PDL Lab5. Text corpus creation and binary classification using DNN

HARISHMITHA SOUNDRANAYAKI
225229113

In [5]:

```
import pandas as pd
import numpy as np
```

In [53]:

```
df= pd.read_csv('newquotes.csv', encoding='latin1')
```

In [54]:

```
df.head()
```

Out[54]:

	Label	Quotes
0	0	Always borrow money from a pessimist. He won t...
1	1	When you change your thoughts, remember to als...
2	0	They say money talks. But, all mine says is go...
3	1	We cannot solve problems with the kind of thin...
4	0	Every day is Friday when you re unemployed.

In [55]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40 entries, 0 to 39
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Label   40 non-null     int64 
 1   Quotes  40 non-null     object
dtypes: int64(1), object(1)
memory usage: 768.0+ bytes
```

In [56]:

```
df.shape
```

Out[56]:

```
(40, 2)
```

In [60]:

```
df['Label'].value_counts()
```

Out[60]:

```
Label
0    20
1    20
Name: count, dtype: int64
```

In [13]:

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Error loading punkt: <urlopen error [Errno 2] No such file
[nltk_data]     or directory>
```

Out[13]:

```
False
```

In [14]:

```
nltk.download('stopwords')
```

```
[nltk_data] Error loading stopwords: <urlopen error [Errno 2] No such
[nltk_data]     file or directory>
```

Out[14]:

```
False
```

In [15]:

```
from nltk.corpus import stopwords
stop_words = stopwords.words("english")
```

In [16]:

```
import regex as re
```

In [18]:

```
df['Quotes'] = df['Quotes'].apply(lambda x: ' '.join([word for word in x.split() if word
```

In [58]:

```
df.head()
```

Out[58]:

	Label	Quotes
0	0	Always borrow money from a pessimist. He won t...
1	1	When you change your thoughts, remember to als...
2	0	They say money talks. But, all mine says is go...
3	1	We cannot solve problems with the kind of thin...
4	0	Every day is Friday when you re unemployed.

In [20]:

```
import string
df['Quotes']=df['Quotes'].str.replace('[{}]'.format(string.punctuation), '')
```

In [59]:

```
df.head()
```

Out[59]:

	Label	Quotes
0	0	Always borrow money from a pessimist. He won t...
1	1	When you change your thoughts, remember to als...
2	0	They say money talks. But, all mine says is go...
3	1	We cannot solve problems with the kind of thin...
4	0	Every day is Friday when you re unemployed.

In [22]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [23]:

```
v = TfidfVectorizer()
x = v.fit_transform(df['Quotes']).toarray()
x = x.astype(float)
```

In [24]:

```
x.dtype
```

Out[24]:

```
dtype('float64')
```

In [25]:

```
x.shape
```

Out[25]:

```
(40, 204)
```

In [26]:

```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam
```

In [27]:

```
y = df['Labels']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=42
```

In [28]:

```
model = Sequential()
model.add(Dense(32, input_dim=204, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['
```

In [29]:

```
model.fit(x_train, y_train, validation_split=0.2, epochs=100, batch_size=10, verbose=1)
```

```
Epoch 1/100
3/3 [==============================] - 1s 174ms/step - loss: 0.6973 - ac
curacy: 0.5417 - val_loss: 0.6697 - val_accuracy: 0.8333
Epoch 2/100
3/3 [==============================] - 0s 39ms/step - loss: 0.6867 - acc
uracy: 0.5417 - val_loss: 0.6714 - val_accuracy: 0.8333
Epoch 3/100
3/3 [==============================] - 0s 36ms/step - loss: 0.6777 - acc
uracy: 0.5417 - val_loss: 0.6729 - val_accuracy: 0.8333
Epoch 4/100
3/3 [==============================] - 0s 32ms/step - loss: 0.6691 - acc
uracy: 0.6667 - val_loss: 0.6741 - val_accuracy: 0.8333
Epoch 5/100
3/3 [==============================] - 0s 30ms/step - loss: 0.6611 - acc
uracy: 0.7917 - val_loss: 0.6756 - val_accuracy: 0.8333
Epoch 6/100
3/3 [==============================] - 0s 31ms/step - loss: 0.6533 - acc
uracy: 0.8333 - val_loss: 0.6771 - val_accuracy: 0.8333
Epoch 7/100
```

In [31]:

```
import matplotlib.pyplot as plt
```

In [32]:

```
def accuracy_plotting(model):
  plt.plot(model.history['accuracy'])
  plt.plot(model.history['val_accuracy'])
  plt.title('Model Accuracy')
  plt.ylabel('accuracy')
  plt.xlabel('epoch')
  plt.legend(['Train', 'Validation'])
  plt.show()
```

In [34]:

```
def Loss_plotting(model):
  plt.plot(model.history['loss'])
  plt.plot(model.history['val_loss'])
  plt.title('Model Loss')
  plt.ylabel('loss')
  plt.xlabel('epoch')
  plt.legend(['Train', 'Validation'])
  plt.show()
```

Model improvement 1

In [35]:

```
def model_improvement1(nodes):
  model = Sequential()
  model.add(Dense(nodes, input_dim=204, activation='relu'))
  model.add(Dense(1, activation='sigmoid'))
  # Compile the model
  model.compile(loss='binary_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=
  return model
```

In [36]:

```
model1 = model_improvement1(8)
```

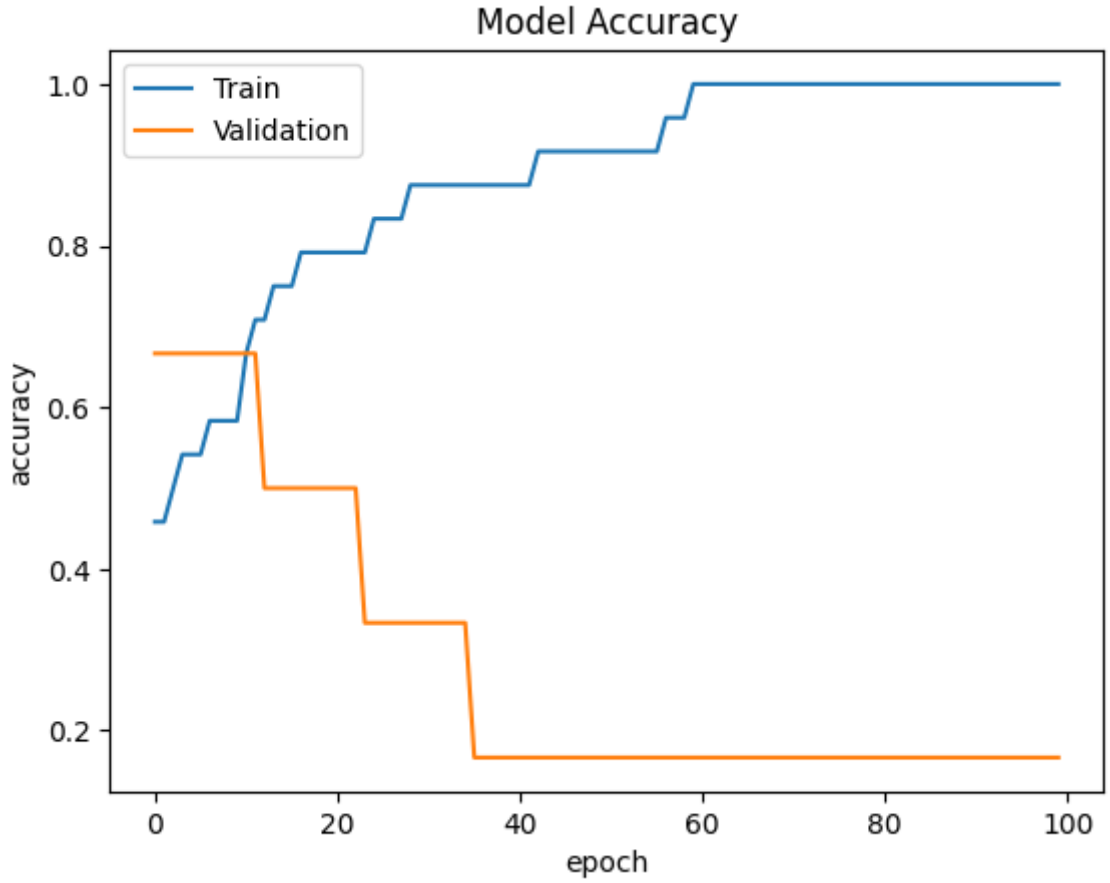
In [37]:

```
history1 = model1.fit(x_train, y_train, validation_split=0.2, epochs=100, batch_size=10,
```

```
Epoch 1/100
3/3 [==============================] - 1s 114ms/step - loss: 0.7066 - ac
curacy: 0.4583 - val_loss: 0.6903 - val_accuracy: 0.6667
Epoch 2/100
3/3 [==============================] - 0s 35ms/step - loss: 0.7011 - acc
uracy: 0.4583 - val_loss: 0.6907 - val_accuracy: 0.6667
Epoch 3/100
3/3 [==============================] - 0s 35ms/step - loss: 0.6966 - acc
uracy: 0.5000 - val_loss: 0.6911 - val_accuracy: 0.6667
Epoch 4/100
3/3 [==============================] - 0s 31ms/step - loss: 0.6919 - acc
uracy: 0.5417 - val_loss: 0.6915 - val_accuracy: 0.6667
Epoch 5/100
3/3 [==============================] - 0s 35ms/step - loss: 0.6872 - acc
uracy: 0.5417 - val_loss: 0.6920 - val_accuracy: 0.6667
Epoch 6/100
3/3 [==============================] - 0s 33ms/step - loss: 0.6830 - acc
uracy: 0.5417 - val_loss: 0.6925 - val_accuracy: 0.6667
Epoch 7/100
```

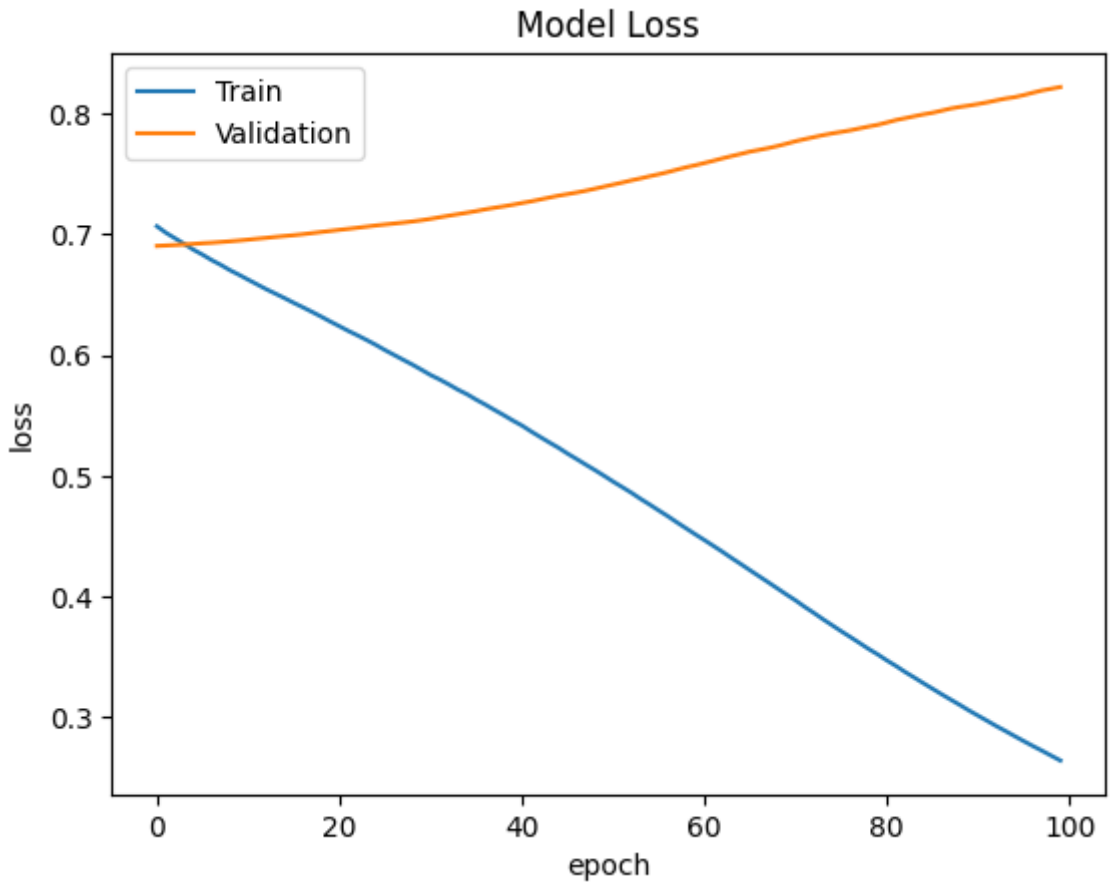
In [38]:

```
accuracy_plotting(history1)
```



