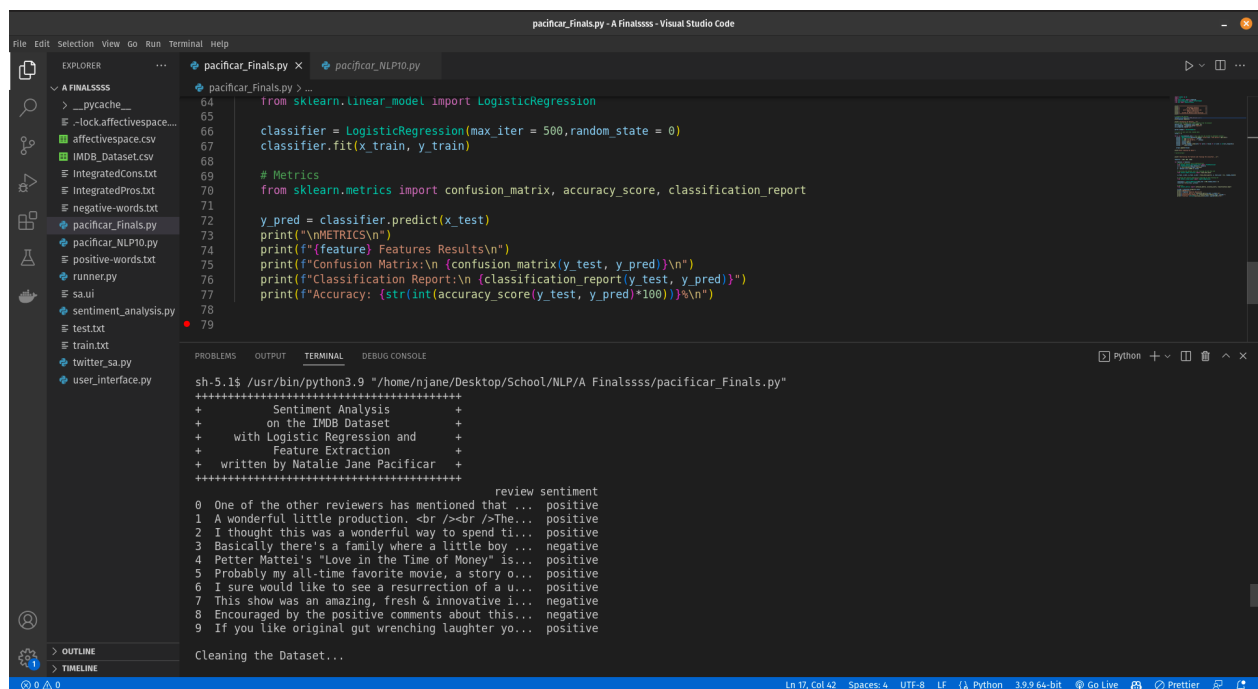


FINALS PROJECT ON NLP

Dataset: IMDB Movie Reviews Dataset

Retrieved from:

