Tokenizer->(BOW+TFIDF)using LSTM & ...

File Edit View Run Add-ons Help

Share | Save Version 3

Cancel Run | Code | Draft Session (5h:15m)

[1]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/100k-courseras-course-reviews-dataset/reviews.csv
/kaggle/input/100k-courseras-course-reviews-dataset/reviews_by_course.csv
```

+ Code + Markdown

[3]:
```
import seaborn as sns
import matplotlib.pyplot as plt
from collections import Counter
import re
from sklearn.feature_extraction.text import CountVectorizer
from wordcloud import WordCloud
import spacy
import nltk
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
```

[4]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import csv
import re
import string
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

[5]:
```
df = pd.read_csv('/kaggle/input/100k-courseras-course-reviews-dataset/reviews.csv')
df.head()
```

[5]:

| | Id | Review | Label |
|---|---|---|---|
| 0 | 0 | good and interesting | 5 |
| 1 | 1 | This class is very helpful to me. Currently, I... | 5 |
| 2 | 2 | likelProf and TAs are helpful and the discussi... | 5 |
| 3 | 3 | Easy to follow and includes a lot basic and im... | 5 |
| 4 | 4 | Really nice teacherIl could got the point eazi... | 4 |

[6]:
```
df2 = pd.read_csv('/kaggle/input/100k-courseras-course-reviews-dataset/reviews_by_course.csv')
df2.head()
```

[6]:

| | CourseId | Review | Label |
|---|---|---|---|
| 0 | 2-speed-it | BOring | 1 |
| 1 | 2-speed-it | Bravo ! | 5 |
| 2 | 2-speed-it | Very goo | 5 |
| 3 | 2-speed-it | Great course - I recommend it for all, especia... | 5 |
| 4 | 2-speed-it | One of the most useful course on IT Management! | 5 |

## We will be working on DF

[7]:
```
df.isnull().sum()
```

[7]:
```
Id        0
Review    0
Label     0
dtype: int64
```

[8]:
```
label_count = df['Label'].value_counts()
print(label_count)
```

```
5    79173
4    18054
3     5071
1     2469
2     2251
Name: Label, dtype: int64
```

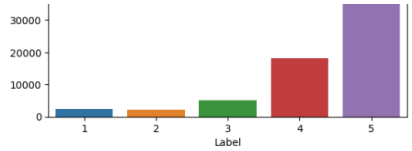
[9]:
```
label_values = df['Label'].values
label_values
```

[9]:
```
array([5, 5, 5, ..., 5, 4, 4])
```

## Data Exploration

**LABEL DISTRIBUTION**

[10]:
```
sns.countplot(x='Label',data=df)
plt.title('Distribution of Labels')
plt.show()
```




[11]:
```
df.shape
```

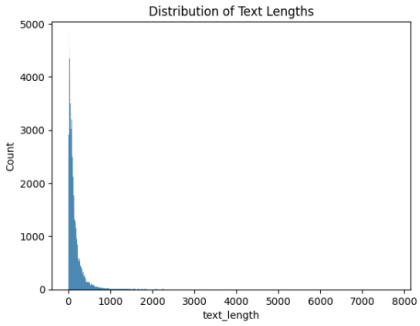
[11]: (107018, 3)

**There are Total 107018 Reviews**

## Length of the Text

[12]:
```
df['text_length'] = df['Review'].apply(len)
```

[13]:
```
sns.histplot(x='text_length', data=df)
plt.title('Distribution of Text Lengths')
plt.show()
```




## Extract and seperate the data based on labels

[14]:
```
train0=df[df['Label']==0]
train1=df[df['Label']==1]
train2=df[df['Label']==2]
train3=df[df['Label']==3]
train4=df[df['Label']==4]
train5=df[df['Label']==5]
```

## Peeking at the Distribution between each label

[15]:
```
train0.shape, train1.shape, train2.shape, train3.shape, train4.shape, train5.shape
```

[15]: ((0, 4), (2469, 4), (2251, 4), (5071, 4), (18054, 4), (79173, 4))

## Reducing the data of each label by 10

[16]:
```
train0=train0[:int(train0.shape[0]/10)]
train1=train1[:int(train1.shape[0]/10)]
train2=train2[:int(train2.shape[0]/10)]
train3=train3[:int(train3.shape[0]/10)]
train4=train4[:int(train4.shape[0]/10)]
train5=train5[:int(train5.shape[0]/10)]
```

[17]:
```
train0.shape, train1.shape, train2.shape, train3.shape, train4.shape, train5.shape
```

[17]: ((0, 4), (246, 4), (225, 4), (507, 4), (1805, 4), (7917, 4))

[18]:
```
df=pd.concat([train0,train1,train2,train3,train4,train5],axis=0)
```

[19]:
```
df.shape
```

[19]: (10700, 4)

## Check for NULL

[20]:
```
total_null = df.isnull().sum().sort_values(ascending = False)
percent = ((df.isnull().sum()/df.isnull().count())*100).sort_values(ascending = False)
print("Total records = ", df.shape[0])

missing_data = pd.concat([total_null,percent.round(2)],axis=1,keys=['Total Missing','In Percent'])
missing_data.head(16)
```

[20]:
```
Total records =  10700
```

| | Total Missing | In Percent |
|---|---|---|
| Id | 0 | 0.0 |
| Review | 0 | 0.0 |
| Label | 0 | 0.0 |
| text_length | 0 | 0.0 |

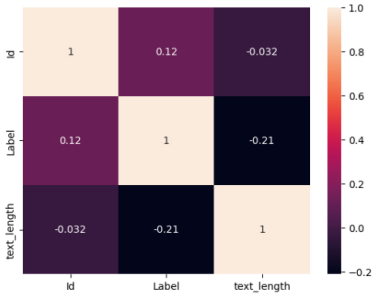
**GOOD NEWS**

[21]:
```
df.head()
```

[21]:

| | Id | Review | Label | text_length |
|---|---|---|---|---|
| 49 | 49 | This course doesn't contain any new informatio... | 1 | 159 |
| 71 | 71 | I do not find very interesting this course. to... | 1 | 186 |
| 79 | 79 | A lot of speaking without any sense. Skip it a... | 1 | 56 |
| 105 | 105 | This course doesn't contain any new informatio... | 1 | 159 |
| 173 | 173 | It's not a course... this is a very short gene... | 1 | 77 |

[22]:
```
import seaborn as sns
ax = sns.heatmap(df.corr(),annot=True)
```

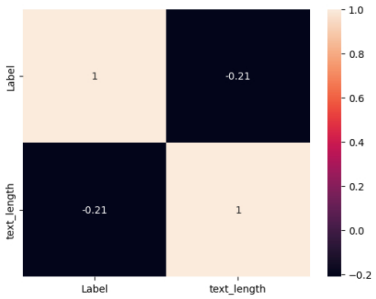



[23]:
```
# We don't need the Id column. Let's drop it !
df.drop(columns=['Id'],inplace=True)
df.head()
```

[23]:

| | Review | Label | text_length |
|---|---|---|---|
| 49 | This course doesn't contain any new informatio... | 1 | 159 |
| 71 | I do not find very interesting this course. to... | 1 | 186 |
| 79 | A lot of speaking without any sense. Skip it a... | 1 | 56 |
| 105 | This course doesn't contain any new informatio... | 1 | 159 |
| 173 | It's not a course... this is a very short gene... | 1 | 77 |

[24]:
```
import seaborn as sns
ax = sns.heatmap(df.corr(),annot=True)
```




## DATA CLEANING

[25]:
```
#remove hashtags
df['Review'].replace( { r"#(\w+)" : '' }, inplace= True, regex = True)
#Remove Mention
df['Review'].replace( { r"@(\w+)" : '' }, inplace= True, regex = True)
#Remove URL
df['Review'].astype(str).replace( { r"http\S+" : '' }, inplace= True, regex = True)

