BOSTON UNIVERSITY

MET CS-688: Web Analytics
Term Project

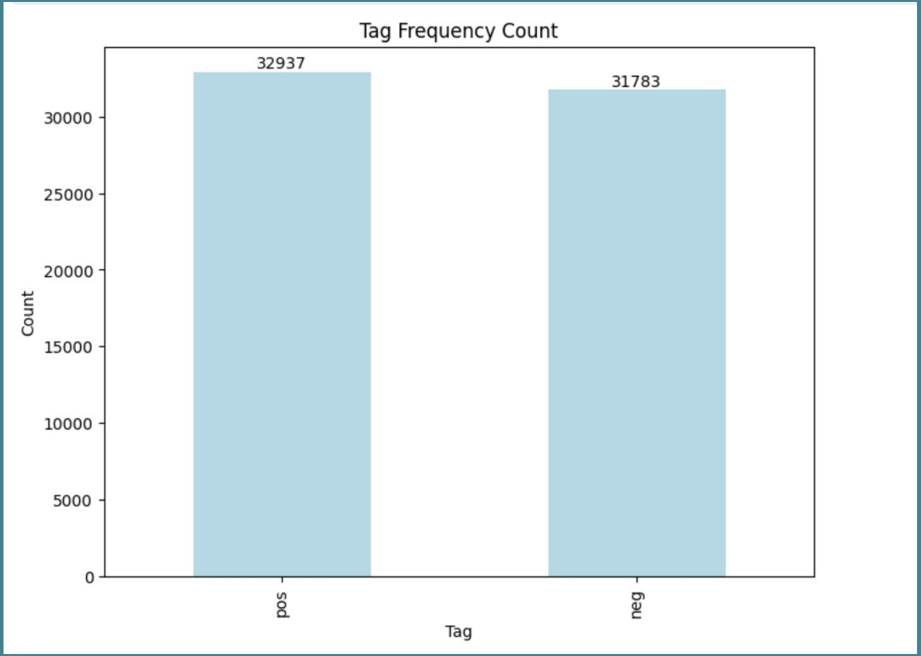
Sentiment Analysis BERT Model Implementation

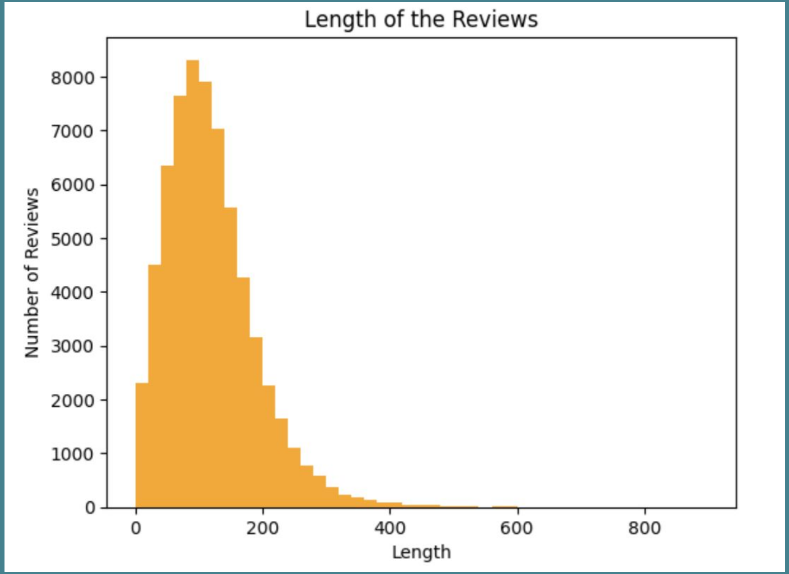
Vaidehi Shah (U90080562)

DATASET

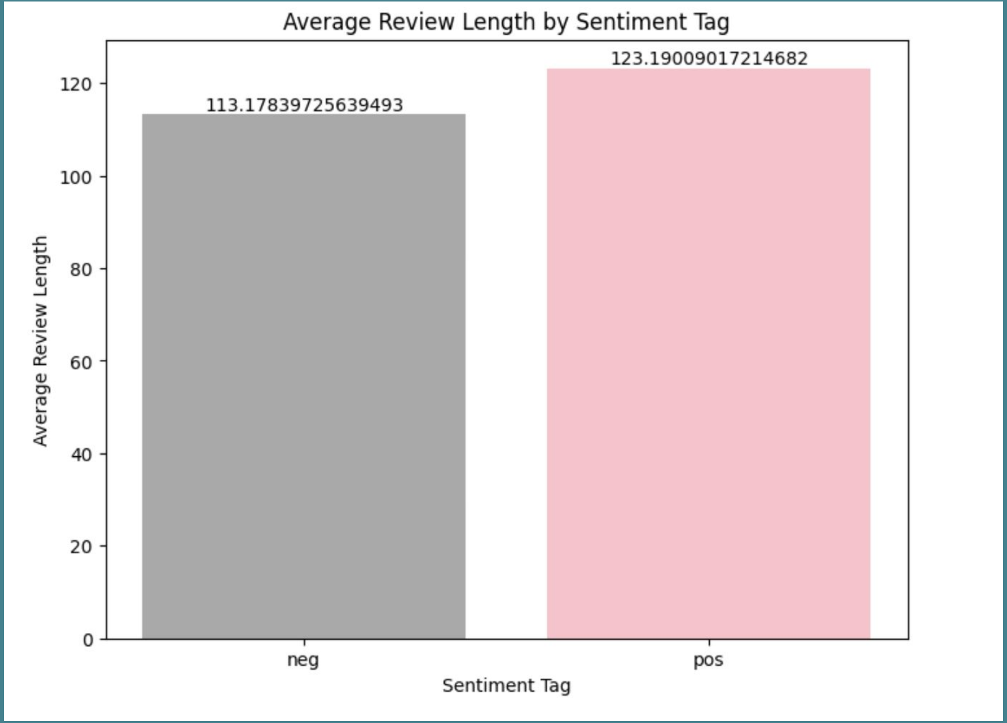
I have selected the "Movie Reviews" dataset available on Kaggle for sentiment analysis. The dataset consists of text-based movie reviews with a corresponding "tag" that categorizes each review as either pos for "positive" or neg for "negative," representing the true sentiment classification of the data according to an automatic rating classifier. The goal of this project is to build a model using natural language processing techniques to accurately predict the sentiment of movie reviews as positive or negative.

