Model prepared by Muhammad Mashhood 24528.

Word2Vec:

Approach 1: Using Spacy's Word Embeddings

1. Data Preparation: Preprocess and clean the text data from the train.csv file.

2. Word Embeddings: Use Spacy to generate word embeddings for the cleaned text data. This step takes approximately 20 minutes.

3. Model Training and Evaluation: Train classification models (e.g., Naive Bayes, KNN) using the generated word embeddings and evaluate their performance on the testing data.

4. Optimization: Fine-tune hyperparameters, such as the number of neighbors for KNN, to optimize model performance.

5. Results: Observe that the KNN model performs best with a number of neighbors in the range of 17-20.

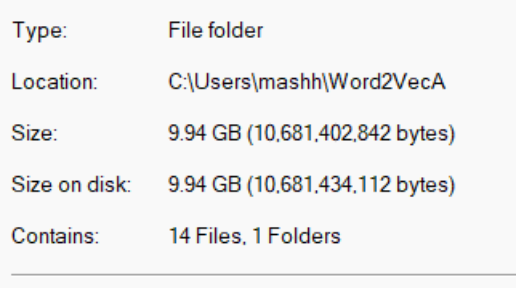
Winning model was linear regression with 84% accuracy.

Approach 2: Custom Word Embeddings using Gensim's Word2Vec(failed)

1. Training Word2Vec Model Use Gensim's Word2Vec to train a custom word2vec model on the train.csv data. This step takes approximately 3 hours due to the large size of the dataset and generated file 8GB approximately in CSV format.

Generated custom word embedding from our training data.

When the code was run to embed our training data according to it:
took 3.5 hours and generated a file of 9.94GB:



Format:

	A	B	C	D	E	F
1	review	sentiment	review_em	review_embedding_avg		
2	SAPS AT SE	negative	[array([			
3	1	2.06E-01				
4	-1.06E-01	-1.17E-01	-8.31E-03	2.00E-01		
5	-7.27E-02	2.10E-01	-1.78E-01	1.13E-01		
6	1.60E-02	-3.64E-01	-9.10E-02	6.79E-02		
7	3.99E-01	1.64E-01	1.76E-01	1.72E-01		
8	-1.72E-01	-2.03E-02	3.60E-01	1.31E-01		
9	1.62E-01	-3.84E-01	1.93E-01	-2.05334440e-01]		
10	dtype=fl	array([-0.2	0.573882	-0.01426	0.318305	-0.0332
11	0.11064	-0.22463	0.275907	-0.38008	-0.0311	
12	0.383385	-0.86332	-0.0733	0.00748	-0.17238	
13	-0.24921	0.118865	-0.63115	0.024886	-0.19421	
14	0.462463	-0.36692	0.167649	-0.43168	-0.06884	

However, when this was run through the Naïve bayes classifier several errors were encountered regarding memory issues which I tried to solve by passing it In chunks(dataset) but after 2 hours of training, nothing yielded.

Approach changed.

My final customized word2vec model:

Pre-Processing the data:

```
]: #parsing html
from bs4 import BeautifulSoup
import re

def parseHtml(html):
  soup = BeautifulSoup(html, 'html.parser')
  return soup.get_text()

def removeDigits(string):
  for i in range(10):
    string=string.replace(str(i),' ')
  return string

#removing html
reviews=list(map(parseHtml, reviews))

#removing digits
reviews=list(map(removeDigits, reviews))
```

```
: #tokenizing
import nltk
nltk.download('punkt')
tokenizedText=[nltk.word_tokenize(item) for item in reviews]
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\mashh\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
: #removing punctuation
punc = '''!()-[]{};:'"\, <>./?@#$%^&*_~'''
tokenizedText= [[word for word in review if word not in punc] for review in tokenizedText]
```

Data splitted:

```
31]: import numpy as np
     #splitting the Dataset into train and test set



       # Convert to a NumPy array
      # Access the shape correctly
     totalRows = len(tokenizedText)
     splitRatio=0.75
     splitPoint=int(splitRatio*totalRows)

     trainReviews=tokenizedText[:splitPoint]
     trainLabels=labels[:splitPoint]
     testReviews=tokenizedText[splitPoint:]
     testLabels=labels[splitPoint:]
```

This function is used to calculate the average embedding of each data item in a dataset based on the word embeddings from a pre-trained Word2Vec model (**`model`**). The **`trainReviews`** and **`testReviews`** datasets are used to generate vectors for training and testing respectively.

```
In [41]: def getVectors(dataset):
           singleDataItemEmbedding=np.zeros(embeddingsSize)
           vectors=[]
           for dataItem in dataset:
             wordCount=0
             for word in dataItem:
               if word in model.wv.key_to_index:
                 singleDataItemEmbedding=singleDataItemEmbedding+model.wv[word]
                 wordCount=wordCount+1

             singleDataItemEmbedding=singleDataItemEmbedding/wordCount
             vectors.append(singleDataItemEmbedding)
           return vectors

         trainReviewVectors=getVectors(trainReviews)
         testReviewVectors=getVectors(testReviews)
```

```
In [42]: #Let's define a function that can display the accuracy, F1-score, label-wise precision, recall, etc

         from sklearn.metrics import accuracy_score
```

Results:

```
####################RESULTS OF NAIVE BAYES CLASSIFIER#################
Accuracy=  0.6169333333333333
##########################################
precision: [0.51776246 0.71727468]
recall: [0.64948454 0.59514801]
fscore: [0.57619118 0.65052913]
support: [3007 4493]
##########################################3
Macro F1  0.6133601556744168
Micro F1  0.6169333333333333

####################RESULTS OF NEURAL NETWORK CLASSIFIER#################
Accuracy=  0.8254666666666667
##########################################
precision: [0.84544008 0.80525751]
recall: [0.81455939 0.83737796]
fscore: [0.8297125  0.82100369]
support: [3915 3585]
##########################################3
Macro F1  0.825358096840683
Micro F1  0.8254666666666667
####################RESULTS OF Random Forest CLASSIFIER#################
Accuracy=  0.7854666666666666
##########################################
precision: [0.76723224 0.80391631]
recall: [0.79834483 0.77341935]
fscore: [0.78247938 0.78837301]
support: [3625 3875]
##########################################3
Macro F1  0.7854261970937771
Micro F1  0.7854666666666666

  ####################RESULTS OF KNN Classifier#################
  Accuracy=  0.6964
  ##########################################
  precision: [0.71818664 0.67435622]
  recall: [0.69054295 0.7028236 ]
  fscore: [0.70409357 0.68829569]
  support: [3923 3577]
  ##########################################3
  Macro F1  0.6961946275682361
  Micro F1  0.6964
```

Time taken:

approximately: 40 minutes were taken to generate the custom word embedding and each model took 15-20 minutes to yield the result.