```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split




def create_data():

    file_path = 'E:/GITHUBREpo/Intro_to_Machine_learning/Neural_network_tuning/diabetes.csv'


    df = pd.read_csv(file_path)
    X = df.drop("Outcome",axis=1)
    Y = df[["Outcome"]]

    scaler = StandardScaler()

    X = scaler.fit_transform(X)

    X_train , X_test ,Y_train, Y_test = train_test_split(X,Y,test_size=0.3,random_state=1)

    return X_train ,X_test , Y_train , Y_test
```

```python
import tensorflow
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense, Dropout

from data_prep import create_data


X_train, X_test, Y_train, Y_test = create_data()

model = Sequential()
model.add(Dense(32, activation='relu', input_dim=8))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy',
              metrics=['accuracy'])
model.fit(X_train, Y_train, batch_size=32, epochs=100,
          validation_data=(X_test, Y_test))
```

```
17/17 [==============================] - 0s 3ms/step - loss: 0.4354 - accuracy: 0.7840 - val_loss: 0.4390 - val_accuracy: 0.8052
Epoch 82/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4347 - accuracy: 0.7877 - val_loss: 0.4394 - val_accuracy: 0.8052
Epoch 83/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4343 - accuracy: 0.7858 - val_loss: 0.4390 - val_accuracy: 0.8095
Epoch 84/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4341 - accuracy: 0.7858 - val_loss: 0.4395 - val_accuracy: 0.8182
Epoch 85/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4337 - accuracy: 0.7877 - val_loss: 0.4391 - val_accuracy: 0.8225
Epoch 86/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4332 - accuracy: 0.7858 - val_loss: 0.4390 - val_accuracy: 0.8052
Epoch 87/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4329 - accuracy: 0.7877 - val_loss: 0.4394 - val_accuracy: 0.8052
Epoch 88/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4326 - accuracy: 0.7858 - val_loss: 0.4390 - val_accuracy: 0.8052
Epoch 89/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4322 - accuracy: 0.7896 - val_loss: 0.4393 - val_accuracy: 0.8009
Epoch 90/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4327 - accuracy: 0.7914 - val_loss: 0.4396 - val_accuracy: 0.8009
Epoch 91/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4320 - accuracy: 0.7896 - val_loss: 0.4392 - val_accuracy: 0.8139
Epoch 92/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4313 - accuracy: 0.7896 - val_loss: 0.4392 - val_accuracy: 0.8009
Epoch 93/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4313 - accuracy: 0.7877 - val_loss: 0.4393 - val_accuracy: 0.8009
Epoch 94/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4308 - accuracy: 0.7896 - val_loss: 0.4398 - val_accuracy: 0.8182
Epoch 95/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4303 - accuracy: 0.7896 - val_loss: 0.4399 - val_accuracy: 0.8052
Epoch 96/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4300 - accuracy: 0.7914 - val_loss: 0.4393 - val_accuracy: 0.8052
Epoch 97/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4296 - accuracy: 0.7896 - val_loss: 0.4393 - val_accuracy: 0.8052
Epoch 98/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4293 - accuracy: 0.7933 - val_loss: 0.4391 - val_accuracy: 0.7965
Epoch 99/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4289 - accuracy: 0.7933 - val_loss: 0.4397 - val_accuracy: 0.8009
Epoch 100/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4291 - accuracy: 0.7952 - val_loss: 0.4399 - val_accuracy: 0.8052
```

optimizer.py

```python
