analysis-nn

May 11, 2020

1 Classification Using Tensorflow Keras

[1]:

```
%load_ext watermark
```

[2]:

```
%watermark -v -m -p numpy,pandas,sklearn,seaborn,matplotlib,tensorflow -g
```

```
/home/hades/anaconda3/envs/test101/lib/python3.7/site-
packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is
deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

```
CPython 3.7.3
IPython 7.9.0

numpy 1.18.1
pandas 1.0.3
sklearn 0.22.1
seaborn 0.9.0
matplotlib 3.1.1
tensorflow 2.2.0

compiler   : GCC 7.3.0
system     : Linux
release    : 4.4.0-18362-Microsoft
machine    : x86_64
processor  : x86_64
CPU cores  : 8
interpreter: 64bit
Git hash   : 2465d217c22cb67de6fc0167c2c295499b5dcf9f
```

[3]:

```
from matplotlib import pyplot as plt
%matplotlib inline

import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, MinMaxScaler, normalize
from sklearn.model_selection import train_test_split
```

```
SEED = 43
np.random.random = SEED
```

[4]:
```
df = pd.read_csv('../datasets/train.csv')
```

[5]:
```
mainTest = pd.read_csv('../datasets/test.csv')
```

1.1 PreScaling Data

[6]:
```
X = df.loc[:, ~df.columns.isin(['blueFirstBlood', 'blueWins'])]
y = df['blueWins']
firstBld = df['blueFirstBlood']
```

[7]:
```
X1 = mainTest.loc[:, ~mainTest.columns.isin(['blueFirstBlood', 'blueWins'])]
y1 = mainTest['blueWins']
firstBld1 = mainTest['blueFirstBlood']
```

[8]:
```
ss = StandardScaler()
mm = MinMaxScaler()
```

[9]:
```
Xstd = ss.fit_transform(X)
Xmm = mm.fit_transform(X)
```

[10]:
```
df_ss = pd.DataFrame(Xstd, columns=X.columns)
df_mm = pd.DataFrame(Xmm, columns=X.columns)
```

[11]:
```
df_ss = pd.concat([df_ss, firstBld], axis=1)
df_mm = pd.concat([df_mm, firstBld], axis=1)
```

[12]:
```
ss1 = StandardScaler()
mm1 = MinMaxScaler()

test_ss = ss1.fit_transform(X1)
test_mm = mm1.fit_transform(X1)

test_ss = pd.DataFrame(test_ss, columns=X1.columns)
test_mm = pd.DataFrame(test_mm, columns=X1.columns)

test_ss = pd.concat([test_ss, firstBld1], axis=1)
test_mm = pd.concat([test_mm, firstBld1], axis=1)
```

1.2 Analysis - UnScaled

[13]:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

[14]:

```
y = keras.utils.to_categorical(y, 2)
```

[15]:

```
X_train, X_test, y_train, y_test = train_test_split(df.loc[:, df.columns !=␣
 ↪'blueWins'], y, test_size=0.33, stratify=y, random_state=SEED)
```

1.2.1 Using 1 Dense Layer softmax

[16]:

```
model = keras.Sequential()
model.add(keras.layers.Dense(2, activation='softmax', input_dim=len(X.columns)␣
 ↪+ 1))
# model.add(keras.layers.Dense(2, activation='relu'))
# model.add(keras.layers.Dense(2, activation='sigmoid'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 2)                 52
=================================================================
Total params: 52
Trainable params: 52
Non-trainable params: 0
_________________________________________________________________
```

[17]:

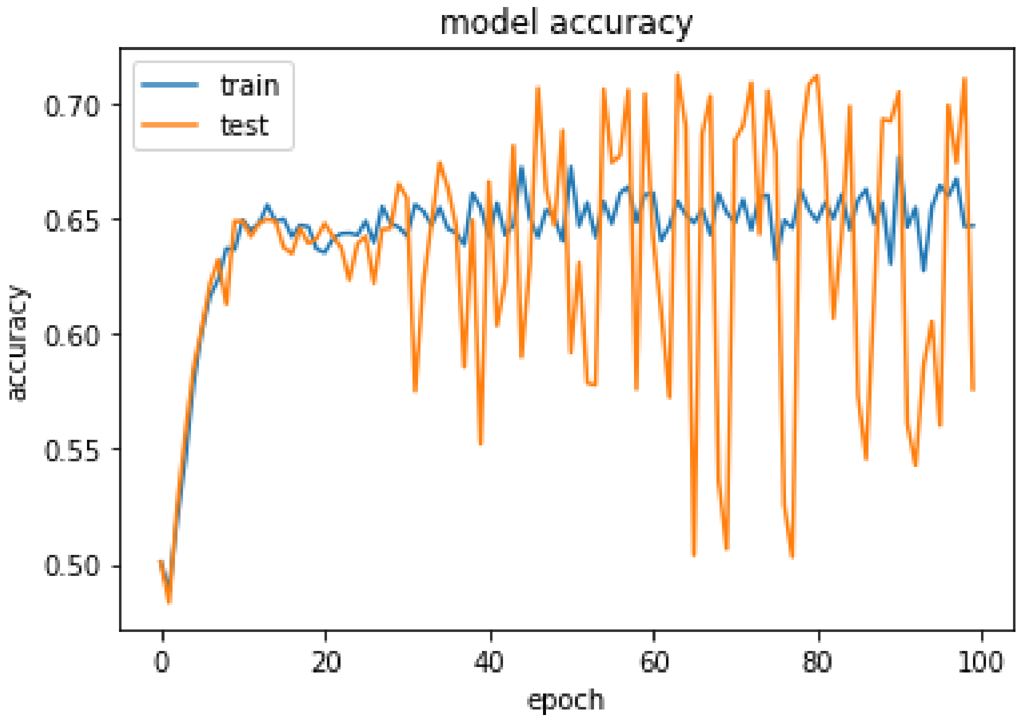
```
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
```

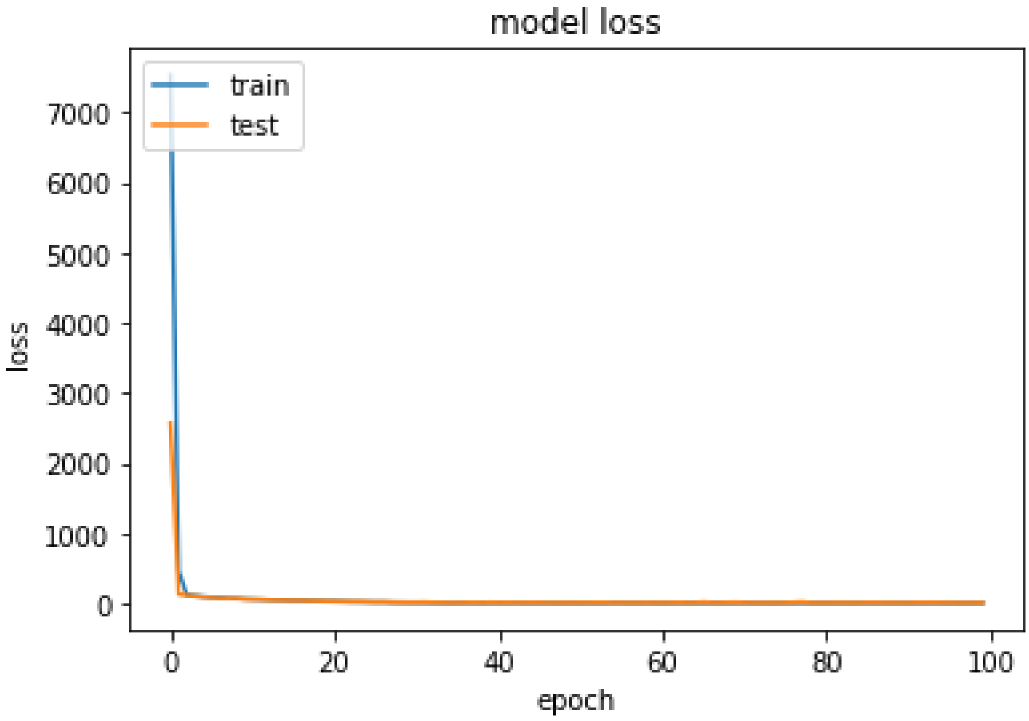
[18]:

```
fits = model.fit(X_train, y_train, epochs=100, validation_data=(X_test,␣
 ↪y_test), verbose=0)
```

[19]:

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





Evaluate Model