In [39]:

```
Loss_plotting(history1)
```



In [40]:

```
model2 = model_improvement1(6)
```

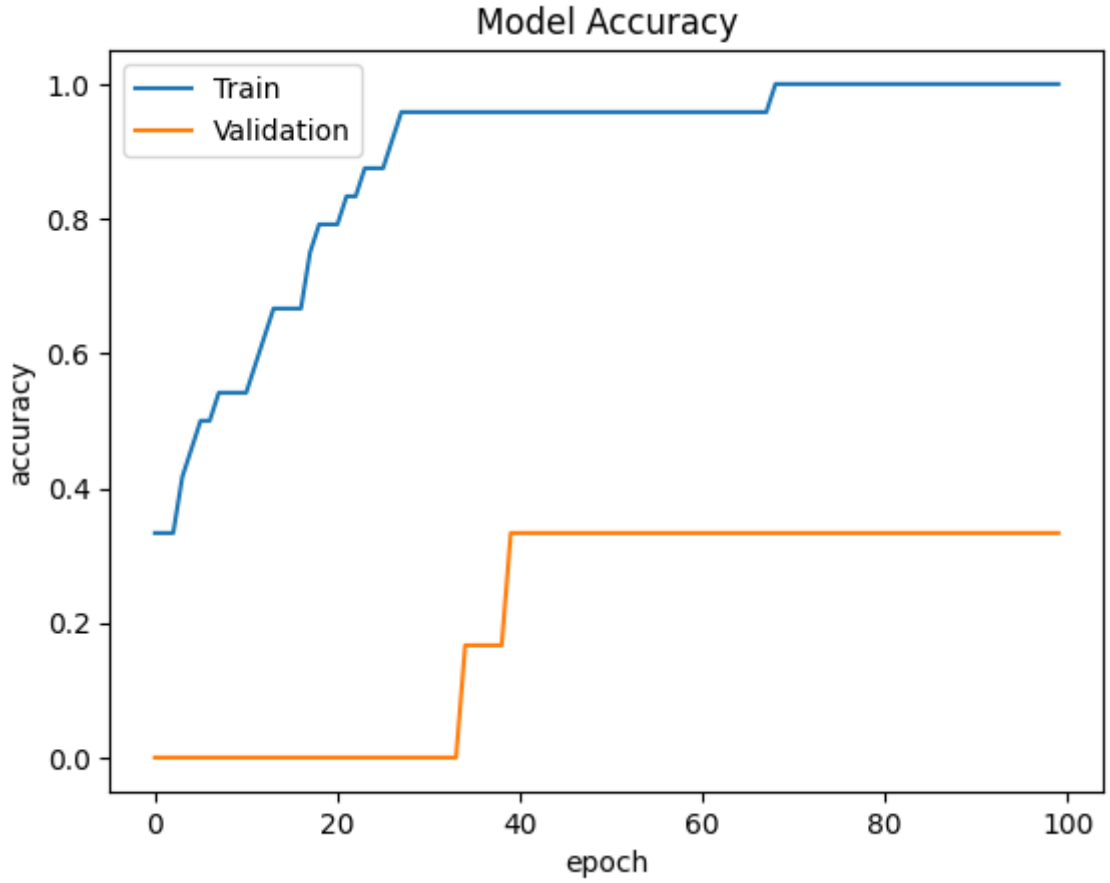
In [41]:

```
history2 = model2.fit(x_train, y_train, validation_split=0.2, epochs=100, batch_size=10,
```

```
Epoch 1/100
3/3 [==============================] - 1s 131ms/step - loss: 0.7236 - ac
curacy: 0.3333 - val_loss: 0.7437 - val_accuracy: 0.0000e+00
Epoch 2/100
3/3 [==============================] - 0s 35ms/step - loss: 0.7183 - acc
uracy: 0.3333 - val_loss: 0.7450 - val_accuracy: 0.0000e+00
Epoch 3/100
3/3 [==============================] - 0s 29ms/step - loss: 0.7132 - acc
uracy: 0.3333 - val_loss: 0.7462 - val_accuracy: 0.0000e+00
Epoch 4/100
3/3 [==============================] - 0s 32ms/step - loss: 0.7084 - acc
uracy: 0.4167 - val_loss: 0.7472 - val_accuracy: 0.0000e+00
Epoch 5/100
3/3 [==============================] - 0s 34ms/step - loss: 0.7039 - acc
uracy: 0.4583 - val_loss: 0.7481 - val_accuracy: 0.0000e+00
Epoch 6/100
3/3 [==============================] - 0s 29ms/step - loss: 0.7001 - acc
uracy: 0.5000 - val_loss: 0.7492 - val_accuracy: 0.0000e+00
Epoch 7/100
```

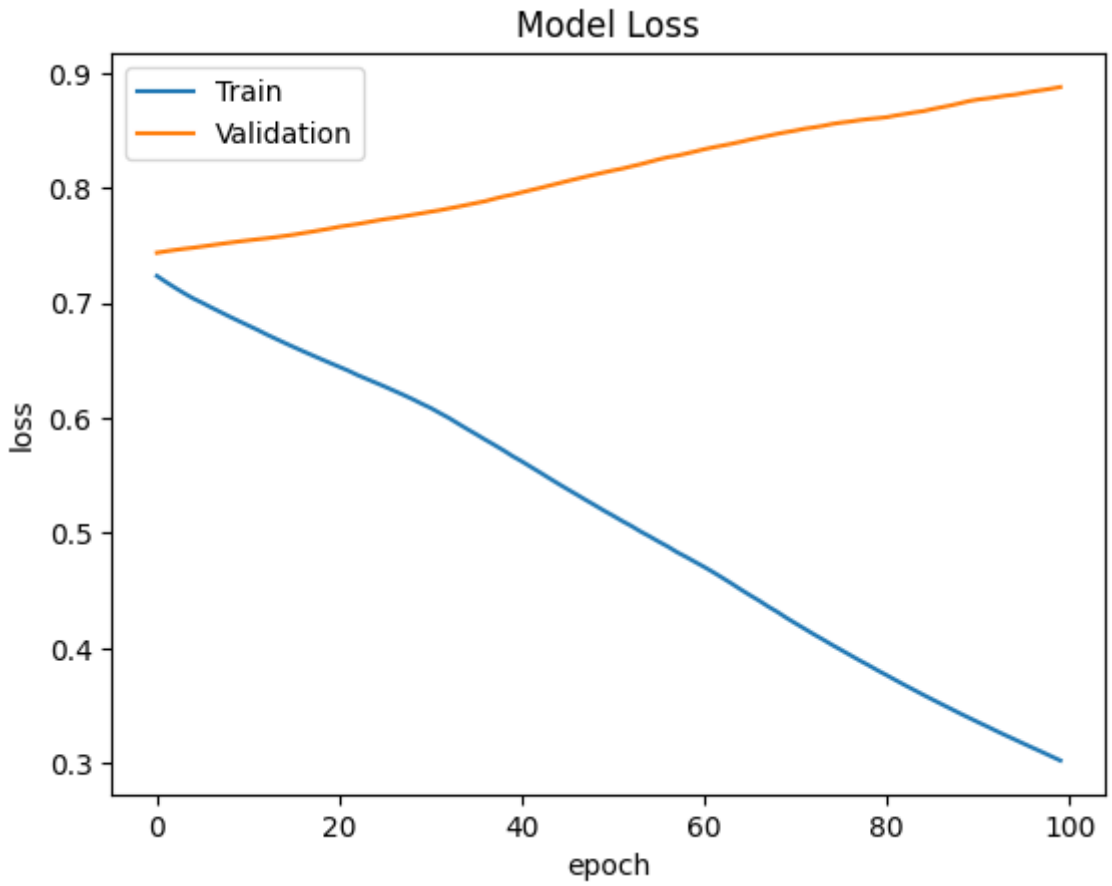
In [42]:

```
accuracy_plotting(history2)
```



