```
from sklearn.model_selection import train_test_split

X_train_full, X_test, y_train_full, y_test = train_test_split(X, y,
test_size=0.2,random_state=42)

X_train, X_valid, y_train, y_valid = train_test_split(X_train_full,
y_train_full,test_size=0.2, random_state=42)


from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_valid = scaler.transform(X_valid)
X_test = scaler.transform(X_test)

import tensorflow as tf
from tensorflow import keras
### Model Initialization
nn_model = keras.models.Sequential([
keras.layers.Input(shape=X_train.shape[1:]),
keras.layers.Dense(10, activation='tanh'),
keras.layers.Dense(1, activation=None)
])
nn_model.summary()
### Model Training
nn_model.compile(loss="mse",
optimizer=keras.optimizers.SGD(learning_rate=1e-2))
nn_model.fit(X_train, y_train, epochs=20, validation_data=(X_valid,
y_valid), verbose=1)
mse_test = nn_model.evaluate(X_test, y_test, verbose=0)


# Model Initialization
model = keras.models.Sequential(
   [keras.layers.Input(shape=X_train.shape[1:]),
    keras.layers.Dense(16,activation='relu'),
    keras.layers.Dense(8,activation='relu'),
    keras.layers.Dense(1,activation='sigmoid')])
model.compile(loss='binary_crossentropy', optimizer='adam',
             metrics=['accuracy'])
```

```
# MNIST
x_train = x_train / 255
x_valid = x_valid / 255
x_test = x_test / 255

model = tf.keras.models.Sequential([
tf.keras.layers.Flatten(input_shape=(28,28)),
   tf.keras.layers.Dense(512, activation="relu"),
   tf.keras.layers.Dense(128, activation="relu"),
   tf.keras.layers.Dense(64, activation="relu"),
   tf.keras.layers.Dense(10, activation="softmax")
])
model.compile(optimizer= 'adam',
            loss= 'sparse_categorical_crossentropy',
            metrics= ['accuracy'])



# A more complex CNN model…
model = tf.keras.models.Sequential([
   keras.layers.Input(shape=(256,256,3)),
   keras.layers.Conv2D(filters=16, kernel_size=(3,3), strides=2,
padding='same',activation='relu'),
   keras.layers.BatchNormalization(),
   keras.layers.Dropout(0.25),
   keras.layers.MaxPooling2D(pool_size=(2,2), strides=2),
   keras.layers.Conv2D(filters=32, kernel_size=(3,3),activation='relu'),
   keras.layers.MaxPooling2D(pool_size=(2,2), strides=2),
   keras.layers.Conv2D(filters=64, kernel_size=(3,3),activation='relu'),
   keras.layers.MaxPooling2D(pool_size=(2,2), strides=2),
   keras.layers.Flatten(),
   keras.layers.Dense(512, activation='relu'),
   keras.layers.Dense(1, activation='sigmoid')
])
from keras.models import load_model
#Load the model
model = load_model('hh_model_no_v.keras')


import tensorflow as tf
```

```
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer
# corpus
example = [
   'This instrument measures the planes altitude',
   'This instrument measures the planes latitude'
]
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd
count_vectorizer = CountVectorizer()
dt_matrix = count_vectorizer.fit_transform(example)

from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(example)

plane_status = [
   'This instrument measures the planes altitude',
   'This instrument measures the planes latitude',
   'Nothing is wrong everybody onboard is doing ok and enjoying dinner'
]
# tokenizer
tokenizer = Tokenizer(num_words = 100, oov_token="out_of_vocab")
tokenizer.fit_on_texts(plane_status)
word_index = tokenizer.word_index

sequences = tokenizer.texts_to_sequences(plane_status)

test_sequences = tokenizer.texts_to_sequences(test_data)
print(word_index)
print(test_sequences)
from tensorflow.keras.preprocessing.sequence import pad_sequences
padded = pad_sequences(sequences,padding='post',maxlen=10)

import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from bs4 import BeautifulSoup
nltk.download("punkt_tab")#tokenization
```

```
nltk.download("stopwords")#pre-crafted stopwords list
nltk.download("wordnet")# lemmatization

stop_words = set(stopwords.words("english"))
lemmatizer = WordNetLemmatizer()
def clean_and_tokenize(text):
    text=text.lower()
    soup=BeautifulSoup(text)
    text=soup.get_text()
    tokens=nltk.word_tokenize(text)
    processed = []
    for tok in tokens:
        if tok in stop_words:
            continue
        lemma = lemmatizer.lemmatize(tok)
        processed.append(lemma)
    return processed
from gensim.models import Word2Vec
w2v_model = Word2Vec(
    sentences=sentences,
    vector_size=100,
    window=5,
    min_count=5,
    sg=0
)
w2v_model.wv["director"]
```