EXPLORATORY DATA ANALYSIS

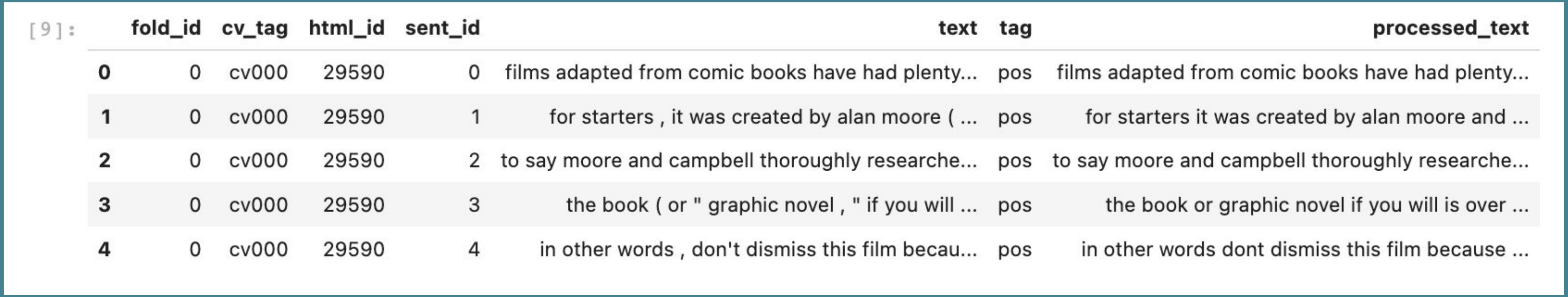




EXPLORATORY DATA ANALYSIS

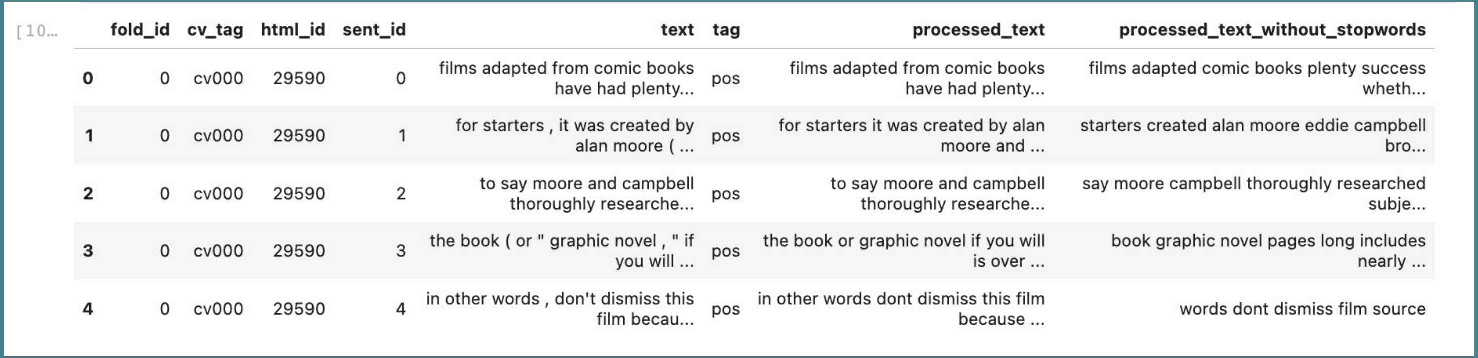


PREPROCESSING DATA



[9]:

	fold_id	cv_tag	html_id	sent_id	text	tag	processed_text
0	0	cv000	29590	0	films adapted from comic books have had plenty...	pos	films adapted from comic books have had plenty...
1	0	cv000	29590	1	for starters , it was created by alan moore (...	pos	for starters it was created by alan moore and ...
2	0	cv000	29590	2	to say moore and campbell thoroughly researche...	pos	to say moore and campbell thoroughly researche...
3	0	cv000	29590	3	the book (or " graphic novel , " if you will ...	pos	the book or graphic novel if you will is over ...
4	0	cv000	29590	4	in other words , don't dismiss this film becau...	pos	in other words dont dismiss this film because ...

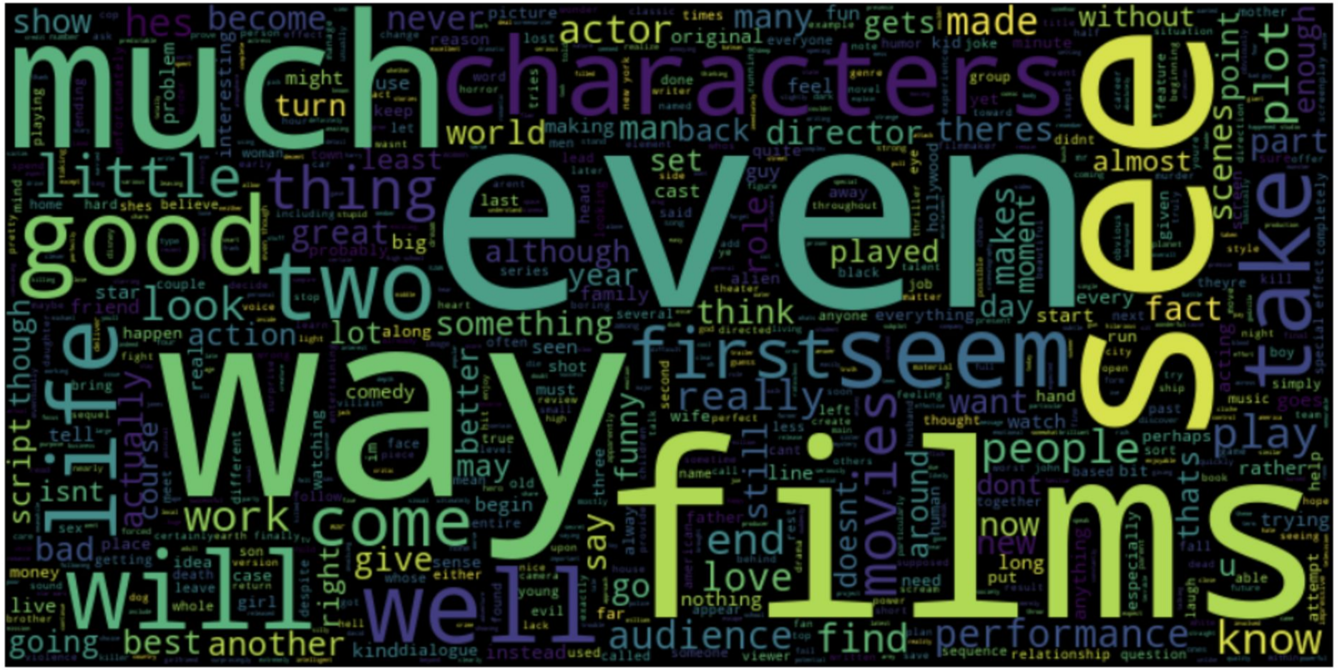


[10...