In [43]:

```
Loss_plotting(history2)
```



In [44]:

```
model3 = model_improvement1(32)
```

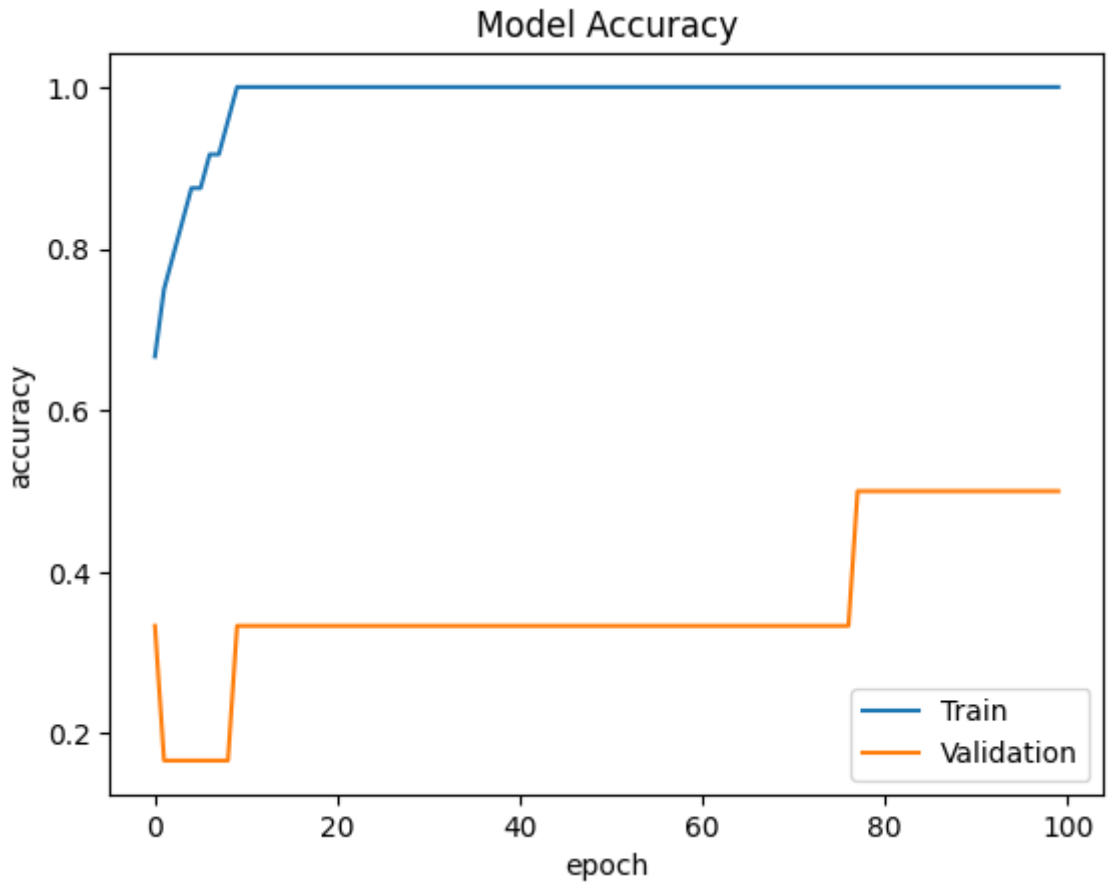
In [45]:

```
history3 = model3.fit(x_train, y_train, validation_split=0.2, epochs=100, batch_size=10,
```

```
Epoch 1/100
3/3 [==============================] - 1s 118ms/step - loss: 0.6849 - ac
curacy: 0.6667 - val_loss: 0.7178 - val_accuracy: 0.3333
Epoch 2/100
3/3 [==============================] - 0s 35ms/step - loss: 0.6755 - acc
uracy: 0.7500 - val_loss: 0.7193 - val_accuracy: 0.1667
Epoch 3/100
3/3 [==============================] - 0s 29ms/step - loss: 0.6673 - acc
uracy: 0.7917 - val_loss: 0.7209 - val_accuracy: 0.1667
Epoch 4/100
3/3 [==============================] - 0s 33ms/step - loss: 0.6598 - acc
uracy: 0.8333 - val_loss: 0.7227 - val_accuracy: 0.1667
Epoch 5/100
3/3 [==============================] - 0s 35ms/step - loss: 0.6513 - acc
uracy: 0.8750 - val_loss: 0.7242 - val_accuracy: 0.1667
Epoch 6/100
3/3 [==============================] - 0s 29ms/step - loss: 0.6441 - acc
uracy: 0.8750 - val_loss: 0.7258 - val_accuracy: 0.1667
Epoch 7/100
```

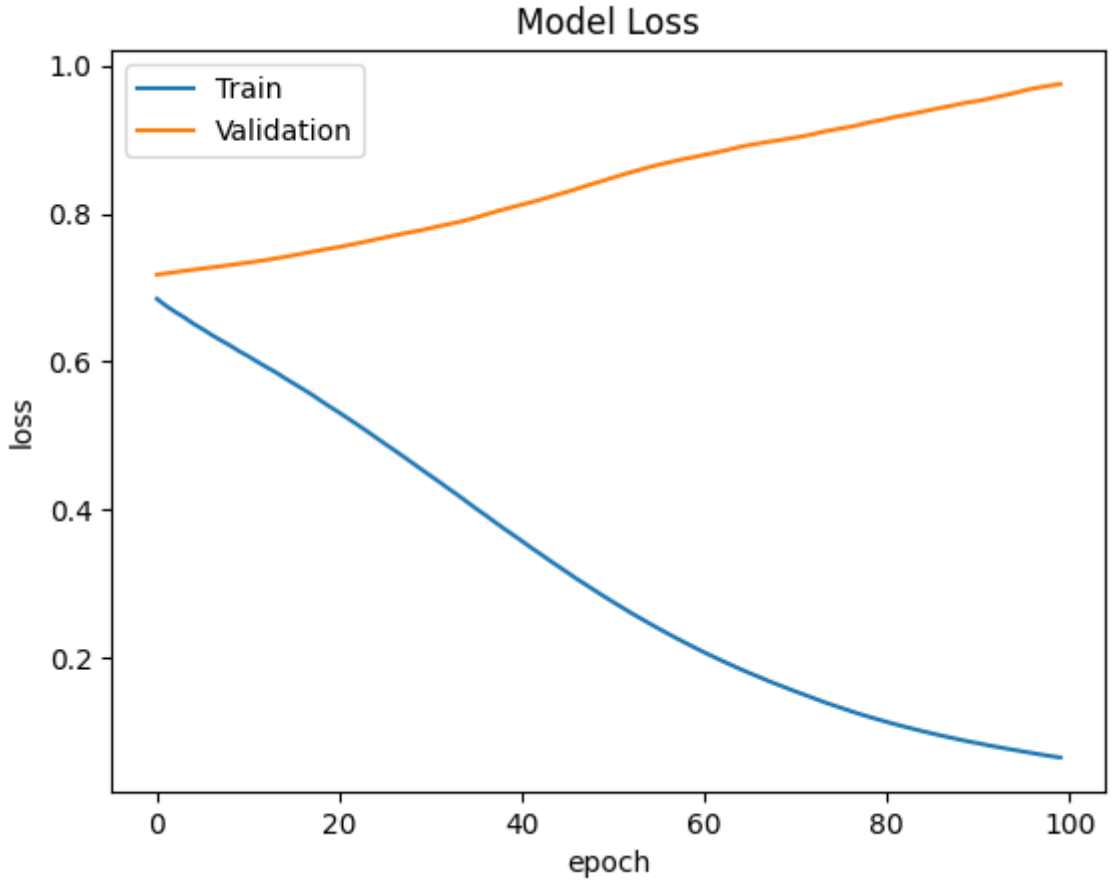
In [46]:

```
accuracy_plotting(history3)
```



In [47]:

```
Loss_plotting(history3)
```



In [48]:

```
model4 = model_improvement1(64)
```

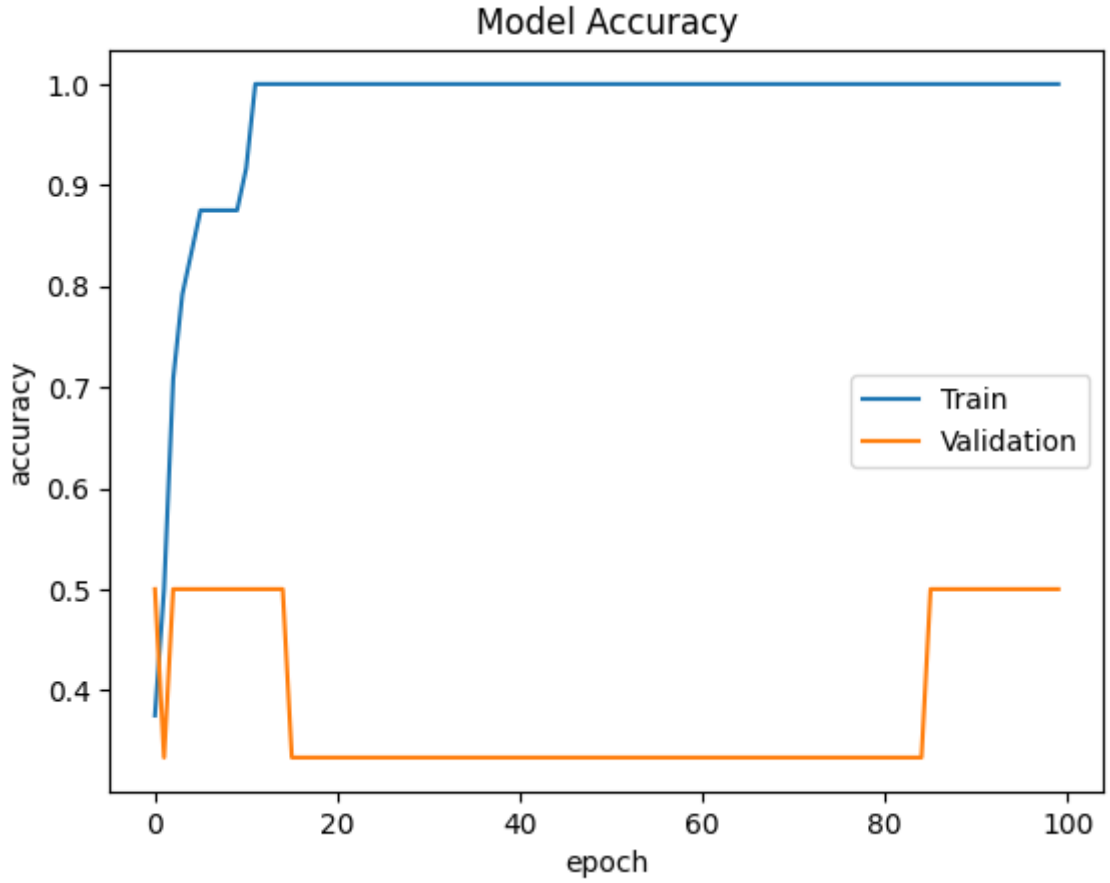
In [49]:

```
history4 = model4.fit(x_train, y_train, validation_split=0.2, epochs=100, batch_size=10,
```

```
Epoch 1/100
3/3 [==============================] - 1s 126ms/step - loss: 0.7017 - ac
curacy: 0.3750 - val_loss: 0.6836 - val_accuracy: 0.5000
Epoch 2/100
3/3 [==============================] - 0s 36ms/step - loss: 0.6884 - acc
uracy: 0.5000 - val_loss: 0.6856 - val_accuracy: 0.3333
Epoch 3/100
3/3 [==============================] - 0s 30ms/step - loss: 0.6772 - acc
uracy: 0.7083 - val_loss: 0.6879 - val_accuracy: 0.5000
Epoch 4/100
3/3 [==============================] - 0s 34ms/step - loss: 0.6667 - acc
uracy: 0.7917 - val_loss: 0.6900 - val_accuracy: 0.5000
Epoch 5/100
3/3 [==============================] - 0s 33ms/step - loss: 0.6570 - acc
uracy: 0.8333 - val_loss: 0.6921 - val_accuracy: 0.5000
Epoch 6/100
3/3 [==============================] - 0s 29ms/step - loss: 0.6470 - acc
uracy: 0.8750 - val_loss: 0.6943 - val_accuracy: 0.5000
Epoch 7/100
```