df['Review']=df['Review'].str.lower()
```

[26]:
```
# Import stopwords with nltk.
from nltk.corpus import stopwords
stop = stopwords.words('english')
```

[27]:
```
df['Review'] = df['Review'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))
```

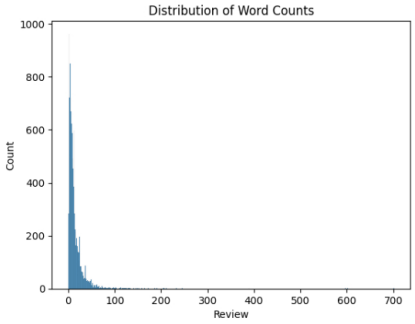
## After Data Cleaning

[28]:
```
df.head()
```

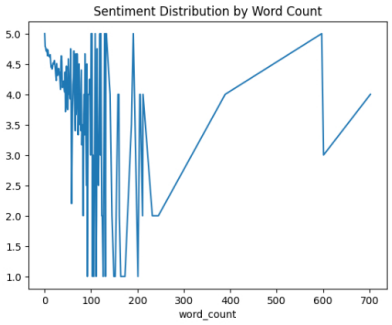
[28]:

| | Review | Label | text_length |
|---|---|---|---|
| 49 | course contain new information. teach excitedl... | 1 | 159 |
| 71 | find interesting course. many interviews. coul... | 1 | 186 |
| 79 | lot speaking without sense. skip cost | 1 | 56 |
| 105 | course contain new information. teach excitedl... | 1 | 159 |
| 173 | course... short general introduction 3d printing | 1 | 77 |

[29]:
```
word_count = df['Review'].apply(lambda x: len(x.split()))
sns.histplot(word_count)
plt.title('Distribution of Word Counts')
plt.show()
```

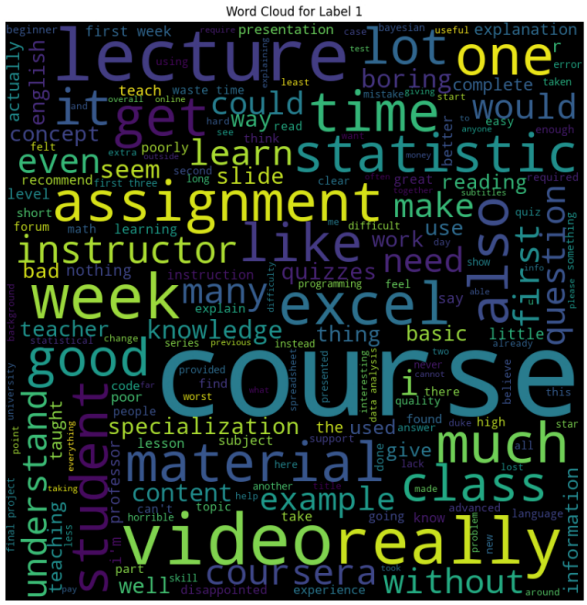



[30]:
```
df['word_count'] = word_count
df.groupby('word_count')['Label'].mean().plot()
plt.title('Sentiment Distribution by Word Count')
plt.show()
```



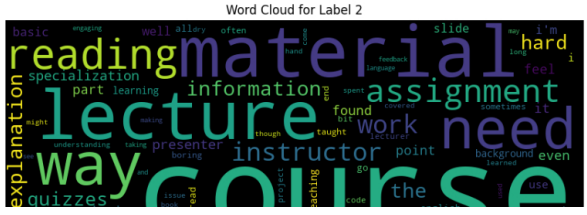

## Label 1

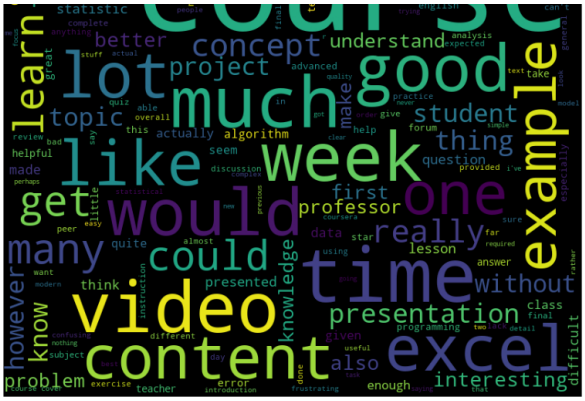
[31]:
```
label1_texts = df[df['Label'] == 1]['Review']
label1_wordcloud = WordCloud(width=800, height=800, background_color='black', stopwords=set()).generate(' '.join(label1_texts))
plt.figure(figsize=(8, 8), facecolor=None)
plt.imshow(label1_wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.title('Word Cloud for Label 1')
plt.show()
```




## Label 2

[32]:
```
label2_texts = df[df['Label'] == 2]['Review']
label2_wordcloud = WordCloud(width=800, height=800, background_color='black', stopwords=set()).generate(' '.join(label2_texts))
plt.figure(figsize=(8, 8), facecolor=None)
plt.imshow(label2_wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.title('Word Cloud for Label 2')
plt.show()

```

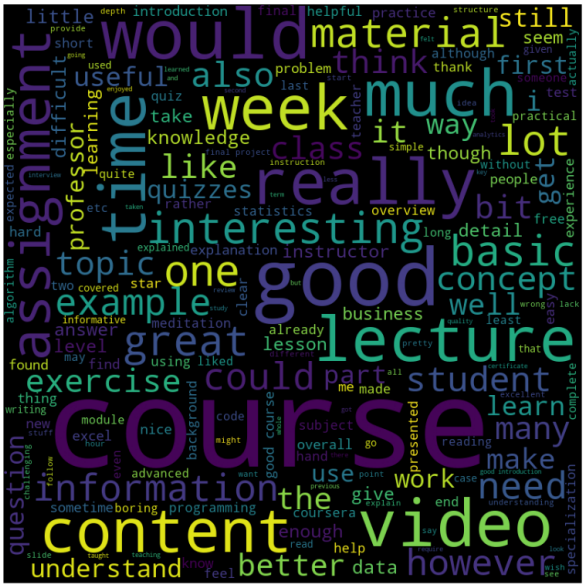





## Label 3

[33]:
```
label3_texts = df[df['Label'] == 3]['Review']
label3_wordcloud = WordCloud(width=800, height=800, background_color='black', stopwords=set()).generate(' '.join(label3_texts))
plt.figure(figsize=(8, 8), facecolor=None)
plt.imshow(label3_wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.title('Word Cloud for Label 3')
plt.show()
```

Word Cloud for Label 3



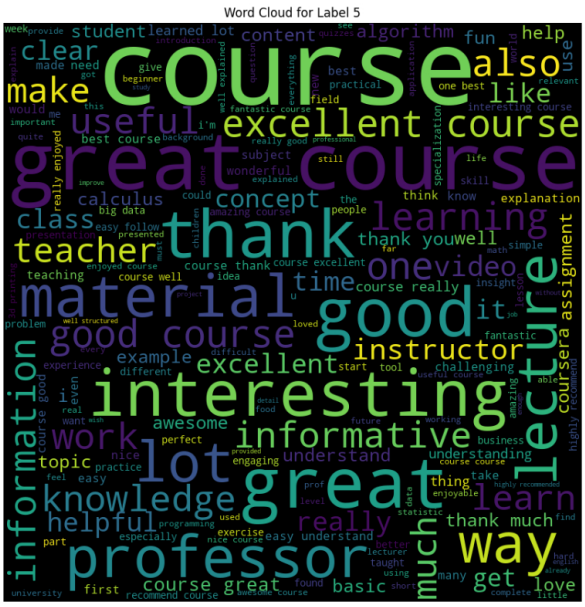

## Label 4

[34]:
```
label4_texts = df[df['Label'] == 4]['Review']
label4_wordcloud = WordCloud(width=800, height=800, background_color='black', stopwords=set()).generate(' '.join(label4_texts))
plt.figure(figsize=(8, 8), facecolor=None)
plt.imshow(label4_wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.title('Word Cloud for Label 4')
plt.show()
```

Word Cloud for Label 4


## Label 5

[35]:
```
label5_texts = df[df['Label'] == 5]['Review']
label5_wordcloud = WordCloud(width=800, height=800, background_color='black', stopwords=set()).generate(' '.join(label5_texts))
plt.figure(figsize=(8, 8), facecolor=None)
plt.imshow(label5_wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.title('Word Cloud for Label 5')
plt.show()
```




## Stemming

[36]:
```
from nltk.corpus import stopwords

