ClassicXORproblem

March 3, 2023

0.1 The Classic XOR problem to motivate NNs

[215]:

```
import tensorflow as tf
```

[216]:

```
## !pip install scikit-learn
## !pip install tensorflow_datasets
## !pip install matplotlib
```

[217]:

```
import numpy as np


from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten

from sklearn.metrics import f1_score, precision_score, recall_score,␣
 ↪confusion_matrix

import tensorflow_datasets as tfds

import matplotlib.pyplot as plt
```

[238]:

```
tf.random.set_seed(1)
np.random.seed(1)
```

[239]:

```
X = np.random.uniform(  low=-1, high=1, size=(200, 2)   )
```

[240]:

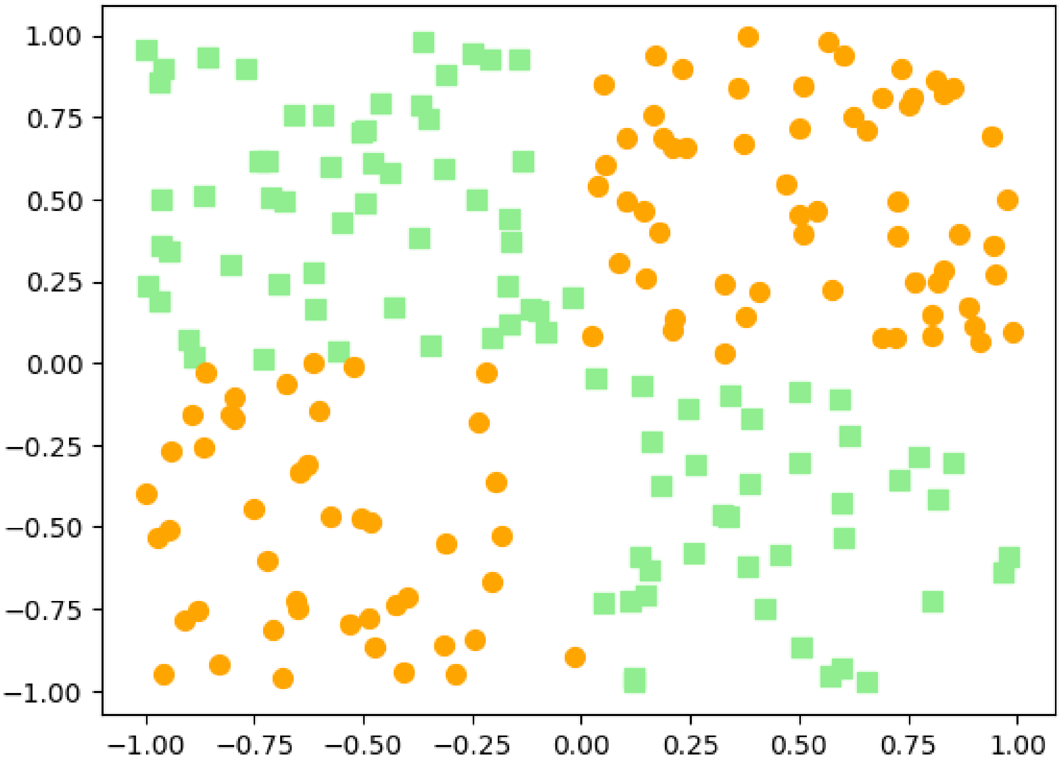
```
y = np.ones(   len(X)    )
```

[241]:

```
## XOR data

y[  X[:, 0] * X[:, 1] < 0   ] = 0
```

[243]:

```
## plt.scatter(X[:, 0], X[:, 1] )


plt.scatter(X[y==0,0],
            X[y==0,1],
            s=50,
            c='lightgreen',
            marker='s',
            label='class0')
plt.scatter(X[y==1,0],
            X[y==1,1],
            s=50,
            c='orange',
            marker='o',
            label='class1')
plt.show()
```



