Assignment 2

In [1]:
```
from keras.datasets import imdb
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_wo
```

In [2]:
```
print(type([max(sequence) for sequence in train_data]))
max([max(sequence) for sequence in train_data])
```

Out[2]:
```
<class 'list'>
9999
```

In [3]:
```
word_index = imdb.get_word_index()

reverse_word_index = dict([(value, key) for (key, value) in word_index.items

decoded_review = ' '.join([reverse_word_index.get(i-3, '?') for i in train_d
decoded_review
```

Out[3]:
```
"? this film was just brilliant casting location scenery story direction ev
eryone's really suited the part they played and you could just imagine bein
g there robert ? is an amazing actor and now the same being director ? fath
er came from the same scottish island as myself so i loved the fact there w
as a real connection with this film the witty remarks throughout the film w
ere great it was just brilliant so much that i bought the film as soon as i
t was released for ? and would recommend it to everyone to watch and the fl
y fishing was amazing really cried at the end it was so sad and you know wh
at they say if you cry at a film it must have been good and this definitely
was also ? to the two little boy's that played the ? of norman and paul the
y were just brilliant children are often left out of the ? list i think bec
ause the stars that play them all grown up are such a big profile for the w
hole film but these children are amazing and should be praised for what the
y have done don't you think the whole story was so lovely because it was tr
ue and was someone's life after all that was shared with us all"
```

In [4]:
```
import numpy as np
def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i,sequence in enumerate(sequences):
        results[i,sequence] = 1
    return results

X_train = vectorize_sequences(train_data)

X_test = vectorize_sequences(test_data)
```

In [5]:
```
X_train[0]
```

Out[5]:
```
array([0., 1., 1., ..., 0., 0., 0.])
```

In [6]:
```
X_train.shape
```

Out[6]:
```
(25000, 10000)
```

In [7]:
```
y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')
```

In [8]:
```
from keras import models
from keras import layers
```

```
model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

In [9]:
```
from keras import optimizers
from keras import losses
from keras import metrics
model.compile(
    optimizer=optimizers.RMSprop(lr=0.001),
    loss = losses.binary_crossentropy,
    metrics = [metrics.binary_accuracy]
)
```

```
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.RMSpro
p` runs slowly on M1/M2 Macs, please use the legacy Keras optimizer instea
d, located at `tf.keras.optimizers.legacy.RMSprop`.
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_ra
te` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.RMSprop.
```

In [10]:
```
X_val = X_train[:10000]
partial_X_train = X_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

In [11]:
```
history = model.fit(
    partial_X_train,
    partial_y_train,
    epochs=20,
    batch_size=512,
    validation_data=(X_val, y_val)
)
```

```
Epoch 1/20
30/30 [==============================] - 2s 44ms/step - loss: 0.5607 - bina
ry_accuracy: 0.7580 - val_loss: 0.4300 - val_binary_accuracy: 0.8570
Epoch 2/20
30/30 [==============================] - 0s 13ms/step - loss: 0.3490 - bina
ry_accuracy: 0.8872 - val_loss: 0.3418 - val_binary_accuracy: 0.8658
Epoch 3/20
30/30 [==============================] - 0s 9ms/step - loss: 0.2588 - binar
y_accuracy: 0.9157 - val_loss: 0.3095 - val_binary_accuracy: 0.8752
Epoch 4/20
30/30 [==============================] - 0s 10ms/step - loss: 0.2086 - bina
ry_accuracy: 0.9299 - val_loss: 0.3189 - val_binary_accuracy: 0.8690
Epoch 5/20
30/30 [==============================] - 0s 17ms/step - loss: 0.1739 - bina
ry_accuracy: 0.9436 - val_loss: 0.2743 - val_binary_accuracy: 0.8876
Epoch 6/20
30/30 [==============================] - 0s 13ms/step - loss: 0.1497 - bina
ry_accuracy: 0.9513 - val_loss: 0.3059 - val_binary_accuracy: 0.8828
Epoch 7/20
30/30 [==============================] - 0s 12ms/step - loss: 0.1278 - bina
ry_accuracy: 0.9610 - val_loss: 0.2914 - val_binary_accuracy: 0.8869
Epoch 8/20
30/30 [==============================] - 0s 12ms/step - loss: 0.1126 - bina
ry_accuracy: 0.9661 - val_loss: 0.3109 - val_binary_accuracy: 0.8783
Epoch 9/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0969 - bina
ry_accuracy: 0.9714 - val_loss: 0.3312 - val_binary_accuracy: 0.8736
Epoch 10/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0840 - bina
ry_accuracy: 0.9766 - val_loss: 0.3312 - val_binary_accuracy: 0.8815
Epoch 11/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0738 - bina
ry_accuracy: 0.9796 - val_loss: 0.3645 - val_binary_accuracy: 0.8778
Epoch 12/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0626 - bina
ry_accuracy: 0.9847 - val_loss: 0.3667 - val_binary_accuracy: 0.8800
Epoch 13/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0542 - bina
ry_accuracy: 0.9869 - val_loss: 0.3855 - val_binary_accuracy: 0.8758
Epoch 14/20
30/30 [==============================] - 1s 36ms/step - loss: 0.0457 - bina
ry_accuracy: 0.9902 - val_loss: 0.4285 - val_binary_accuracy: 0.8750
Epoch 15/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0403 - bina
ry_accuracy: 0.9922 - val_loss: 0.4716 - val_binary_accuracy: 0.8698
Epoch 16/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0333 - bina
ry_accuracy: 0.9939 - val_loss: 0.4605 - val_binary_accuracy: 0.8744
Epoch 17/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0300 - bina
ry_accuracy: 0.9943 - val_loss: 0.4802 - val_binary_accuracy: 0.8691
Epoch 18/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0237 - bina
ry_accuracy: 0.9964 - val_loss: 0.5454 - val_binary_accuracy: 0.8567
Epoch 19/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0222 - bina
ry_accuracy: 0.9972 - val_loss: 0.5230 - val_binary_accuracy: 0.8701
Epoch 20/20
30/30 [==============================] - 1s 39ms/step - loss: 0.0197 - bina
ry_accuracy: 0.9967 - val_loss: 0.5470 - val_binary_accuracy: 0.8720
```