nltk.download('stopwords')
ps = PorterStemmer()
df['Review'] = df['Review'].apply(lambda x: ' '.join([ps.stem(word) for word in x.split() if word not in set(stopwords.words('english'))]))
```

```
[nltk_data] Downloading package stopwords to /usr/share/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

## Punctuation Removal

[37]:
```
import string
nlp = spacy.load('en_core_web_sm')
punctuations = string.punctuation
df['Review'] = df['Review'].apply(lambda x: ' '.join([token.text for token in nlp(x) if not token.is_punct]))
```

[38]:
```
df.head()
```

[38]:

| | Review | Label | text_length | word_count |
|---|---|---|---|---|
| 49 | cours contain new information teach excitedli ... | 1 | 159 | 14 |
| 71 | find interest course manl interviews could wor... | 1 | 186 | 14 |
| 79 | lot speak without sense skip cost | 1 | 56 | 6 |
| 105 | cours contain new information teach excitedli ... | 1 | 159 | 14 |
| 173 | course short gener introduct 3d print | 1 | 77 | 6 |

## Randomization

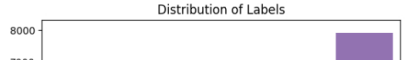
[39]:
```
#randomization
df = df.sample(frac=1).reset_index(drop=True)
df.head()
```

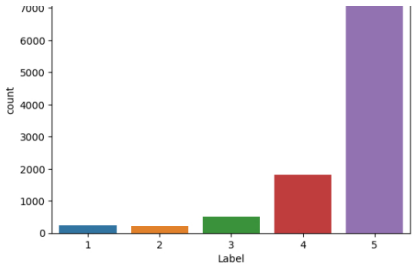
[39]:

| | Review | Label | text_length | word_count |
|---|---|---|---|---|
| 0 | good science great explanations simpl recip ti... | 4 | 694 | 72 |
| 1 | incred informative definit appropri beginners ... | 5 | 298 | 27 |
| 2 | awesom cours interest practic exercises | 5 | 57 | 5 |
| 3 | good must engag | 5 | 24 | 3 |
| 4 | great class i 'm happi choic special take cour... | 5 | 235 | 18 |

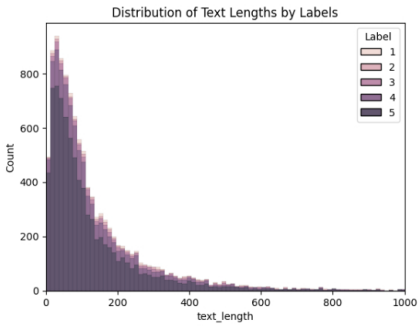
## EDA for final Dataset

[40]:
```
sns.countplot(x='Label',data=df)
plt.title('Distribution of Labels')
plt.show()
```

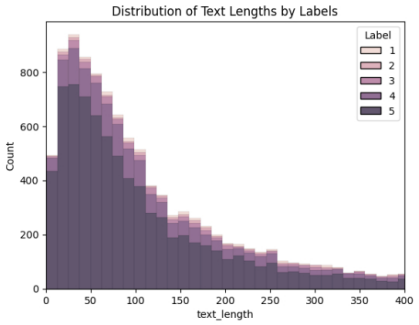





[41]:
```
sns.histplot(x='text_length', data=df, hue='Label', multiple='stack')
plt.title('Distribution of Text Lengths by Labels')
plt.xlim(0, 1000)
plt.show()
```




[42]:
```
sns.histplot(x='text_length', data=df, hue='Label', multiple='stack')
plt.title('Distribution of Text Lengths by Labels')
plt.xlim(0, 400)
plt.show()
```




[43]:
```
df.shape
```

[43]:
```
(10700, 4)
```

[44]:
```
df['Label'].value_counts()
```

[44]:
```
5    7917
4    1805
3     507
1     246
2     225
Name: Label, dtype: int64
```

## Defining X and Y

[45]:
```
y = df['Label']
x = df['Review']
```

## Vectorization using TFIDF

[46]:
```
# Creating a word corpus for vectorization
corpus = []
for i in range(x.shape[0]):
    corpus.append(x.iloc[i])
```

[47]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import csv
import re
import string
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

[48]:
```
vectorizer1 = TfidfVectorizer(max_features=1000)
X1 = vectorizer1.fit_transform(x)
feature_names1 = vectorizer1.get_feature_names_out()
denselist1 = X1.todense().tolist()
train = pd.DataFrame(denselist1, columns=feature_names1)
```

[49]:
```
# splitting the training and testing part from the data
X_train, X_test, y_train, y_test = train_test_split(train, y, test_size=0.2, random_state=0)

```

[50]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

[50]: ((8560, 1000), (2140, 1000), (8560,), (2140,))

[51]:
```
df.head()
```

[51]:

| | Review | Label | text_length | word_count |
|---|---|---|---|---|
| 0 | good science great explanations simpl recip ti... | 4 | 694 | 72 |
| 1 | incred informative definit appropri beginners ... | 5 | 298 | 27 |
| 2 | awesom cours interest practic exercises | 5 | 57 | 5 |
| 3 | good must engag | 5 | 24 | 3 |
| 4 | great class i 'm happi choic special take cour... | 5 | 235 | 18 |

[85]:
```
df.head()
```

[85]:

| | Review | Label | text_length | word_count |
|---|---|---|---|---|
| 0 | good science great explanations simpl recip ti... | 4 | 694 | 72 |
| 1 | incred informative definit appropri beginners ... | 5 | 298 | 27 |
| 2 | awesom cours interest practic exercises | 5 | 57 | 5 |
| 3 | good must engag | 5 | 24 | 3 |
| 4 | great class i 'm happi choic special take cour... | 5 | 235 | 18 |

[87]:
```
df1 = df.copy()
```

## LSTM Testing

[52]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
import matplotlib.pyplot as plt

```

```
X_train, y_train = np.array(X_train), np.array(y_train)
```

[57]:
```
X_train.shape
```

[57]: (8560, 1000)

[58]:
```
X_train.size
```

[58]: 8560000

[70]:
```
y_train
```

[70]: array([5, 4, 3, ..., 4, 4, 5])

[71]:
```
y_train.shape
```

[71]: (8560,)