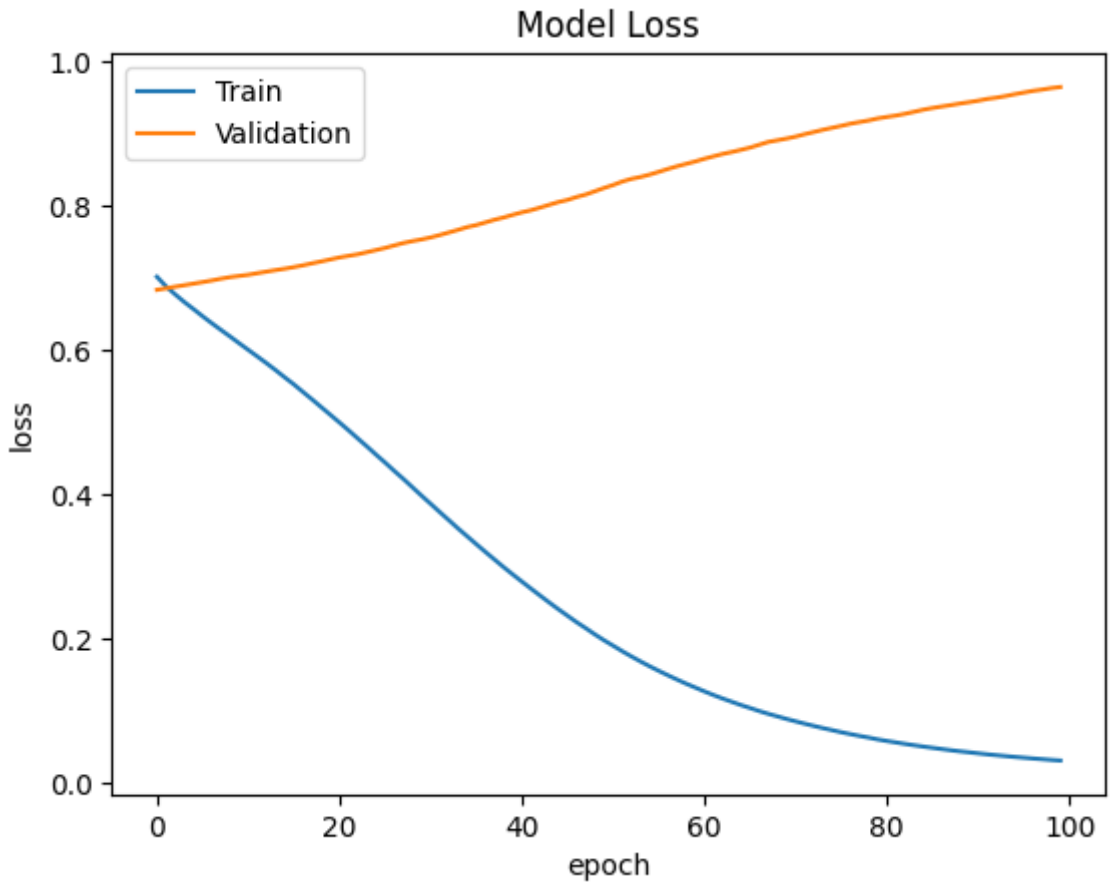
In [50]:

```
accuracy_plotting(history4)
```



In [51]:

```
Loss_plotting(history4)
```



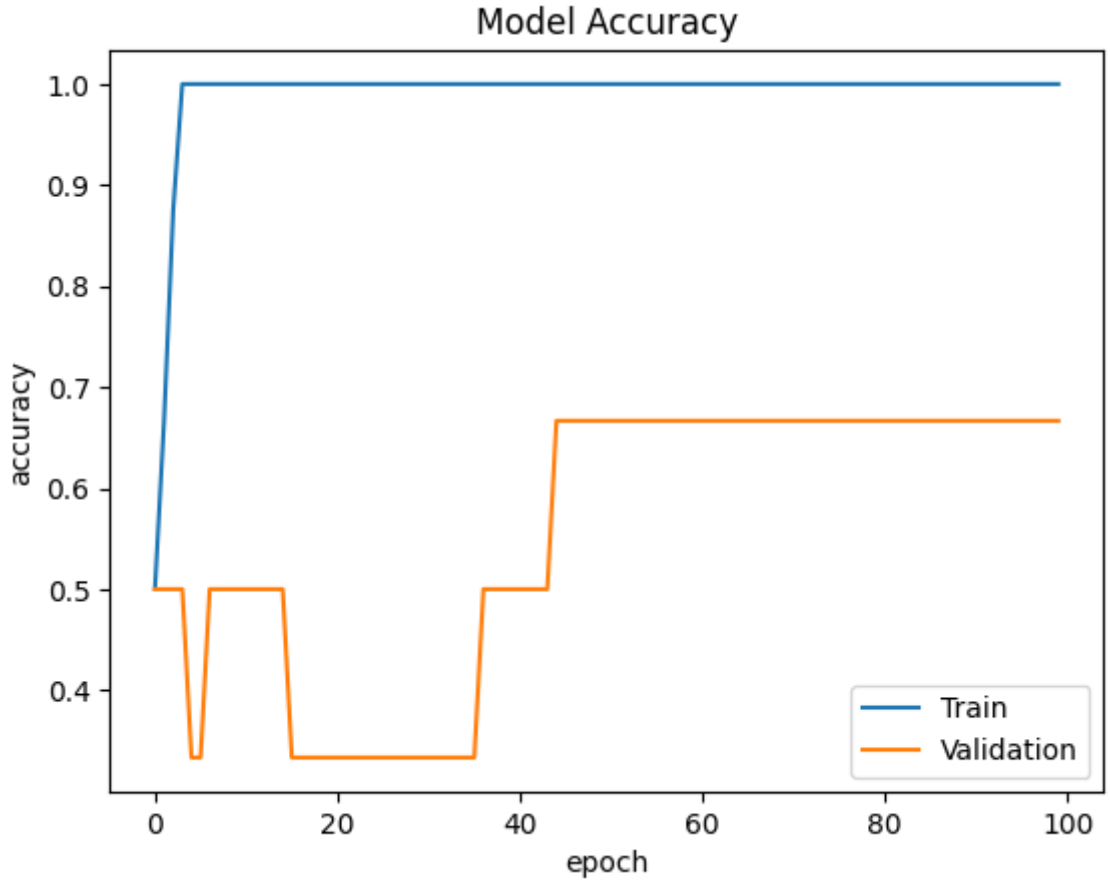
In [52]:

```
model5 = model_improvement1(128)
history5 = model5.fit(x_train, y_train, validation_split=0.2, epochs=100, batch_size=10,
```

```
Epoch 1/100
3/3 [==============================] - 1s 121ms/step - loss: 0.6988 - ac
curacy: 0.4167 - val_loss: 0.6726 - val_accuracy: 0.6667
Epoch 2/100
3/3 [==============================] - 0s 35ms/step - loss: 0.6781 - acc
uracy: 0.5833 - val_loss: 0.6750 - val_accuracy: 0.6667
Epoch 3/100
3/3 [==============================] - 0s 30ms/step - loss: 0.6622 - acc
uracy: 0.6667 - val_loss: 0.6779 - val_accuracy: 0.5000
Epoch 4/100
3/3 [==============================] - 0s 29ms/step - loss: 0.6463 - acc
uracy: 0.8750 - val_loss: 0.6809 - val_accuracy: 0.5000
Epoch 5/100
3/3 [==============================] - 0s 30ms/step - loss: 0.6310 - acc
uracy: 0.9167 - val_loss: 0.6841 - val_accuracy: 0.5000
Epoch 6/100
3/3 [==============================] - 0s 34ms/step - loss: 0.6169 - acc
uracy: 0.9167 - val_loss: 0.6874 - val_accuracy: 0.5000
Epoch 7/100
```

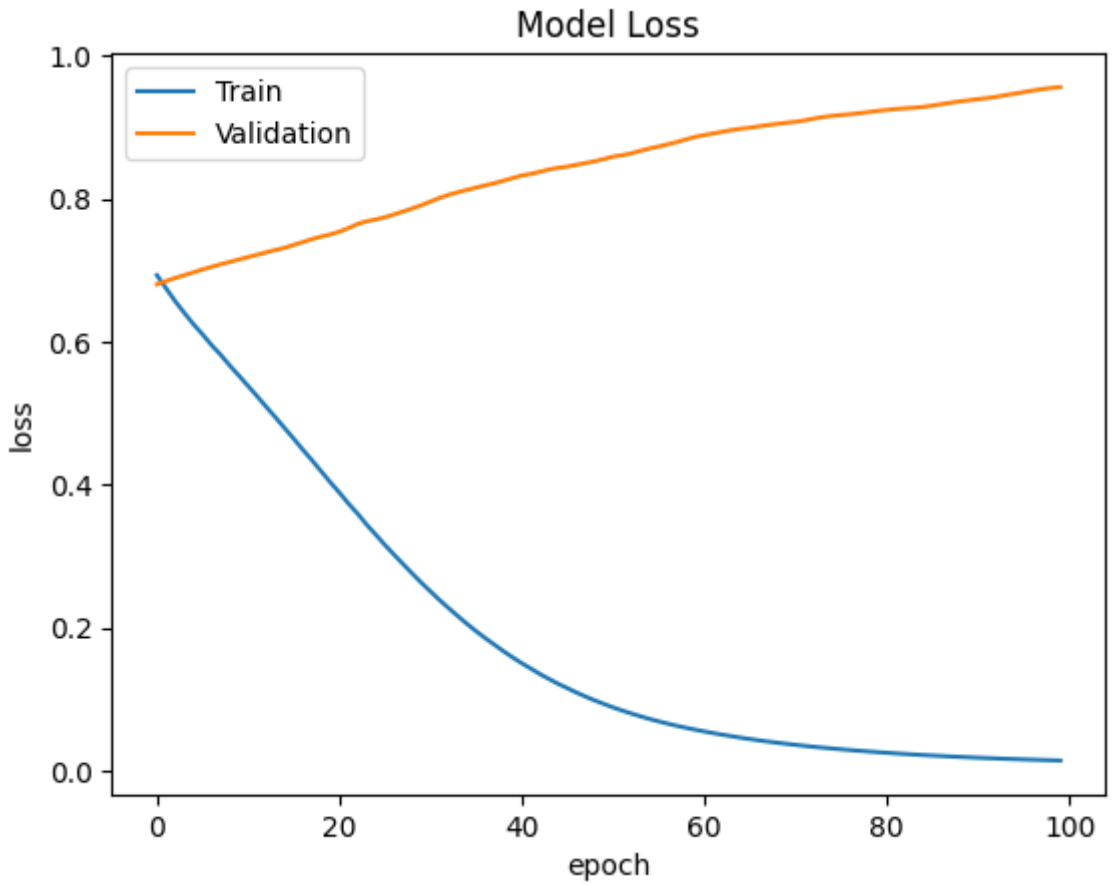
In [67]:

```
accuracy_plotting(history5)
```