```
[20]: y_true = keras.utils.to_categorical(y1, 2)
      model.evaluate(mainTest.loc[:, mainTest.columns!='blueWins'], y_true)
```

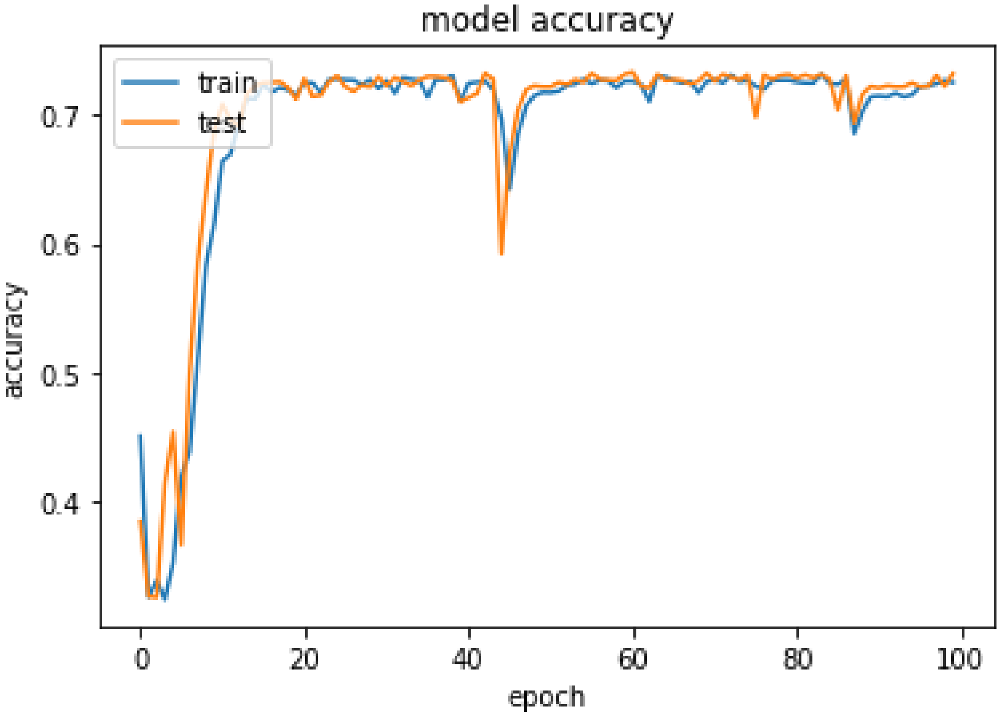
```
93/93 [==============================] - 0s 989us/step - loss: 4.0897 -
accuracy: 0.6059
```

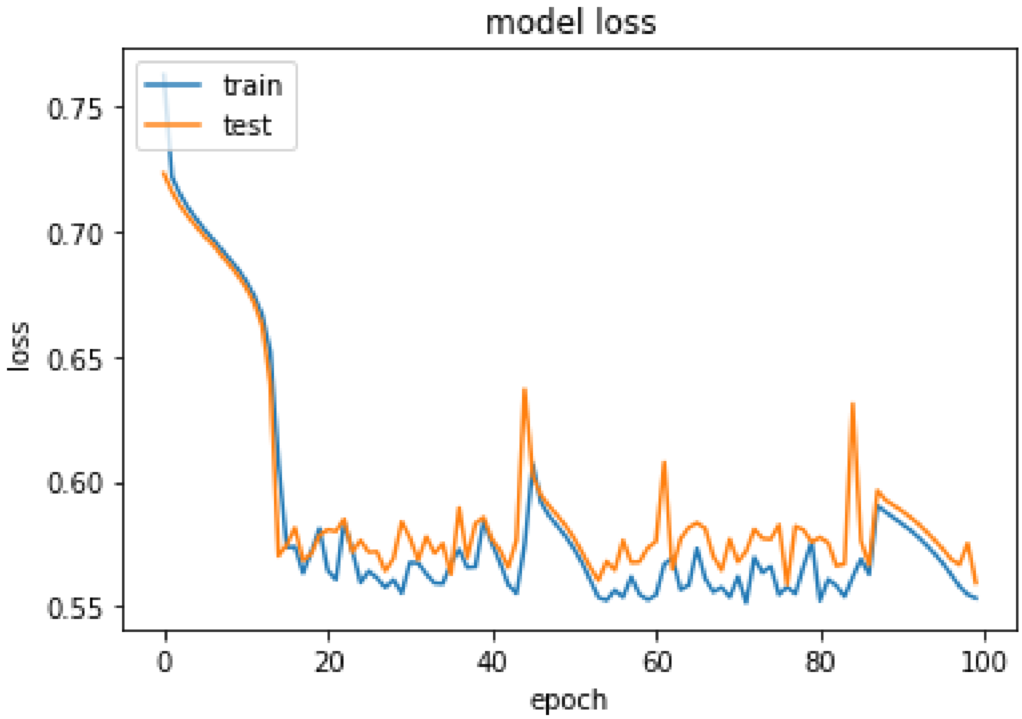
```
[20]: [4.0897297859191895, 0.6059378981590271]
```

1.2.2 Using 1 Dense Layer relu

```
[21]: model = keras.Sequential()
      model.add(keras.layers.Dense(2, activation='relu', input_dim=len(X_test.
       ↪columns)))
      model.summary()
      model.compile(optimizer='adam', loss='categorical_crossentropy',␣
       ↪metrics=['accuracy'])
      fits = model.fit(X_train, y_train, epochs=100, validation_data=(X_test,␣
       ↪y_test), verbose=0)
      plt.plot(fits.history['accuracy'])
      plt.plot(fits.history['val_accuracy'])
      plt.title('model accuracy')
      plt.ylabel('accuracy')
      plt.xlabel('epoch')
      plt.legend(['train', 'test'], loc='upper left')
      plt.show()

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

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 2)                 52
=================================================================
Total params: 52
Trainable params: 52
Non-trainable params: 0
_________________________________________________________________
```





Evalute

[22]:
```
y_true = keras.utils.to_categorical(y1, 2)
model.evaluate(mainTest.loc[:, mainTest.columns!='blueWins'], y_true)
```

```
93/93 [==============================] - 0s 984us/step - loss: 0.5521 -
accuracy: 0.7233
```

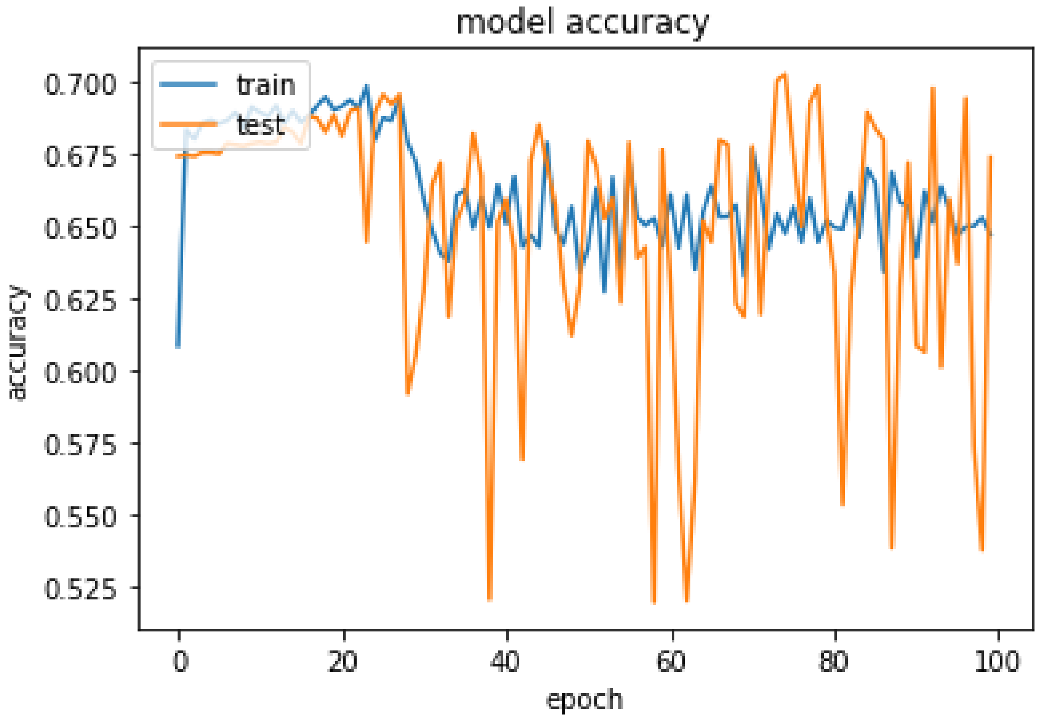
[22]:
```
[0.5521367788314819, 0.7233468294143677]
```

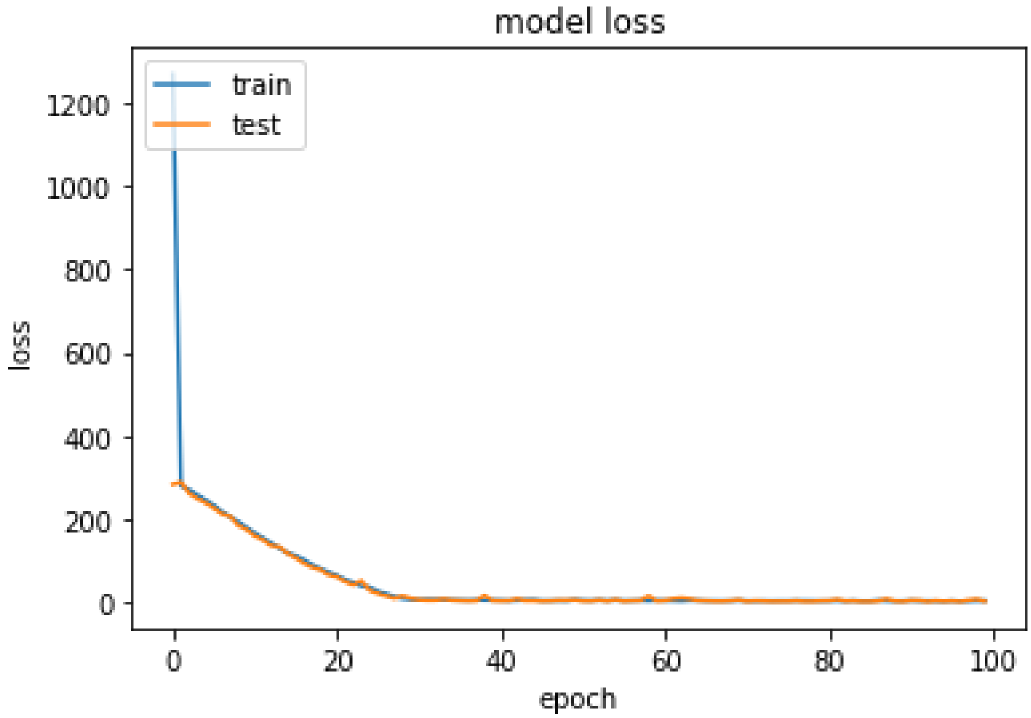
1.2.3 Using 2 Dense Layer - relu and softmax

[23]:
```
model = keras.Sequential()
model.add(keras.layers.Dense(2, activation='relu', input_dim=len(X_test.
 ↪columns)))
model.add(keras.layers.Dense(2, activation='softmax'))
model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
fits = model.fit(X_train, y_train, epochs=100, validation_data=(X_test,␣
 ↪y_test), verbose=0)
plt.plot(fits.history['accuracy'])
plt.plot(fits.history['val_accuracy'])
```