	fold_id	cv_tag	html_id	sent_id	text	tag	processed_text	processed_text_without_stopwords
0	0	cv000	29590	0	films adapted from comic books have had plenty...	pos	films adapted from comic books have had plenty...	films adapted comic books plenty success wheth...
1	0	cv000	29590	1	for starters , it was created by alan moore (...	pos	for starters it was created by alan moore and ...	starters created alan moore eddie campbell bro...
2	0	cv000	29590	2	to say moore and campbell thoroughly researche...	pos	to say moore and campbell thoroughly researche...	say moore campbell thoroughly researched subje...
3	0	cv000	29590	3	the book (or " graphic novel , " if you will ...	pos	the book or graphic novel if you will is over ...	book graphic novel pages long includes nearly ...
4	0	cv000	29590	4	in other words , don't dismiss this film becau...	pos	in other words dont dismiss this film because ...	words dont dismiss film source

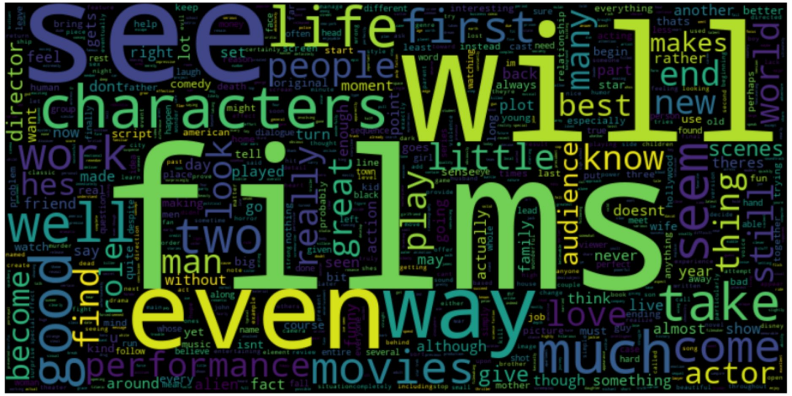
WORD CLOUDS

```
[22]: plot_wordcloud(df, "processed_text")
```

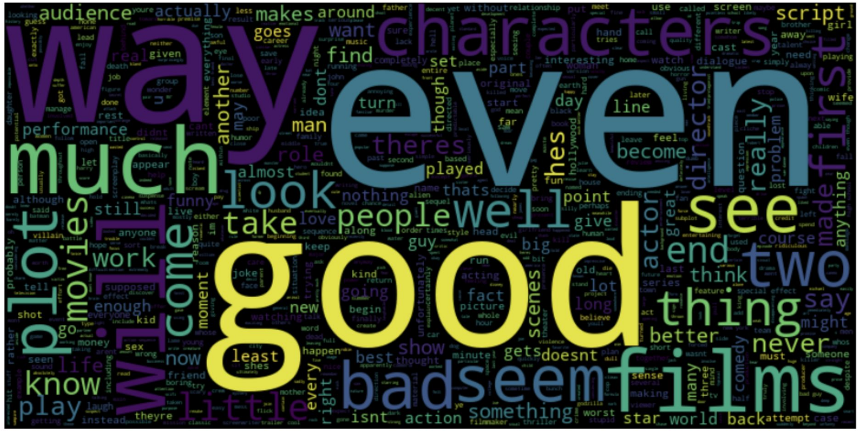


WORD CLOUDS

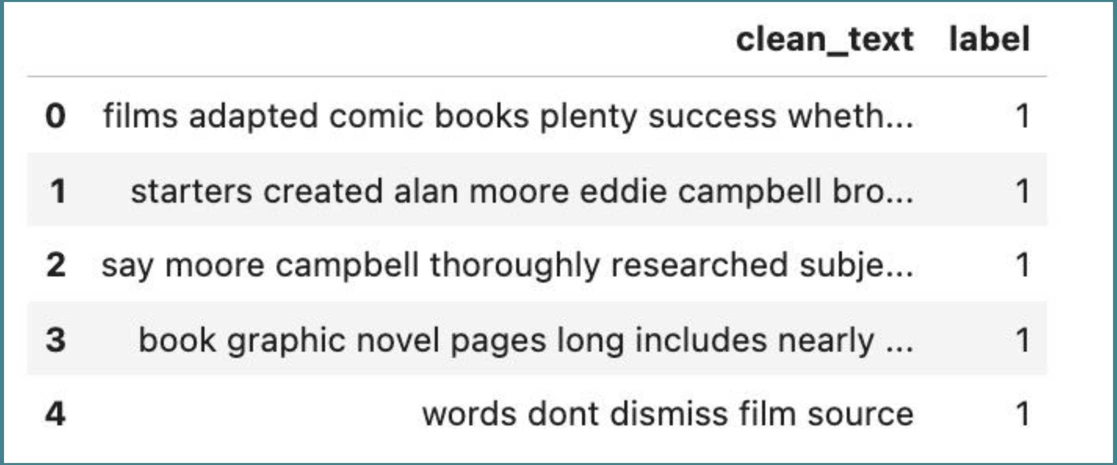
```
[23]: plot_wordcloud(df_positive, "processed_text")
```



```
plot_wordcloud(df_negative, "processed_text")
```