[242]:

```
X_train = X[:100, :]
y_train = y[:100]

X_test  = X[100:, :]
y_test  = y[100:]
```

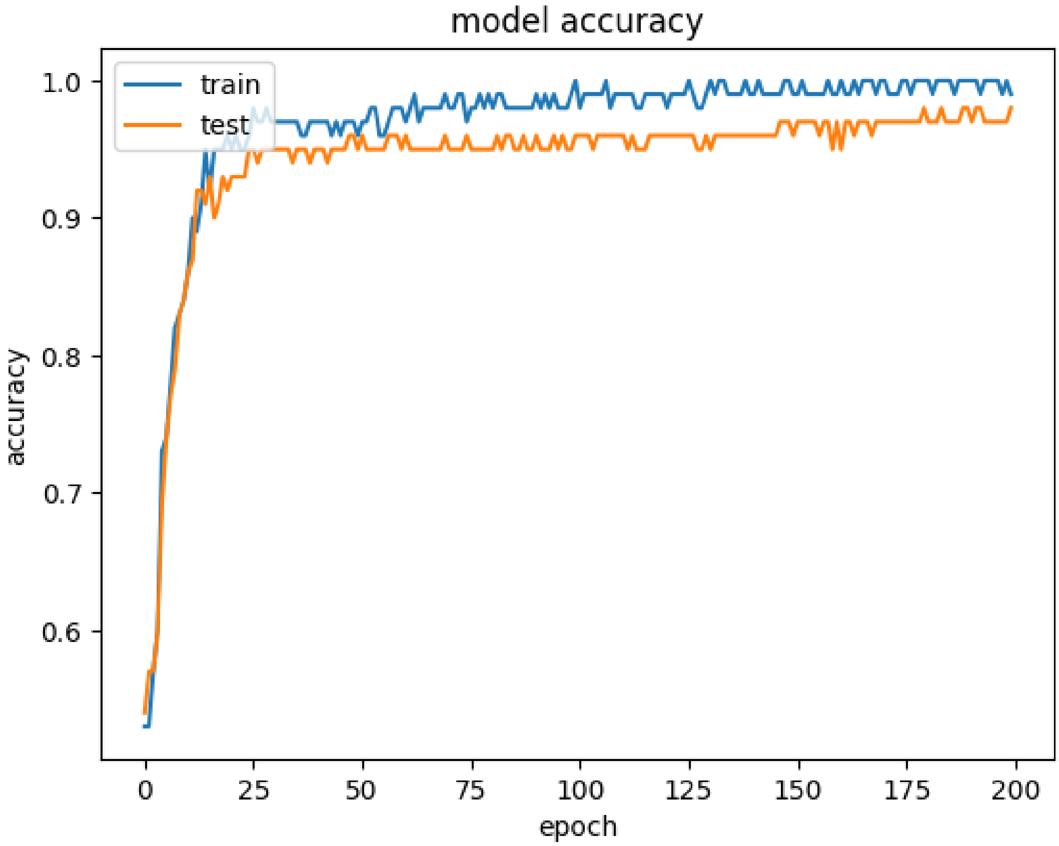
[224]:

```
model = Sequential()