```python
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

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

```
Model: "sequential_2"
-----------------------------------------------------------------
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 2)                 52
-----------------------------------------------------------------
dense_3 (Dense)              (None, 2)                 6
=================================================================
Total params: 58
Trainable params: 58
Non-trainable params: 0
-----------------------------------------------------------------
```





Evaluate

[24]:
```
y_true = keras.utils.to_categorical(y1, 2)
model.evaluate(mainTest.loc[:, mainTest.columns!='blueWins'], y_true)
```

```
93/93 [==============================] - 0s 1ms/step - loss: 2.8175 - accuracy:
0.6947
```

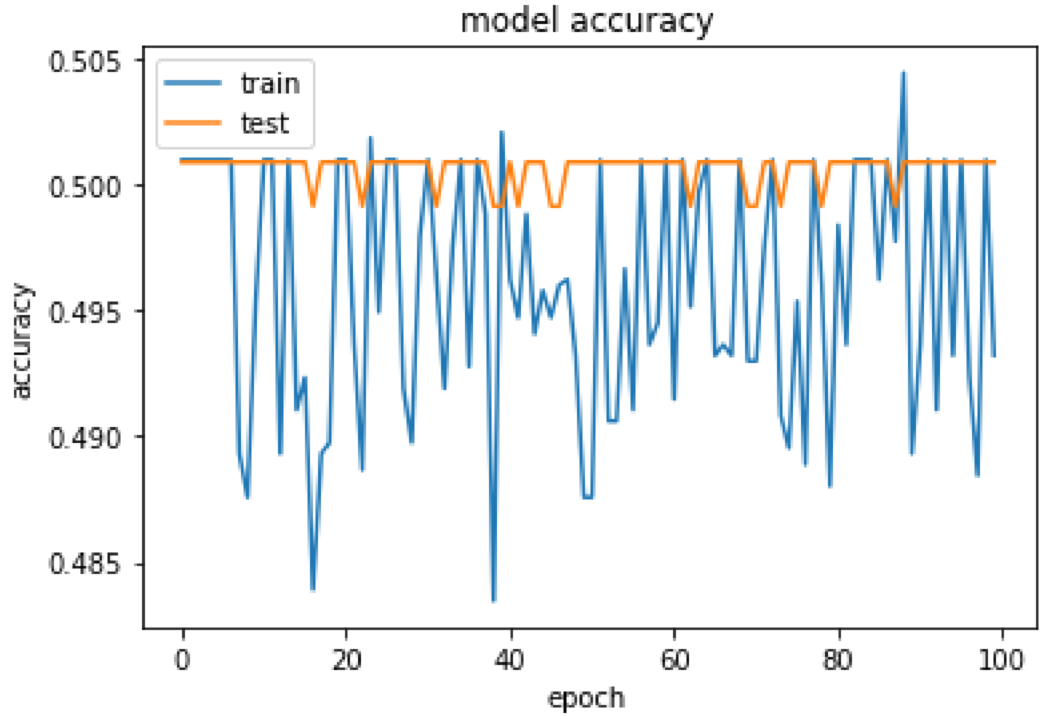
[24]:
```
[2.8174984455108643, 0.6946693658828735]
```

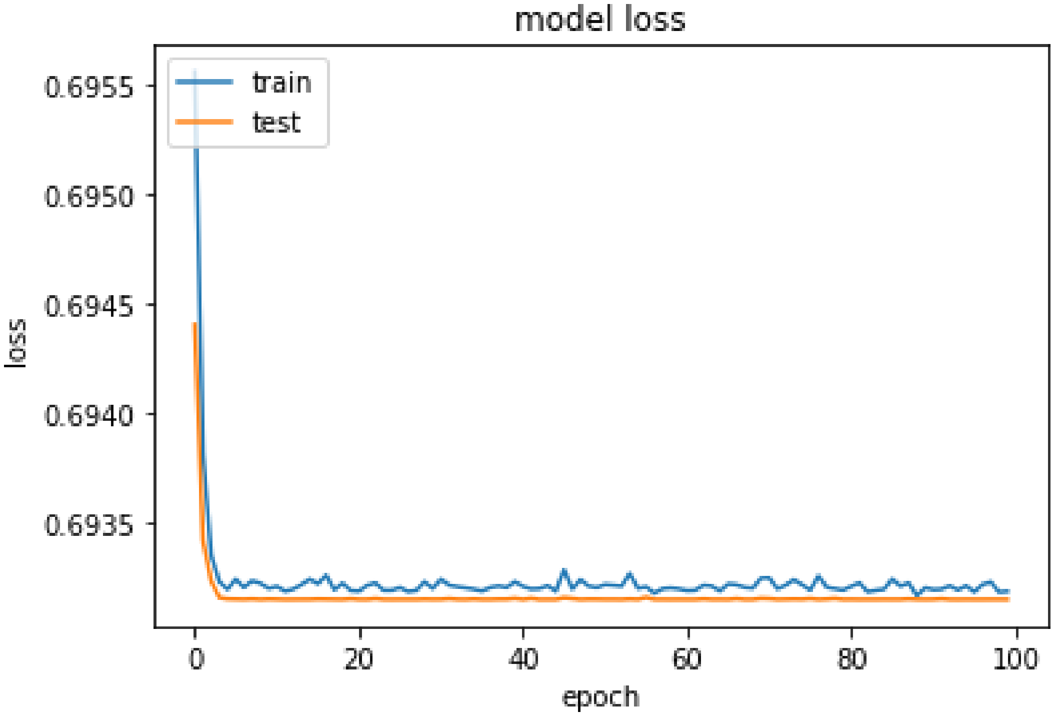
1.2.4 Using 2 Dense Layer - softmax and sigmoid

[25]:
```
model = keras.Sequential()
model.add(keras.layers.Dense(2, activation='softmax', input_dim=len(X_test.
 ↪columns)))
model.add(keras.layers.Dense(2, activation='sigmoid'))
model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
```

```
fits = model.fit(X_train, y_train, epochs=100, validation_data=(X_test, y_test), verbose=0)
plt.plot(fits.history['accuracy'])
plt.plot(fits.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

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

# Evaluate
y_true = keras.utils.to_categorical(y1, 2)
model.evaluate(mainTest.loc[:, mainTest.columns!='blueWins'], y_true)
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 2)                 52
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 6
=================================================================
Total params: 58
Trainable params: 58
Non-trainable params: 0
_________________________________________________________________
```


model accuracy
train
test
0.505
0.500
0.495
0.490
0.485
accuracy
0
20
40
60
80
100
epoch

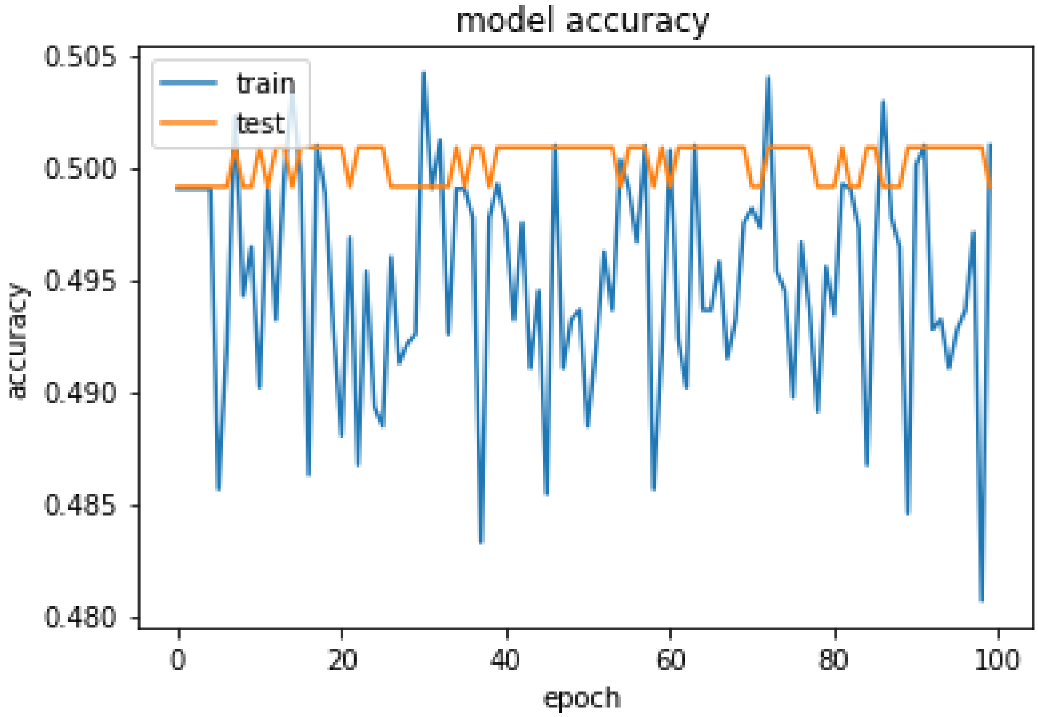

model loss
train
test
0.6955
0.6950
0.6945
0.6940
0.6935
loss
0
20
40
60
80
100
epoch