```
# Reshaping X_train for efficient modelling
X_train = np.reshape(X_train, (X_train.shape[0],X_train.shape[1],1))
```

[61]:
```
X_train
```

[61]:
```
array([[[0.],
        [0.],
        [0.],
        ...,
        [0.],
        [0.],
        [0.]],

       [[0.],
        [0.],
        [0.],
        ...,
        [0.],
        [0.],
        [0.]],

       [[0.],
        [0.],
        [0.],
        ...,
        [0.],
        [0.],
        [0.]],

       ...,

       [[0.],
        [0.],
        [0.],
        ...,
        [0.],
        [0.],
        [0.]],

       [[0.],
        [0.],
        [0.],
        ...,
        [0.],
        [0.],
        [0.]],

       [[0.],
        [0.],
        [0.],
        ...,
        [0.],
        [0.],
        [0.]]])
```

[62]:
```
X_train.shape[1]
```

[62]: 1000

[63]:
```
X_train.shape[2]
```

[63]: 1

[64]:
```
X_train.shape[0]
```

[64]: 8560

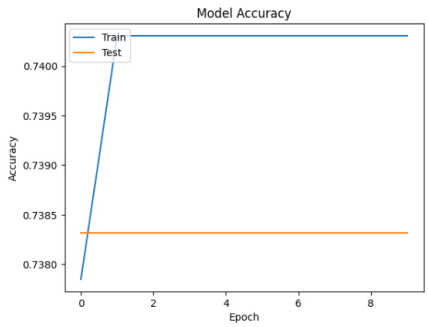
[65]:
```
X_test = np.array(X_test)
```

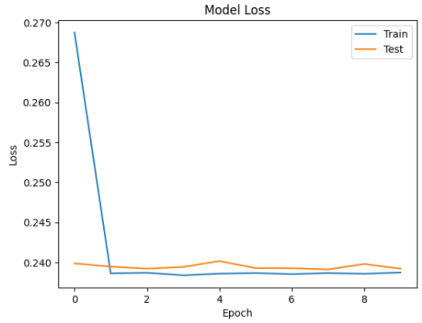
[66]:
```
# Reshaping X_test for efficient modelling
X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
```

```
from keras.utils import to_categorical

# Convert labels to one-hot encoded format
y_train_encoded = to_categorical(y_train, num_classes=6)
y_test_encoded = to_categorical(y_test, num_classes=6)

# Creating the LSTM model
model = Sequential()
model.add(LSTM(64, input_shape=(X_train.shape[1], 1)))
model.add(Dense(6, activation='sigmoid'))

# Compiling the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Training the model
history = model.fit(X_train, y_train_encoded, epochs=10, batch_size=32, validation_data=(X_test, y_test_encoded))
```

```
Epoch 1/10
268/268 [==============================] - 134s 488ms/step - loss: 0.2688 - accuracy: 0.7379 - val_loss: 0.2399 - val_accuracy: 0.7383
Epoch 2/10
268/268 [==============================] - 130s 487ms/step - loss: 0.2386 - accuracy: 0.7403 - val_loss: 0.2395 - val_accuracy: 0.7383
Epoch 3/10
268/268 [==============================] - 130s 487ms/step - loss: 0.2387 - accuracy: 0.7403 - val_loss: 0.2392 - val_accuracy: 0.7383
Epoch 4/10
268/268 [==============================] - 130s 485ms/step - loss: 0.2384 - accuracy: 0.7403 - val_loss: 0.2394 - val_accuracy: 0.7383
Epoch 5/10
268/268 [==============================] - 132s 492ms/step - loss: 0.2386 - accuracy: 0.7403 - val_loss: 0.2402 - val_accuracy: 0.7383
Epoch 6/10
268/268 [==============================] - 132s 494ms/step - loss: 0.2387 - accuracy: 0.7403 - val_loss: 0.2393 - val_accuracy: 0.7383
Epoch 7/10
268/268 [==============================] - 131s 488ms/step - loss: 0.2385 - accuracy: 0.7403 - val_loss: 0.2393 - val_accuracy: 0.7383
Epoch 8/10
268/268 [==============================] - 132s 492ms/step - loss: 0.2387 - accuracy: 0.7403 - val_loss: 0.2391 - val_accuracy: 0.7383
Epoch 9/10
 45/268 [====>.........................] - ETA: 1:41 - loss: 0.2308 - accuracy: 0.7500
```

```
# Plotting accuracy and loss curves
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Test'], loc='upper right')
plt.show()
```




+ Code + Markdown

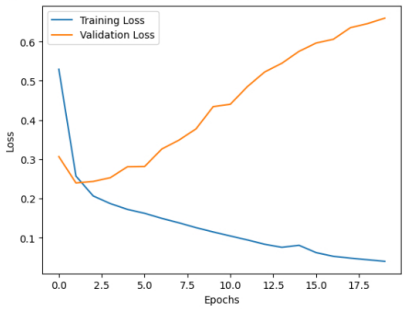
## TFIDF + Neural Networks

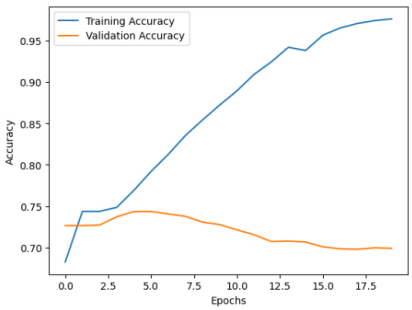
## RELU + RELU + TANH

[77]:
```
import numpy as np
from tensorflow import keras
from tensorflow.keras import layers
```

```
input_dim = X_train.shape[1]


model = keras.Sequential()
model.add(layers.Dense(64, input_dim=input_dim, activation='relu'))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(6, activation='tanh'))


model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])


history=model.fit(X_train, y_train_encoded , epochs=20, batch_size=32, validation_split=0.2)