model.add(Dense(16, input_shape=(2,), activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

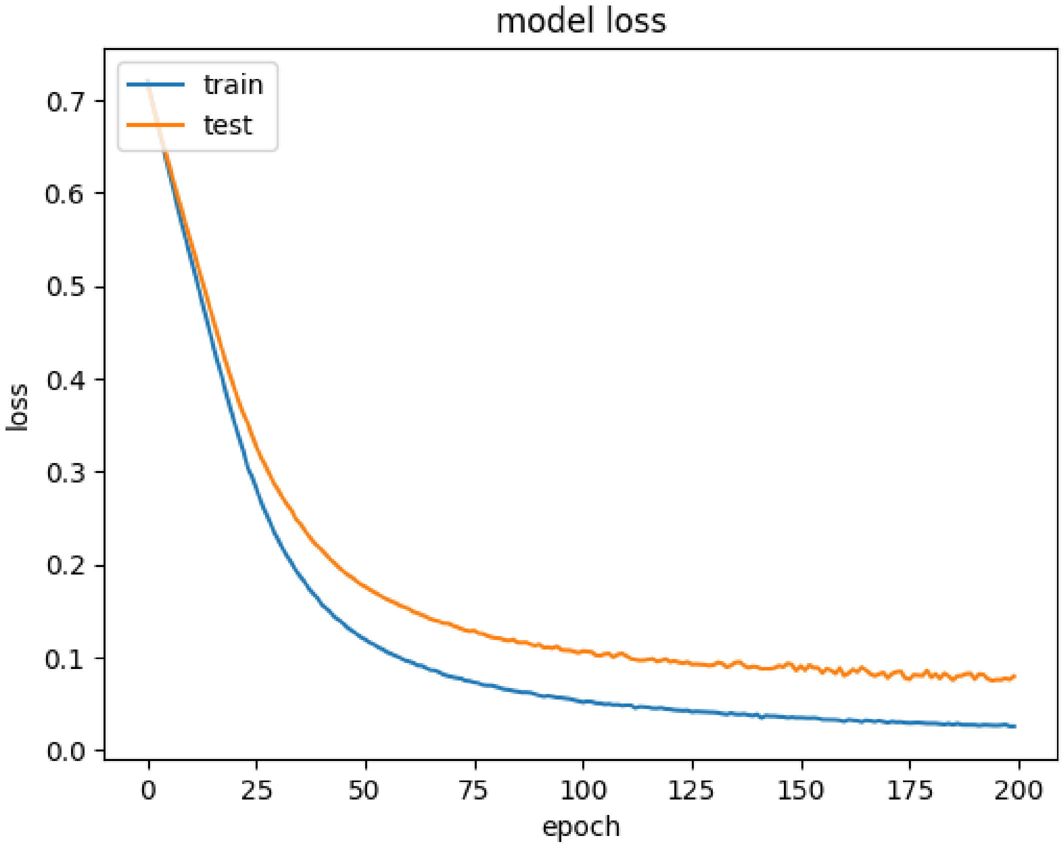
```
optimizer = tf.keras.optimizers.SGD()