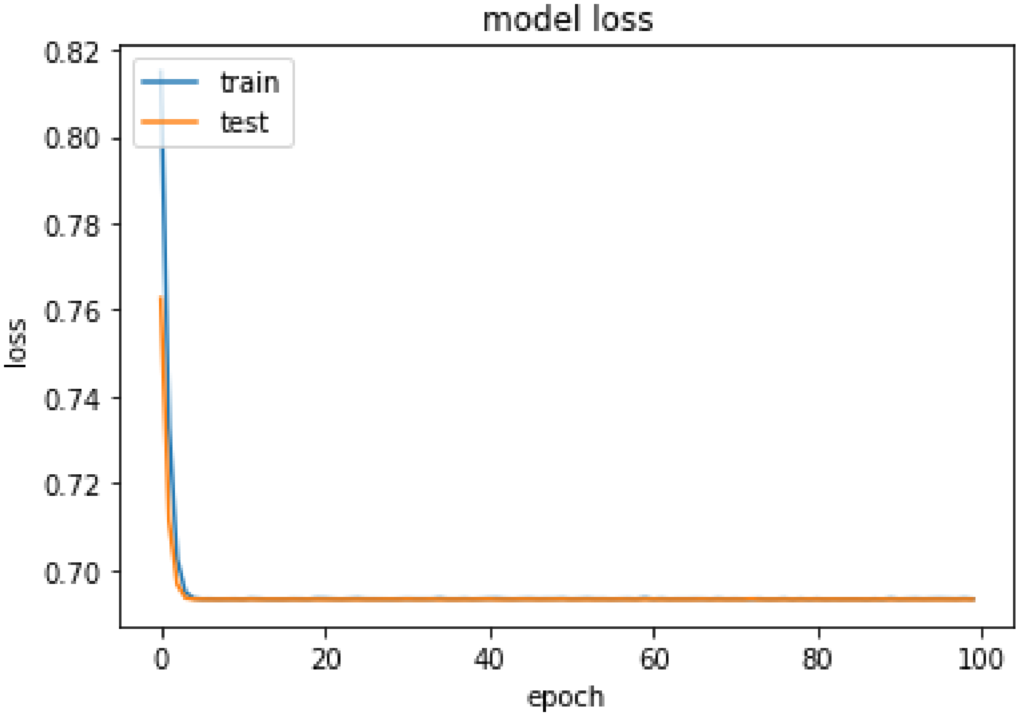
```
93/93 [==============================] - 0s 957us/step - loss: 0.6931 -
accuracy: 0.5010
```

```
[25]: [0.6931451559066772, 0.5010121464729309]
```

```
[26]: model = keras.Sequential()
      model.add(keras.layers.Dense(2, activation='softmax', input_dim=len(X_test.
       ↪columns)))
      model.add(keras.layers.Dense(2, activation='softmax'))
      model.summary()
      model.compile(optimizer='adam', loss='categorical_crossentropy',␣
       ↪metrics=['accuracy'])
      fits = model.fit(X_train, y_train, epochs=100, validation_data=(X_test,␣
       ↪y_test), verbose=0)
      plt.plot(fits.history['accuracy'])
      plt.plot(fits.history['val_accuracy'])
      plt.title('model accuracy')
      plt.ylabel('accuracy')
      plt.xlabel('epoch')
      plt.legend(['train', 'test'], loc='upper left')
      plt.show()

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

      # Evaluate
      y_true = keras.utils.to_categorical(y1, 2)
      model.evaluate(mainTest.loc[:, mainTest.columns!='blueWins'], y_true)
```

```
Model: "sequential_4"

-----------------------------------------------------------------
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 2)                 52

-----------------------------------------------------------------
dense_7 (Dense)              (None, 2)                 6
=================================================================
Total params: 58
Trainable params: 58
Non-trainable params: 0

-----------------------------------------------------------------
```


model accuracy
train
test
0.505
0.500
0.495
0.490
0.485
0.480
accuracy
0
20
40
60
80
100
epoch

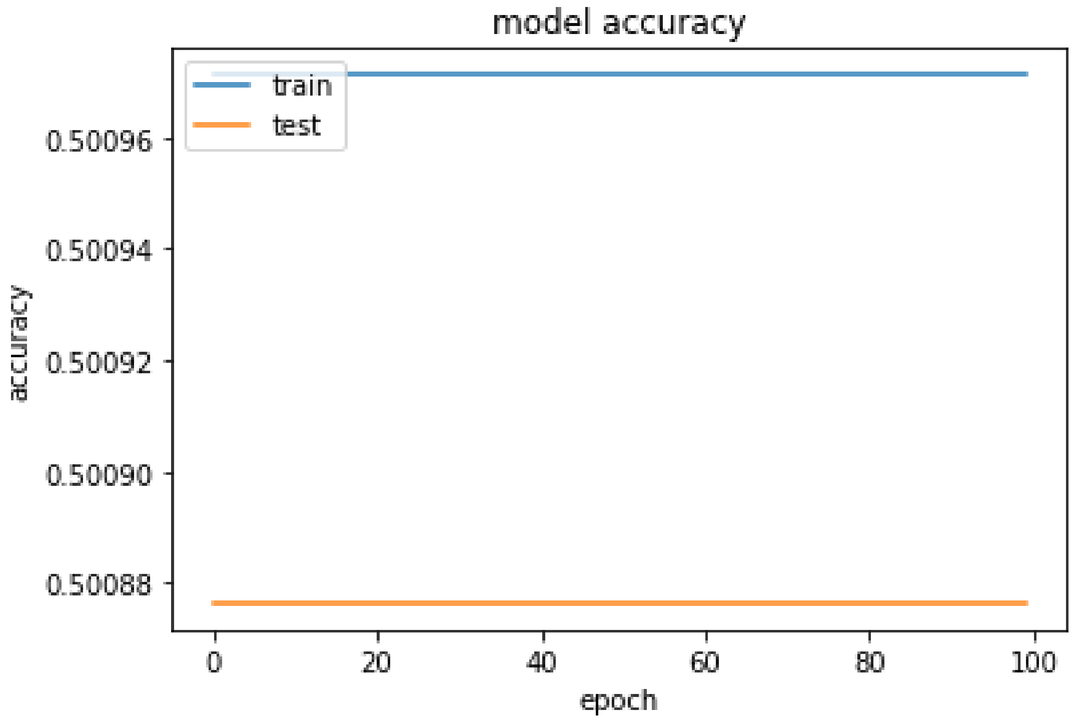

model loss
train
test
0.82
0.80
0.78
0.76
0.74
0.72
0.70
loss
0
20
40
60
80
100
epoch

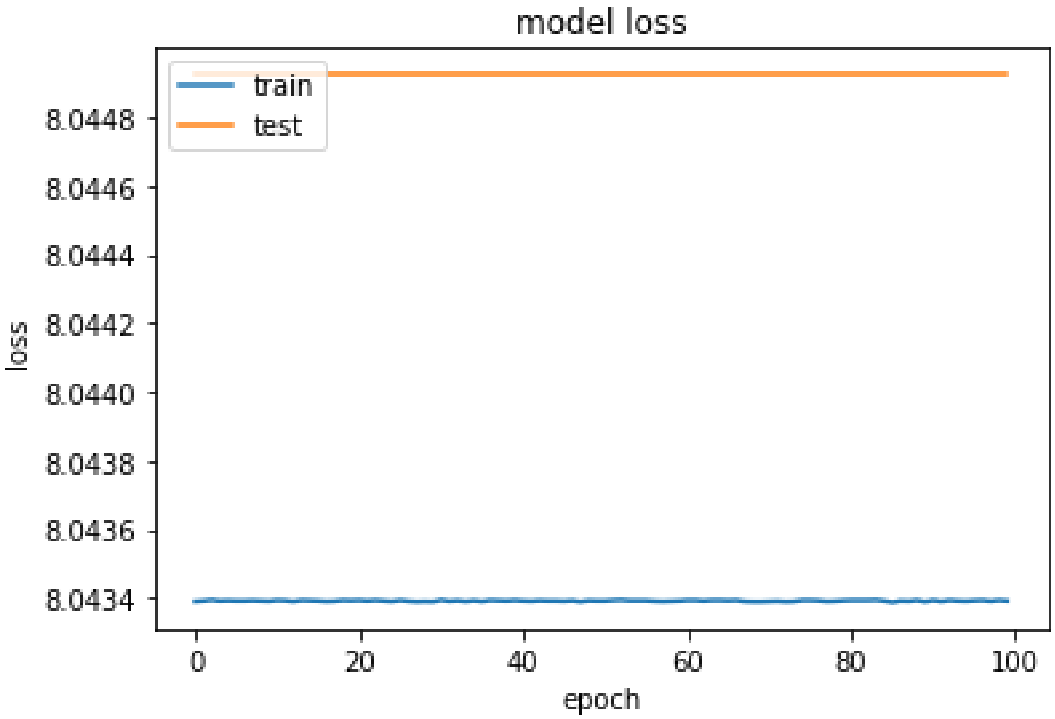
```
93/93 [==============================] - 0s 922us/step - loss: 0.6931 - accuracy: 0.4990
```