In [68]:

```
Loss_plotting(history5)
```



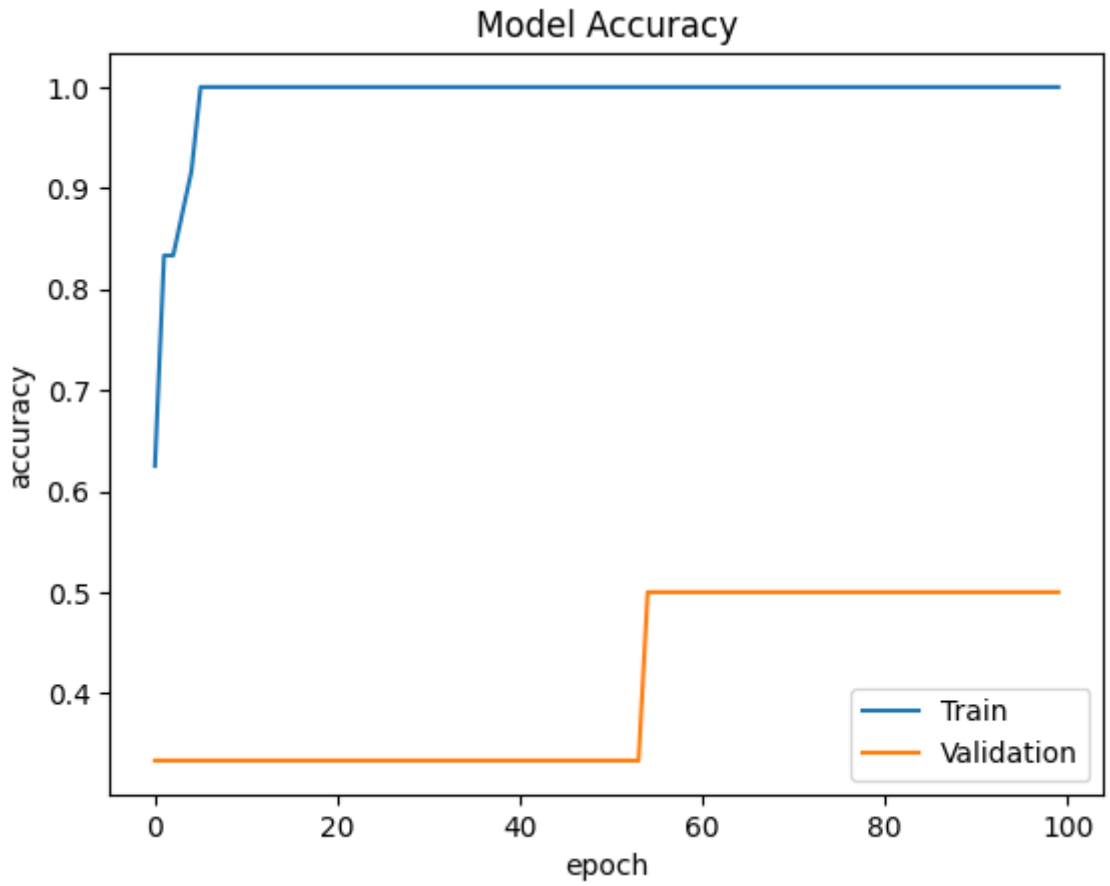
In [69]:

```
model6 = model_improvement1(256)
history6 = model6.fit(x_train, y_train, validation_split=0.2, epochs=100, batch_size=10,
```

```
Epoch 1/100
3/3 [==============================] - 1s 153ms/step - loss: 0.6918 - ac
curacy: 0.6250 - val_loss: 0.7008 - val_accuracy: 0.3333
Epoch 2/100
3/3 [==============================] - 0s 30ms/step - loss: 0.6659 - acc
uracy: 0.8333 - val_loss: 0.7060 - val_accuracy: 0.3333
Epoch 3/100
3/3 [==============================] - 0s 38ms/step - loss: 0.6424 - acc
uracy: 0.8333 - val_loss: 0.7099 - val_accuracy: 0.3333
Epoch 4/100
3/3 [==============================] - 0s 37ms/step - loss: 0.6210 - acc
uracy: 0.8750 - val_loss: 0.7144 - val_accuracy: 0.3333
Epoch 5/100
3/3 [==============================] - 0s 36ms/step - loss: 0.6006 - acc
uracy: 0.9167 - val_loss: 0.7191 - val_accuracy: 0.3333
Epoch 6/100
3/3 [==============================] - 0s 39ms/step - loss: 0.5821 - acc
uracy: 1.0000 - val_loss: 0.7245 - val_accuracy: 0.3333
Epoch 7/100
```

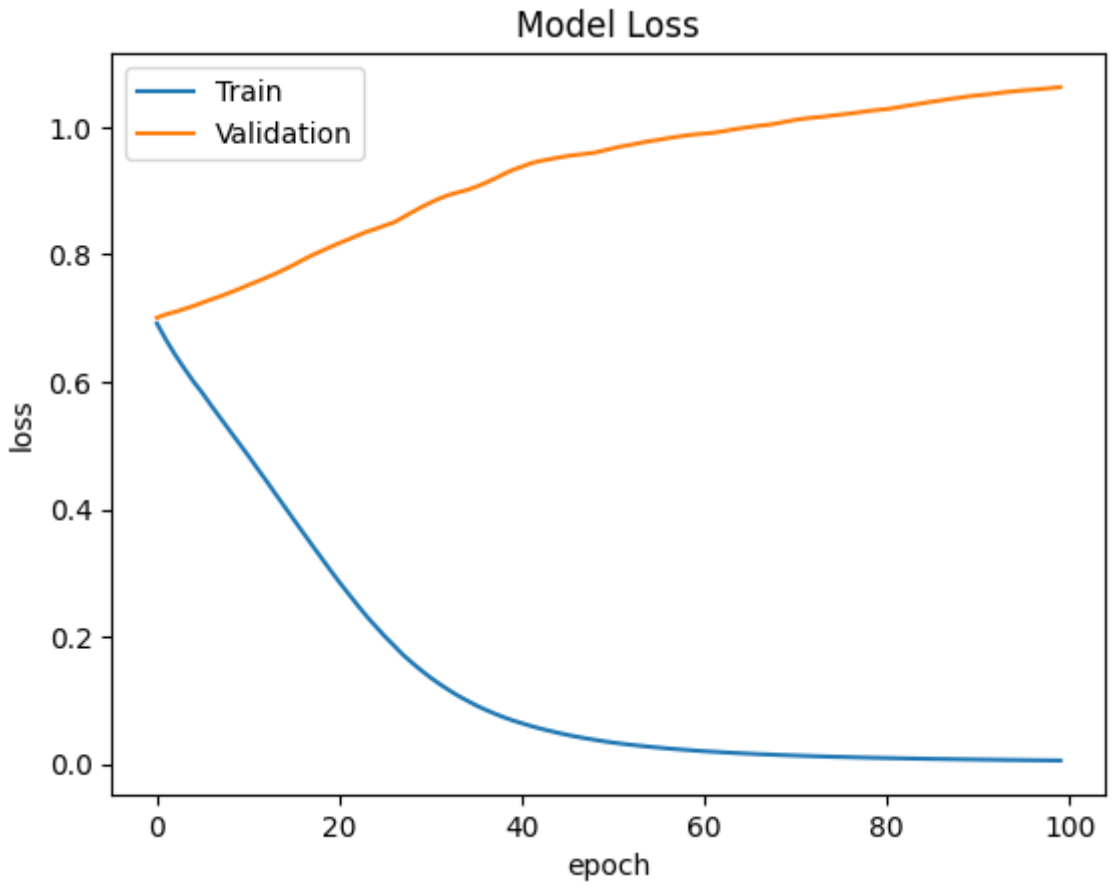
In [70]:

```
accuracy_plotting(history6)
```



In [71]:

```
Loss_plotting(history6)
```



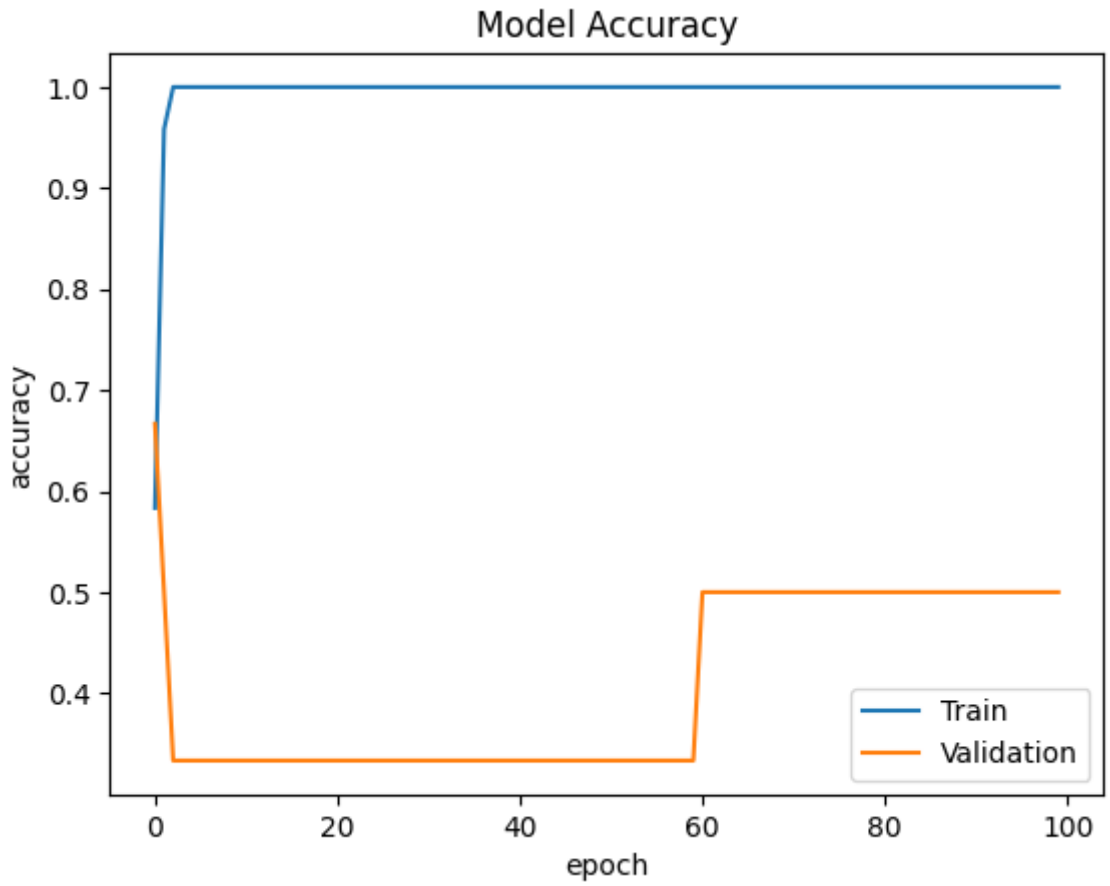
In [72]:

```
model7 = model_improvement1(512)
history7 = model7.fit(x_train, y_train, validation_split=0.2, epochs=100, batch_size=10,
```

```
Epoch 1/100
3/3 [==============================] - 1s 195ms/step - loss: 0.6929 - ac
curacy: 0.5833 - val_loss: 0.6874 - val_accuracy: 0.6667
Epoch 2/100
3/3 [==============================] - 0s 41ms/step - loss: 0.6552 - acc
uracy: 0.9583 - val_loss: 0.6917 - val_accuracy: 0.5000
Epoch 3/100
3/3 [==============================] - 0s 41ms/step - loss: 0.6251 - acc
uracy: 1.0000 - val_loss: 0.6958 - val_accuracy: 0.3333
Epoch 4/100
3/3 [==============================] - 0s 65ms/step - loss: 0.5979 - acc
uracy: 1.0000 - val_loss: 0.7013 - val_accuracy: 0.3333
Epoch 5/100
3/3 [==============================] - 0s 39ms/step - loss: 0.5701 - acc
uracy: 1.0000 - val_loss: 0.7061 - val_accuracy: 0.3333
Epoch 6/100
3/3 [==============================] - 0s 38ms/step - loss: 0.5438 - acc
uracy: 1.0000 - val_loss: 0.7121 - val_accuracy: 0.3333
Epoch 7/100
```