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

```

```
Epoch 1/20
214/214 [==============================] - 2s 5ms/step - loss: 0.5292 - accuracy: 0.6825 - val_loss: 0.3066 - val_accuracy: 0.7266
Epoch 2/20
214/214 [==============================] - 1s 3ms/step - loss: 0.2569 - accuracy: 0.7437 - val_loss: 0.2395 - val_accuracy: 0.7266
Epoch 3/20
214/214 [==============================] - 1s 3ms/step - loss: 0.2065 - accuracy: 0.7437 - val_loss: 0.2433 - val_accuracy: 0.7272
Epoch 4/20
214/214 [==============================] - 1s 3ms/step - loss: 0.1869 - accuracy: 0.7485 - val_loss: 0.2529 - val_accuracy: 0.7371
Epoch 5/20
214/214 [==============================] - 1s 3ms/step - loss: 0.1721 - accuracy: 0.7691 - val_loss: 0.2808 - val_accuracy: 0.7436
Epoch 6/20
214/214 [==============================] - 1s 3ms/step - loss: 0.1621 - accuracy: 0.7919 - val_loss: 0.2814 - val_accuracy: 0.7436
Epoch 7/20
214/214 [==============================] - 1s 3ms/step - loss: 0.1492 - accuracy: 0.8125 - val_loss: 0.3259 - val_accuracy: 0.7407
Epoch 8/20
214/214 [==============================] - 1s 3ms/step - loss: 0.1380 - accuracy: 0.8353 - val_loss: 0.3486 - val_accuracy: 0.7377
Epoch 9/20
214/214 [==============================] - 1s 3ms/step - loss: 0.1255 - accuracy: 0.8541 - val_loss: 0.3774 - val_accuracy: 0.7307
Epoch 10/20
214/214 [==============================] - 1s 3ms/step - loss: 0.1146 - accuracy: 0.8721 - val_loss: 0.4341 - val_accuracy: 0.7278
Epoch 11/20
214/214 [==============================] - 1s 3ms/step - loss: 0.1044 - accuracy: 0.8892 - val_loss: 0.4401 - val_accuracy: 0.7214
Epoch 12/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0941 - accuracy: 0.9090 - val_loss: 0.4854 - val_accuracy: 0.7155
Epoch 13/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0833 - accuracy: 0.9241 - val_loss: 0.5223 - val_accuracy: 0.7074
Epoch 14/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0755 - accuracy: 0.9417 - val_loss: 0.5445 - val_accuracy: 0.7079
Epoch 15/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0807 - accuracy: 0.9378 - val_loss: 0.5748 - val_accuracy: 0.7068
Epoch 16/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0619 - accuracy: 0.9562 - val_loss: 0.5961 - val_accuracy: 0.7009
Epoch 17/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0526 - accuracy: 0.9650 - val_loss: 0.6056 - val_accuracy: 0.6986
Epoch 18/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0479 - accuracy: 0.9704 - val_loss: 0.6352 - val_accuracy: 0.6980
Epoch 19/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0438 - accuracy: 0.9739 - val_loss: 0.6456 - val_accuracy: 0.6998
Epoch 20/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0397 - accuracy: 0.9759 - val_loss: 0.6594 - val_accuracy: 0.6992
```




[78]:
```
model.summary()
```

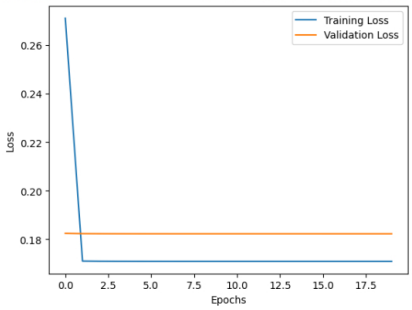
```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_7 (Dense)             (None, 64)                64064

 dense_8 (Dense)             (None, 32)                2080

 dense_9 (Dense)             (None, 6)                 198

=================================================================
Total params: 66,342
Trainable params: 66,342
Non-trainable params: 0
_________________________________________________________________
```

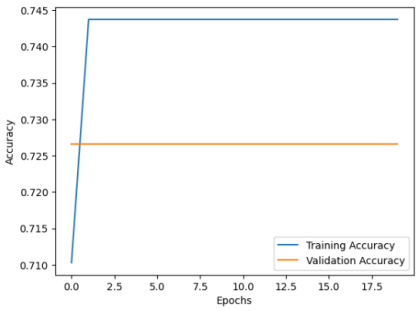
## RELU + RELU + RELU + TANH

[83]:
```
input_dim = X_train.shape[1]

model = keras.Sequential()
model.add(layers.Dense(128, input_dim=input_dim, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(6, activation='tanh'))


model.compile(loss='hinge', optimizer='adam', metrics=['accuracy'])


history=model.fit(X_train, y_train_encoded, epochs=20, batch_size=32, validation_split=0.2)

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
```

```
plt.legend()
plt.show()

plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
Epoch 1/20
214/214 [==============================] - 2s 5ms/step - loss: 0.2711 - accuracy: 0.7103 - val_loss: 0.1824 - val_accuracy: 0.7266
Epoch 2/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1710 - accuracy: 0.7437 - val_loss: 0.1823 - val_accuracy: 0.7266
Epoch 3/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1709 - accuracy: 0.7437 - val_loss: 0.1823 - val_accuracy: 0.7266
Epoch 4/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1709 - accuracy: 0.7437 - val_loss: 0.1823 - val_accuracy: 0.7266
Epoch 5/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1709 - accuracy: 0.7437 - val_loss: 0.1823 - val_accuracy: 0.7266
Epoch 6/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1709 - accuracy: 0.7437 - val_loss: 0.1822 - val_accuracy: 0.7266
Epoch 7/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1709 - accuracy: 0.7437 - val_loss: 0.1822 - val_accuracy: 0.7266
Epoch 8/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1709 - accuracy: 0.7437 - val_loss: 0.1822 - val_accuracy: 0.7266
Epoch 9/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1709 - accuracy: 0.7437 - val_loss: 0.1822 - val_accuracy: 0.7266
Epoch 10/20
214/214 [==============================] - 1s 5ms/step - loss: 0.1709 - accuracy: 0.7437 - val_loss: 0.1822 - val_accuracy: 0.7266
Epoch 11/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1709 - accuracy: 0.7437 - val_loss: 0.1822 - val_accuracy: 0.7266
Epoch 12/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1709 - accuracy: 0.7437 - val_loss: 0.1822 - val_accuracy: 0.7266
Epoch 13/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1709 - accuracy: 0.7437 - val_loss: 0.1822 - val_accuracy: 0.7266
Epoch 14/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1709 - accuracy: 0.7437 - val_loss: 0.1822 - val_accuracy: 0.7266
Epoch 15/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1709 - accuracy: 0.7437 - val_loss: 0.1822 - val_accuracy: 0.7266
Epoch 16/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1709 - accuracy: 0.7437 - val_loss: 0.1822 - val_accuracy: 0.7266
Epoch 17/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1709 - accuracy: 0.7437 - val_loss: 0.1822 - val_accuracy: 0.7266
Epoch 18/20
214/214 [==============================] - 1s 5ms/step - loss: 0.1709 - accuracy: 0.7437 - val_loss: 0.1822 - val_accuracy: 0.7266
Epoch 19/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1709 - accuracy: 0.7437 - val_loss: 0.1822 - val_accuracy: 0.7266
Epoch 20/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1709 - accuracy: 0.7437 - val_loss: 0.1822 - val_accuracy: 0.7266
```



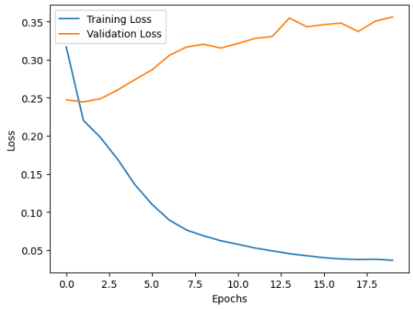

## RELU + RELU + RELU + RELU + TANH

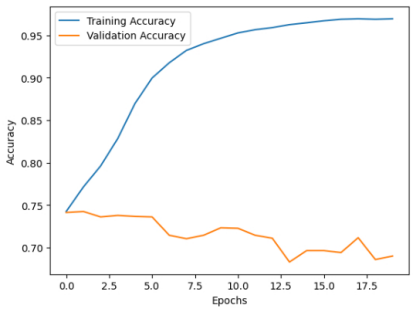
[82]:

```
input_dim = X_train.shape[1]

model = keras.Sequential()
model.add(layers.Dense(32, input_dim=input_dim, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(6, activation='tanh'))

model.compile(loss='squared_hinge', optimizer='adam', metrics=['accuracy'])

history=model.fit(X_train, y_train_encoded, epochs=20, batch_size=32, validation_split=0.2)