[26]:
```
[0.6931474208831787, 0.4989878535270691]
```

[27]:
```
model = keras.Sequential()
model.add(keras.layers.Dense(2, activation='sigmoid', input_dim=len(X_test.columns)))
# model.add(keras.layers.Dense(2, activation='softmax'))
model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
fits = model.fit(X_train, y_train, epochs=100, validation_data=(X_test, y_test), verbose=0)
plt.plot(fits.history['accuracy'])
plt.plot(fits.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

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

# Evaluate
y_true = keras.utils.to_categorical(y1, 2)
model.evaluate(mainTest.loc[:, mainTest.columns!='blueWins'], y_true)
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_8 (Dense)              (None, 2)                 52
=================================================================
Total params: 52
Trainable params: 52
Non-trainable params: 0
_________________________________________________________________
```


model accuracy
train
test
accuracy
0.50096
0.50094
0.50092
0.50090
0.50088
0
20
40
60
80
100
epoch


model loss
train
test
loss
8.0448
8.0446
8.0444
8.0442
8.0440
8.0438
8.0436
8.0434
0
20
40
60
80
100
epoch

```
93/93 [==============================] - 0s 1ms/step - loss: 8.0427 - accuracy:
0.5010
```

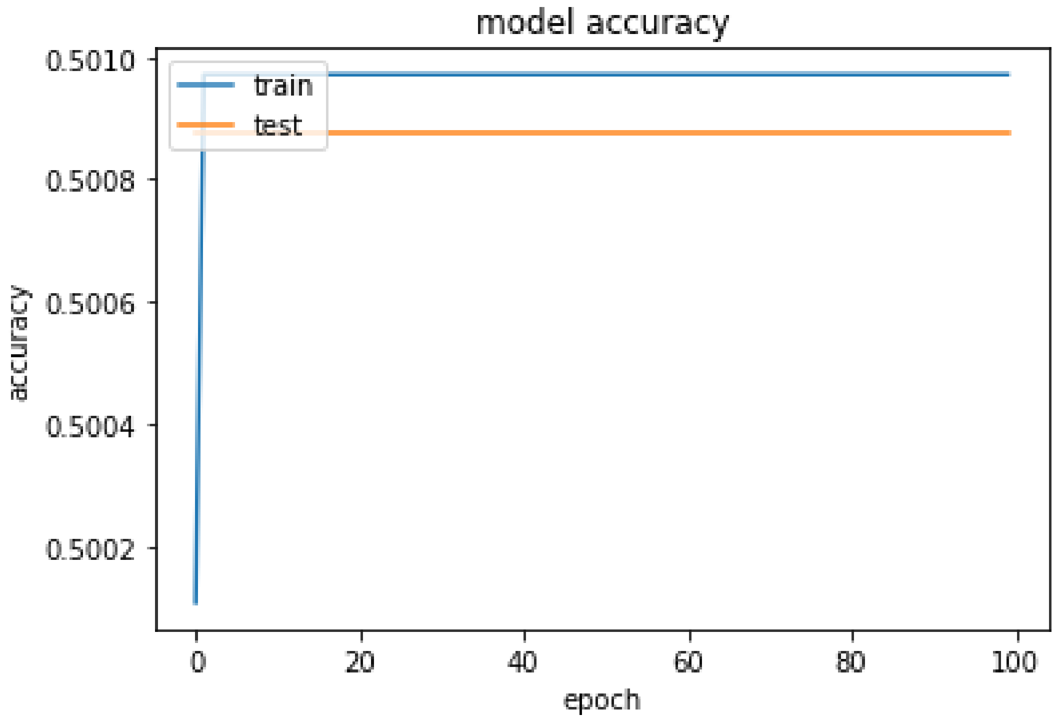
[27]:
```
[8.04273509979248, 0.5010121464729309]
```

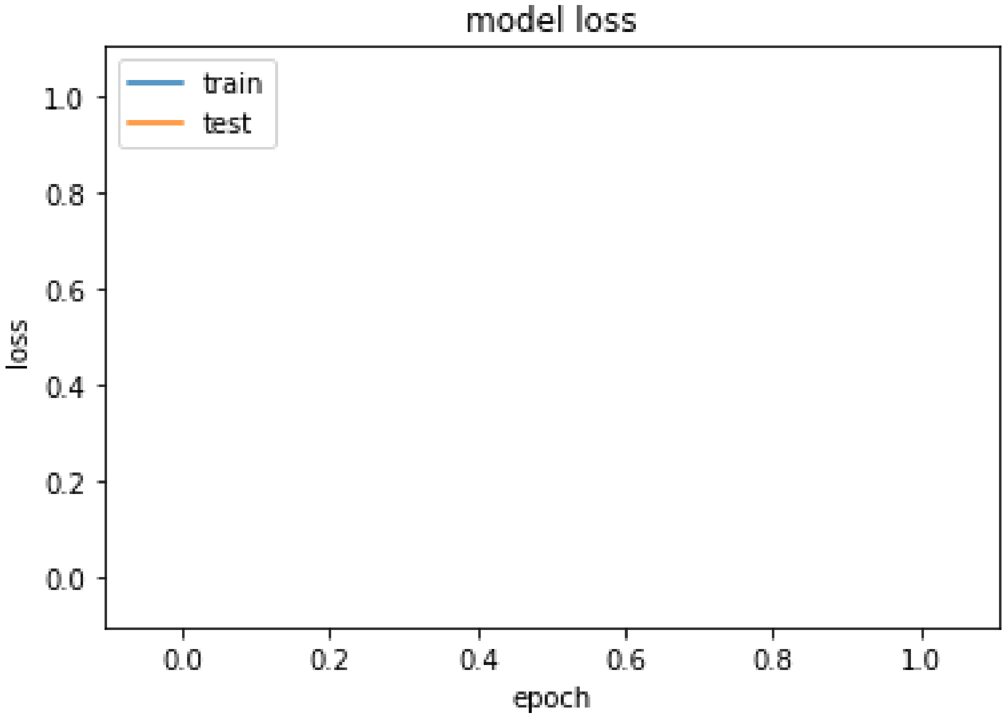
1.3 Standard Scale

[28]:
```
# y_cat = keras.utils.to_categorical(y, 2)
X_train, X_test, y_train, y_test = train_test_split(df_ss, y, test_size=0.33,␣
 ↪stratify=y, random_state=SEED)
```

[29]:
```
model = keras.Sequential()
model.add(keras.layers.Dense(2, activation='relu', input_dim=len(X_test.
 ↪columns)))
# model.add(keras.layers.Dense(2, activation='sigmoid'))
model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
fits = model.fit(X_train, y_train, epochs=100, validation_data=(X_test,␣
 ↪y_test), verbose=0)
plt.plot(fits.history['accuracy'])
plt.plot(fits.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

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

# Evaluate
y_true = keras.utils.to_categorical(y1, 2)
model.evaluate(test_ss, y_true)
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_9 (Dense)              (None, 2)                 52
=================================================================
Total params: 52
Trainable params: 52
Non-trainable params: 0
_________________________________________________________________
```