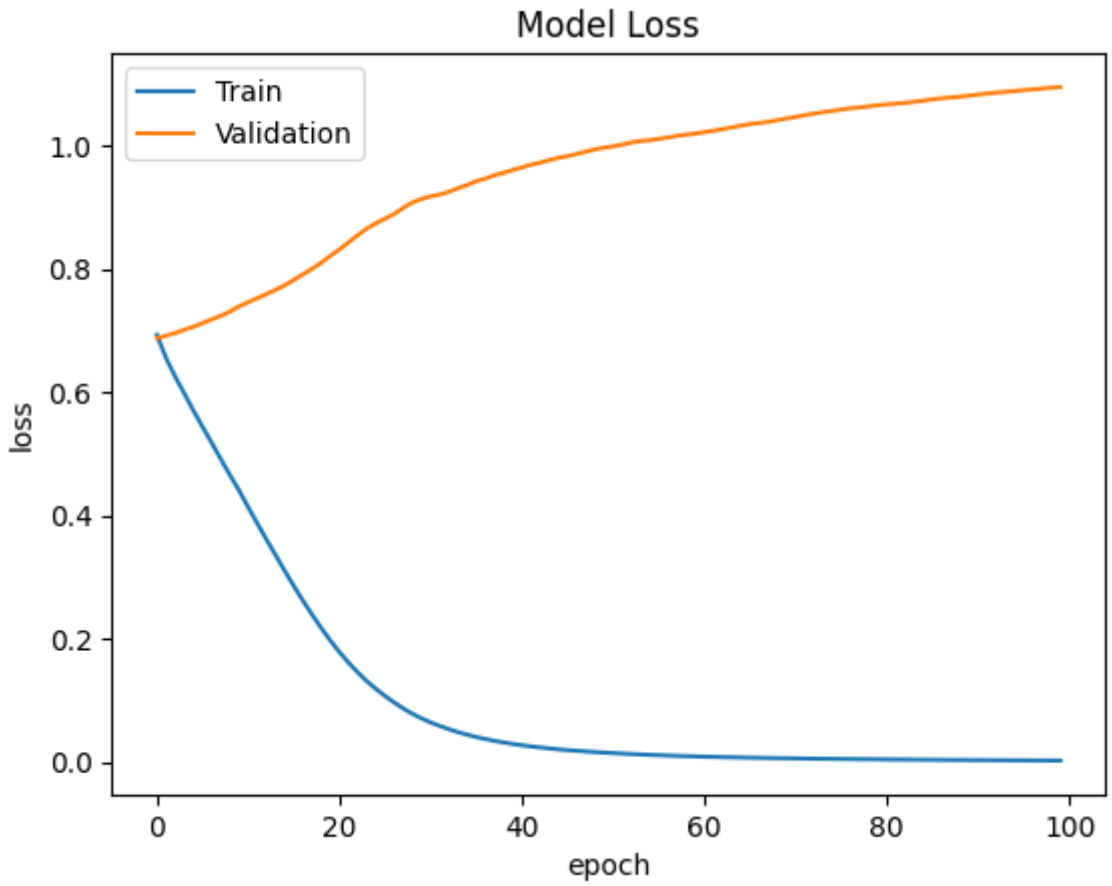
In [73]:

```
accuracy_plotting(history7)
```



In [74]:

```
Loss_plotting(history7)
```



Model Improvement 2

In [75]:

```
#2 Layers with 32 nodes
model = Sequential()
model.add(Dense(32, input_dim=204, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(loss='binary_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['
```

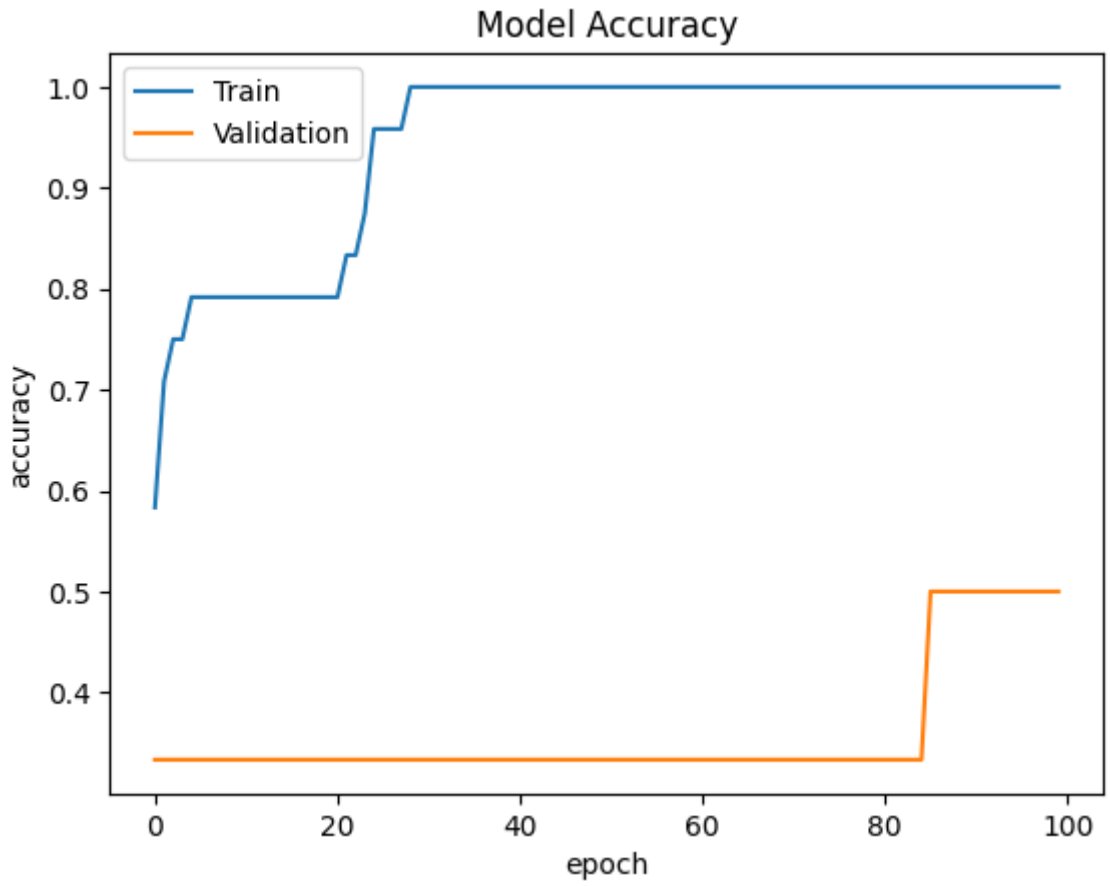
In [76]:

```
hist1 = model.fit(x_train, y_train, validation_split=0.2, epochs=100, batch_size=10, verb
```

```
Epoch 1/100
3/3 [==============================] - 1s 175ms/step - loss: 0.6850 - ac
curacy: 0.5833 - val_loss: 0.6951 - val_accuracy: 0.3333
Epoch 2/100
3/3 [==============================] - 0s 54ms/step - loss: 0.6778 - acc
uracy: 0.7083 - val_loss: 0.6962 - val_accuracy: 0.3333
Epoch 3/100
3/3 [==============================] - 0s 46ms/step - loss: 0.6719 - acc
uracy: 0.7500 - val_loss: 0.6981 - val_accuracy: 0.3333
Epoch 4/100
3/3 [==============================] - 0s 19ms/step - loss: 0.6667 - acc
uracy: 0.7500 - val_loss: 0.6999 - val_accuracy: 0.3333
Epoch 5/100
3/3 [==============================] - 0s 27ms/step - loss: 0.6609 - acc
uracy: 0.7917 - val_loss: 0.7011 - val_accuracy: 0.3333
Epoch 6/100
3/3 [==============================] - 0s 28ms/step - loss: 0.6556 - acc
uracy: 0.7917 - val_loss: 0.7031 - val_accuracy: 0.3333
Epoch 7/100
```

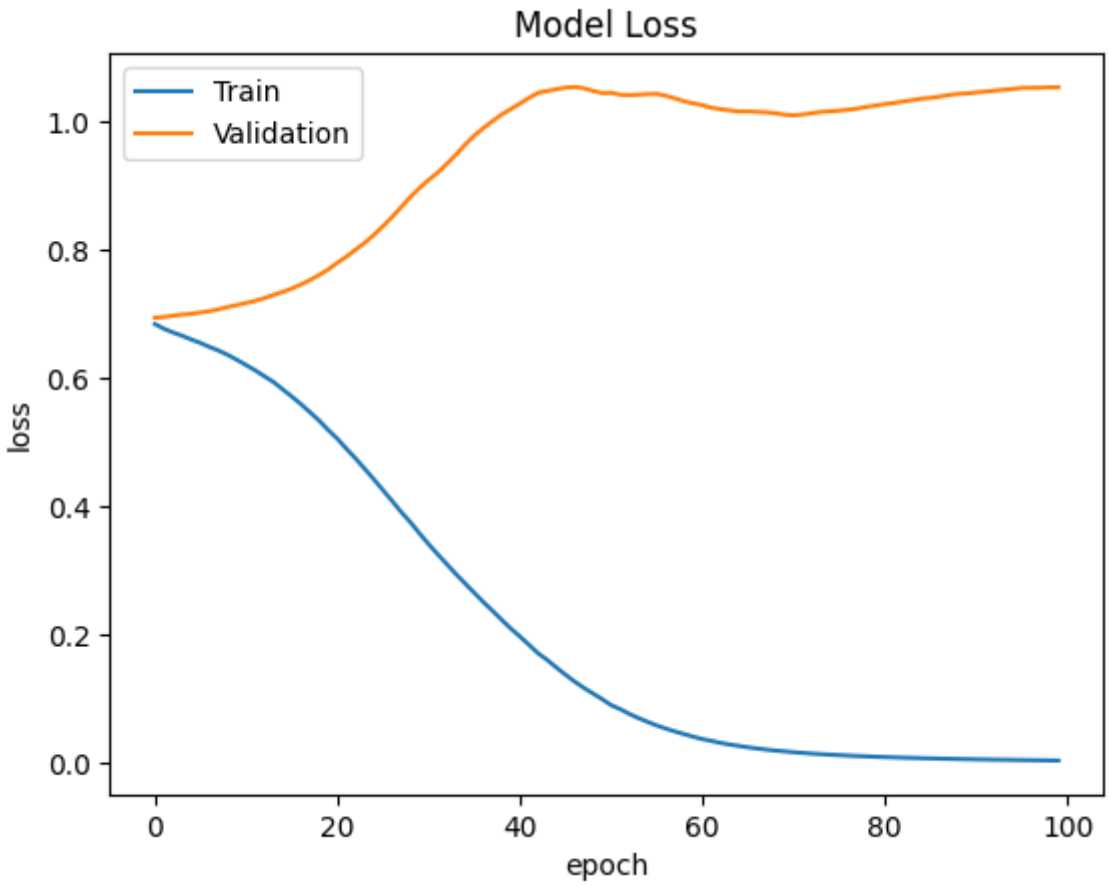
In [77]:

```
accuracy_plotting(hist1)
```



In [78]:

```
Loss_plotting(hist1)
```



In [79]:

```
#3 layers with 32 nodes
model = Sequential()
model.add(Dense(32, input_dim=204, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(loss='binary_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['
```