"""# steps for tuning the layer
    1) how to select appropiate optimizer
    2) Number of nodes in a layer
    3) how to select the number of hidden layers in a model
    4) all in model( with all the necesary features)"""



step1 = " SELECT THE APPROPIATE OPTIMIZER"
import tensorflow
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense, Dropout
import keras_tuner as kt

from data_prep import create_data

def build_model(hp):

    model = Sequential()

    model.add(Dense(32,activation='relu',input_dim = 8))
    model.add(Dense(1,activation ='sigmoid'))

    optimizer = hp.Choice('optimizer',values =['adam','sgd','rmsprop','adadelta'])

    model.compile(optimizer=optimizer,loss = 'binary_crossentropy',metrics=['accuracy'])

    return model

tuner = kt.RandomSearch(build_model,objective='val_accuracy',max_trials=5)

X_train, X_test, Y_train, Y_test = create_data()

tuner.search(X_train,Y_train ,epochs =5,validation_data =(X_test,Y_test))


print(tuner.get_best_hyperparameters()[0].values)
model = tuner.get_best_models(num_models=1)[0]

print(model.summary())
model.fit(X_train,Y_train,batch_size=32,epochs=100,initial_epoch=6,validation_data=(X_test,Y_test))

```

```
Search: Running Trial #4

Value             |Best Value So Far |Hyperparameter
adadelta          |rmsprop           |optimizer

Epoch 1/5
17/17 [==============================] - 1s 22ms/step - loss: 0.6611 - accuracy: 0.6443 - val_loss: 0.6363 - val_accuracy: 0.6623
Epoch 2/5
17/17 [==============================] - 0s 3ms/step - loss: 0.6611 - accuracy: 0.6443 - val_loss: 0.6362 - val_accuracy: 0.6623
Epoch 3/5
17/17 [==============================] - 0s 3ms/step - loss: 0.6610 - accuracy: 0.6443 - val_loss: 0.6362 - val_accuracy: 0.6623
Epoch 4/5
17/17 [==============================] - 0s 3ms/step - loss: 0.6610 - accuracy: 0.6443 - val_loss: 0.6361 - val_accuracy: 0.6623
Epoch 5/5
17/17 [==============================] - 0s 3ms/step - loss: 0.6609 - accuracy: 0.6443 - val_loss: 0.6361 - val_accuracy: 0.6623
Trial 4 Complete [00h 00m 01s]
val_accuracy: 0.6623376607894897

Best val_accuracy So Far: 0.7835497856140137
Total elapsed time: 00h 00m 07s
{'optimizer': 'rmsprop'}
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32)                288

 dense_1 (Dense)             (None, 1)                 33

=================================================================
```

```python
step2 = " SELECT THE APPROPIATE NEURONS "
import tensorflow
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense, Dropout
import keras_tuner as kt
from data_prep import create_data

X_train, X_test, Y_train, Y_test = create_data()

def build_model(hp):
    model = Sequential()

    units = hp.Int('units',min_value =8,max_value =128,step =8)

    model.add(Dense(units=units,activation='relu',input_dim = 8))
    model.add(Dense(1,activation='sigmoid'))

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

    return model

tuner = kt.RandomSearch(build_model,
                        objective='val_accuracy', max_trials=5,directory ='mydir',project_name = 'Dipayan')
tuner.search(X_train,Y_train,epochs =5,validation_data =(X_test,Y_test))
print(tuner.get_best_hyperparameters()[0].values)
model = tuner.get_best_models(num_models=1)[0]
#running the model
model.fit(X_train,Y_train,batch_size=32,epochs=100,initial_epoch=6)
```

```
Epoch 5/5
17/17 [==============================] - 0s 10ms/step - loss: 0.6263 - accuracy: 0.6760 - val_loss: 0.5766 - val_accuracy: 0.6970
Trial 4 Complete [00h 00m 02s]
val_accuracy: 0.6969696879386902

Best val_accuracy So Far: 0.7878788113594055
Total elapsed time: 00h 00m 07s

Search: Running Trial #5

Value             |Best Value So Far |Hyperparameter
112               |128               |units

Epoch 1/5
17/17 [==============================] - 1s 18ms/step - loss: 0.6963 - accuracy: 0.5382 - val_loss: 0.6526 - val_accuracy: 0.6234
Epoch 2/5
17/17 [==============================] - 0s 9ms/step - loss: 0.6193 - accuracy: 0.6927 - val_loss: 0.5853 - val_accuracy: 0.7532
Epoch 3/5
17/17 [==============================] - 0s 10ms/step - loss: 0.5720 - accuracy: 0.7281 - val_loss: 0.5411 - val_accuracy: 0.7792
Epoch 4/5
17/17 [==============================] - 0s 9ms/step - loss: 0.5417 - accuracy: 0.7449 - val_loss: 0.5122 - val_accuracy: 0.7835
Epoch 5/5
17/17 [==============================] - 0s 9ms/step - loss: 0.5193 - accuracy: 0.7449 - val_loss: 0.4915 - val_accuracy: 0.7879
Trial 5 Complete [00h 00m 02s]
val_accuracy: 0.7878788113594055

Best val_accuracy So Far: 0.7878788113594055
Total elapsed time: 00h 00m 09s
{'units': 128}
```

```python
step2 = " SELECT THE NUMBER OF LAYERS IN A MODEL"
import tensorflow
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense, Dropout
import keras_tuner as kt
from data_prep import create_data

X_train, X_test, Y_train, Y_test = create_data()


def build_model(hp):

    model = Sequential()

    model.add(Dense(96,activation ='relu',input_dim =8))

    for i in range(hp.Int('num_layers',min_value = 1,max_value = 10)):

        model.add(Dense(96,activation ='relu'))

    model.add(Dense(1,activation='sigmoid'))

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

    return model


tuner = kt.RandomSearch(build_model,objective='val_accuracy',max_trials=5,directory ='layer_dir',project_name = 'num_of_layers')

tuner.search(X_train,Y_train,epochs =5,validation_data = (X_test,Y_test))
print(tuner.get_best_hyperparameters()[0].values)
model = tuner.get_best_models(num_models=1)[0]
model.fit(X_train,Y_train,epochs=100,initial_epoch=6,validation_data=(X_test,Y_test))
```

```
17/17 [==============================] - 0s 5ms/step - loss: 0.4669 - accuracy: 0.7728 - val_loss: 0.4798 - val_accuracy: 0.7835
Epoch 5/5
17/17 [==============================] - 0s 12ms/step - loss: 0.4526 - accuracy: 0.7821 - val_loss: 0.4656 - val_accuracy: 0.8009
Trial 4 Complete [00h 00m 03s]
val_accuracy: 0.8008658289909363

Best val_accuracy So Far: 0.8051947951316833
Total elapsed time: 00h 00m 11s

Search: Running Trial #5

Value             |Best Value So Far |Hyperparameter
3                 |10                |num_layers

Epoch 1/5
17/17 [==============================] - 1s 20ms/step - loss: 0.6401 - accuracy: 0.6425 - val_loss: 0.5612 - val_accuracy: 0.7186
Epoch 2/5
17/17 [==============================] - 0s 10ms/step - loss: 0.5252 - accuracy: 0.7132 - val_loss: 0.4692 - val_accuracy: 0.7835
Epoch 3/5
17/17 [==============================] - 0s 10ms/step - loss: 0.4770 - accuracy: 0.7747 - val_loss: 0.4577 - val_accuracy: 0.7922
Epoch 4/5
17/17 [==============================] - 0s 4ms/step - loss: 0.4761 - accuracy: 0.7635 - val_loss: 0.4610 - val_accuracy: 0.7922
Epoch 5/5
17/17 [==============================] - 0s 4ms/step - loss: 0.4505 - accuracy: 0.7896 - val_loss: 0.4415 - val_accuracy: 0.7922
Trial 5 Complete [00h 00m 02s]
val_accuracy: 0.7922077775001526

Best val_accuracy So Far: 0.8051947951316833
Total elapsed time: 00h 00m 14s
{'num_layers': 10}
```

```
X_train, X_test, Y_train, Y_test = create_data()