<https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews>



```
pacifcar_Finals.py - A Finalssss - Visual Studio Code

EXPLORER
  A Finalssss
    __pycache__
    .lock.affectivespace...
    affectivespace.csv
    IMDB_Dataset.csv
    IntegratedCons.txt
    IntegratedPros.txt
    negative-words.txt
    pacifcar_Finals.py
    pacifcar_NLP10.py
    positive-words.txt
    runner.py
    saui
    sentiment_analysis.py
    test.txt
    train.txt
    twitter_sa.py
    user_interface.py

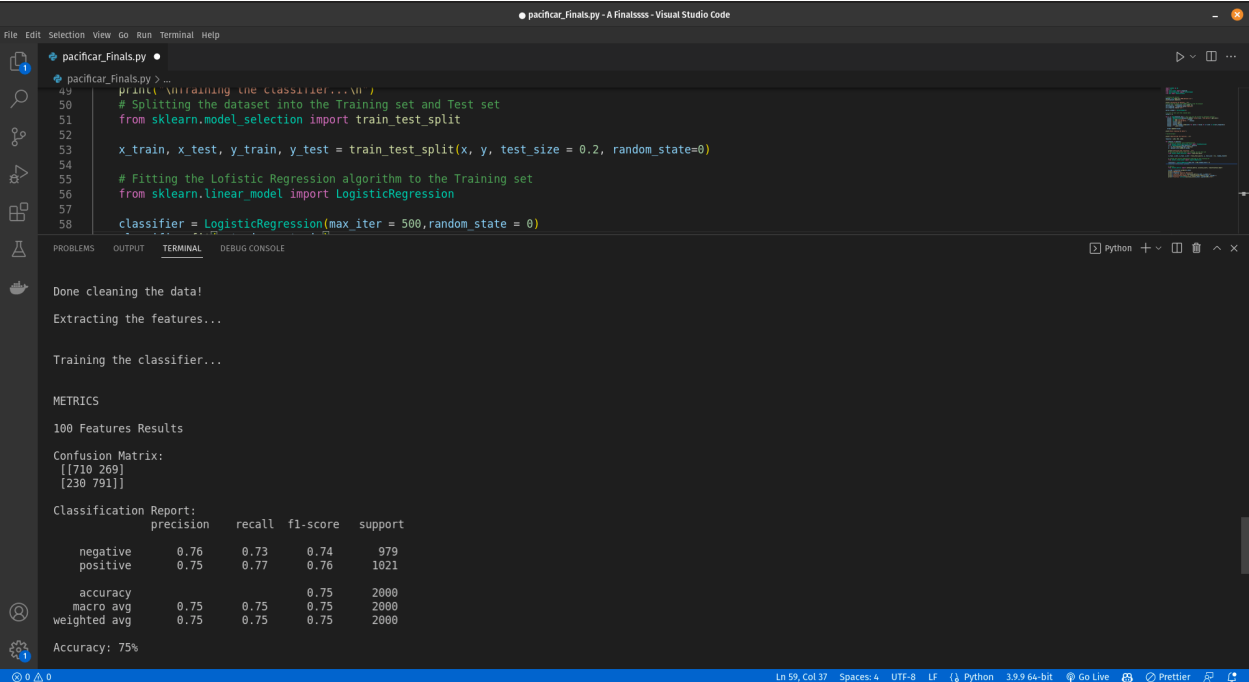
pacifcar_Finals.py
64 from sklearn.linear_model import LogisticRegression
65
66 classifier = LogisticRegression(max_iter = 500, random_state = 0)
67 classifier.fit(x_train, y_train)
68
69 # Metrics
70 from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
71
72 y_pred = classifier.predict(x_test)
73 print("\nMetrics\n")
74 print(f"Feature: Features Results\n")
75 print(f"Confusion Matrix:\n {confusion_matrix(y_test, y_pred)}\n")
76 print(f"Classification Report:\n {classification_report(y_test, y_pred)}")
77 print(f"Accuracy: {str(int(accuracy_score(y_test, y_pred)*100))}\n")
78
79

TERMINAL
sh-5.1$ /usr/bin/python3.9 "/home/njane/Desktop/School/NLP/A Finalssss/pacifcar_Finals.py"
+-----+
+ Sentiment Analysis +
+ on the IMDB Dataset +
+ with Logistic Regression and +
+ Feature Extraction +
+ written by Natalie Jane Pacificar +
+-----+
review sentiment
0 One of the other reviewers has mentioned that ... positive
1 A wonderful little production. <br /><br />The... positive
2 I thought this was a wonderful way to spend ti... positive
3 Basically there's a family where a little boy ... negative
4 Petter Mattei's "Love in the Time of Money" is... positive
5 Probably my all-time favorite movie, a story o... positive
6 I sure would like to see a resurrection of a u... positive
7 This show was an amazing, fresh & innovative l... negative
8 Encouraged by the positive comments about this... negative
9 If you like original gut wrenching laughter yo... positive

Cleaning the Dataset...
```

1. Cleaned the dataset by removing the stopwords, parsing the data with BeautifulSoup, using regex to remove unnecessary characters, and using the Porter Stemmer to reduce the words into their lemmas.
2. Extracted Features to increase the accuracy of our algorithm. I tried different numbers of features to see if there are significant changes in our scores.
3. For the classifier, I used Logistic Regression to fit the algorithm to our training set as well as to predict the sentiment.