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
Epoch 1/20
214/214 [==============================] - 2s 4ms/step - loss: 0.3167 - accuracy: 0.7424 - val_loss: 0.2473 - val_accuracy: 0.7412
Epoch 2/20
214/214 [==============================] - 1s 3ms/step - loss: 0.2203 - accuracy: 0.7713 - val_loss: 0.2445 - val_accuracy: 0.7424
Epoch 3/20
214/214 [==============================] - 1s 3ms/step - loss: 0.1977 - accuracy: 0.7960 - val_loss: 0.2488 - val_accuracy: 0.7360
Epoch 4/20
214/214 [==============================] - 1s 3ms/step - loss: 0.1691 - accuracy: 0.8284 - val_loss: 0.2604 - val_accuracy: 0.7377
Epoch 5/20
214/214 [==============================] - 1s 3ms/step - loss: 0.1360 - accuracy: 0.8696 - val_loss: 0.2738 - val_accuracy: 0.7366
Epoch 6/20
214/214 [==============================] - 1s 3ms/step - loss: 0.1100 - accuracy: 0.8998 - val_loss: 0.2867 - val_accuracy: 0.7360
Epoch 7/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0896 - accuracy: 0.9178 - val_loss: 0.3056 - val_accuracy: 0.7144
Epoch 8/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0765 - accuracy: 0.9324 - val_loss: 0.3165 - val_accuracy: 0.7103
Epoch 9/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0689 - accuracy: 0.9403 - val_loss: 0.3202 - val_accuracy: 0.7144
Epoch 10/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0624 - accuracy: 0.9467 - val_loss: 0.3153 - val_accuracy: 0.7231
Epoch 11/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0578 - accuracy: 0.9530 - val_loss: 0.3211 - val_accuracy: 0.7225
Epoch 12/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0528 - accuracy: 0.9568 - val_loss: 0.3280 - val_accuracy: 0.7144
Epoch 13/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0491 - accuracy: 0.9593 - val_loss: 0.3304 - val_accuracy: 0.7109
Epoch 14/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0454 - accuracy: 0.9628 - val_loss: 0.3546 - val_accuracy: 0.6828
Epoch 15/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0428 - accuracy: 0.9650 - val_loss: 0.3431 - val_accuracy: 0.6963
Epoch 16/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0402 - accuracy: 0.9673 - val_loss: 0.3460 - val_accuracy: 0.6963
Epoch 17/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0385 - accuracy: 0.9690 - val_loss: 0.3480 - val_accuracy: 0.6939
Epoch 18/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0378 - accuracy: 0.9696 - val_loss: 0.3369 - val_accuracy: 0.7114
Epoch 19/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0381 - accuracy: 0.9690 - val_loss: 0.3505 - val_accuracy: 0.6857
```

```
Epoch 20/20
214/214 [==============================] - 1s 3ms/step - loss: 0.0368 - accuracy: 0.9696 - val_loss: 0.3559 - val_accuracy: 0.6898
```



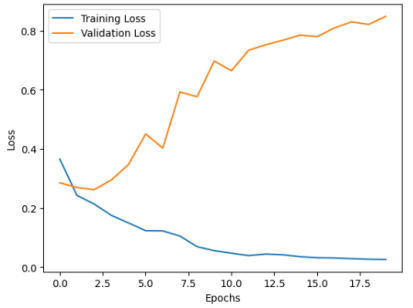



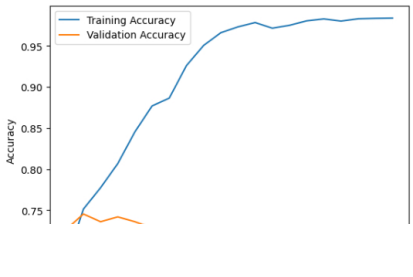

## RELU + RELU + RELU + RELU + TANH

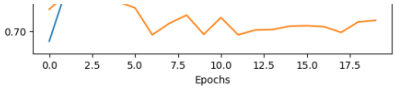
[84]:

```
input_dim = X_train.shape[1]

model = keras.Sequential()
model.add(layers.Dense(128, input_dim=input_dim, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(6, activation='tanh'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

history = model.fit(X_train, y_train_encoded, epochs=20, batch_size=32, validation_split=0.2)

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
Epoch 1/20
214/214 [==============================] - 2s 5ms/step - loss: 0.3650 - accuracy: 0.6878 - val_loss: 0.2851 - val_accuracy: 0.7266
Epoch 2/20
214/214 [==============================] - 1s 4ms/step - loss: 0.2423 - accuracy: 0.7513 - val_loss: 0.2691 - val_accuracy: 0.7453
Epoch 3/20
214/214 [==============================] - 1s 4ms/step - loss: 0.2132 - accuracy: 0.7773 - val_loss: 0.2618 - val_accuracy: 0.7360
Epoch 4/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1750 - accuracy: 0.8067 - val_loss: 0.2945 - val_accuracy: 0.7418
Epoch 5/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1493 - accuracy: 0.8454 - val_loss: 0.3468 - val_accuracy: 0.7360
Epoch 6/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1231 - accuracy: 0.8769 - val_loss: 0.4503 - val_accuracy: 0.7284
Epoch 7/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1224 - accuracy: 0.8864 - val_loss: 0.4027 - val_accuracy: 0.6957
Epoch 8/20
214/214 [==============================] - 1s 4ms/step - loss: 0.1048 - accuracy: 0.9260 - val_loss: 0.5922 - val_accuracy: 0.7097
Epoch 9/20
214/214 [==============================] - 1s 4ms/step - loss: 0.0689 - accuracy: 0.9505 - val_loss: 0.5763 - val_accuracy: 0.7196
Epoch 10/20
214/214 [==============================] - 1s 4ms/step - loss: 0.0553 - accuracy: 0.9660 - val_loss: 0.6969 - val_accuracy: 0.6963
Epoch 11/20
214/214 [==============================] - 1s 4ms/step - loss: 0.0469 - accuracy: 0.9731 - val_loss: 0.6640 - val_accuracy: 0.7167
Epoch 12/20
214/214 [==============================] - 1s 4ms/step - loss: 0.0389 - accuracy: 0.9784 - val_loss: 0.7335 - val_accuracy: 0.6957
Epoch 13/20
214/214 [==============================] - 1s 4ms/step - loss: 0.0440 - accuracy: 0.9715 - val_loss: 0.7520 - val_accuracy: 0.7015
Epoch 14/20
214/214 [==============================] - 1s 4ms/step - loss: 0.0416 - accuracy: 0.9750 - val_loss: 0.7673 - val_accuracy: 0.7021
Epoch 15/20
214/214 [==============================] - 1s 4ms/step - loss: 0.0350 - accuracy: 0.9804 - val_loss: 0.7850 - val_accuracy: 0.7062
Epoch 16/20
214/214 [==============================] - 1s 4ms/step - loss: 0.0317 - accuracy: 0.9828 - val_loss: 0.7794 - val_accuracy: 0.7068
Epoch 17/20
214/214 [==============================] - 1s 4ms/step - loss: 0.0308 - accuracy: 0.9801 - val_loss: 0.8090 - val_accuracy: 0.7056
Epoch 18/20
214/214 [==============================] - 1s 4ms/step - loss: 0.0284 - accuracy: 0.9829 - val_loss: 0.8297 - val_accuracy: 0.6986
Epoch 19/20
214/214 [==============================] - 1s 4ms/step - loss: 0.0264 - accuracy: 0.9835 - val_loss: 0.8210 - val_accuracy: 0.7114
Epoch 20/20
214/214 [==============================] - 1s 4ms/step - loss: 0.0254 - accuracy: 0.9838 - val_loss: 0.8482 - val_accuracy: 0.7132
```






## BOW + Neural Networks

[89]:
```
df1.head()
```

[89]:

| | Review | Label | text_length | word_count |
|---|---|---|---|---|
| 0 | good science great explanations simpl recip tl... | 4 | 694 | 72 |
| 1 | incred informative definit appropri beginners ... | 5 | 298 | 27 |
| 2 | awesom cours interest practic exercises | 5 | 57 | 5 |
| 3 | good must engag | 5 | 24 | 3 |
| 4 | great class i 'm happi choic special take cour... | 5 | 235 | 18 |

[95]:
```
x1 = df1['Review']
y1 = df1['Label']
```

[96]:
```
bow['labelxyz']=df1.Label
```

[91]:
```
# Creating a word corpus for vectorization
corpus = []
for i in range(x1.shape[0]):
    corpus.append(x1.iloc[i])
```

[92]:
```
vectorizer_train = CountVectorizer(max_features=1000)
vocabulary_train = vectorizer_train.fit_transform(corpus)
bow = pd.DataFrame(vocabulary_train.toarray(), columns = vectorizer_train.get_feature_names_out())
```

[93]:
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

## Training & Testing

[97]:
```
bow_zero = bow[bow.labelxyz == 0]
bow_one = bow[bow.labelxyz == 1]
bow_two = bow[bow.labelxyz == 2]
bow_three = bow[bow.labelxyz == 3]
bow_four = bow[bow.labelxyz == 4]
bow_five = bow[bow.labelxyz == 5]
```

[98]:
```
X_train_zero = bow_zero.sample(frac=0.8, random_state=0)
X_test_zero = bow_zero.drop(X_train_zero.index)
X_train_one = bow_one.sample(frac=0.8, random_state=0)
X_test_one = bow_one.drop(X_train_one.index)
X_train_two = bow_two.sample(frac=0.8, random_state=0)
X_test_two = bow_two.drop(X_train_two.index)
X_train_three = bow_three.sample(frac=0.8, random_state=0)
X_test_three = bow_three.drop(X_train_three.index)
X_train_four = bow_four.sample(frac=0.8, random_state=0)
X_test_four = bow_four.drop(X_train_four.index)
X_train_five = bow_five.sample(frac=0.8, random_state=0)
X_test_five = bow_five.drop(X_train_five.index)
```

[99]:
```
X_train_df = pd.concat([X_train_zero, X_train_one, X_train_two, X_train_three, X_train_four, X_train_five], axis = 0)
bow_train = X_train_df.drop(['labelxyz'], axis = 1)
y_train1 = list(X_train_df.labelxyz)
X_test_df = pd.concat([X_test_zero, X_test_one, X_test_two, X_test_three, X_test_four, X_test_five], axis = 0)
bow_test = X_test_df.drop(['labelxyz'], axis = 1)
y_test1 = list(X_test_df.labelxyz)
```