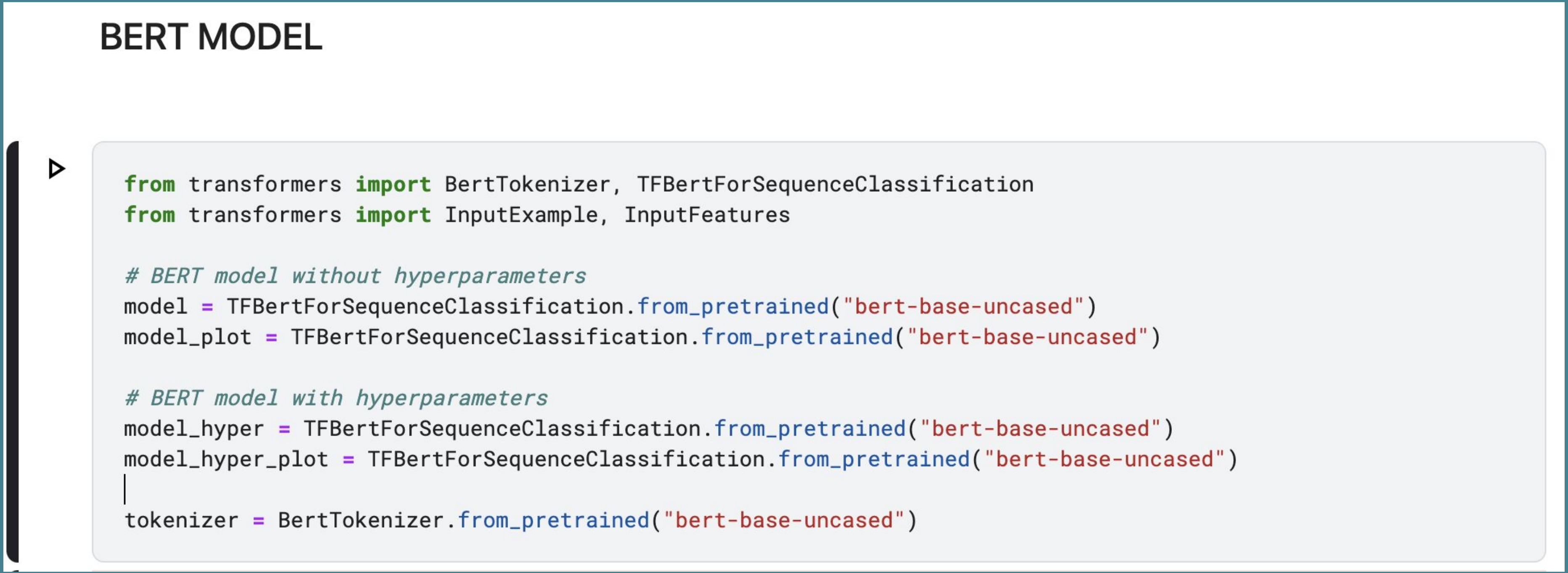


PROCESSED DATAFRAME



	clean_text	label
0	films adapted comic books plenty success wheth...	1
1	starters created alan moore eddie campbell bro...	1
2	say moore campbell thoroughly researched subje...	1
3	book graphic novel pages long includes nearly ...	1
4	words dont dismiss film source	1