model.compile(
    optimizer=optimizer,
    loss=tf.keras.losses.BinaryCrossentropy(),
    metrics=[tf.keras.metrics.BinaryAccuracy()]
)

hist = model.fit(X_train, y_train, validation_data=(X_test, y_test),␣
  ↪epochs=200, batch_size=2, verbose=0)
```

[225]:

```
history = hist


# summarize history for accuracy
plt.plot(history.history['binary_accuracy'])
plt.plot(history.history['val_binary_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



[226]:
```
history = hist

# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



[227]:

```
train_loss, train_accuracy = model.evaluate(X_train, y_train)
test_loss, test_accuracy = model.evaluate(X_test, y_test)

y_pred = model.predict(X_test).round()


f1 = f1_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print('\nTraining accuracy:', train_accuracy)
print('Test accuracy:', test_accuracy)
print("\nF1 score:", f1)
print("Precision score:", precision)
print("Recall score:", recall)
print("Confusion matrix:\n", conf_matrix)
```

```
4/4 [==============================] - 0s 2ms/step - loss: 0.0228 -
binary_accuracy: 1.0000
```

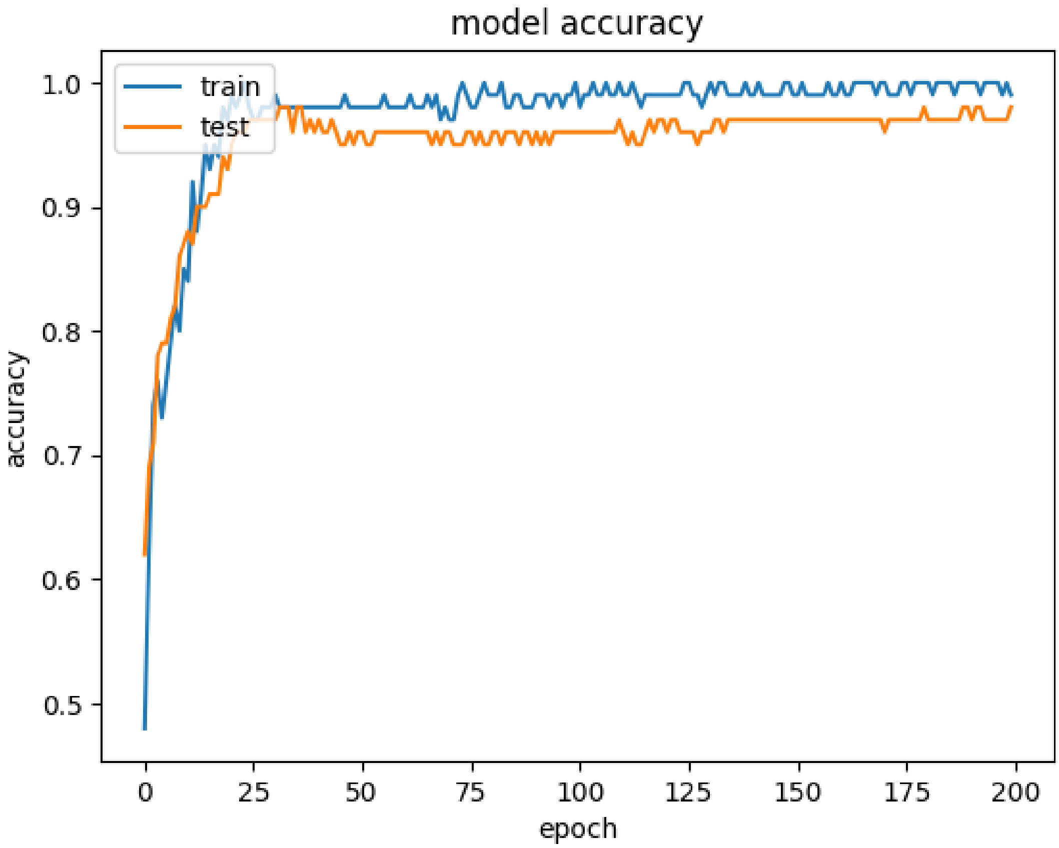
```
4/4 [==============================] - 0s 2ms/step - loss: 0.0793 -
binary_accuracy: 0.9800
4/4 [==============================] - 0s 1ms/step

Training accuracy: 1.0
Test accuracy: 0.9800000190734863

F1 score: 0.9814814814814815
Precision score: 0.9636363636363636
Recall score: 1.0
Confusion matrix:
 [[45  2]
 [ 0 53]]
```

[244]:

```
model = tf.keras.Sequential()
model.add( tf.keras.layers.Dense(units=16, input_shape=(2,), activation='relu'))
model.add(  tf.keras.layers.Dense(units=8, activation='relu'))
model.add(  tf.keras.layers.Dense(units=1, activation='sigmoid'))

optimizer = tf.keras.optimizers.SGD(learning_rate=0.01)

model.compile(
    optimizer=optimizer,
    loss=tf.keras.losses.BinaryCrossentropy(),
    metrics=[tf.keras.metrics.BinaryAccuracy()]
)