```
# Convert labels to one-hot encoded format
y_train_encoded = to_categorical(y_train1, num_classes=6)
y_test_encoded = to_categorical(y_test1, num_classes=6)
```

[116]:
```
bow_train, y_train1 = np.array(bow_train), np.array(y_train1)
```

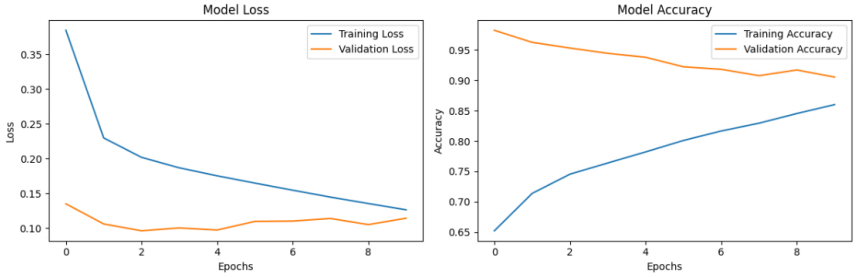
[117]:
```
bow_test, y_test1 = np.array(bow_test), np.array(y_test1)
```

## RELU + SIGMOID

[105]:
```
from sklearn.model_selection import train_test_split
import tensorflow as tf
from sklearn.feature_extraction.text import CountVectorizer

model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(bow_train.shape[1],)),
    tf.keras.layers.Dense(6, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

history = model.fit(bow_train, y_train_encoded, epochs=10, batch_size=32, validation_split=0.2)

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
```

```
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```

```
Epoch 1/10
214/214 [==============================] - 1s 4ms/step - loss: 0.3845 - accuracy: 0.6522 - val_loss: 0.1351 - val_accuracy: 0.9825
Epoch 2/10
214/214 [==============================] - 1s 3ms/step - loss: 0.2298 - accuracy: 0.7138 - val_loss: 0.1062 - val_accuracy: 0.9626
Epoch 3/10
214/214 [==============================] - 1s 3ms/step - loss: 0.2019 - accuracy: 0.7455 - val_loss: 0.0964 - val_accuracy: 0.9533
Epoch 4/10
214/214 [==============================] - 1s 3ms/step - loss: 0.1869 - accuracy: 0.7639 - val_loss: 0.1004 - val_accuracy: 0.9445
Epoch 5/10
214/214 [==============================] - 1s 3ms/step - loss: 0.1754 - accuracy: 0.7821 - val_loss: 0.0974 - val_accuracy: 0.9381
Epoch 6/10
214/214 [==============================] - 1s 3ms/step - loss: 0.1649 - accuracy: 0.8010 - val_loss: 0.1099 - val_accuracy: 0.9224
Epoch 7/10
214/214 [==============================] - 1s 3ms/step - loss: 0.1547 - accuracy: 0.8166 - val_loss: 0.1103 - val_accuracy: 0.9183
Epoch 8/10
214/214 [==============================] - 1s 3ms/step - loss: 0.1447 - accuracy: 0.8294 - val_loss: 0.1140 - val_accuracy: 0.9078
Epoch 9/10
214/214 [==============================] - 1s 3ms/step - loss: 0.1355 - accuracy: 0.8454 - val_loss: 0.1053 - val_accuracy: 0.9171
Epoch 10/10
214/214 [==============================] - 1s 3ms/step - loss: 0.1265 - accuracy: 0.8601 - val_loss: 0.1144 - val_accuracy: 0.9054
```

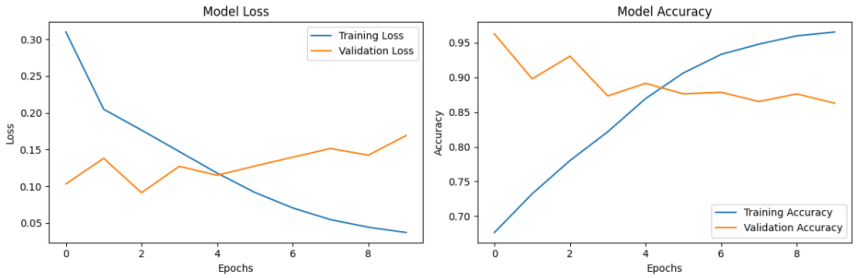



## RELU + RELU + SIGMOID

[110]:

```
from sklearn.model_selection import train_test_split
import tensorflow as tf
from sklearn.feature_extraction.text import CountVectorizer

model = tf.keras.Sequential([
    tf.keras.layers.Dense(128, activation='relu', input_shape=(bow_train.shape[1],)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(6, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

history = model.fit(bow_train, y_train_encoded, epochs=10, batch_size=32, validation_split=0.2)

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```

