238702774048]
```

```
model.evaluate(X_dev, y_dev)
```

```
377/377 [==============================] - 1s 1ms/step - loss: 0.5566 - accura
```

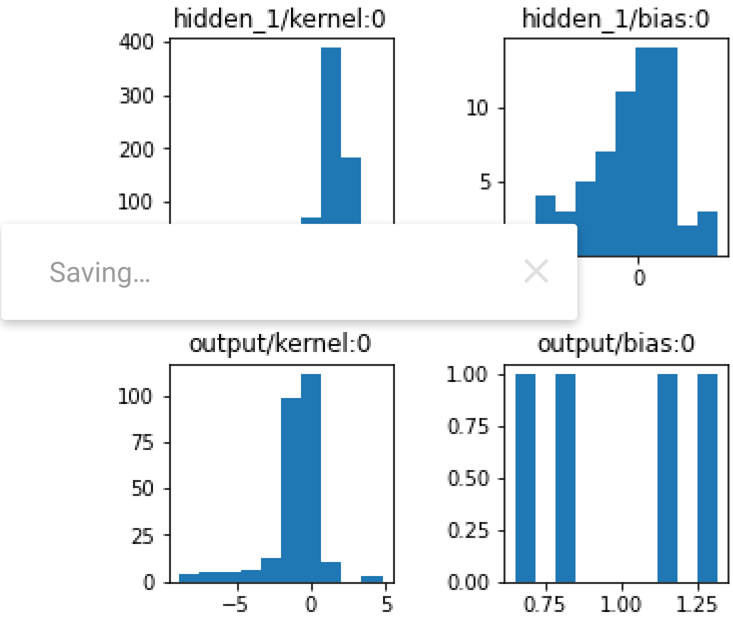
```
[0.5565788745880127, 0.7726706862449646]
```

```
np.argmax(model.predict(np.expand_dims(X_test.to_numpy()[0], axis=0))[0])
```

```
1/1 [==============================] - 0s 17ms/step
0
```

```
# HOMEWORK - Create the same 2-layer NN with 100N in L-1 for spiral dataset, Keras
# spiral dataset - https://drive.google.com/uc?id=1dLOPwh01o3k8p_hK633ixhD1ehz6nNWk
```

```
# Plot histograms of weight and bias values
import matplotlib.pyplot as plt
fig, axes = plt.subplots(2, 2, figsize=(5,5))
fig.subplots_adjust(hspace=0.5, wspace=0.5)

# get the weights from the layers
weight_layers = [layer for layer in model.layers]

for i, layer in enumerate(weight_layers):
    for j in [0, 1]:
        axes[i, j].hist(layer.weights[j].numpy().flatten(), align='left')
        axes[i, j].set_title(layer.weights[j].name)
```



✓ 1s completed at 16:10 ● ×

Saving... ×