

✓ I. Import the dataset. Define x and y.

```
import pandas as pd
#Pandas is a powerful open-source data analysis and manipulation library for Python.
#It provides data structures and functions for efficiently working with structured data, such as tables and time series data.

dataset = pd.read_csv('airdata.csv')

#In google colab, we run files cell by cell.

#We'll set up x and y attributes.
#y will determine if the tumor is benign or malign.
#x will tell all the feautres on the column titles.
x = dataset.drop(columns=["id"]) # we're saying the "x" data is everying except in the column.

y = dataset["id"]
```

✓ II. Now we need to split the data into a training set and a testing set.

```
# This part is important because we see AI does well on data that it sees but falls apart when seeing new data.
# To mitigate this, we'll seperate a part of our data aside to be tested on later.
# What we're looking for is that the algorithm will be given a data that it has not seen before and see how it does.

from sklearn.model_selection import train_test_split #We're importing it. The scikit-learn/sklearn is a free and open-source machine learning

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2) # 20% of our data will be in the testing set. It's normal to divide
```

✓ III. Now let's build and train the model.

```
# To do this we'll be using TensorFlow Keras.
#Keras is an open-source library that provides a Python interface for artificial neural networks.
#Keras was first independent software, then integrated into the TensorFlow library, and later supporting more.

import tensorflow as tf


model = tf.keras.models.Sequential()

#We'll start adding layers to our module.
#Refer to the neural network picture.
#The Input layer is our x values with all attributes.

model.add(tf.keras.layers.Dense(256, input_shape=x_train.shape[1:], activation='sigmoid'))
#model.add(tf.keras.layers.Dense(256, input_shape=x_train.shape, activation='sigmoid')) #256 neurons is the power of 2 that we set.
#we're saying we'll input something that the size of "x_train", which means all of the x feautres on the dataset.
#The output will be 256 neurons.
#Sigmoid function: we take all the values from neural network and plotting them between 0 and 1. It readuces the model's complexiy and makes

model.add(tf.keras.layers.Dense(256, activation='sigmoid'))

model.add(tf.keras.layers.Dense(1, activation='sigmoid')) #This is the output layer.
# 1 dense neuron because the final value will be one single value between 1 and 0 on the diagnosis column.
```

 /usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to `super().__init__(activity_regularizer=activity_regularizer, **kwargs)`

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
#The optimizer we'll be using is called "adam".
#This "binary_crossentropy" function is useful when using "yes" (1) or "no" (0) sort of results.
```

```
#We want to classify as many chemical pollutants as possible so we use "accuracy" metrics.
```


✓ V. Fit our data.

```
model.fit(x_train, y_train, epochs=11)
#epochs means "how many times the algorithm iterates over the same data".
#Algorithm learns going over and over on the same data.
```

 [Show hidden output](#)

Evaluate the model.

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

 [Show hidden output](#)

Start coding or [generate](#) with AI.