hist = model.fit(
    X_train,
    y_train,
    validation_data=(X_test, y_test),
    epochs=200,
    batch_size=2,
    verbose=0
)
```

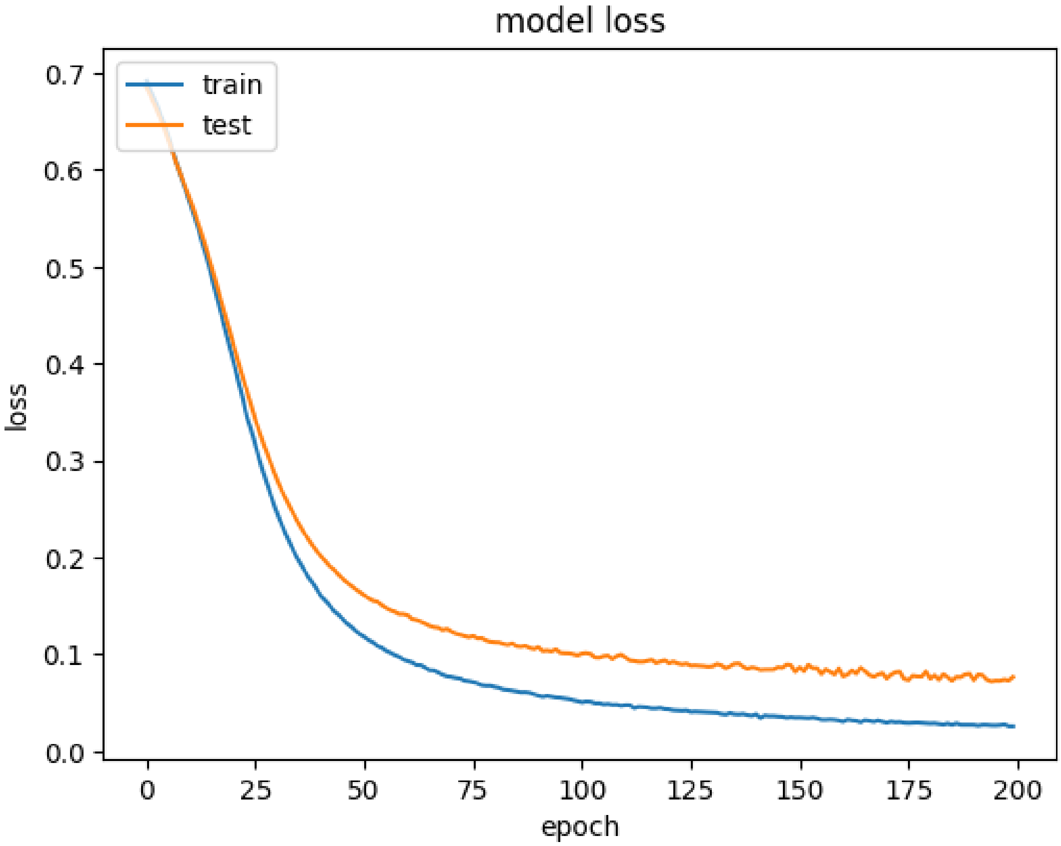
[245]:

```
history = hist


# summarize history for accuracy
plt.plot(history.history['binary_accuracy'])
plt.plot(history.history['val_binary_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



[246]:

```
history = hist

# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



[247]:
```
train_loss, train_accuracy = model.evaluate(X_train, y_train)
test_loss, test_accuracy = model.evaluate(X_test, y_test)

y_pred = model.predict(X_test).round().flatten()



f1 = f1_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print('\nTraining accuracy:', train_accuracy)
print('Test accuracy:', test_accuracy)
print("\nF1 score:", f1)
print("Precision score:", precision)
print("Recall score:", recall)
print("Confusion matrix:\n", conf_matrix)
```