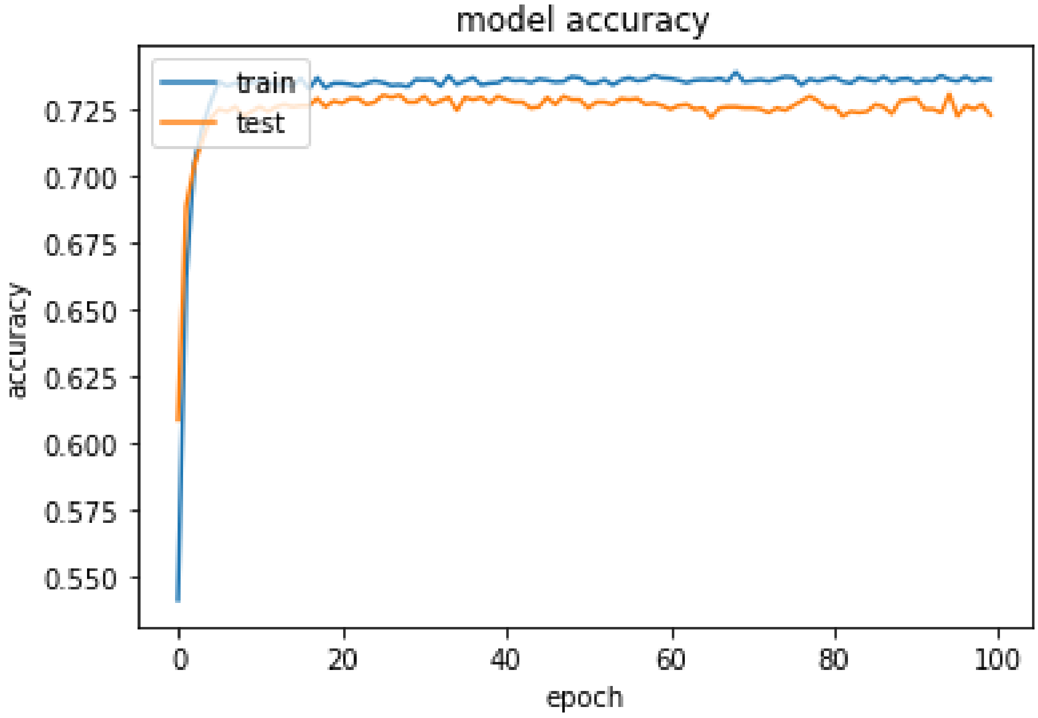


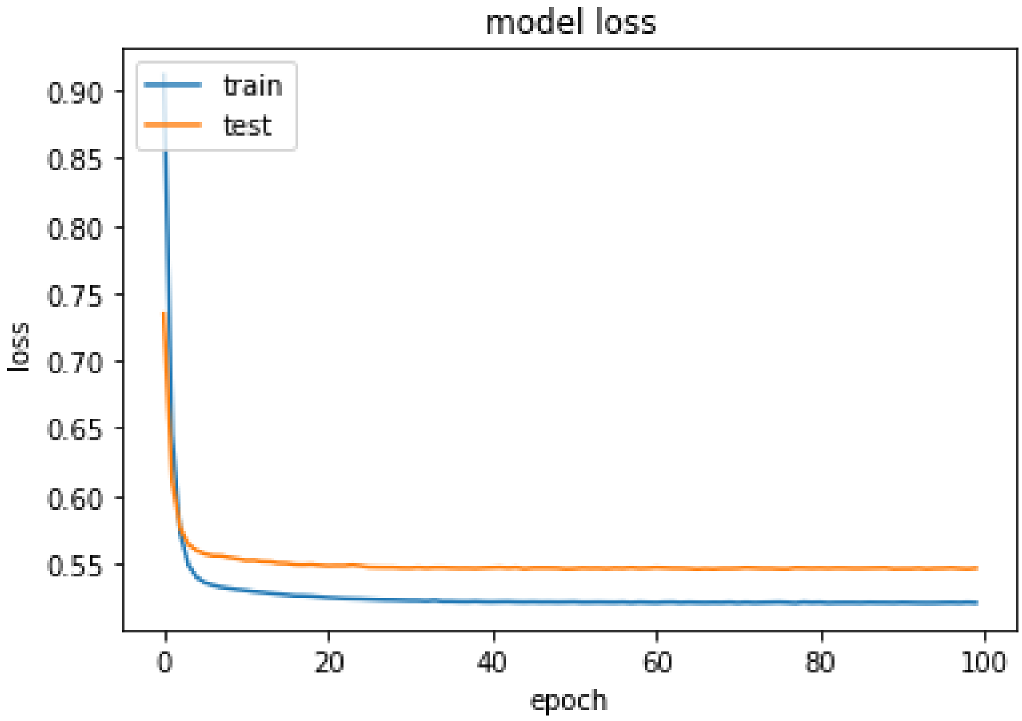
```
93/93 [==============================] - 0s 854us/step - loss: nan - accuracy:
0.5010
```

[29]:
```
[nan, 0.5010121464729309]
```

[30]:
```
model = keras.Sequential()
model.add(keras.layers.Dense(2, activation='softmax', input_dim=len(X_test.
 ↪columns)))
# model.add(keras.layers.Dense(2, activation='sigmoid'))
model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
fits = model.fit(X_train, y_train, epochs=100, validation_data=(X_test,␣
 ↪y_test), verbose=0)
plt.plot(fits.history['accuracy'])
plt.plot(fits.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

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

# Evaluate
y_true = keras.utils.to_categorical(y1, 2)
model.evaluate(test_ss, y_true)
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_10 (Dense)             (None, 2)                 52
=================================================================
Total params: 52
Trainable params: 52
Non-trainable params: 0
_________________________________________________________________
```





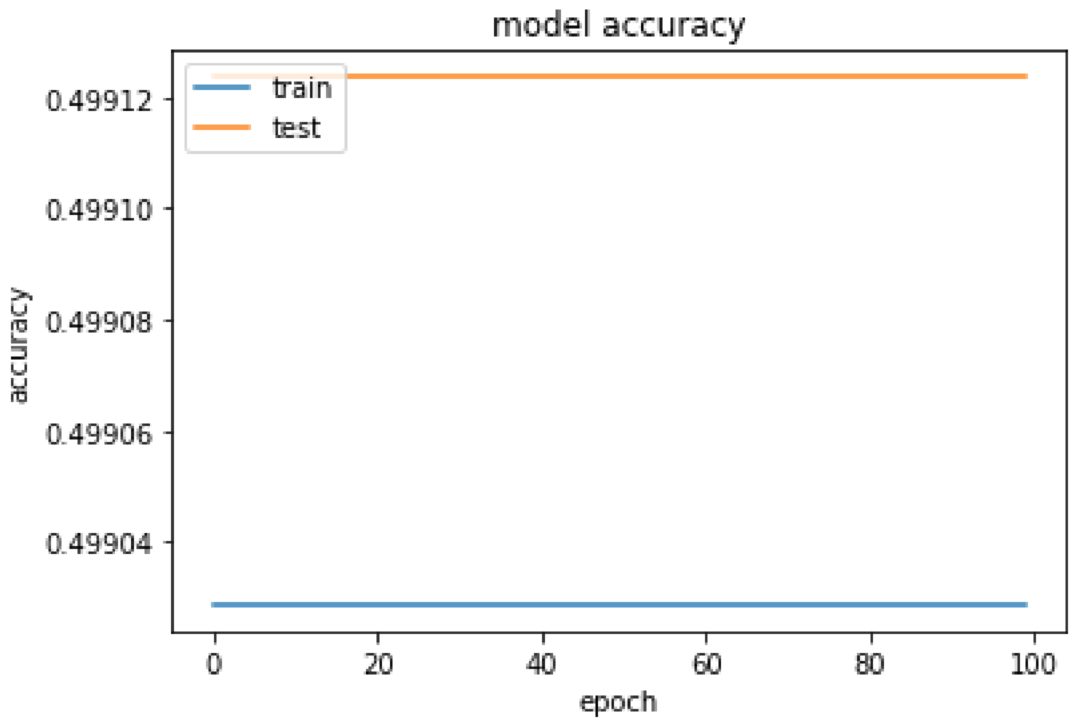
```
93/93 [==============================] - 0s 904us/step - loss: 0.5300 -
accuracy: 0.7244
```

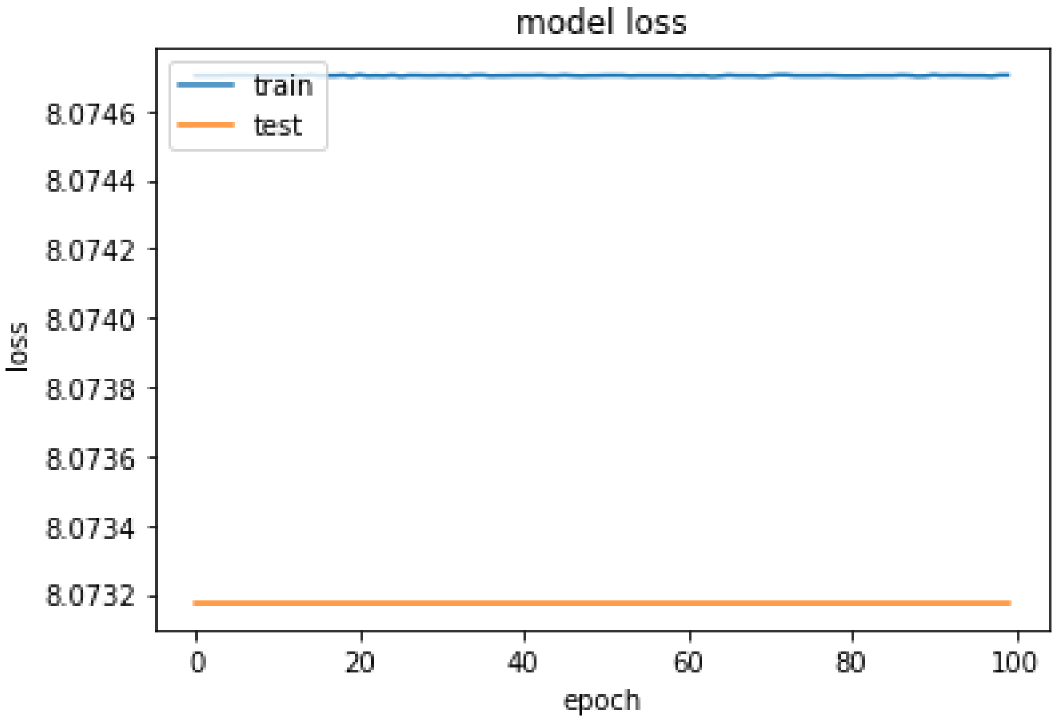
[30]:
```
[0.5299619436264038, 0.7243589758872986]
```

[31]:
```
model = keras.Sequential()
model.add(keras.layers.Dense(2, activation='softmax', input_dim=len(X_test.
 ↪columns)))
model.add(keras.layers.Dense(2, activation='relu'))
model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
fits = model.fit(X_train, y_train, epochs=100, validation_data=(X_test,␣
 ↪y_test), verbose=0)
plt.plot(fits.history['accuracy'])
plt.plot(fits.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

plt.plot(fits.history['loss'])
```