method = " Tune everything at once "
def build_model(hp):
    model = Sequential()

    counter = 0

    for i in range (hp.Int("num_layers",min_value =1 ,max_value = 10)):

        if counter == 0:

            model.add(
                Dense(
                hp.Int('units' + str(i),min_value =8,max_value = 128,step =8),
                      activation = hp.Choice('activation'+str(i),values =['relu','tanh','sigmoid']),
                      input_dim = 8
                )
            )
                model.add(Dropout(hp.Choice('Dropout'+ str(i),values = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9])))

        else:
            model.add(
                Dense(
                hp.Int('units' + str(i),min_value = 8,max_value = 128,step =8),
                activation=hp.Choice('activation_'+str(i) , values=['relu', 'tanh','sigmoid'])
                )
            )
                model.add(Dropout(hp.Choice('Dropout'+ str(i),values = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9])))
            counter+=1

    model.add(Dense(1,activation='sigmoid'))
    model.compile(optimizer=hp.Choice('optimizer',values = ['rmsprop','adam','sgd','nadam','adadelta']),
                  loss ='binary_crossentropy',
                  metrics=['accuracy'])
    return model

tuner = kt.RandomSearch(build_model,
                        objective='val_accuracy', max_trials=5,
                        directory ='final_all',
                        project_name = 'final')

tuner.search(X_train,Y_train,epochs = 5,validation_data =(X_test,Y_test))

print(tuner.get_best_hyperparameters()[0].values)
model = tuner.get_best_models(num_models=1)[0]

model.fit(X_train,Y_train,batch_size=32,epochs=200,initial_epoch=6,validation_data=(X_test,Y_test))
```

```
tanh            |relu            |activation_4
0.2             |0.6             |Dropout4
104             |64              |units5
relu            |tanh            |activation_5
0.9             |0.9             |Dropout5
120             |112             |units6
sigmoid         |tanh            |activation_6
0.8             |0.6             |Dropout6
104             |None            |units7
sigmoid         |None            |activation_7
0.1             |None            |Dropout7

Epoch 1/5
17/17 [==============================] - 3s 23ms/step - loss: 0.8602 - accuracy: 0.4972 - val_loss: 0.6585 - val_accuracy: 0.6320
Epoch 2/5
17/17 [==============================] - 0s 5ms/step - loss: 0.7725 - accuracy: 0.5605 - val_loss: 0.6549 - val_accuracy: 0.6320
Epoch 3/5
17/17 [==============================] - 0s 4ms/step - loss: 0.7293 - accuracy: 0.5847 - val_loss: 0.6552 - val_accuracy: 0.6320
Epoch 4/5
17/17 [==============================] - 0s 5ms/step - loss: 0.7304 - accuracy: 0.6089 - val_loss: 0.6555 - val_accuracy: 0.6320
Epoch 5/5
17/17 [==============================] - 0s 5ms/step - loss: 0.7346 - accuracy: 0.5810 - val_loss: 0.6563 - val_accuracy: 0.6320
Trial 5 Complete [00h 00m 03s]
val_accuracy: 0.6320346593856812

Best val_accuracy So Far: 0.7878788113594055
Total elapsed time: 00h 00m 13s
{'num_layers': 1, 'units0': 112, 'activation0': 'tanh', 'Dropout0': 0.9, 'optimizer': 'rmsprop', 'units1': 24, 'activation_1': 'sigmoid', 'Dropout1': 0.4, 'units2': 120, 'activation_2':
 'sigmoid', 'Dropout2': 0.4, 'units3': 40, 'activation_3': 'relu', 'Dropout3': 0.7, 'units4': 96, 'activation_4': 'relu', 'Dropout4': 0.6, 'units5': 64, 'activation_5': 'tanh', 'Dropout
5': 0.9, 'units6': 112, 'activation_6': 'tanh', 'Dropout6': 0.6}
```