```
4/4 [==============================] - 0s 2ms/step - loss: 0.0229 -
binary_accuracy: 1.0000
```

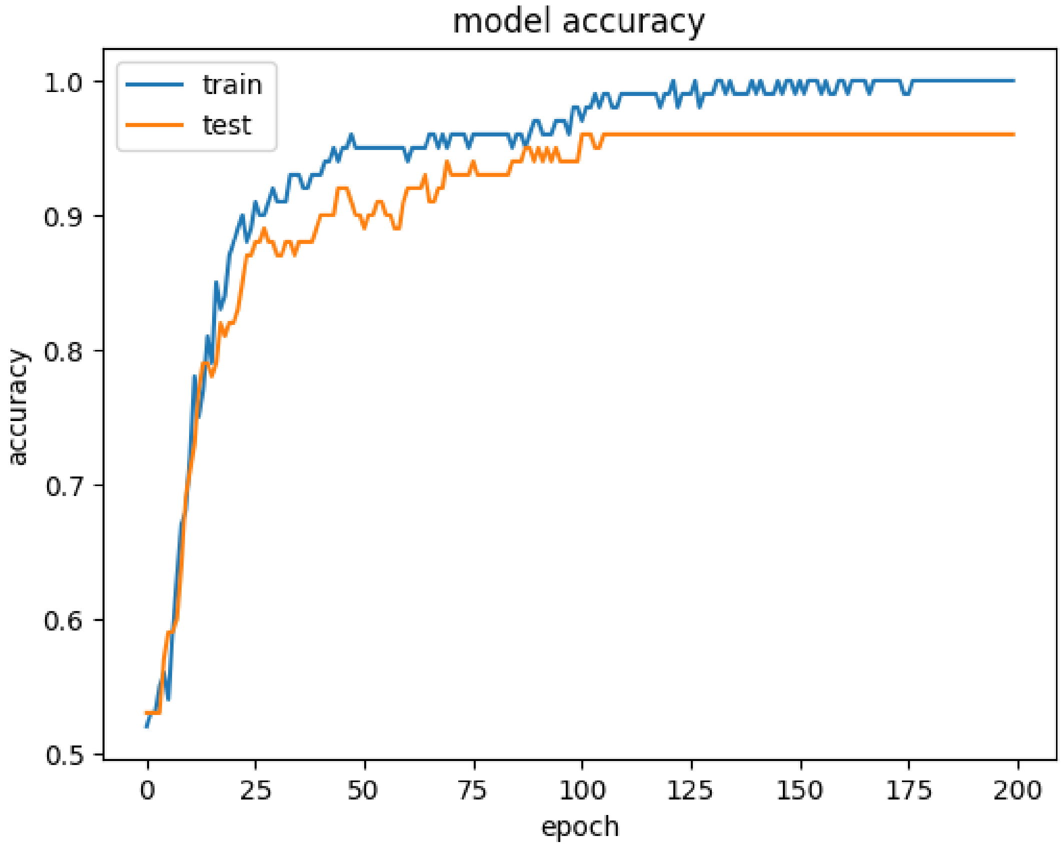
```
4/4 [==============================] - 0s 1ms/step - loss: 0.0765 -
binary_accuracy: 0.9800
4/4 [==============================] - 0s 1ms/step

