In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

In [ ]:

In [2]:

```
import pandas as pd

# Read the CSV file into a DataFrame
data = pd.read_csv('your_file.csv')

from sklearn.preprocessing import LabelEncoder

# Initialize the LabelEncoder
label_encoder = LabelEncoder()

# Fit and transform the 'sentiment' column
data['sentiment_encoded'] = label_encoder.fit_transform(data['Sentiment'])

# Display the DataFrame with the encoded sentiment column
data
```

Out[2]:

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes	Sentiment	Tokenized	Without_Stop
0	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	5	i feel so lucky to have found this used (phone...	1.0	positive	['i', 'feel', 'so', 'lucky', 'to', 'have', 'fo...	['feel', 'found', 'us 'p
1	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	nice phone, nice up grade from my pantach revu...	0.0	positive	['nice', 'phone', ',', 'nice', 'up', 'grade', ...	['nice', 'pho 'nice', '
2	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	5	very pleased	0.0	positive	['very', 'pleased']	['pl
3	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	it works good but it goes slow sometimes but i...	0.0	positive	['it', 'works', 'good', 'but', 'it', 'goes', '...	['works', 'goes', 'somet
4	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	great phone to replace my lost phone. the only...	0.0	positive	['great', 'phone', 'to', 'replace', 'my', 'los...	['great', 'p 'replace 'ph
...	...	...	...	...	...	...	...	...	
349465	samsung convoy u640 phone for verizon wireless...	samsung	79.95	5	great phone. large keys, best flip phone i hav...	0.0	positive	['great', 'phone', '.', 'large', 'keys', ',', ...	['great', 'pho 'large', 'key
349466	samsung convoy u640 phone for verizon wireless...	samsung	79.95	5	pros...works great, very durable, easy to navi...	0.0	positive	['pros', '...', 'works', 'great', ',', 'very',...	['pros', '...', ' 'great', ',', 'c
349467	samsung convoy	samsung	79.95	5	just as described	0.0	positive	['just', 'as', 'described',	['described', 'p

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes	Sentiment	Tokenized	Without_Stop
	u640 phone for verizon wireless...				perfect for the price			'perfect', 'for', ...	
349468	samsung convoy u640 phone for verizon wireless...	samsung	79.95	1	would not work	0.0	negative	['would', 'not', 'work']	['would',
349469	samsung convoy u640 phone for verizon wireless...	samsung	79.95	3	speaker phone doesn't work, but phone works good	0.0	neutral	['speaker', 'phone', 'does', "n't", 'work', ',...	['speaker', 'p "n't", 'w

349470 rows × 12 columns

In [3]:
```
data.head()
```

Out[3]:

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes	Sentiment	Tokenized	Without_Stopwords
0	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	5	i feel so lucky to have found this used (phone...	1.0	positive	['i', 'feel', 'so', 'lucky', 'to', 'have', 'fo...	['feel', 'lucky', 'found', 'used', '(', 'phone...
1	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	nice phone, nice up grade from my pantach revu...	0.0	positive	['nice', 'phone', ',', 'nice', 'up', 'grade', ...	['nice', 'phone', ',', 'nice', 'grade', 'panta...
2	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	5	very pleased	0.0	positive	['very', 'pleased']	['pleased']
3	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	it works good but it goes slow sometimes but i...	0.0	positive	['it', 'works', 'good', 'but', 'it', 'goes', '...	['works', 'good', 'goes', 'slow', 'sometimes',...
4	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	great phone to replace my lost phone. the only...	0.0	positive	['great', 'phone', 'to', 'replace', 'my', 'los...	['great', 'phone', 'replace', 'lost', 'phone',...

In [4]:

```
df = data[['Reviews', 'sentiment_encoded']]
df.head()
```

Out[4]:

	Reviews	sentiment_encoded
0	i feel so lucky to have found this used (phone...	2
1	nice phone, nice up grade from my pantach revu...	2
2	very pleased	2
3	it works good but it goes slow sometimes but i...	2
4	great phone to replace my lost phone. the only...	2

In [5]:
```
df.shape
```

Out[5]:
```
(349470, 2)
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 349470 entries, 0 to 349469
Data columns (total 2 columns):
 #   Column             Non-Null Count   Dtype 
---  ------             --------------   ----- 
 0   Reviews            349470 non-null  object
 1   sentiment_encoded  349470 non-null  int32 
dtypes: int32(1), object(1)
memory usage: 4.0+ MB
```