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

# Evaluate
y_true = keras.utils.to_categorical(y1, 2)
model.evaluate(test_ss, y_true)
```

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_11 (Dense)             (None, 2)                 52
_________________________________________________________________
dense_12 (Dense)             (None, 2)                 6
=================================================================
Total params: 58
Trainable params: 58
Non-trainable params: 0
_________________________________________________________________
```





```
93/93 [==============================] - 0s 2ms/step - loss: 8.0754 - accuracy:
0.4990
```

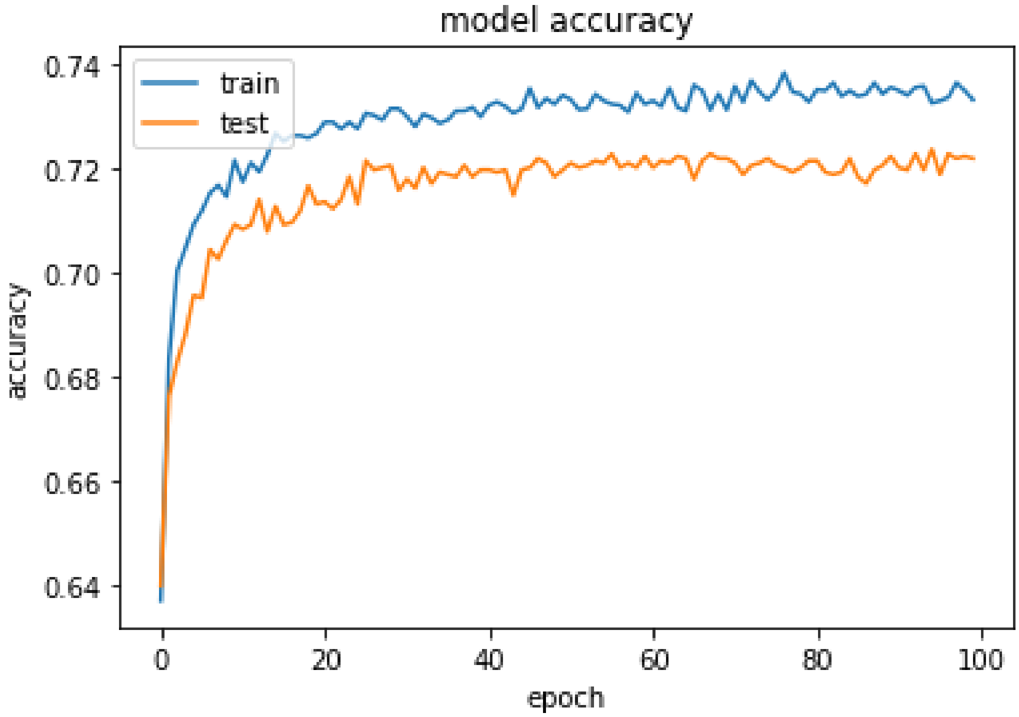
[31]:
```
[8.075363159179688, 0.4989878535270691]
```

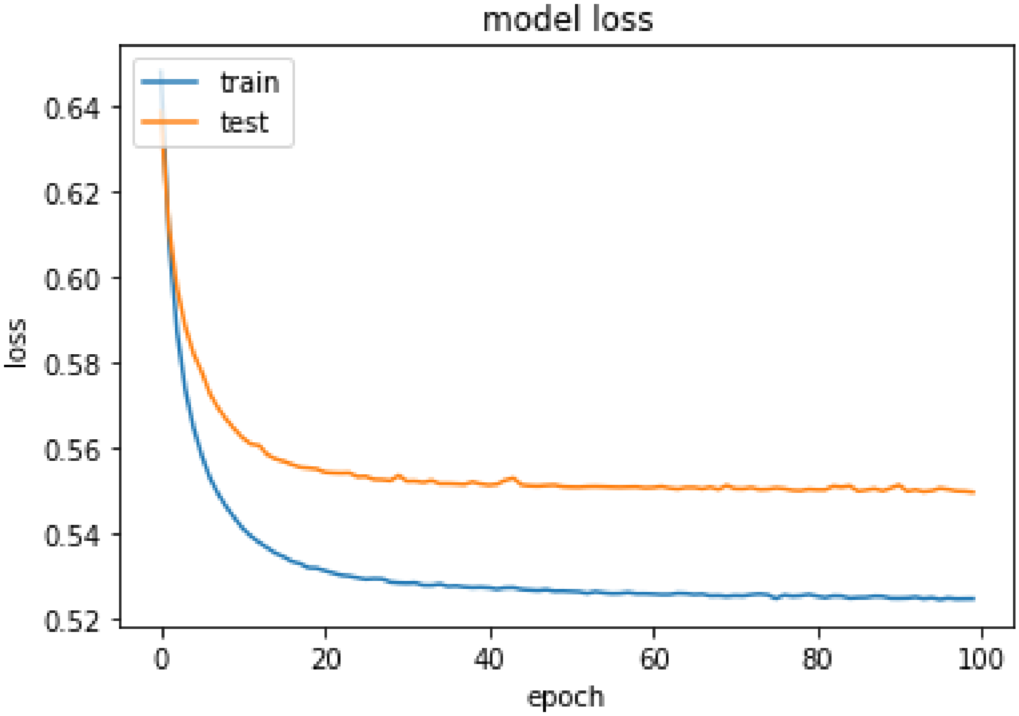
1.4 Min Max

[32]:
```
# y_cat = keras.utils.to_categorical(y, 2)
X_train, X_test, y_train, y_test = train_test_split(df_mm, y, test_size=0.33,␣
 ↪stratify=y, random_state=SEED)
```

[33]:
```
model = keras.Sequential()
model.add(keras.layers.Dense(2, activation='softmax', input_dim=len(X_test.
 ↪columns)))
# model.add(keras.layers.Dense(2, activation='sigmoid'))
model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
fits = model.fit(X_train, y_train, epochs=100, validation_data=(X_test,␣
 ↪y_test), verbose=0)
plt.plot(fits.history['accuracy'])
plt.plot(fits.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

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

# Evaluate
y_true = keras.utils.to_categorical(y1, 2)
model.evaluate(test_mm, y_true)
```

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_13 (Dense)             (None, 2)                 52
=================================================================
Total params: 52
Trainable params: 52
Non-trainable params: 0
_________________________________________________________________
```


model accuracy
accuracy
epoch
train
test
0.74
0.72
0.70
0.68
0.66
0.64
0
20
40
60
80
100

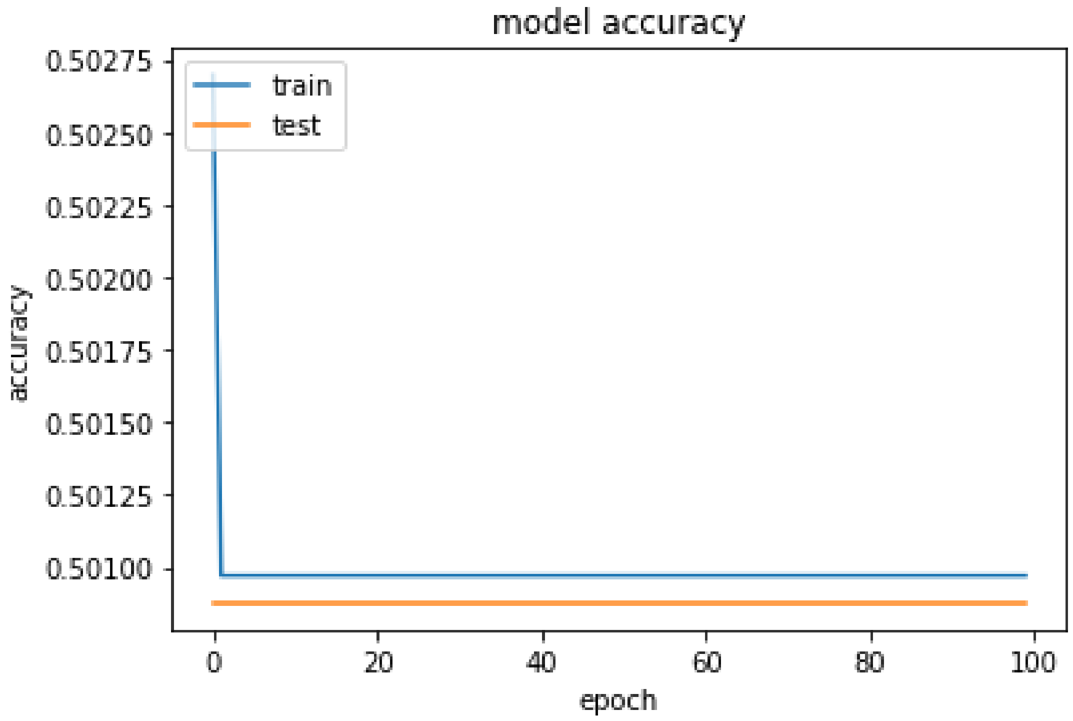

model loss
loss
epoch
train
test
0.64
0.62
0.60
0.58
0.56
0.54
0.52
0
20
40
60
80
100

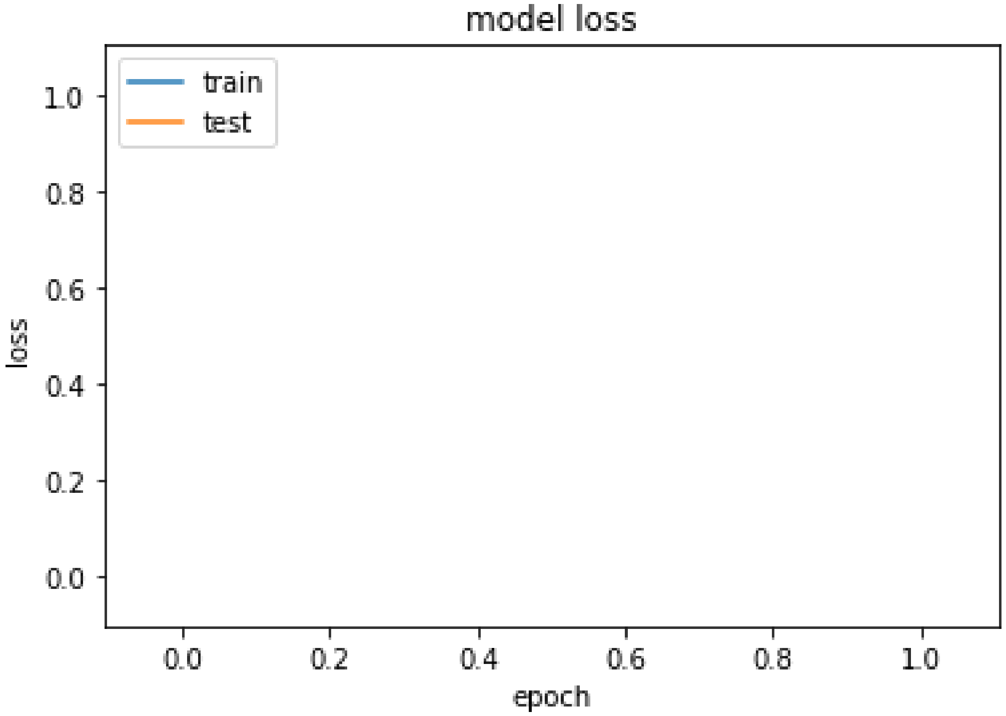
```
93/93 [==============================] - 0s 1ms/step - loss: 0.5907 - accuracy:
0.6856
```

[33]:
```
[0.590725839138031, 0.6855600476264954]
```

[34]:
```python
model = keras.Sequential()
model.add(keras.layers.Dense(2, activation='relu', input_dim=len(X_test.
 ↪columns)))
# model.add(keras.layers.Dense(2, activation='sigmoid'))
model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
fits = model.fit(X_train, y_train, epochs=100, validation_data=(X_test,␣
 ↪y_test), verbose=0)
plt.plot(fits.history['accuracy'])
plt.plot(fits.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

