```
#
# Purdue University Global
#
# IN400 - AI: Deep Learning and Machine Learning
#
# Unit 2 Assignment Part 1 / Module 2 Competency Assessment Part 1
#
# Exploring Machine Learning Tools With TensorFlow
#
# PyCharm Code
#

# NOTE: Ignore any "could not load dynamic library libcudart ..."
# warnings produced during execution. The Codio environment does
# not have GPU capabilites.

# Import the applicable libraries and set the default parameters.
# If the libraries do not exist, they must be installed in the Python environment.

from pandas import read_csv

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

# Data File Location

dataFile = '/home/codio/workspace/data/IN400/ionosphere.data'

# Output Header

print('\nUnit 2 Assignment Part 1 / Module 2 Competency Assessment Part 1\n')

from datetime import datetime

print(datetime.now().strftime("%m/%d/%Y %H:%M:%S"), '\n')

# Load the dataset from the specified location

df = read_csv(dataFile, header=None)

# Peek at the data

print("DATA SNIPPET")

print(df.head())

print()

# Explore the data

print("DATA STATISTICS")

print(df.describe())

print()

# Split the data frame into input (features) and output (Class) columns

X = df.values[:, :-1]

y = df.values[:, -1]

# Make sure the values of all features are of type float

X = X.astype('float32')

# Change the Class column from strings to integer - good: g to 1 and bad: b to 0

y = LabelEncoder().fit_transform(y)

# Split the dataset into an 80% training set and a 20% test.
# The split happens for both the feature part and the class part

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

print("ARRAY SHAPES: TRAINING X-DATA / TEST X-DATA / TRAINING Y-DATA / TEST Y-DATA")

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

print()

# Get the number of feature columns

n_features = X_train.shape[1]

# Define the parameters of the model

model = Sequential()

model.add(Dense(10, activation='relu', kernel_initializer='he_normal', input_shape=(n_features,)))

model.add(Dense(8, activation='relu', kernel_initializer='he_normal'))

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

# Compile the model

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

# Fit the model

model.fit(X_train, y_train, epochs=150, batch_size=32, verbose=0)

# Evaluate the model

Accuracy = model.evaluate(X_test, y_test, verbose=0)

print()

print('Model Accuracy: %.3f' % Accuracy[1])

print()

# Test prediction by creating a new row that covers all the feature columns

row = [1,0,0.99539,-0.05889,0.85243,0.02306,0.83398,-0.37708,1,0.03760,0.85243,-0.17755,0.59755,-0.44945,0.60536,
       -0.38223,0.84356,-0.38542,0.58212,-0.32192,0.56971,-0.29674,0.36946,-0.47357,0.56811,-0.51171,0.41078,-0.46168,
       0.21266,-0.34090,0.42267,-0.54487,0.18641,-0.45300]

TestPred = model.predict([row])

print('Test Prediction: %.3f' % TestPred)
```