In [7]:
```
#Test and Train dataframes
```

In [8]:
```
%%time
import pandas as pd
from sklearn.model_selection import train_test_split

# Assuming 'data' is your DataFrame containing 'Reviews' and 'sentiment_encoded' colum
df = data[['Reviews', 'sentiment_encoded']]

# Split the data into training (60%) and temporary data (40%)
X_train_temp, X_temp, Y_train_temp, Y_temp = train_test_split(df['Reviews'], df['senti

# Split the temporary data into testing (50%) and validation (50%)
X_test, X_validation, Y_test, Y_validation = train_test_split(X_temp, Y_temp, test_siz

print("Train:", X_train_temp.shape, Y_train_temp.shape)
print("Test:", X_test.shape, Y_test.shape)
print("Validation:", X_validation.shape, Y_validation.shape)
```

```
Train: (209682,) (209682,)
Test: (69894,) (69894,)
Validation: (69894,) (69894,)
CPU times: total: 15.6 ms
Wall time: 69.9 ms
```

In [ ]:

In [9]:
```
#Building a model
```

In [10]:
```
from simpletransformers.classification import ClassificationModel
```

```
# Create a TransformerModel
model = ClassificationModel('bert', 'bert-base-cased', num_labels=3, args={'reprocess_
```

```
Some weights of BertForSequenceClassification were not initialized from the model che
ckpoint at bert-base-cased and are newly initialized: ['classifier.weight', 'classifi
er.bias']
You should probably TRAIN this model on a down-stream task to be able to use it for p
redictions and inference.
```

In [23]:

```
train_df = pd.DataFrame({
    'text': X_train_temp[:20000].replace(r'\n', ' ', regex=True),
    'label': Y_train_temp[:20000]
})

test_df = pd.DataFrame({
    'text': X_test[-8000:].replace(r'\n', ' ', regex=True),
    'label': Y_test[-8000:]
})

validation_df = pd.DataFrame({
    'text': X_validation[-8000:].replace(r'\n', ' ', regex=True),
    'label': Y_validation[-8000:]
})
```

In [12]:

```
train_df.shape
```

Out[12]:

```
(20000, 2)
```

In [13]:

```
test_df.shape
```

Out[13]:

```
(8000, 2)
```

In [14]:

```
validation_df.shape
```

Out[14]:

```
(8000, 2)
```

In [15]:

```
model.train_model(train_df)
```

```
C:\Users\SACHIN\anaconda3\Lib\site-packages\simpletransformers\classification\classif
ication_model.py:612: UserWarning: Dataframe headers not specified. Falling back to u
sing column 0 as text and column 1 as labels.
  warnings.warn(
  0%|          | 0/20000 [00:00<?, ?it/s]
Epoch:   0%|          | 0/1 [00:00<?, ?it/s]
Running Epoch 0 of 1:   0%|          | 0/2500 [00:00<?, ?it/s]
```

Out[15]:

```
(2500, 0.4227670201926492)
```

In [ ]:

In [ ]:

In [17]:

```
result_test, model_outputs_test, wrong_predictions_test = model.eval_model(test_df)
```

```
C:\Users\SACHIN\anaconda3\Lib\site-packages\simpletransformers\classification\classif
ication_model.py:1454: UserWarning: Dataframe headers not specified. Falling back to
using column 0 as text and column 1 as labels.
  warnings.warn(
  0%|          | 0/8000 [00:00<?, ?it/s]
Running Evaluation:   0%|          | 0/1000 [00:00<?, ?it/s]
```

In [ ]:

In [24]:

```
result_validation, model_outputs_validation, wrong_predictions_validation = model.eval
```

```
C:\Users\SACHIN\anaconda3\Lib\site-packages\simpletransformers\classification\classif
ication_model.py:1454: UserWarning: Dataframe headers not specified. Falling back to
using column 0 as text and column 1 as labels.
  warnings.warn(
  0%|          | 0/8000 [00:00<?, ?it/s]
Running Evaluation:   0%|          | 0/1000 [00:00<?, ?it/s]
```

In [ ]:

```
#Model Evaluation test
```

In [26]:

```
result_test
```

Out[26]:

```
{'mcc': 0.7407787985957375, 'eval_loss': 0.35076412666449325}
```

In [27]:

```
model_outputs_test
```

Out[27]:

```
array([[-3.09651875, -1.62132001,  4.50514841],
       [-1.52153051, -0.06079921,  1.81366718],
       [ 2.50336242, -0.79744315, -1.93051732],
       ...,
       [-2.14424634, -1.27033639,  3.46602249],
       [-3.15373158, -1.51070786,  4.44351864],
       [ 0.44362289,  0.6996659 , -0.63016003]])
```