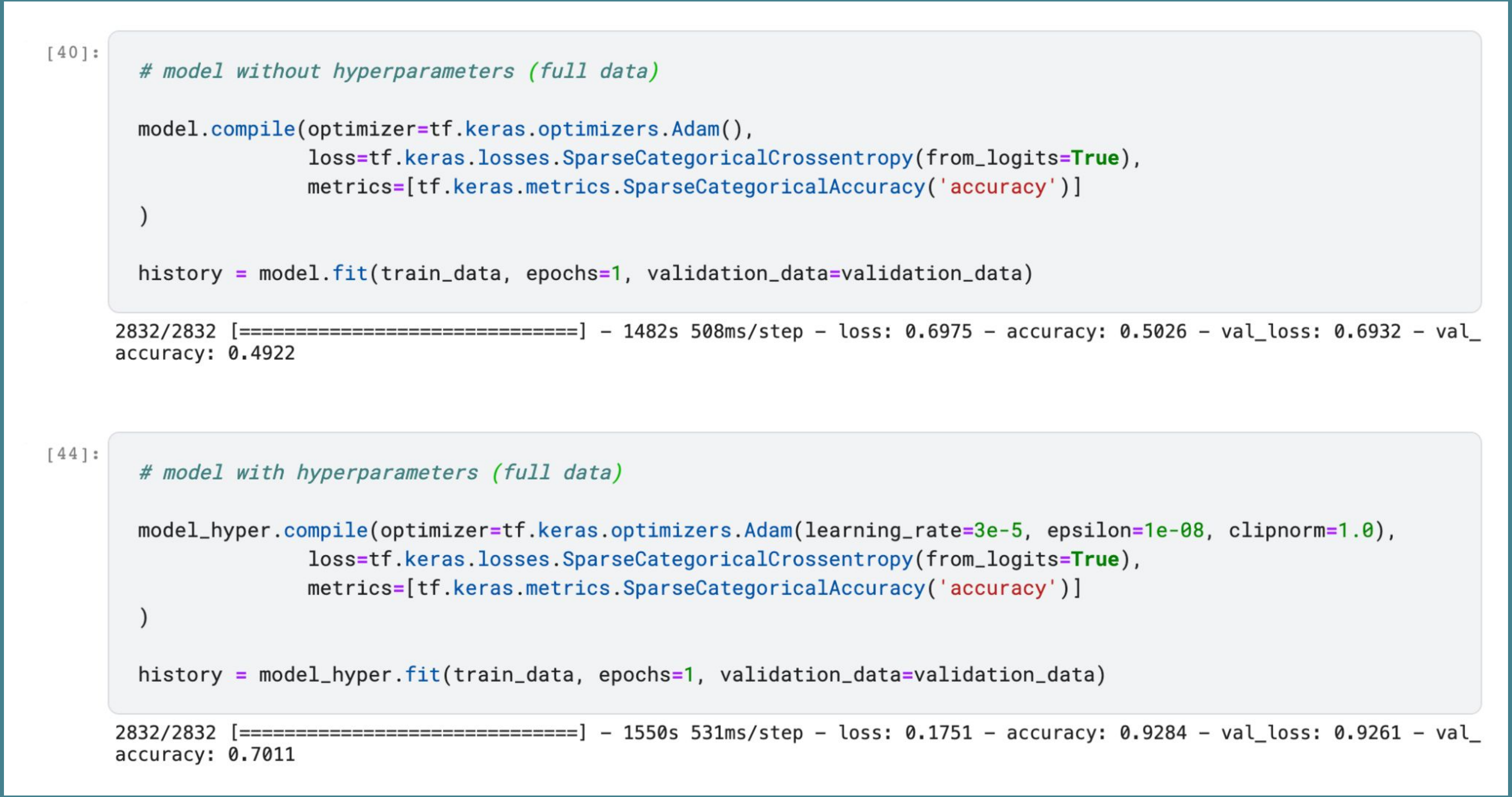
BERT MODEL IMPLEMENTATION



BERT MODEL

```python
from transformers import BertTokenizer, TFBertForSequenceClassification
from transformers import InputExample, InputFeatures

# BERT model without hyperparameters
model = TFBertForSequenceClassification.from_pretrained("bert-base-uncased")
model_plot = TFBertForSequenceClassification.from_pretrained("bert-base-uncased")

# BERT model with hyperparameters
model_hyper = TFBertForSequenceClassification.from_pretrained("bert-base-uncased")
model_hyper_plot = TFBertForSequenceClassification.from_pretrained("bert-base-uncased")

tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
```

UNCASED BERT MODEL



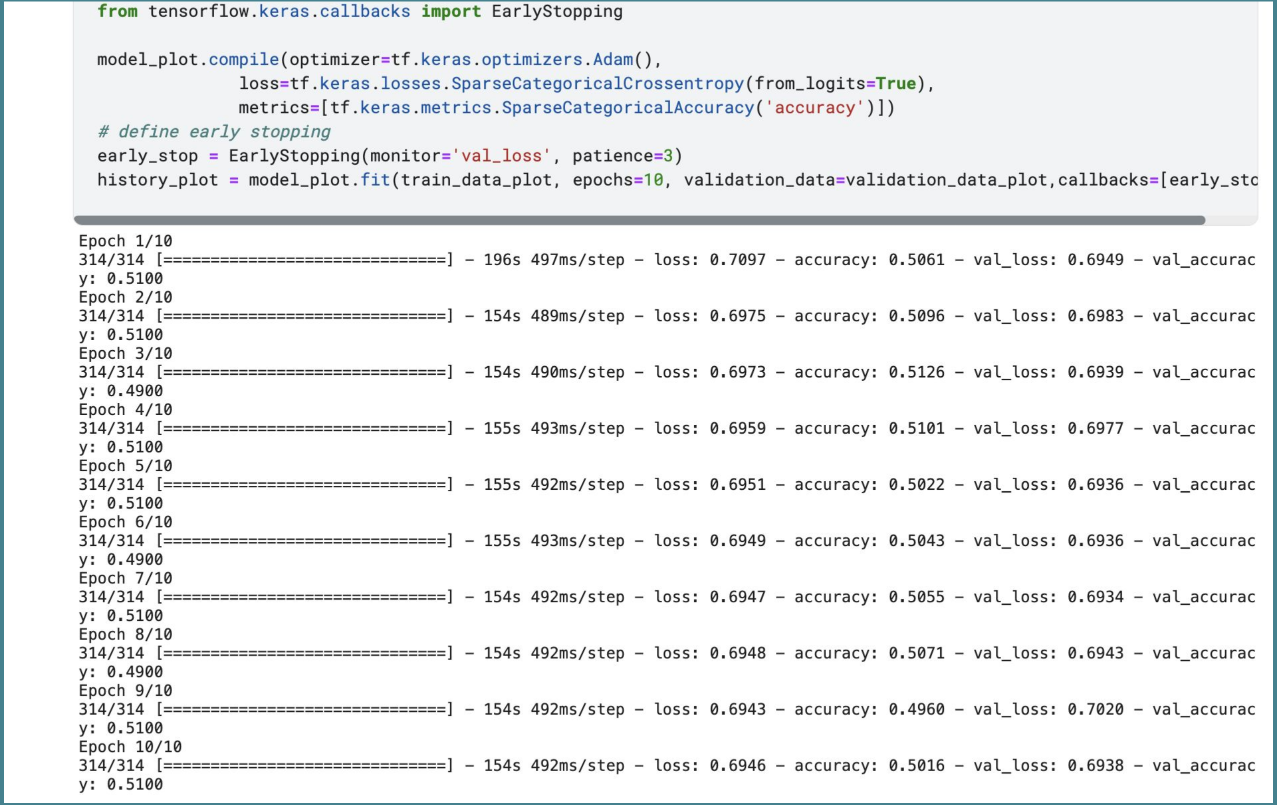
```
[40]:
# model without hyperparameters (full data)

model.compile(optimizer=tf.keras.optimizers.Adam(),
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=[tf.keras.metrics.SparseCategoricalAccuracy('accuracy')]
)

history = model.fit(train_data, epochs=1, validation_data=validation_data)
```

```
2832/2832 [==============================] - 1482s 508ms/step - loss: 0.6975 - accuracy: 0.5026 - val_loss: 0.6932 - val_
accuracy: 0.4922
```

```
[44]:
# model with hyperparameters (full data)

model_hyper.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=3e-5, epsilon=1e-08, clipnorm=1.0),
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=[tf.keras.metrics.SparseCategoricalAccuracy('accuracy')]
)

history = model_hyper.fit(train_data, epochs=1, validation_data=validation_data)
```

```
2832/2832 [==============================] - 1550s 531ms/step - loss: 0.1751 - accuracy: 0.9284 - val_loss: 0.9261 - val_
accuracy: 0.7011
```

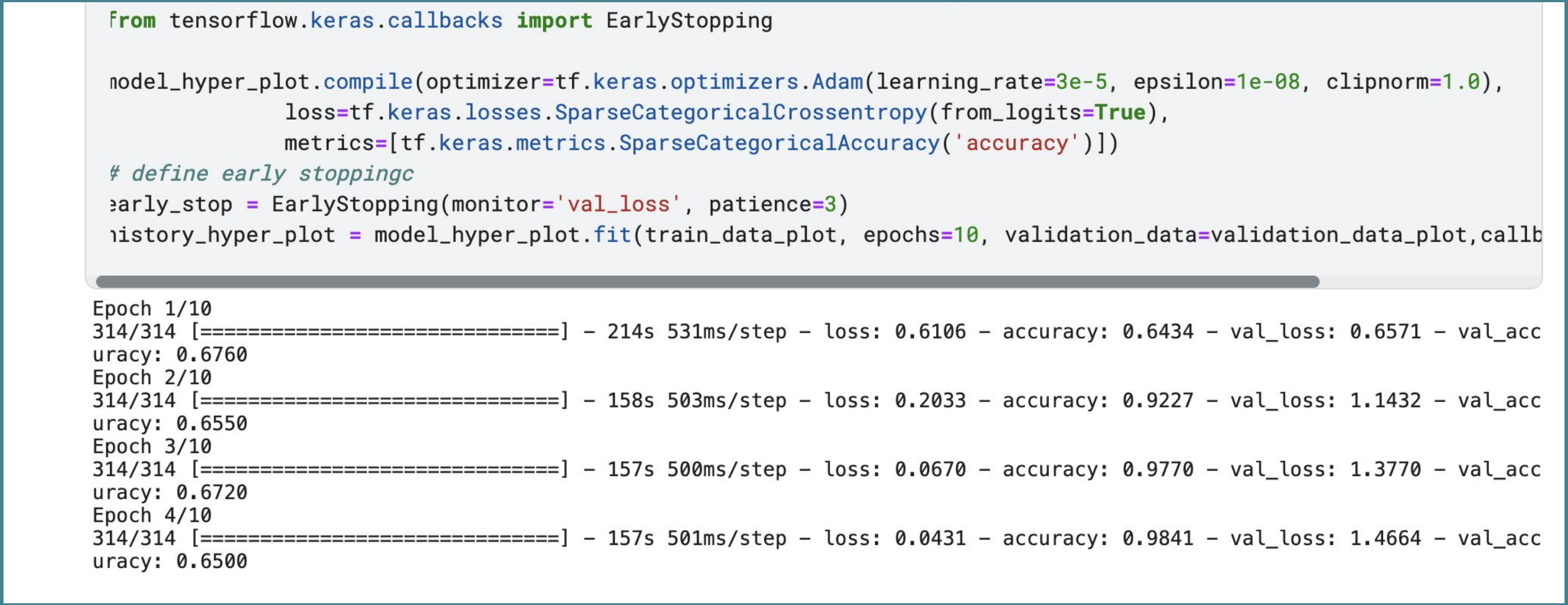
MODEL W/O HYPERPARAMETERS WITH 10 EPOCHS



```
from tensorflow.keras.callbacks import EarlyStopping

model_plot.compile(optimizer=tf.keras.optimizers.Adam(),
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=[tf.keras.metrics.SparseCategoricalAccuracy('accuracy')])
# define early stopping
early_stop = EarlyStopping(monitor='val_loss', patience=3)
history_plot = model_plot.fit(train_data_plot, epochs=10, validation_data=validation_data_plot,callbacks=[early_sto
```

```
Epoch 1/10
314/314 [==============================] - 196s 497ms/step - loss: 0.7097 - accuracy: 0.5061 - val_loss: 0.6949 - val_accurac
y: 0.5100
Epoch 2/10
314/314 [==============================] - 154s 489ms/step - loss: 0.6975 - accuracy: 0.5096 - val_loss: 0.6983 - val_accurac
y: 0.5100
Epoch 3/10
314/314 [==============================] - 154s 490ms/step - loss: 0.6973 - accuracy: 0.5126 - val_loss: 0.6939 - val_accurac
y: 0.4900
Epoch 4/10
314/314 [==============================] - 155s 493ms/step - loss: 0.6959 - accuracy: 0.5101 - val_loss: 0.6977 - val_accurac
y: 0.5100
Epoch 5/10
314/314 [==============================] - 155s 492ms/step - loss: 0.6951 - accuracy: 0.5022 - val_loss: 0.6936 - val_accurac
y: 0.5100
Epoch 6/10
314/314 [==============================] - 155s 493ms/step - loss: 0.6949 - accuracy: 0.5043 - val_loss: 0.6936 - val_accurac
y: 0.4900
Epoch 7/10
314/314 [==============================] - 154s 492ms/step - loss: 0.6947 - accuracy: 0.5055 - val_loss: 0.6934 - val_accurac
y: 0.5100
Epoch 8/10
314/314 [==============================] - 154s 492ms/step - loss: 0.6948 - accuracy: 0.5071 - val_loss: 0.6943 - val_accurac
y: 0.4900
Epoch 9/10
314/314 [==============================] - 154s 492ms/step - loss: 0.6943 - accuracy: 0.4960 - val_loss: 0.7020 - val_accurac
y: 0.5100
Epoch 10/10
314/314 [==============================] - 154s 492ms/step - loss: 0.6946 - accuracy: 0.5016 - val_loss: 0.6938 - val_accurac
y: 0.5100
```

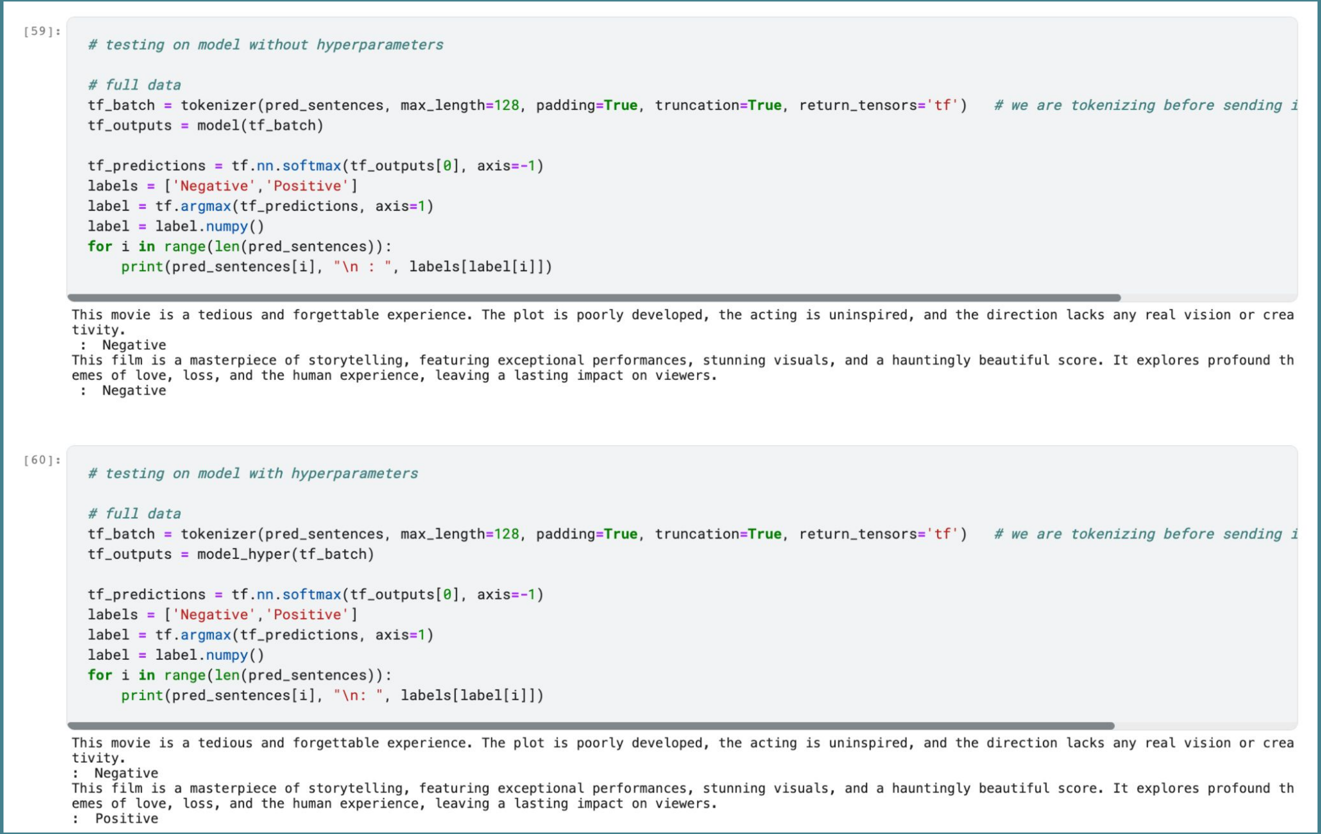
MODEL W/ HYPERPARAMETERS WITH 10 EPOCHS



```
from tensorflow.keras.callbacks import EarlyStopping

nodel_hyper_plot.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=3e-5, epsilon=1e-08, clipnorm=1.0),
             loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
             metrics=[tf.keras.metrics.SparseCategoricalAccuracy('accuracy')])
# define early stoppingc
early_stop = EarlyStopping(monitor='val_loss', patience=3)
nistory_hyper_plot = model_hyper_plot.fit(train_data_plot, epochs=10, validation_data=validation_data_plot,callb
```

```
Epoch 1/10
314/314 [==============================] - 214s 531ms/step - loss: 0.6106 - accuracy: 0.6434 - val_loss: 0.6571 - val_acc
uracy: 0.6760
Epoch 2/10
314/314 [==============================] - 158s 503ms/step - loss: 0.2033 - accuracy: 0.9227 - val_loss: 1.1432 - val_acc
uracy: 0.6550
Epoch 3/10
314/314 [==============================] - 157s 500ms/step - loss: 0.0670 - accuracy: 0.9770 - val_loss: 1.3770 - val_acc
uracy: 0.6720
Epoch 4/10
314/314 [==============================] - 157s 501ms/step - loss: 0.0431 - accuracy: 0.9841 - val_loss: 1.4664 - val_acc
uracy: 0.6500
```

RESULTS



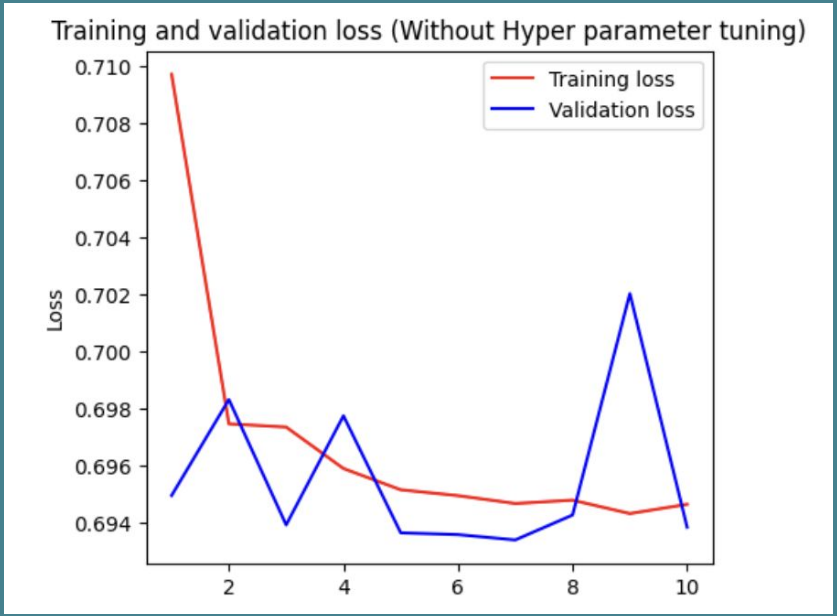
```
[59]:
# testing on model without hyperparameters

# full data
tf_batch = tokenizer(pred_sentences, max_length=128, padding=True, truncation=True, return_tensors='tf')   # we are tokenizing before sending i
tf_outputs = model(tf_batch)

tf_predictions = tf.nn.softmax(tf_outputs[0], axis=-1)
labels = ['Negative','Positive']
label = tf.argmax(tf_predictions, axis=1)
label = label.numpy()
for i in range(len(pred_sentences)):
    print(pred_sentences[i], "\n : ", labels[label[i]])
```

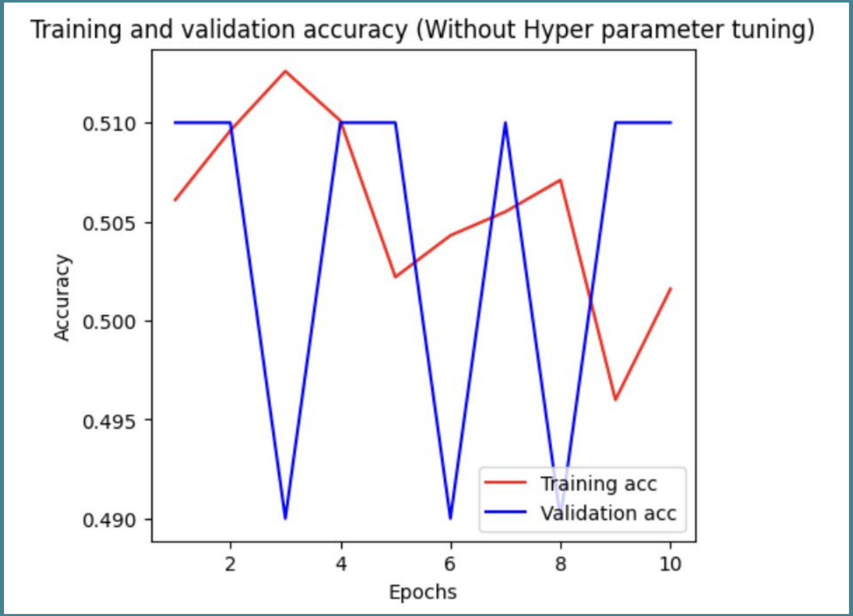
```
This movie is a tedious and forgettable experience. The plot is poorly developed, the acting is uninspired, and the direction lacks any real vision or crea
tivity.
 :  Negative
This film is a masterpiece of storytelling, featuring exceptional performances, stunning visuals, and a hauntingly beautiful score. It explores profound th
emes of love, loss, and the human experience, leaving a lasting impact on viewers.
 :  Negative
```

```
[60]:
# testing on model with hyperparameters

# full data
tf_batch = tokenizer(pred_sentences, max_length=128, padding=True, truncation=True, return_tensors='tf')   # we are tokenizing before sending i
tf_outputs = model_hyper(tf_batch)

tf_predictions = tf.nn.softmax(tf_outputs[0], axis=-1)
labels = ['Negative','Positive']
label = tf.argmax(tf_predictions, axis=1)
label = label.numpy()
for i in range(len(pred_sentences)):
    print(pred_sentences[i], "\n: ", labels[label[i]])
```

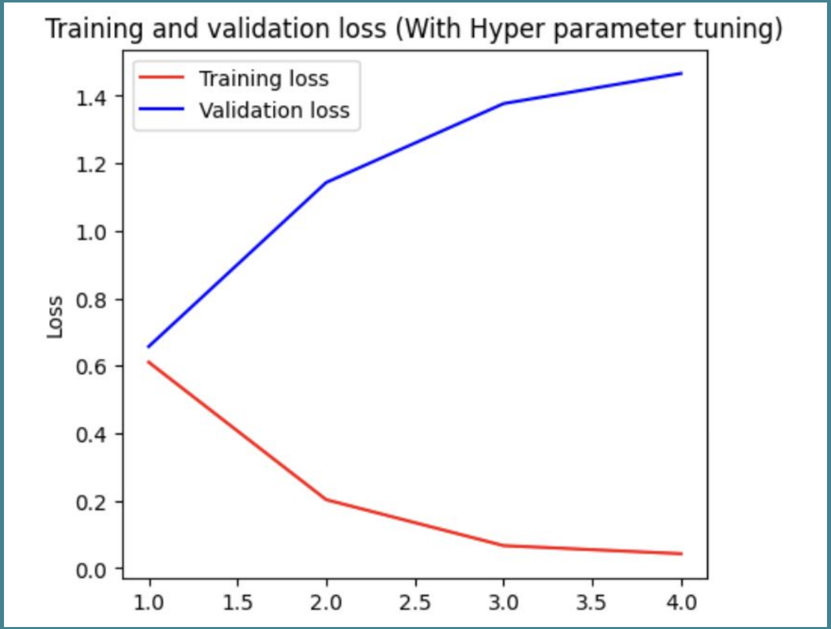
```
This movie is a tedious and forgettable experience. The plot is poorly developed, the acting is uninspired, and the direction lacks any real vision or crea
tivity.
:  Negative
This film is a masterpiece of storytelling, featuring exceptional performances, stunning visuals, and a hauntingly beautiful score. It explores profound th
emes of love, loss, and the human experience, leaving a lasting impact on viewers.
:  Positive
```

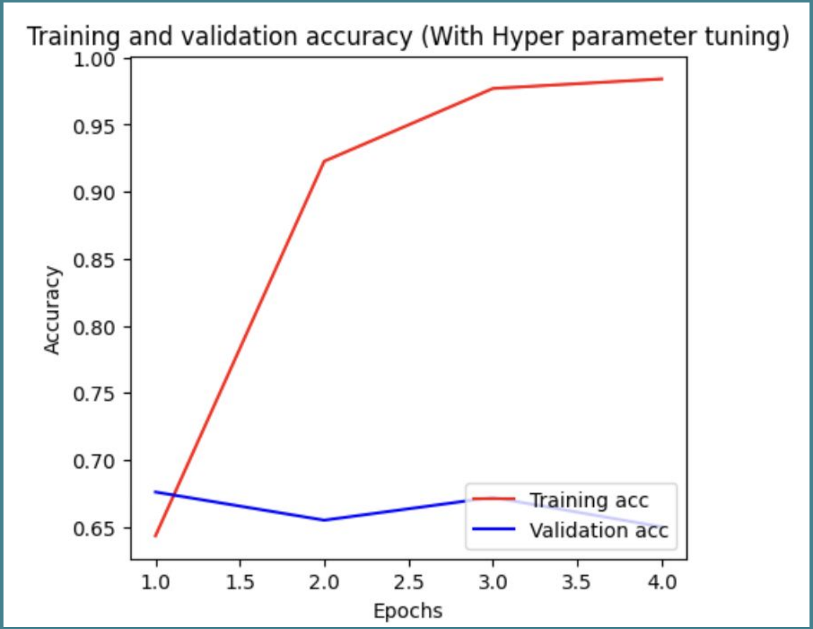
GRAPHS FOR MODEL WITHOUT HYPERPARAMETER





GRAPHS FOR MODEL WITH HYPERPARAMETER





CONCLUSION

- ➔ This project on sentiment analysis of movie reviews using BERT model involved preprocessing the data, exploring the dataset, training and testing the model, and analyzing the results.
- ➔ The model trained with hyperparameters achieved a significantly higher accuracy of 92%, while the model without hyperparameters had a low accuracy of 50%. Although there was overfitting in the model with hyperparameters, it gave more accurate results in predicting the sentiment of the movie reviews.
- ➔ This project demonstrates the effectiveness of natural language processing techniques and pre-trained models in sentiment analysis tasks.

THANK YOU!