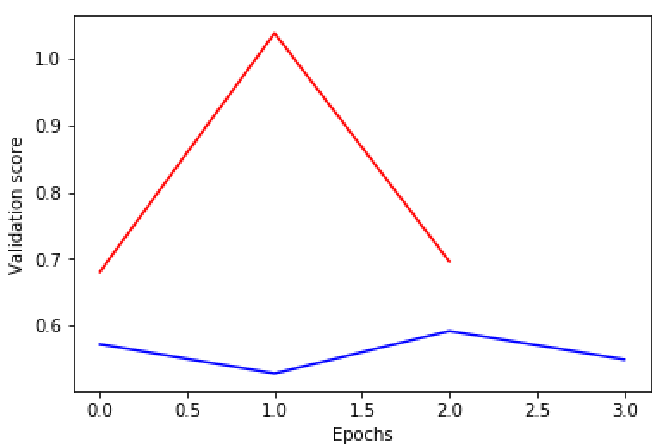
In [12]:

```python
# Import necessary modules
from keras.layers import Dense
from keras.models import Sequential
from keras.utils import to_categorical
import pandas as pd


pred_data = pd.read_csv('pred_data.csv').values

df = pd.read_csv('titanic_all_numeric.csv')

predictors = df.drop(['survived'], axis=1).values

n_cols = predictors.shape[1]

# Convert the target to categorical: target
target = to_categorical(df.survived)

# Import the SGD optimizer
from keras.optimizers import SGD

input_shape = (n_cols,)

def get_new_model(input_shape = input_shape):
  model = Sequential()
  model.add(Dense(100, activation='relu', input_shape = input_shape))
  model.add(Dense(100, activation='relu'))
  model.add(Dense(2, activation='softmax'))
  return(model)


# Create list of learning rates: lr_to_test
lr_to_test = [0.000001, 0.01, 1]

# Loop over learning rates
for lr in lr_to_test:
  print('\n\nTesting model with learning rate: %f\n'%lr )

  # Build new model to test, unaffected by previous models
  model = get_new_model()

  # Create SGD optimizer with specified learning rate: my_optimizer
  my_optimizer = SGD(lr=lr)

  # Compile the model
  model.compile(optimizer=my_optimizer, loss='categorical_crossentropy', metrics=['accuracy'
])

  # Fit the model
  model.fit(predictors, target, validation_split=0.3, epochs=10)
```

```
Testing model with learning rate: 0.000001

Train on 623 samples, validate on 268 samples
Epoch 1/10
623/623 [==============================] - 1s 2ms/step - loss: 2.5209 - acc: 0.3949 - val_los
s: 2.4016 - val_acc: 0.3582
Epoch 2/10
623/623 [==============================] - 0s 61us/step - loss: 2.4932 - acc: 0.3949 - val_lo
ss: 2.3730 - val_acc: 0.3582
Epoch 3/10
623/623 [==============================] - 0s 70us/step - loss: 2.4659 - acc: 0.3949 - val_lo
ss: 2.3449 - val_acc: 0.3582
Epoch 4/10
623/623 [==============================] - 0s 62us/step - loss: 2.4391 - acc: 0.3949 - val_lo
ss: 2.3168 - val_acc: 0.3582
Epoch 5/10
623/623 [==============================] - 0s 69us/step - loss: 2.4123 - acc: 0.3949 - val_lo
ss: 2.2891 - val_acc: 0.3582
Epoch 6/10
623/623 [==============================] - 0s 77us/step - loss: 2.3859 - acc: 0.3949 - val_lo
ss: 2.2613 - val_acc: 0.3582
Epoch 7/10
623/623 [==============================] - 0s 66us/step - loss: 2.3594 - acc: 0.3949 - val_lo
ss: 2.2342 - val_acc: 0.3582
Epoch 8/10
623/623 [==============================] - 0s 62us/step - loss: 2.3333 - acc: 0.3949 - val_lo
ss: 2.2073 - val_acc: 0.3582
Epoch 9/10
623/623 [==============================] - 0s 59us/step - loss: 2.3075 - acc: 0.3949 - val_lo
ss: 2.1807 - val_acc: 0.3582
Epoch 10/10
623/623 [==============================] - 0s 66us/step - loss: 2.2819 - acc: 0.3949 - val_lo
ss: 2.1542 - val_acc: 0.3582


Testing model with learning rate: 0.010000

Train on 623 samples, validate on 268 samples
Epoch 1/10
623/623 [==============================] - 1s 2ms/step - loss: 2.2539 - acc: 0.5634 - val_los
s: 0.6835 - val_acc: 0.6754
Epoch 2/10
623/623 [==============================] - 0s 59us/step - loss: 0.8159 - acc: 0.6372 - val_lo
ss: 0.5663 - val_acc: 0.7164
Epoch 3/10
623/623 [==============================] - 0s 62us/step - loss: 0.6969 - acc: 0.6453 - val_lo
ss: 0.5929 - val_acc: 0.7164
Epoch 4/10
623/623 [==============================] - 0s 66us/step - loss: 0.7203 - acc: 0.6565 - val_lo
ss: 0.6054 - val_acc: 0.6903
Epoch 5/10
623/623 [==============================] - 0s 66us/step - loss: 0.6729 - acc: 0.6388 - val_lo
ss: 0.9221 - val_acc: 0.6418
Epoch 6/10
623/623 [==============================] - 0s 72us/step - loss: 0.6558 - acc: 0.6501 - val_lo
ss: 0.7797 - val_acc: 0.6493
Epoch 7/10
623/623 [==============================] - 0s 64us/step - loss: 0.6421 - acc: 0.6677 - val_lo
ss: 0.5595 - val_acc: 0.7090
Epoch 8/10
623/623 [==============================] - 0s 67us/step - loss: 0.6279 - acc: 0.6726 - val_lo
ss: 0.5585 - val_acc: 0.7239
Epoch 9/10
623/623 [==============================] - 0s 70us/step - loss: 0.6201 - acc: 0.6790 - val_lo
ss: 0.5771 - val_acc: 0.7015
```

```
Epoch 10/10
623/623 [==============================] - 0s 70us/step - loss: 0.6158 - acc: 0.6838 - val_lo