Training accuracy: 1.0
Test accuracy: 0.9800000190734863

F1 score: 0.9814814814814815
Precision score: 0.9636363636363636
Recall score: 1.0
Confusion matrix:
 [[45  2]
 [ 0 53]]
```

[232]:

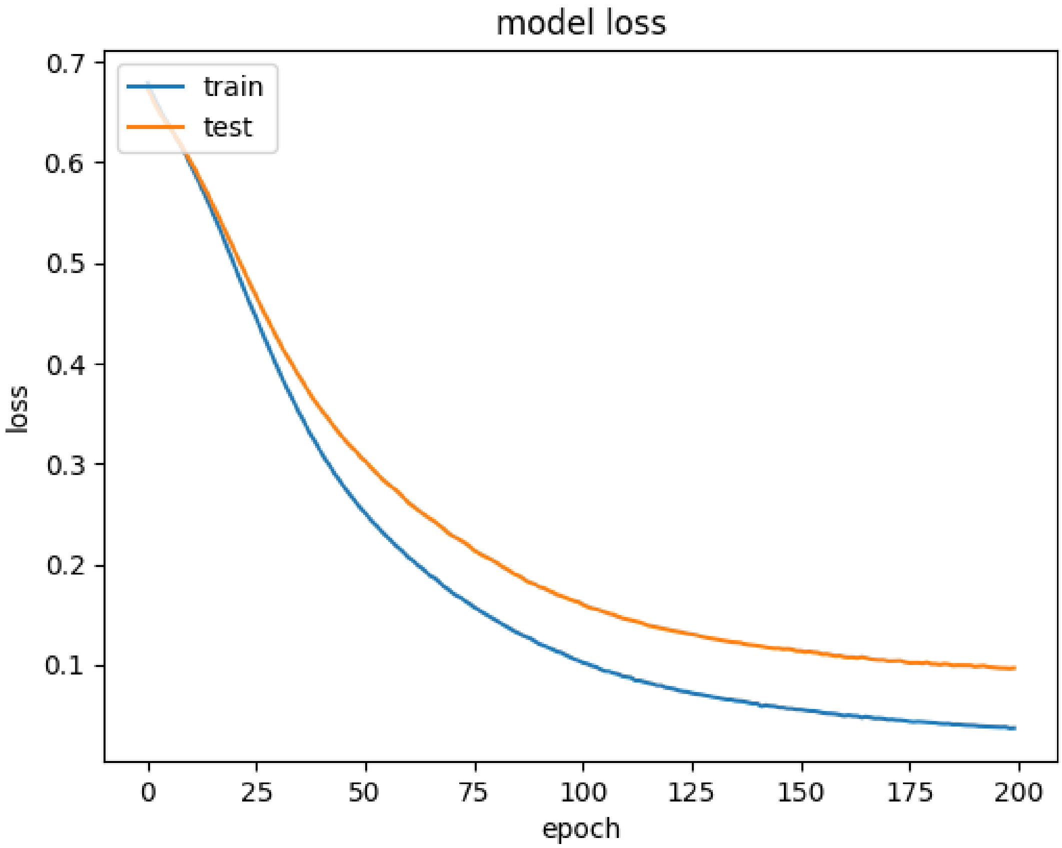
```
model = Sequential()
model.add(Dense(units=64, input_shape=(2,), activation='relu'))
model.add(Dense(units=32, activation='relu'))
model.add(Dense(units=16, activation='relu'))
model.add(Dense(units=1, activation='sigmoid'))

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

history = model.fit(X_train, y_train, validation_data=(X_test, y_test),␣
 ↪epochs=500,
                    batch_size=16, verbose=0)
```

[233]:

```
history = hist



# summarize history for accuracy
plt.plot(history.history['binary_accuracy'])
plt.plot(history.history['val_binary_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



[234]:
```
history = hist

# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



[235]:
```
train_loss, train_accuracy = model.evaluate(X_train, y_train)
test_loss, test_accuracy = model.evaluate(X_test, y_test)

y_pred = model.predict(X_test).round()


f1 = f1_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print('\nTraining accuracy:', train_accuracy)
print('Test accuracy:', test_accuracy)
print("\nF1 score:", f1)
print("Precision score:", precision)
print("Recall score:", recall)
print("Confusion matrix:\n", conf_matrix)
```

```
4/4 [==============================] - 0s 2ms/step - loss: 2.0055e-04 -
accuracy: 1.0000
```

```
4/4 [==============================] - 0s 2ms/step - loss: 0.0593 - accuracy:
0.9800
4/4 [==============================] - 0s 995us/step

Training accuracy: 1.0
Test accuracy: 0.9800000190734863

F1 score: 0.9814814814814815
Precision score: 0.9636363636363636
Recall score: 1.0
Confusion matrix:
 [[45  2]
 [ 0 53]]
```