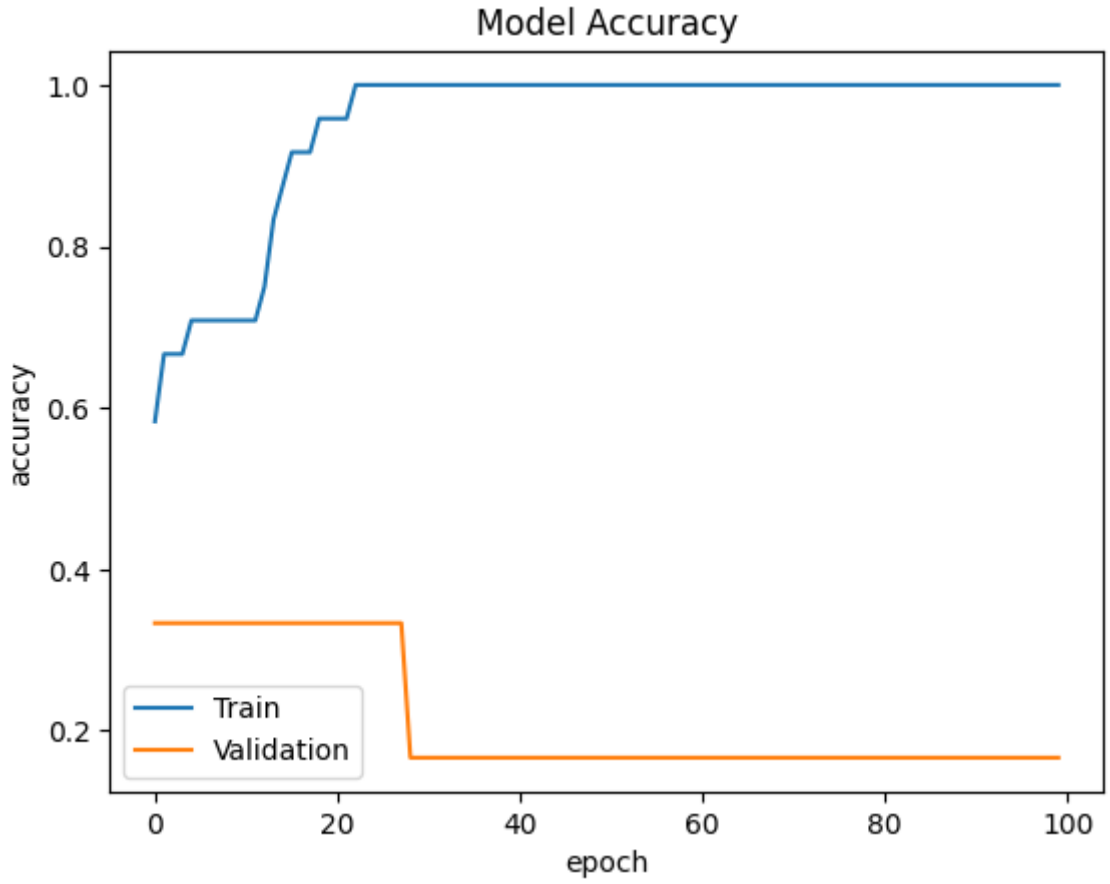
In [80]:

```
hist2 = model.fit(x_train, y_train, validation_split=0.2, epochs=100, batch_size=10, verb
```

```
Epoch 1/100
3/3 [==============================] - 2s 179ms/step - loss: 0.6889 - ac
curacy: 0.5833 - val_loss: 0.6999 - val_accuracy: 0.3333
Epoch 2/100
3/3 [==============================] - 0s 35ms/step - loss: 0.6817 - acc
uracy: 0.6667 - val_loss: 0.7017 - val_accuracy: 0.3333
Epoch 3/100
3/3 [==============================] - 0s 41ms/step - loss: 0.6763 - acc
uracy: 0.6667 - val_loss: 0.7034 - val_accuracy: 0.3333
Epoch 4/100
3/3 [==============================] - 0s 37ms/step - loss: 0.6706 - acc
uracy: 0.6667 - val_loss: 0.7049 - val_accuracy: 0.3333
Epoch 5/100
3/3 [==============================] - 0s 37ms/step - loss: 0.6649 - acc
uracy: 0.7083 - val_loss: 0.7074 - val_accuracy: 0.3333
Epoch 6/100
3/3 [==============================] - 0s 33ms/step - loss: 0.6590 - acc
uracy: 0.7083 - val_loss: 0.7110 - val_accuracy: 0.3333
Epoch 7/100
```

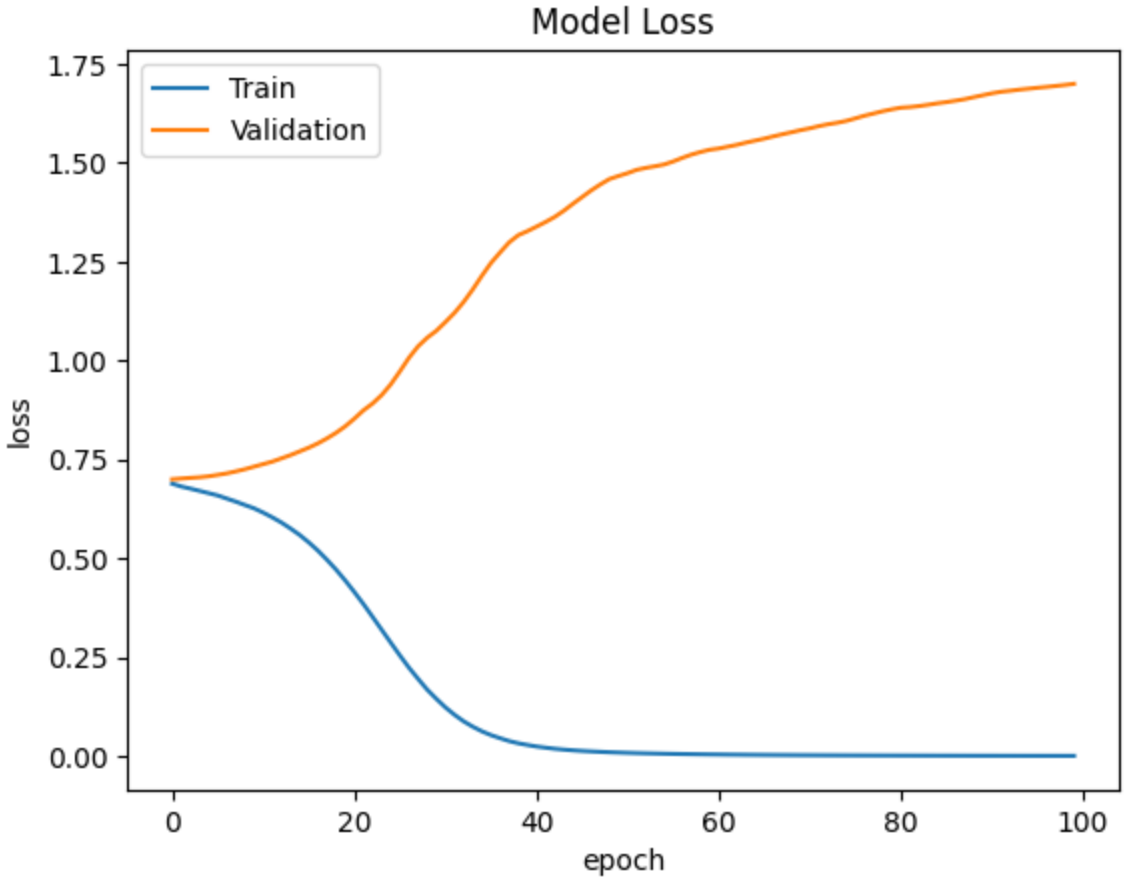
In [81]:

```
accuracy_plotting(hist2)
```



In [82]:

```
Loss_plotting(hist2)
```



In [83]:

```
# 4 layers with 32 nodes
model = Sequential()
model.add(Dense(32, input_dim=204, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(loss='binary_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['
```

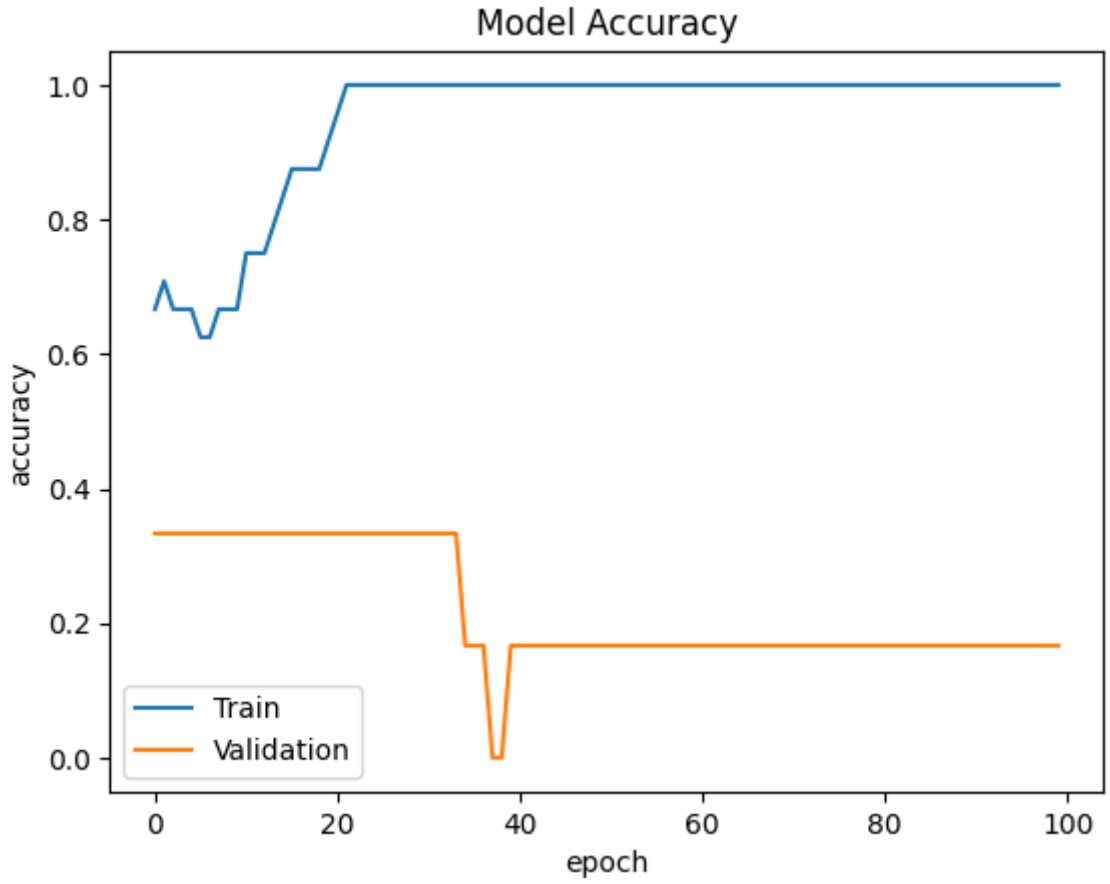
In [84]:

```
hist3 = model.fit(x_train, y_train, validation_split=0.2, epochs=100, batch_size=10, verb
```

```
Epoch 1/100
3/3 [==============================] - 2s 353ms/step - loss: 0.6894 - ac
curacy: 0.6667 - val_loss: 0.6982 - val_accuracy: 0.3333
Epoch 2/100
3/3 [==============================] - 0s 48ms/step - loss: 0.6820 - acc
uracy: 0.7083 - val_loss: 0.7023 - val_accuracy: 0.3333
Epoch 3/100
3/3 [==============================] - 0s 68ms/step - loss: 0.6745 - acc
uracy: 0.6667 - val_loss: 0.7063 - val_accuracy: 0.3333
Epoch 4/100
3/3 [==============================] - 0s 62ms/step - loss: 0.6673 - acc
uracy: 0.6667 - val_loss: 0.7105 - val_accuracy: 0.3333
Epoch 5/100
3/3 [==============================] - 0s 49ms/step - loss: 0.6604 - acc
uracy: 0.6667 - val_loss: 0.7156 - val_accuracy: 0.3333
Epoch 6/100
3/3 [==============================] - 0s 35ms/step - loss: 0.6521 - acc
uracy: 0.6250 - val_loss: 0.7217 - val_accuracy: 0.3333
Epoch 7/100
```