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

# Evaluate
y_true = keras.utils.to_categorical(y1, 2)
model.evaluate(test_mm, y_true)
```

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_14 (Dense)             (None, 2)                 52
=================================================================
Total params: 52
Trainable params: 52
Non-trainable params: 0
_________________________________________________________________
```


model accuracy
train
test
0.50275
0.50250
0.50225
0.50200
0.50175
0.50150
0.50125
0.50100
accuracy
0
20
40
60
80
100
epoch


model loss
train
test
1.0
0.8
0.6
0.4
0.2
0.0
loss
0.0
0.2
0.4
0.6
0.8
1.0
epoch

```
93/93 [==============================] - 0s 936us/step - loss: nan - accuracy:
0.5010
```

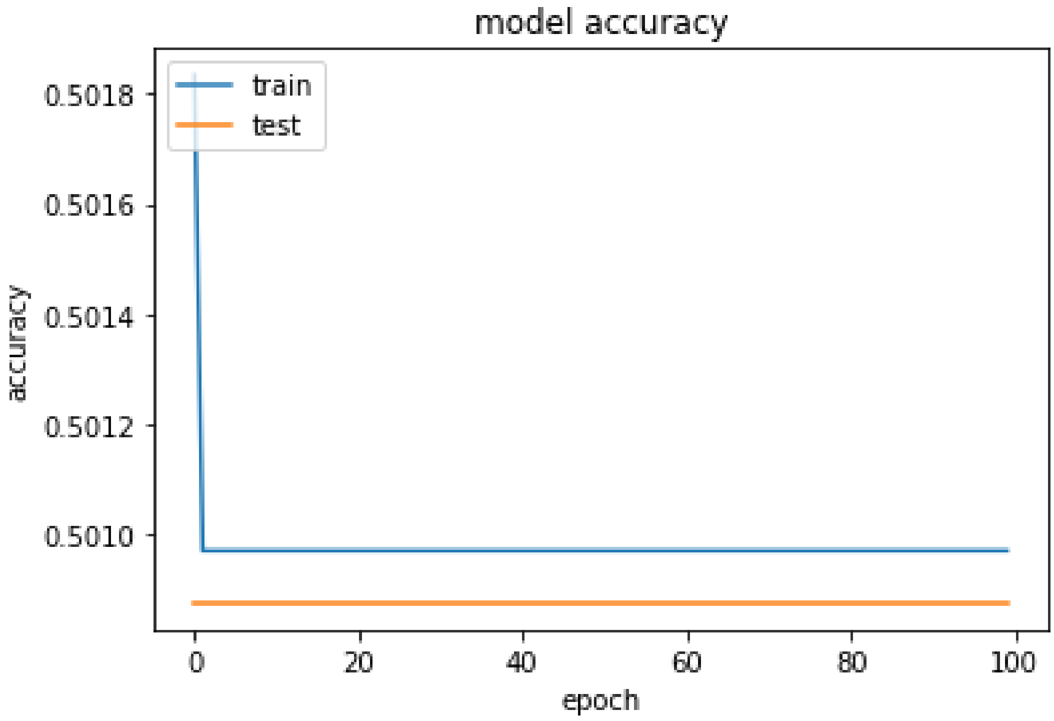
[34]:

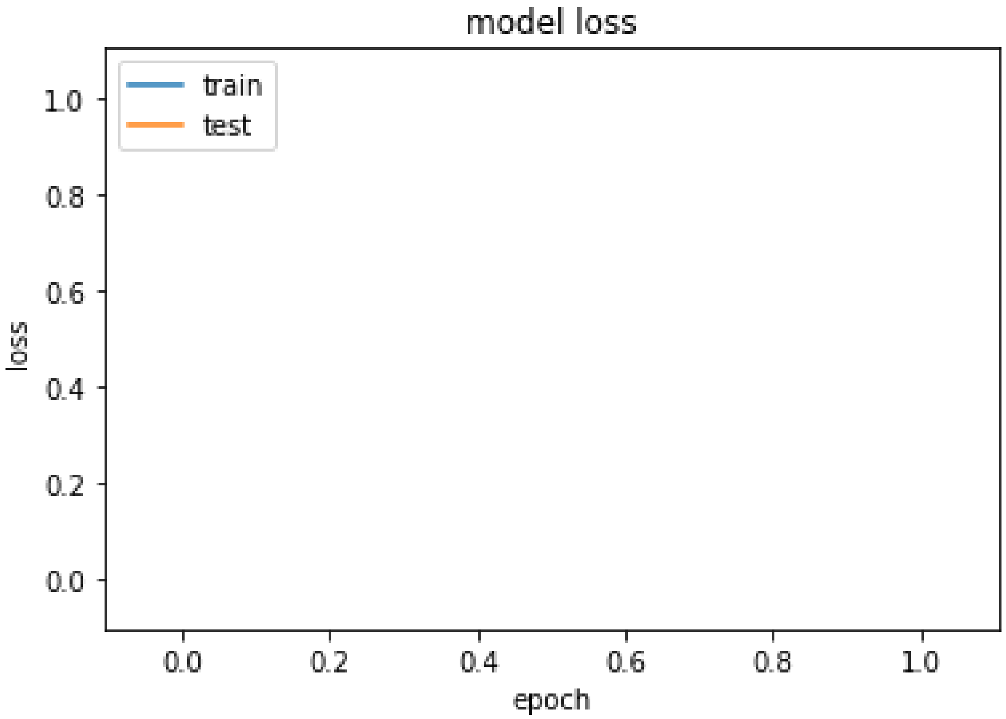
```
[nan, 0.5010121464729309]
```

[35]:

```python
model = keras.Sequential()
model.add(keras.layers.Dense(2, activation='softmax', input_dim=len(X_test.
 ↪columns)))
model.add(keras.layers.Dense(2, activation='relu'))
model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
fits = model.fit(X_train, y_train, epochs=100, validation_data=(X_test,␣
 ↪y_test), verbose=0)
plt.plot(fits.history['accuracy'])
plt.plot(fits.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

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

# Evaluate
y_true = keras.utils.to_categorical(y1, 2)
model.evaluate(test_mm, y_true)
```

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_15 (Dense)             (None, 2)                 52
_________________________________________________________________
dense_16 (Dense)             (None, 2)                 6
=================================================================
Total params: 58
Trainable params: 58
Non-trainable params: 0
_________________________________________________________________
```


model accuracy
train
test
accuracy
0.5018
0.5016
0.5014
0.5012
0.5010
epoch
0
20
40
60
80
100


model loss
train
test
loss
1.0
0.8
0.6
0.4
0.2
0.0
epoch
0.0
0.2
0.4
0.6
0.8
1.0

```
93/93 [==============================] - 0s 1ms/step - loss: nan - accuracy:
0.5010
```

```
[35]: [nan, 0.5010121464729309]
```

1.4.1 Conclusion

The best configuration seems to be 1 dense layer with activation function `softmax` for Scaled data where as `relu` performed better with unscaled data.

```
[ ]:
```