Our algorithm training on **100** Features. It only achieved **75%** accuracy.



```
pacificar_Finals.py > ...
49 print("\nTraining the classifier...\n")
50 # Splitting the dataset into the Training set and Test set
51 from sklearn.model_selection import train_test_split
52
53 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state=0)
54
55 # Fitting the Logistic Regression algorithm to the Training set
56 from sklearn.linear_model import LogisticRegression
57
58 classifier = LogisticRegression(max_iter = 500, random_state = 0)
```

Done cleaning the data!
Extracting the features...
Training the classifier...

METRICS

100 Features Results

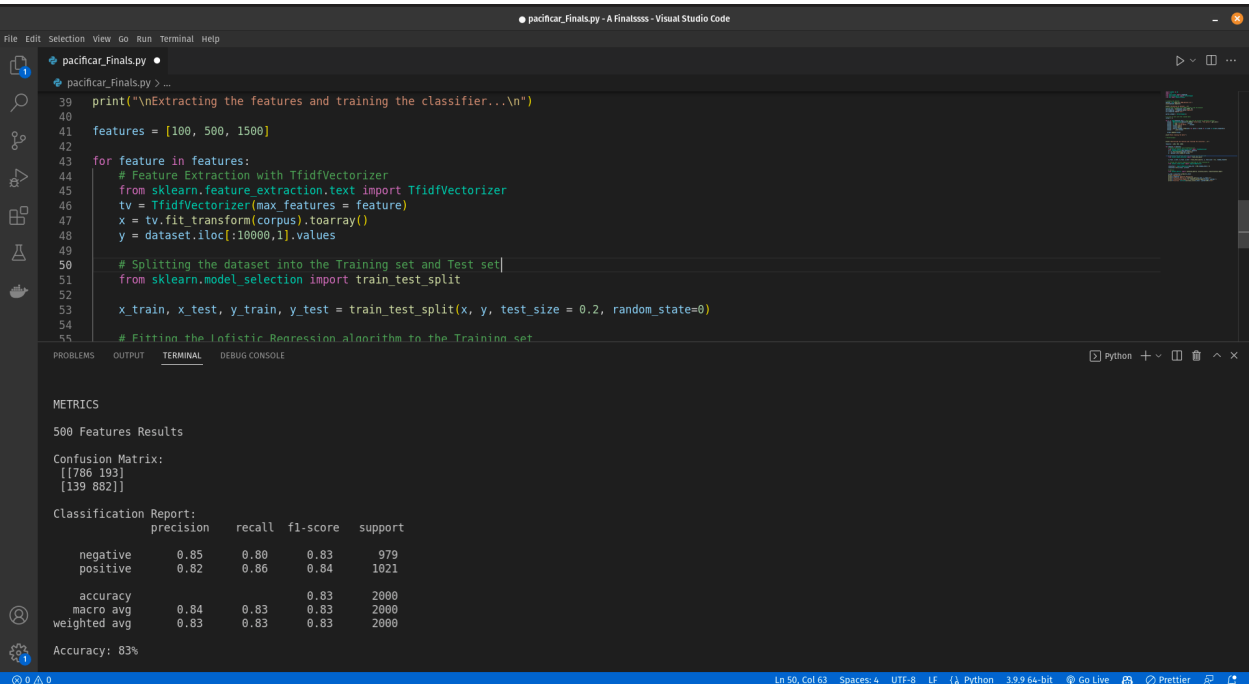
Confusion Matrix:
[[710 269]
 [230 791]]

Classification Report:

	precision	recall	f1-score	support
negative	0.76	0.73	0.74	979
positive	0.75	0.77	0.76	1021
accuracy			0.75	2000
macro avg	0.75	0.75	0.75	2000
weighted avg	0.75	0.75	0.75	2000

Accuracy: 75%

Our algorithm training on **500** Features. It achieved **83%** accuracy.



```
pacificar_Finals.py > ...
39 print("\nExtracting the features and training the classifier...\n")
40
41 features = [100, 500, 1500]
42
43 for feature in features:
44     # Feature Extraction with TfidfVectorizer
45     from sklearn.feature_extraction.text import TfidfVectorizer
46     tv = TfidfVectorizer(max_features = feature)
47     x = tv.fit_transform(corpus).toarray()
48     y = dataset.iloc[:,10000,1].values
49
50 # Splitting the dataset into the Training set and Test set
51 from sklearn.model_selection import train_test_split
52
53 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state=0)
54
55 # Fitting the Logistic Regression algorithm to the Training set
```