```
Epoch 1/10
214/214 [==============================] - 2s 5ms/step - loss: 0.3098 - accuracy: 0.6765 - val_loss: 0.1034 - val_accuracy: 0.9626
Epoch 2/10
214/214 [==============================] - 1s 4ms/step - loss: 0.2049 - accuracy: 0.7323 - val_loss: 0.1382 - val_accuracy: 0.8978
Epoch 3/10
214/214 [==============================] - 1s 4ms/step - loss: 0.1765 - accuracy: 0.7802 - val_loss: 0.0915 - val_accuracy: 0.9305
Epoch 4/10
214/214 [==============================] - 1s 4ms/step - loss: 0.1474 - accuracy: 0.8217 - val_loss: 0.1273 - val_accuracy: 0.8733
Epoch 5/10
214/214 [==============================] - 1s 4ms/step - loss: 0.1181 - accuracy: 0.8695 - val_loss: 0.1151 - val_accuracy: 0.8914
Epoch 6/10
214/214 [==============================] - 1s 4ms/step - loss: 0.0918 - accuracy: 0.9064 - val_loss: 0.1277 - val_accuracy: 0.8762
Epoch 7/10
214/214 [==============================] - 1s 4ms/step - loss: 0.0708 - accuracy: 0.9331 - val_loss: 0.1397 - val_accuracy: 0.8786
Epoch 8/10
214/214 [==============================] - 1s 4ms/step - loss: 0.0548 - accuracy: 0.9479 - val_loss: 0.1515 - val_accuracy: 0.8651
Epoch 9/10
214/214 [==============================] - 1s 4ms/step - loss: 0.0446 - accuracy: 0.9598 - val_loss: 0.1424 - val_accuracy: 0.8762
Epoch 10/10
214/214 [==============================] - 1s 4ms/step - loss: 0.0373 - accuracy: 0.9652 - val_loss: 0.1692 - val_accuracy: 0.8628
```

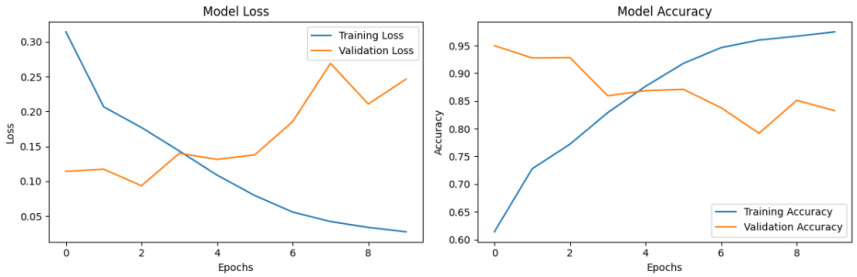



[111]:

```
from sklearn.model_selection import train_test_split
import tensorflow as tf
from sklearn.feature_extraction.text import CountVectorizer

model = tf.keras.Sequential([
    tf.keras.layers.Dense(128, activation='relu', input_shape=(bow_train.shape[1],)),
    tf.keras.layers.Dense(64, activation='relu'),
     tf.keras.layers.Dense(32, activation='relu'),
```

```
    tf.keras.layers.Dense(16, activation='relu'),
    tf.keras.layers.Dense(6, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

history = model.fit(bow_train, y_train_encoded, epochs=10, batch_size=32, validation_split=0.2)

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```

```
Epoch 1/10
214/214 [==============================] - 2s 5ms/step - loss: 0.3140 - accuracy: 0.6139 - val_loss: 0.1142 - val_accuracy: 0.9498
Epoch 2/10
214/214 [==============================] - 1s 4ms/step - loss: 0.2066 - accuracy: 0.7277 - val_loss: 0.1173 - val_accuracy: 0.9276
Epoch 3/10
214/214 [==============================] - 1s 4ms/step - loss: 0.1770 - accuracy: 0.7721 - val_loss: 0.0935 - val_accuracy: 0.9282
Epoch 4/10
214/214 [==============================] - 1s 4ms/step - loss: 0.1434 - accuracy: 0.8290 - val_loss: 0.1400 - val_accuracy: 0.8593
Epoch 5/10
214/214 [==============================] - 1s 4ms/step - loss: 0.1087 - accuracy: 0.8765 - val_loss: 0.1313 - val_accuracy: 0.8687
Epoch 6/10
214/214 [==============================] - 1s 4ms/step - loss: 0.0794 - accuracy: 0.9178 - val_loss: 0.1378 - val_accuracy: 0.8710
Epoch 7/10
214/214 [==============================] - 1s 4ms/step - loss: 0.0558 - accuracy: 0.9463 - val_loss: 0.1856 - val_accuracy: 0.8377
Epoch 8/10
214/214 [==============================] - 1s 4ms/step - loss: 0.0422 - accuracy: 0.9598 - val_loss: 0.2689 - val_accuracy: 0.7916
Epoch 9/10
214/214 [==============================] - 1s 4ms/step - loss: 0.0338 - accuracy: 0.9669 - val_loss: 0.2107 - val_accuracy: 0.8511
Epoch 10/10
214/214 [==============================] - 1s 4ms/step - loss: 0.0275 - accuracy: 0.9746 - val_loss: 0.2464 - val_accuracy: 0.8325
```




## BOW + LSTM

```
from keras.utils import to_categorical

# Convert labels to one-hot encoded format
y_train_encoded = to_categorical(y_train1, num_classes=6)
y_test_encoded = to_categorical(y_test1, num_classes=6)
# Creating the LSTM model
model = Sequential()
model.add(LSTM(64, input_shape=(bow_train.shape[1], 1)))
model.add(Dense(6, activation='sigmoid'))

# Compiling the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Training the model
history = model.fit(bow_train, y_train_encoded, epochs=10, batch_size=32, validation_data=(bow_test, y_test_encoded))
```

```
Epoch 1/10
268/268 [==============================] - 133s 487ms/step - loss: 0.2694 - accuracy: 0.7373 - val_loss: 0.2386 - val_accuracy: 0.7401
Epoch 2/10
268/268 [==============================] - 131s 488ms/step - loss: 0.2391 - accuracy: 0.7399 - val_loss: 0.2388 - val_accuracy: 0.7401
Epoch 3/10
268/268 [==============================] - 131s 488ms/step - loss: 0.2392 - accuracy: 0.7399 - val_loss: 0.2389 - val_accuracy: 0.7401
Epoch 4/10
268/268 [==============================] - 130s 487ms/step - loss: 0.2393 - accuracy: 0.7399 - val_loss: 0.2387 - val_accuracy: 0.7401
Epoch 5/10
268/268 [==============================] - 130s 484ms/step - loss: 0.2391 - accuracy: 0.7399 - val_loss: 0.2389 - val_accuracy: 0.7401
Epoch 6/10
268/268 [==============================] - 130s 486ms/step - loss: 0.2387 - accuracy: 0.7399 - val_loss: 0.2397 - val_accuracy: 0.7401
Epoch 7/10
268/268 [==============================] - 130s 485ms/step - loss: 0.2389 - accuracy: 0.7399 - val_loss: 0.2382 - val_accuracy: 0.7401
Epoch 8/10
213/268 [======================>.......] - ETA: 24s - loss: 0.2382 - accuracy: 0.7408
```

[ ]:

```
# Plotting accuracy and loss curves
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Test'], loc='upper right')
plt.show()
```