In [28]:

```
lst_test = []
for arr in model_outputs_test:
    lst_test.append(np.argmax(arr))
```

In [29]:

```
true_test = test_df['label'].tolist()
predicted_test = lst_test
```

In [30]:

```
import sklearn
mat_test = sklearn.metrics.confusion_matrix(true_test , predicted_test)
mat_test
```

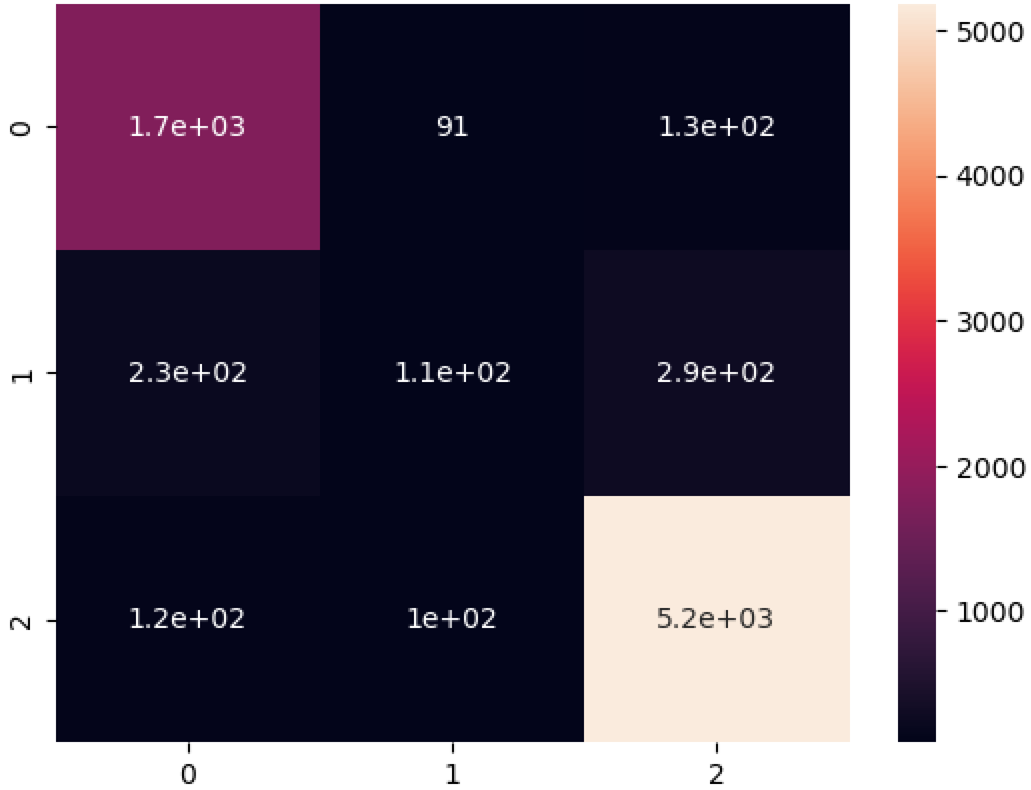
Out[30]:

```
array([[1749,   91,  127],
       [ 227,  106,  286],
       [ 125,  105, 5184]], dtype=int64)
```

In [31]:

```
df_cm_test = pd.DataFrame(mat_test, range(3), range(3))

sns.heatmap(df_cm_test, annot=True)
plt.show()
```



In [32]:
```
sklearn.metrics.classification_report(y_true=true_test,y_pred=predicted_test,target_na
```

Out[32]:
```
'              precision    recall  f1-score   support\n\n     neutral    0.83246
0.88917   0.85988      1967\n    negative    0.35099   0.17124   0.23018       619\n
positive    0.92621   0.95752   0.94160      5414\n\n    accuracy
0.87987      8000\n   macro avg    0.70322   0.67264   0.67722      8000\nweighted av
g    0.85865   0.87987   0.86646      8000\n'
```

In [33]:
```
sklearn.metrics.accuracy_score(true_test,predicted_test)
```

Out[33]:
```
0.879875
```

In [34]:
```
from sklearn.metrics import classification_report, accuracy_score, precision_score, re

# Calculate accuracy
accuracy = accuracy_score(true_test, predicted_test)

# Calculate precision, recall, and f1-score
precision = precision_score(true_test, predicted_test, average='weighted')
recall = recall_score(true_test, predicted_test, average='weighted')
f1 = f1_score(true_test, predicted_test, average='weighted')

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Accuracy: 0.879875
Precision: 0.8586523308064559
Recall: 0.879875
F1 Score: 0.8664644261886704
```