METRICS

500 Features Results

Confusion Matrix:
[[786 193]
 [139 882]]

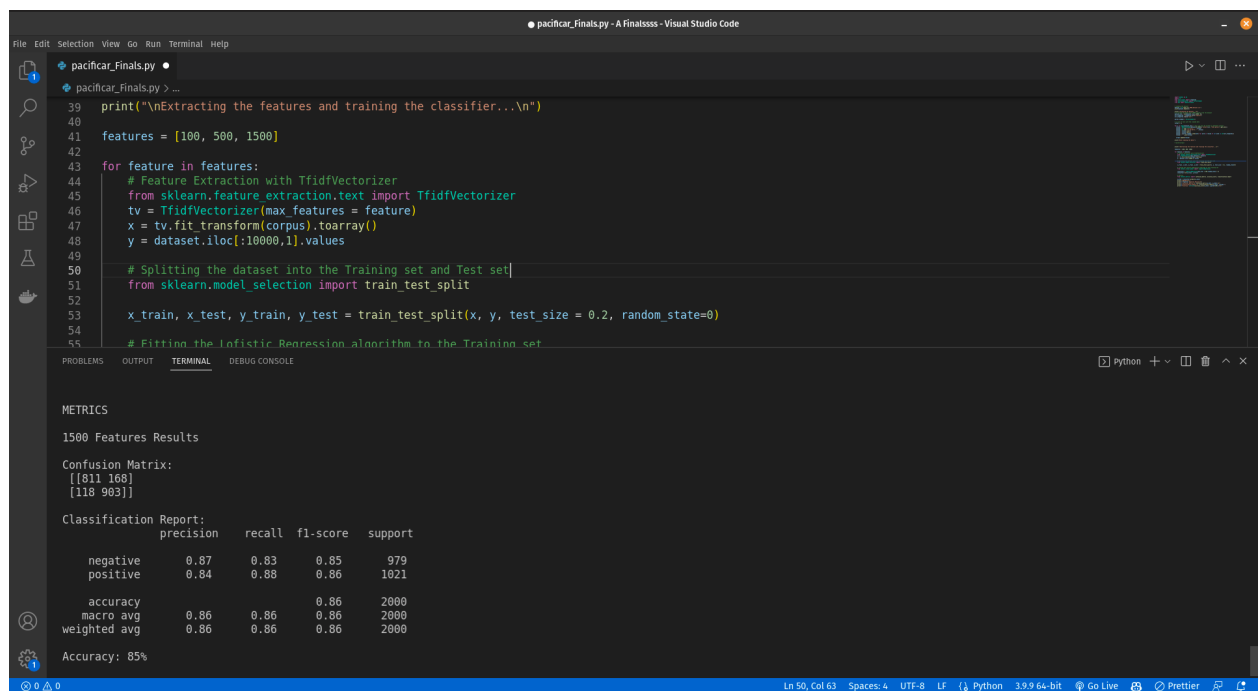
Classification Report:

	precision	recall	f1-score	support
negative	0.85	0.80	0.83	979
positive	0.82	0.86	0.84	1021
accuracy			0.83	2000
macro avg	0.84	0.83	0.83	2000
weighted avg	0.83	0.83	0.83	2000

Accuracy: 83%

Our algorithm training on **1500** Features. It achieved **85%** accuracy.

Natalie Jane Pacificar BSCS 3-B



```
39 print("\nExtracting the features and training the classifier...\n")
40
41 features = [100, 500, 1500]
42
43 for feature in features:
44     # Feature Extraction with TfidfVectorizer
45     from sklearn.feature_extraction.text import TfidfVectorizer
46     tv = TfidfVectorizer(max_features = feature)
47     x = tv.fit_transform(corpus).toarray()
48     y = dataset.iloc[:10000,1].values
49
50     # Splitting the dataset into the Training set and Test set
51     from sklearn.model_selection import train_test_split
52
53     x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state=0)
54
55     # Fitting the Logistic Regression algorithm to the Training set
```

METRICS

1500 Features Results

Confusion Matrix:

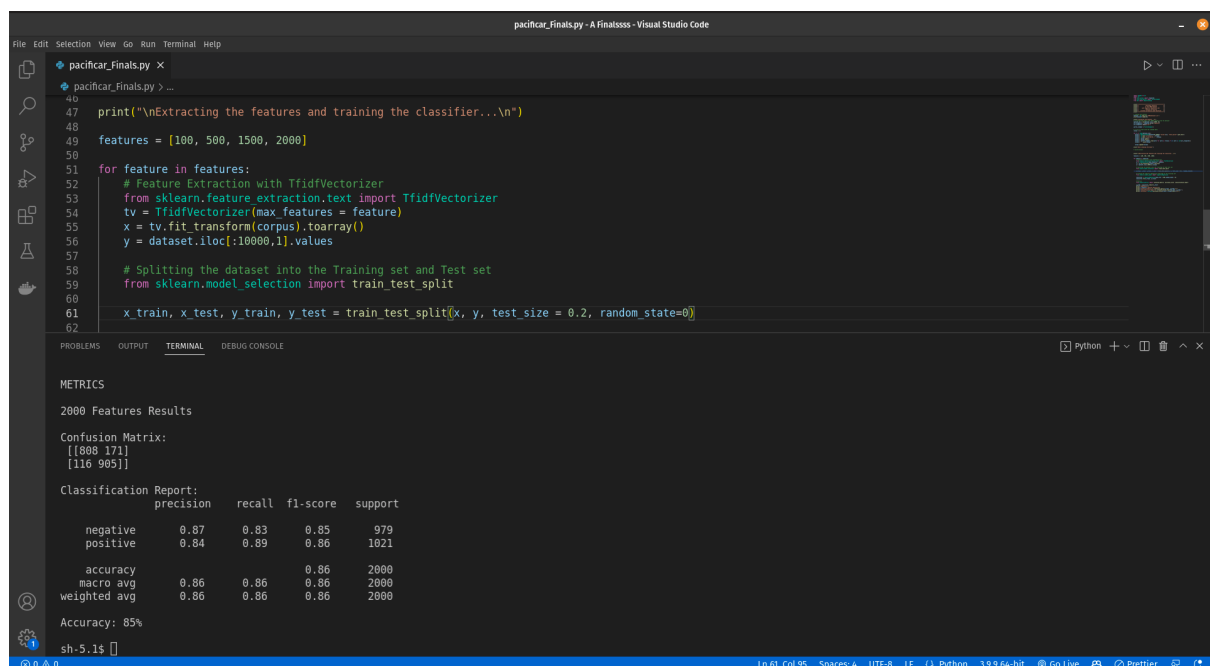
```
[[811 168]
 [118 903]]
```

Classification Report:

	precision	recall	f1-score	support
negative	0.87	0.83	0.85	979
positive	0.84	0.88	0.86	1021
accuracy			0.86	2000
macro avg	0.86	0.86	0.86	2000
weighted avg	0.86	0.86	0.86	2000

Accuracy: 85%

Our algorithm training on 2000 Features. It still achieved **85%** accuracy.



```
47 print("\nExtracting the features and training the classifier...\n")
48
49 features = [100, 500, 1500, 2000]
50
51 for feature in features:
52     # Feature Extraction with TfidfVectorizer
53     from sklearn.feature_extraction.text import TfidfVectorizer
54     tv = TfidfVectorizer(max_features = feature)
55     x = tv.fit_transform(corpus).toarray()
56     y = dataset.iloc[:10000,1].values
57
58     # Splitting the dataset into the Training set and Test set
59     from sklearn.model_selection import train_test_split
60
61     x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state=0)
62
```

METRICS

2000 Features Results

Confusion Matrix:

```
[[808 171]
 [116 905]]
```

Classification Report:

	precision	recall	f1-score	support
negative	0.87	0.83	0.85	979
positive	0.84	0.89	0.86	1021
accuracy			0.86	2000
macro avg	0.86	0.86	0.86	2000
weighted avg	0.86	0.86	0.86	2000

Accuracy: 85%

We trained our algorithm with different numbers of features. We found out that the algorithm reached its peak with **1500 features**. The number of False Positives and False Negatives was greatly reduced which means that the accuracy also increased together with the scores in the classification report as we increase the number of features up to 1500 features.