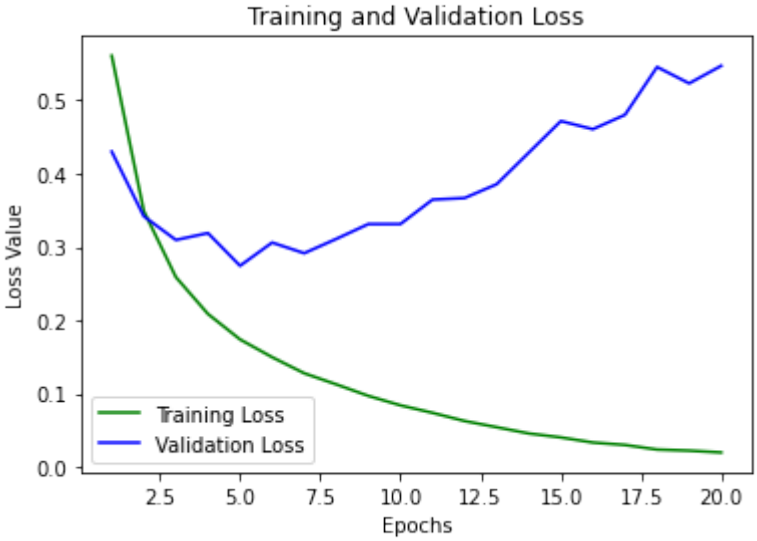
In [12]:

```
history_dict = history.history
history_dict.keys()
```

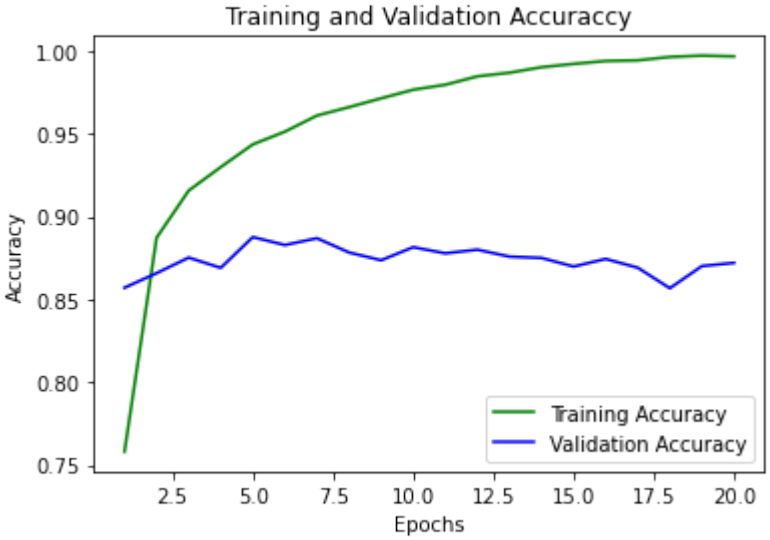
Out[12]:
```
dict_keys(['loss', 'binary_accuracy', 'val_loss', 'val_binary_accuracy'])
```

In [13]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [14]:
```
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, 'g', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
plt.legend()
plt.show()
```



In [15]:
```
acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, acc_values, 'g', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'b', label="Validation Accuracy")
plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



In [16]:
```
model.fit(
    partial_X_train,
    partial_y_train,
    epochs=3,
    batch_size=512,
    validation_data=(X_val, y_val)
)
```

```
Epoch 1/3
30/30 [==============================] - 1s 40ms/step - loss: 0.0142 - bina
ry_accuracy: 0.9988 - val_loss: 0.5647 - val_binary_accuracy: 0.8706
Epoch 2/3
30/30 [==============================] - 0s 12ms/step - loss: 0.0101 - bina
ry_accuracy: 0.9997 - val_loss: 0.6371 - val_binary_accuracy: 0.8682
Epoch 3/3
30/30 [==============================] - 0s 9ms/step - loss: 0.0122 - binar
y_accuracy: 0.9989 - val_loss: 0.6161 - val_binary_accuracy: 0.8665
```

Out[16]:
```
<keras.src.callbacks.History at 0x30b74dd50>
```

In [17]:
```
np.set_printoptions(suppress=True)
result = model.predict(X_test)
```

```
782/782 [==============================] - 1s 729us/step
```

In [18]:
```
result
```

Out[18]:
```
array([[0.00862332],
       [0.99999994],
       [0.39848855],
       ...,
       [0.00114856],
       [0.00454546],
       [0.970272  ]], dtype=float32)
```

In [19]:
```
y_pred = np.zeros(len(result))
for i, score in enumerate(result):
    y_pred[i] = 1 if score > 0.5 else 0
```

In [20]:
```
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(y_pred, y_test)
mae
```

Out[20]:
```
0.1456
```