In [35]:
```
#Give your statement
```

In [85]:

```python
def sentiment(text):
    result = model.predict([text])
    pos = np.where(result[1][0] == np.amax(result[1][0]))[0][0]
    if pos == 0:
        print("This statement is Negative")
    elif pos == 2:
        print("This statement is Positive")
    elif pos == 1:
        print("This statement is Neutral")
```

In [86]:

```python
sentiment("a pretty capable, durable texting phone, unlocked as advertised, but the ve
```

```
  0%|          | 0/1 [00:00<?, ?it/s]
  0%|          | 0/1 [00:00<?, ?it/s]
This statement is Neutral
```

In [87]:

```python
sentiment("Do not buy it!. It feels like a pretended toy in your hand. If you want it
```

```
  0%|          | 0/1 [00:00<?, ?it/s]
  0%|          | 0/1 [00:00<?, ?it/s]
This statement is Negative
```

In [88]:

```python
sentiment("I went through lots of reviews for different phone before buying it. I four
```

```
  0%|          | 0/1 [00:00<?, ?it/s]
  0%|          | 0/1 [00:00<?, ?it/s]
This statement is Positive
```

In [41]:

```python
# validation
```

In [43]:

```python
lst_validation = []
for arr in model_outputs_validation:
    lst_validation.append(np.argmax(arr))

true_validation = validation_df['label'].tolist()
predicted_validation = lst_validation

import sklearn
mat_validation = sklearn.metrics.confusion_matrix(true_validation , predicted_validati
mat_validation
```

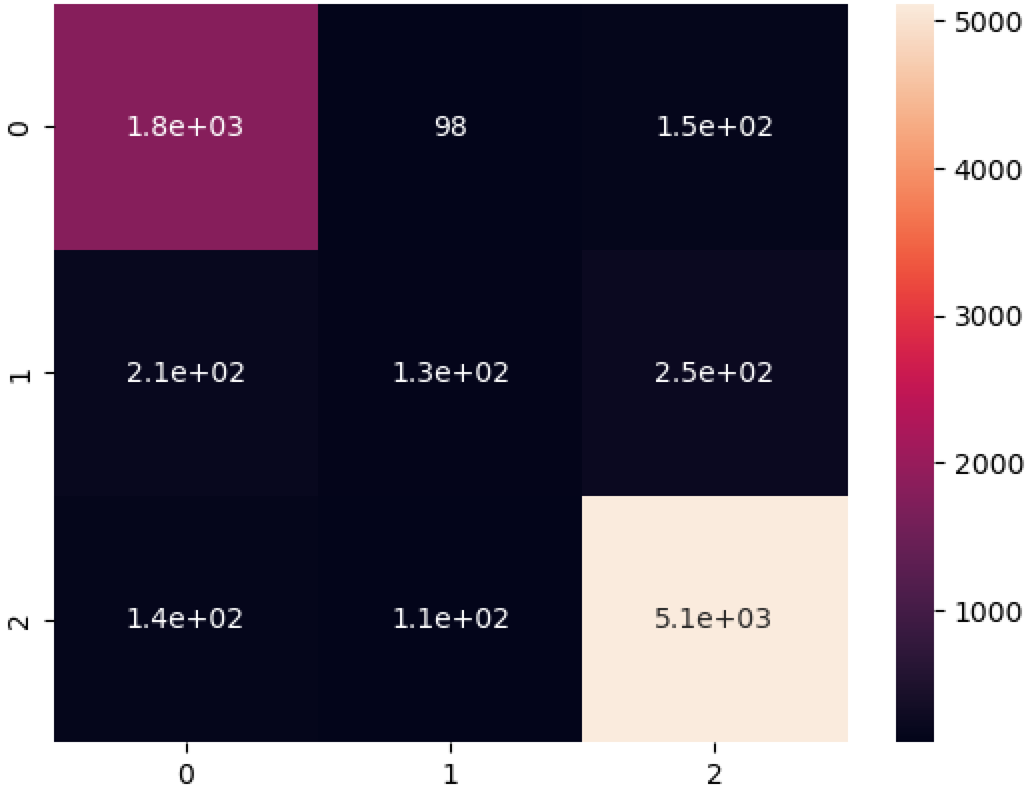
Out[43]:

```
array([[1792,   98,  148],
       [ 213,  129,  252],
       [ 140,  110, 5118]], dtype=int64)
```

In [72]:

```python
df_cm_validation = pd.DataFrame(mat_validation, range(3), range(3))

sns.heatmap(df_cm_validation, annot=True)
plt.show()
```