ss: 0.5829 - val_acc: 0.7052


Testing model with learning rate: 1.000000

Train on 623 samples, validate on 268 samples
Epoch 1/10
623/623 [==============================] - 1s 2ms/step - loss: 6.2358 - acc: 0.6035 - val_los
s: 5.7736 - val_acc: 0.6418
Epoch 2/10
623/623 [==============================] - 0s 66us/step - loss: 6.3644 - acc: 0.6051 - val_lo
ss: 5.7736 - val_acc: 0.6418
Epoch 3/10
623/623 [==============================] - 0s 61us/step - loss: 6.3644 - acc: 0.6051 - val_lo
ss: 5.7736 - val_acc: 0.6418
Epoch 4/10
623/623 [==============================] - 0s 64us/step - loss: 6.3644 - acc: 0.6051 - val_lo
ss: 5.7736 - val_acc: 0.6418
Epoch 5/10
623/623 [==============================] - 0s 69us/step - loss: 6.3644 - acc: 0.6051 - val_lo
ss: 5.7736 - val_acc: 0.6418
Epoch 6/10
623/623 [==============================] - 0s 59us/step - loss: 6.3644 - acc: 0.6051 - val_lo
ss: 5.7736 - val_acc: 0.6418
Epoch 7/10
623/623 [==============================] - 0s 66us/step - loss: 6.3644 - acc: 0.6051 - val_lo
ss: 5.7736 - val_acc: 0.6418
Epoch 8/10
623/623 [==============================] - 0s 59us/step - loss: 6.3644 - acc: 0.6051 - val_lo
ss: 5.7736 - val_acc: 0.6418
Epoch 9/10
623/623 [==============================] - 0s 62us/step - loss: 6.3644 - acc: 0.6051 - val_lo
ss: 5.7736 - val_acc: 0.6418
Epoch 10/10
623/623 [==============================] - 0s 62us/step - loss: 6.3644 - acc: 0.6051 - val_lo
ss: 5.7736 - val_acc: 0.6418
```

In [13]:
```
# Import necessary modules
import matplotlib.pyplot as plt

# Import EarlyStopping
from keras.callbacks import EarlyStopping

# Specify the model
model_1 = Sequential()
model_1.add(Dense(100, activation='relu', input_shape = input_shape))
model_1.add(Dense(100, activation='relu'))
model_1.add(Dense(2, activation='softmax'))

# Compile the model
model_1.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Define early_stopping_monitor
early_stopping_monitor = EarlyStopping(patience=2)

# Create the new model: model_2
model_2 = Sequential()

# Add the first and second layers
model_2.add(Dense(100, activation='relu', input_shape=input_shape))
model_2.add(Dense(100, activation='relu'))

# Add the output layer
model_2.add(Dense(2, activation='softmax'))

# Compile model_2
model_2.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Fit model_1
model_1_training = model_1.fit(predictors, target, epochs=15, validation_split=0.2, callbacks
=[early_stopping_monitor], verbose=False)

# Fit model_2
model_2_training = model_2.fit(predictors, target, epochs=15, validation_split=0.2, callbacks
=[early_stopping_monitor], verbose=False)

# Create the plot
plt.plot(model_1_training.history['val_loss'], 'r', model_2_training.history['val_loss'], 'b'
)
plt.xlabel('Epochs')
plt.ylabel('Validation score')
plt.show()
```



In [14]:

```
# Define early_stopping_monitor
early_stopping_monitor = EarlyStopping(patience=2)


# Specify the model
model_1 = Sequential()
model_1.add(Dense(100, activation='relu', input_shape = input_shape))
model_1.add(Dense(100, activation='relu'))
model_1.add(Dense(2, activation='softmax'))

# Compile the model
model_1.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# The input shape to use in the first hidden layer
input_shape = (n_cols,)

# Create the new model: model_2
model_2 = Sequential()

# Add the first, second, and third hidden layers
model_2.add(Dense(50, activation='relu', input_shape=input_shape))
model_2.add(Dense(50, activation='relu'))
model_2.add(Dense(50, activation='relu'))

# Add the output layer
model_2.add(Dense(2, activation='softmax'))

# Compile model_2
model_2.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Fit model 1
model_1_training = model_1.fit(predictors, target, epochs=20, validation_split=0.4, callbacks
=[early_stopping_monitor], verbose=False)

# Fit model 2
model_2_training = model_2.fit(predictors, target, epochs=20, validation_split=0.4, callbacks
=[early_stopping_monitor], verbose=False)

# Create the plot
plt.plot(model_1_training.history['val_loss'], 'r', model_2_training.history['val_loss'], 'b'
)
plt.xlabel('Epochs')
plt.ylabel('Validation score')
plt.show()
```