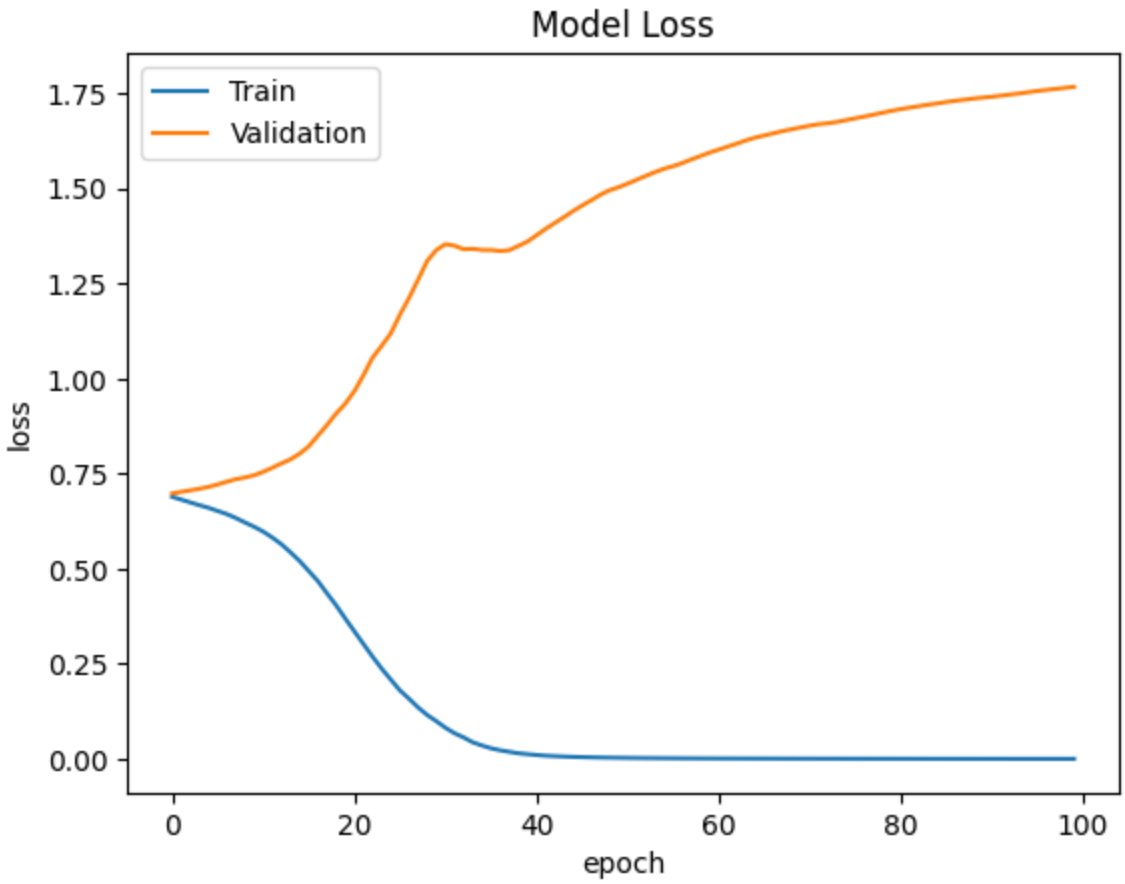
In [85]:

```
accuracy_plotting(hist3)
```



In [86]:

```
Loss_plotting(hist3)
```



In [87]:

```
# 5 layers with 32 nodes
model = Sequential()
model.add(Dense(32, input_dim=204, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(loss='binary_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['
```

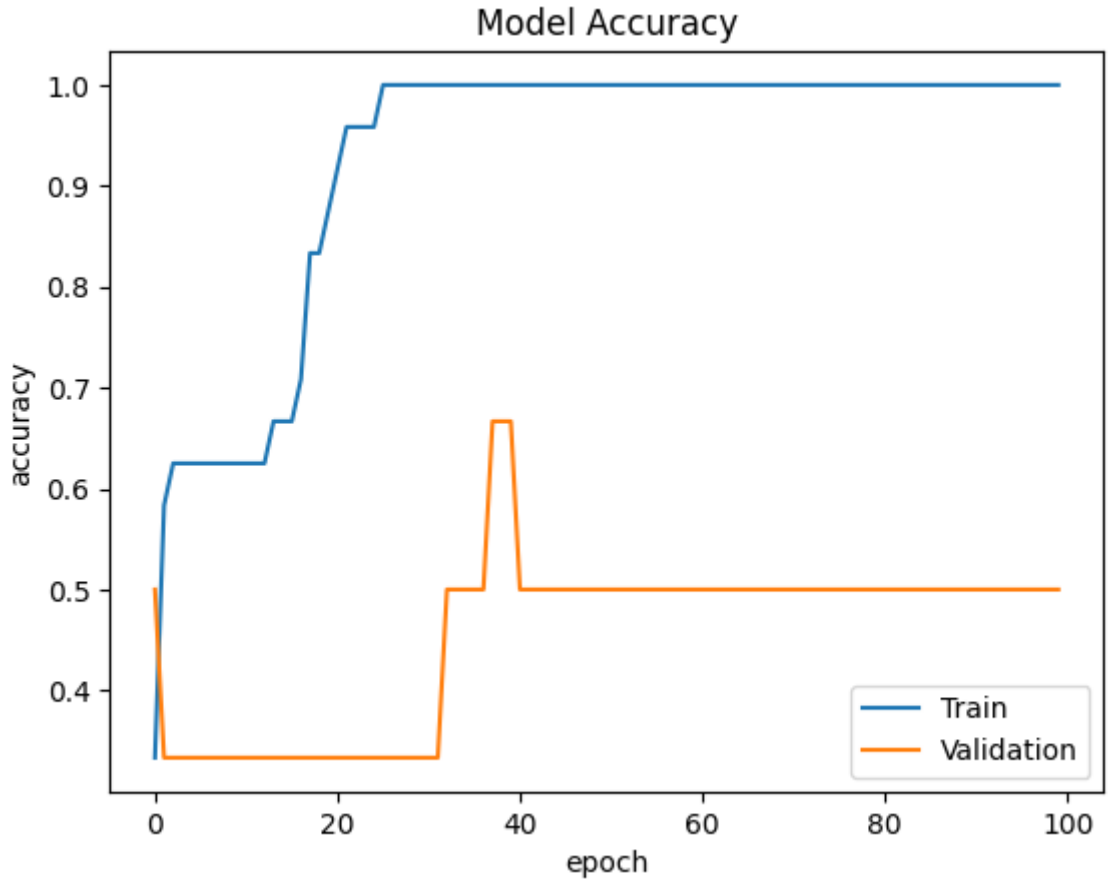
In [88]:

```
hist4 = model.fit(x_train, y_train, validation_split=0.2, epochs=100, batch_size=10, verb
```

```
Epoch 1/100
3/3 [==============================] - 1s 193ms/step - loss: 0.6972 - ac
curacy: 0.3333 - val_loss: 0.6935 - val_accuracy: 0.5000
Epoch 2/100
3/3 [==============================] - 0s 39ms/step - loss: 0.6920 - acc
uracy: 0.5833 - val_loss: 0.6962 - val_accuracy: 0.3333
Epoch 3/100
3/3 [==============================] - 0s 46ms/step - loss: 0.6889 - acc
uracy: 0.6250 - val_loss: 0.6975 - val_accuracy: 0.3333
Epoch 4/100
3/3 [==============================] - 0s 43ms/step - loss: 0.6862 - acc
uracy: 0.6250 - val_loss: 0.6985 - val_accuracy: 0.3333
Epoch 5/100
3/3 [==============================] - 0s 39ms/step - loss: 0.6840 - acc
uracy: 0.6250 - val_loss: 0.6996 - val_accuracy: 0.3333
Epoch 6/100
3/3 [==============================] - 0s 45ms/step - loss: 0.6816 - acc
uracy: 0.6250 - val_loss: 0.7009 - val_accuracy: 0.3333
Epoch 7/100
```

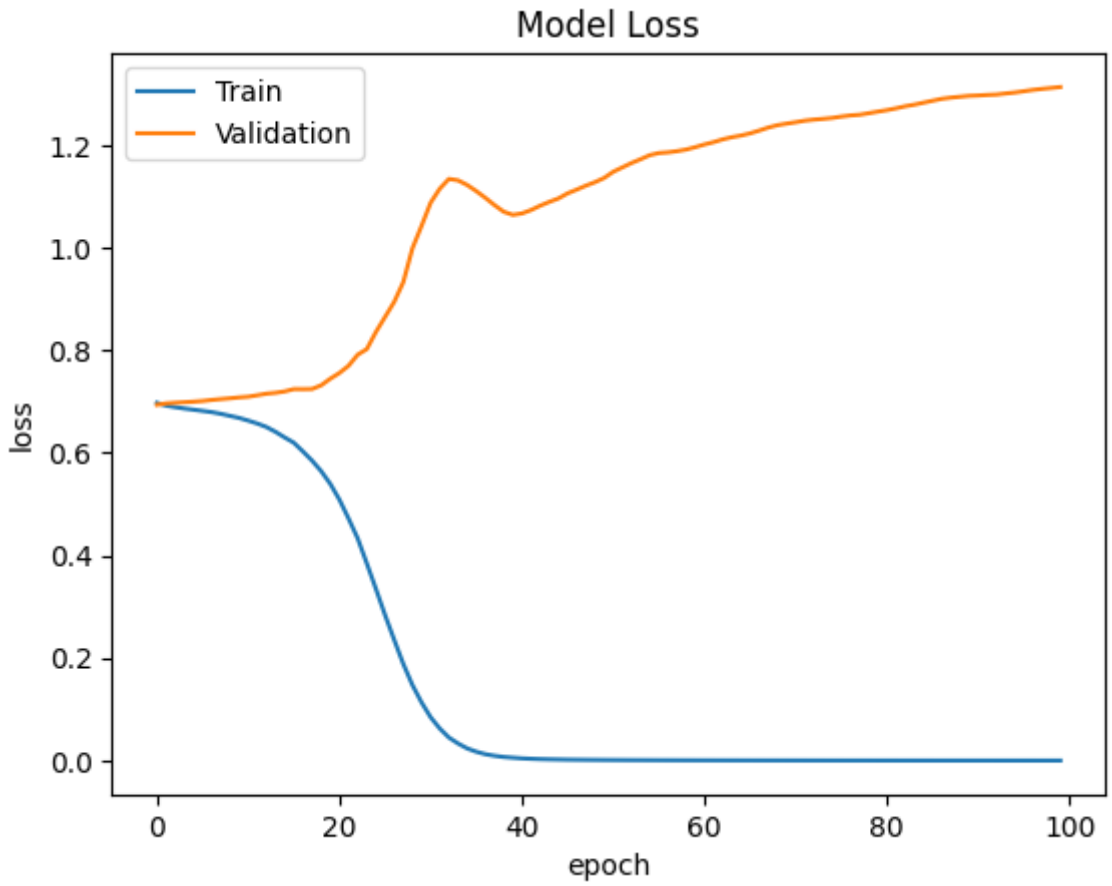
In [89]:

```
accuracy_plotting(hist4)
```



In [91]:

```
Loss_plotting(hist4)
```



In [ ]:

```

```