In [73]:

```
sklearn.metrics.classification_report(y_true=true_validation,y_pred=predicted_validati
```

Out[73]:

```
'              precision    recall  f1-score   support\n\n     neutral    0.83543
0.87929   0.85680      2038\n    negative    0.38279   0.21717   0.27712       594\n
positive    0.92751   0.95343   0.94029      5368\n\n    accuracy
0.87987      8000\n   macro avg    0.71524   0.68330   0.69140      8000\nweighted av
g    0.86361   0.87987   0.86978      8000\n'
```

In [74]:

```
from sklearn.metrics import classification_report, accuracy_score, precision_score, re

# Calculate accuracy
accuracy = accuracy_score(true_validation, predicted_validation)

# Calculate precision, recall, and f1-score
precision = precision_score(true_validation, predicted_validation, average='weighted')
recall = recall_score(true_validation, predicted_validation, average='weighted')
f1 = f1_score(true_test, predicted_validation, average='weighted')

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Accuracy: 0.879875
Precision: 0.8636074021626677
Recall: 0.879875
F1 Score: 0.527322826895247
```

In [78]:

```
import pandas as pd

# Assuming you have three DataFrames: train_df, validation_df, and test_df

# Export train_df to CSV
train_df.to_csv('train_data.csv', index=False)  # Use index=False to exclude the inde
```

```
# Export validation_df to CSV
validation_df.to_csv('validation_data.csv', index=False)

# Export test_df to CSV
test_df.to_csv('test_data.csv', index=False)
```

In [82]:

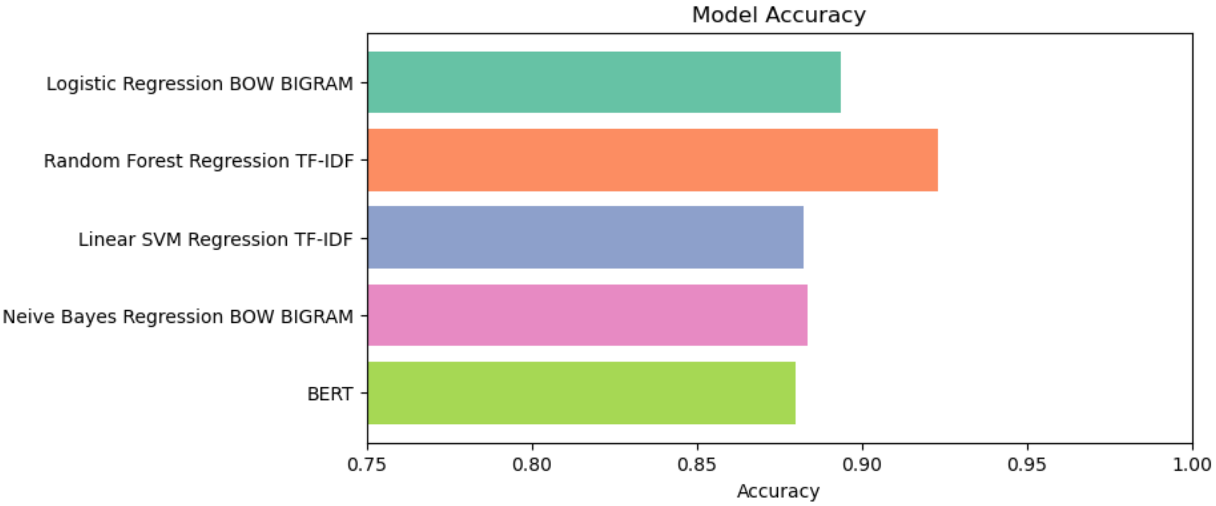
```
import matplotlib.pyplot as plt
import numpy as np

# Data
models = ["Logistic Regression BOW BIGRAM", "Random Forest Regression TF-IDF",
          "Linear SVM Regression TF-IDF", "Neive Bayes Regression BOW BIGRAM", "BERT"]
accuracy = [0.8938, 0.9231, 0.8822, 0.8836, 0.8799]

# Define light colors
colors = ['#66c2a5', '#fc8d62', '#8da0cb', '#e78ac3', '#a6d854']

# Create a horizontal bar plot
plt.figure(figsize=(8, 4))  # Adjust the figure size as needed
plt.barh(models, accuracy, color=colors)
plt.xlim(0.75, 1)  # Set the x-axis range
plt.xlabel('Accuracy')
plt.title('Model Accuracy')
plt.gca().invert_yaxis()  # Invert the y-axis for better readability

# Show the plot
plt.show()
```



In [ ]: