

## **Table of Contents**

1. Problem Definition .....	1
2. Project Objectives .....	1
3. Intelligent system methodology.....	2
4. Description of implementation .....	2
5. Test Results .....	4
6. Discussion and comments on the experience gained .....	4
7. References .....	9

## 1. Problem Definition

**Problem Definition** Users have written over hundreds of reviews for each movie. The reviews are expressed in the natural language, along with a column describing the overall sentiment of that review wither positive or negative. To make a better-informed decision, user has to go through each of them, which is a time-consuming activity that user is highly unlikely to invest time in.

Sentiment analysis is a natural language processing (NLP) technique used to determine whether data is positive, negative, or neutral. Sentiment analysis is often performed on textual data to help businesses monitor brand and product sentiment in customer feedback, and understand customer needs. Using sentiment analysis, we can find the state of mind of the reviewer while providing the review and understand if the person was “happy”, “sad”, “angry” and so on.

In this project we're helping users making a better-informed decision by using Sentiment Analysis on a set of movie reviews given by reviewers and try to understand what their overall reaction to the movie was, i.e. if they liked the movie or they hated it. We aim to utilize the relationships of the words in the review to predict the overall polarity of the review.

## 2. Project Objectives

The main objective is to determine the polarity of that text (polarity is whether the author expresses positive or negative opinion).

- Learn how to read a big Dataset for movie review to work on.
- Learn Preprocessing techniques used to simplify the dataset with removing any non-useful word from the data set or any symbol.
- Learn to take a sample of dataset because the size of dataset is massive.
- Learn to give weights to words to operate on it and fit data set in table.
- Learn to train and test the machine to operate.
- Get the best model that has the biggest accuracy.

### 3. Intelligent system methodology

Our method of sentiment analysis is based upon machine learning

#### Pre-Processing Phase:

Text is messy, people love to throw in attempts at expressing themselves more clearly by adding extravagant punctuation and spelling words incorrectly.

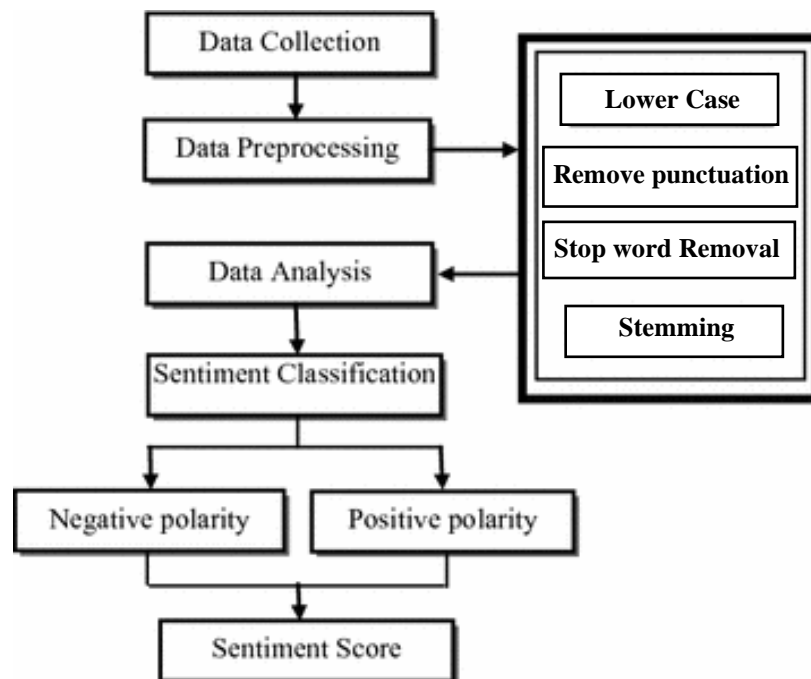
Pre-Processing is a process of preparing the raw data and making it suitable for a machine learning model. This is done to remove the unwanted words or symbols. These words / symbols do not affect the outcome but can slow down the processing of algorithm.

#### Feature Extraction:

How should we approach transforming word representations to numerical versions that a model can interpret?

#### Classification Phase:

Based on different algorithms these keywords are put under certain category.



### 4. Description of implementation

The dataset used for this task was collected from Kaggle IMDB dataset which have 50K movie reviews for natural language processing or Text analytics. This is a dataset for binary sentiment classification.

## Sentiment Analysis phases:

### Dataset preprocessing involves the following steps:

In order to perform data preprocessing using Python, we need to import some predefined Python libraries. These libraries are used to perform some specific jobs.

**NumPy:** NumPy Python library is used for including any type of mathematical operation in the code. It also supports to add large, multidimensional arrays and matrices.

**Pandas:** Pandas library is one of the most famous Python libraries and used for importing and managing the datasets.

Also, from python libraries we used to implement the text preprocessing tasks is nltk and re.

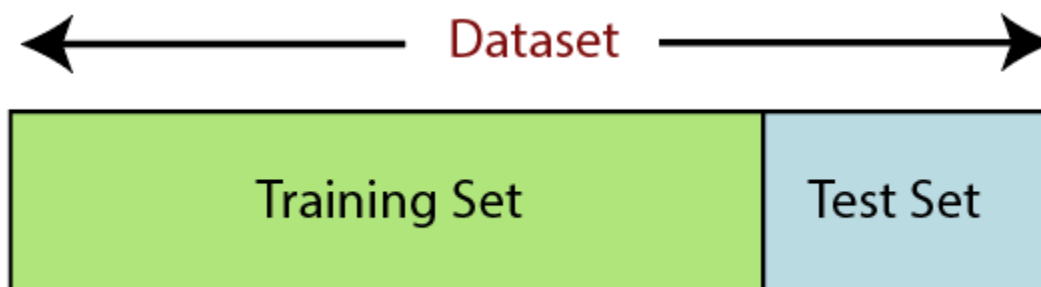
Basic pre-processing for text consists of removing non-alphabetic characters, stop words (a set of very common words like the, a, and, etc.) and changing all words to lowercase: Words like Book and book mean the same but when not converted to the lower case those two are represented as two different words in the vector space model. We also need to remove HTML tags from the movie reviews using regular expression.

### Feature Extraction:

Scikit-learn's CountVectorizer is used to convert a collection of text documents to a vector of term/token counts. It also enables the pre-processing of text data prior to generating the vector representation. This functionality makes it a highly flexible feature representation module for text.

### Splitting the dataset:

Every dataset for Machine Learning model must be split into two separate sets: training set and test set.



We split data: test data 10% and train data 90%

## **Classification Phase**

We have used **Random Forest Classifier** using the **Scikit-Learn library** of Python which is a supervised Machine learning algorithm used for classification, regression, and other tasks using decision trees.

The Random Forest classifier creates a set of decision trees from a randomly selected subset of the training set then it collects the votes from different decision trees to decide the final prediction.

## **5. Test Results**

We have tried many classifier techniques as Logistic Regression with accuracy 88.73%, KN Neighbors Classifier with accuracy 59%, Decision Tree Classifier with accuracy 77.9% and finally we have chosen using Random Forest classifier technique, as it has the higher accuracy with 89%

## **6. Discussion and comments on the experience gained**

Finding the polarity of the reviews can help in various domain. Intelligent systems can be developed which can provide the users with comprehensive reviews of movies, products, services etc. without requiring the user to go through individual reviews, he can directly take decisions based on the results provided by the intelligent systems.

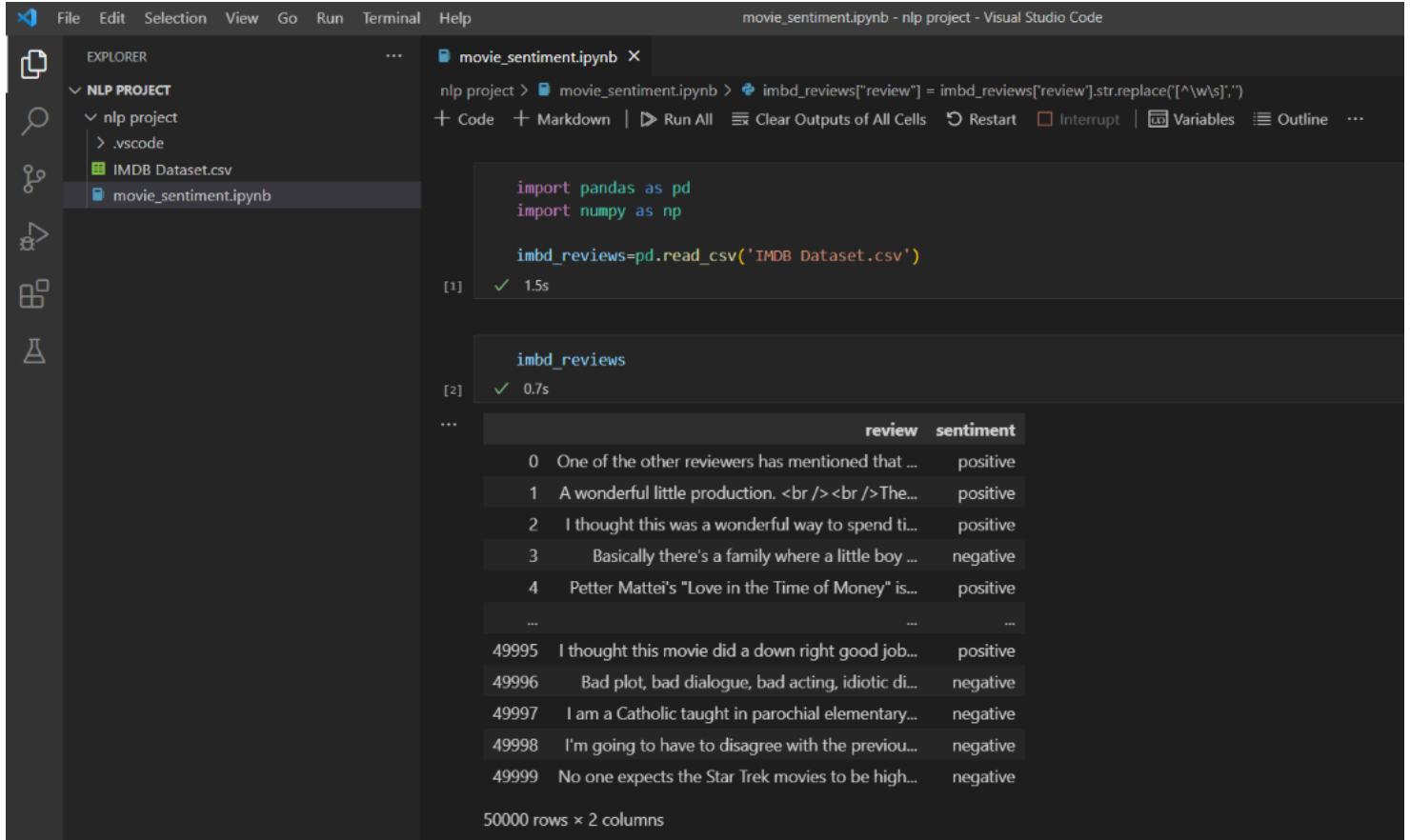
### **To what extent were the project objectives met?**

In conclusion, through this project, we learned the preprocessing techniques, Feature extraction using count vector and classification technique Random Forest Classifier

### **If you get the opportunity to do it again, what would you do differently - e.g., use a different methodology and/or software tool?**

One of the major improvements that can be incorporated as we move ahead in this project is using another algorithm in feature extraction as Tf-Idf, or Word2Vec. Another point of improvement can be to model this problem as a multi-class classification problem where we classify the sentiments of reviewer in more than binary fashion like “Happy”, “Bored”, “Afraid”, etc. instead of complete positive/negative. Furthermore, we may apply sentiment classification algorithms to detect fake reviews.

## 7. Code and Output



```
File Edit Selection View Go Run Terminal Help
movie_sentiment.ipynb - nlp project - Visual Studio Code
```

EXPLORER

- NLP PROJECT
  - .vscode
  - IMDB Dataset.csv
  - movie\_sentiment.ipynb

```
movie_sentiment.ipynb X
nlp project > movie_sentiment.ipynb > imbd_reviews["review"] = imbd_reviews["review"].str.replace("[^\w\s]","")
+ Code + Markdown | Run All Clear Outputs of All Cells Restart Interrupt Variables Outline ...
```

```
import pandas as pd
import numpy as np

imbd_reviews=pd.read_csv('IMDB Dataset.csv')
```

[1] ✓ 1.5s

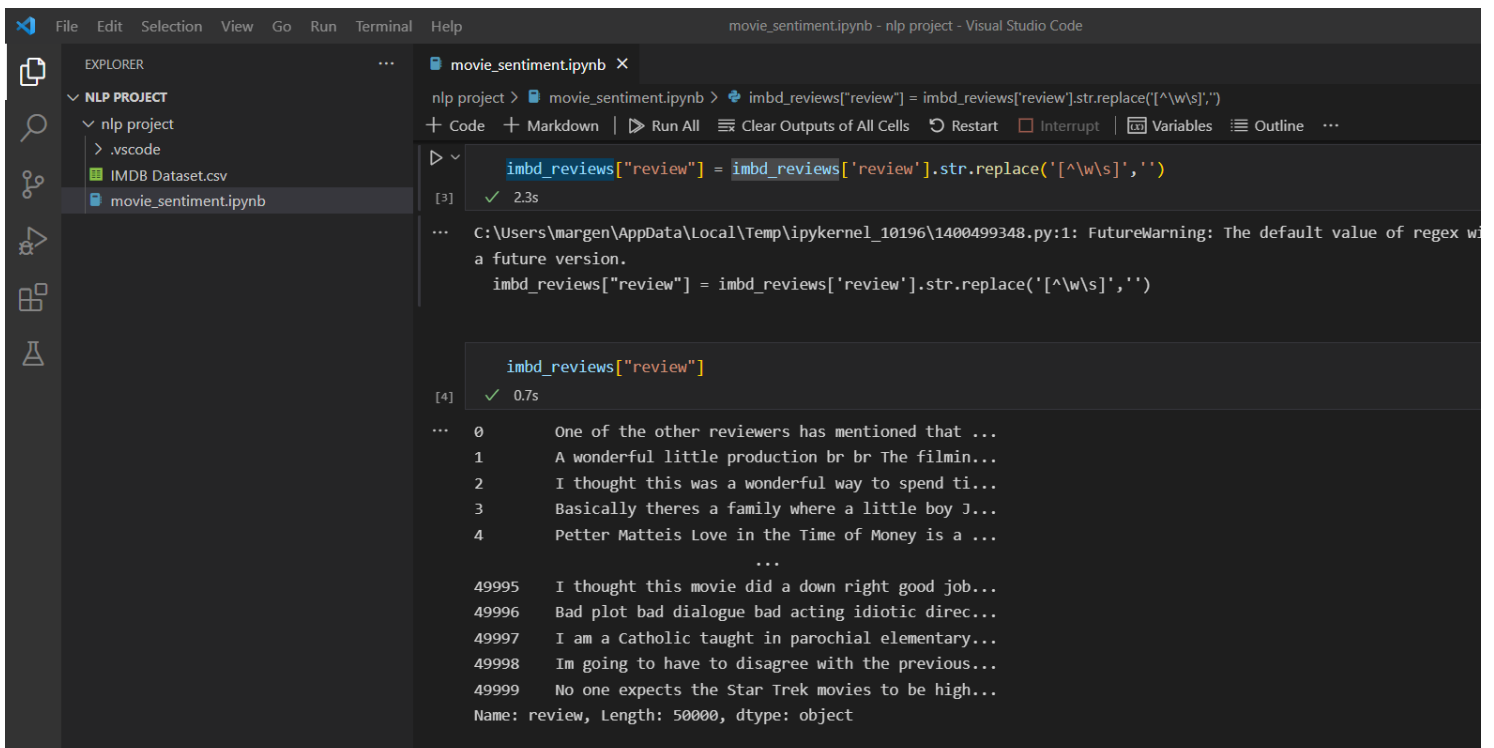
```
imbd_reviews
```

[2] ✓ 0.7s

...

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production.     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
...	...	...
49995	I thought this movie did a down right good job...	positive
49996	Bad plot, bad dialogue, bad acting, idiotic di...	negative
49997	I am a Catholic taught in parochial elementary...	negative
49998	I'm going to have to disagree with the previou...	negative
49999	No one expects the Star Trek movies to be high...	negative

50000 rows × 2 columns



```
File Edit Selection View Go Run Terminal Help
movie_sentiment.ipynb - nlp project - Visual Studio Code
```

EXPLORER

- NLP PROJECT
  - .vscode
  - IMDB Dataset.csv
  - movie\_sentiment.ipynb

```
movie_sentiment.ipynb X
nlp project > movie_sentiment.ipynb > imbd_reviews["review"] = imbd_reviews["review"].str.replace("[^\w\s]","")
+ Code + Markdown | Run All Clear Outputs of All Cells Restart Interrupt Variables Outline ...
```

```
imbd_reviews["review"] = imbd_reviews["review"].str.replace('[^\w\s]','')
```

[3] ✓ 2.3s

... C:\Users\margen\AppData\Local\Temp\ipykernel\_10196\1400499348.py:1: FutureWarning: The default value of regex will be changed to True in a future version.

```
imbd_reviews["review"] = imbd_reviews["review"].str.replace('[^\w\s]','')
```

```
imbd_reviews["review"]
```

[4] ✓ 0.7s

...

0	One of the other reviewers has mentioned that ...
1	A wonderful little production br br The filmin...
2	I thought this was a wonderful way to spend ti...
3	Basically theres a family where a little boy J...
4	Petter Matteis Love in the Time of Money is a ...
...	...
49995	I thought this movie did a down right good job...
49996	Bad plot bad dialogue bad acting idiotic direc...
49997	I am a Catholic taught in parochial elementary...
49998	Im going to have to disagree with the previous...
49999	No one expects the Star Trek movies to be high...

Name: review, Length: 50000, dtype: object

Visual Studio Code interface showing a Jupyter Notebook for NLP project. The Explorer pane on the left shows the project structure: NLP PROJECT, nlp project, .vscode, IMDB Dataset.csv, and movie\_sentiment.ipynb. The main editor displays the notebook content, which includes a cell with the following code:

```
imdb_reviews['review'] = imdb_reviews['review'].str.replace('[^\w\s]','')
imdb_reviews['review'] = imdb_reviews['review'].str.lower()
```

The output of the second cell shows a preview of the `imdb_reviews` DataFrame:

```
imdb_reviews
```

	review	sentiment
0	one of the other reviewers has mentioned that ...	positive
1	a wonderful little production br br the filmin...	positive
2	i thought this was a wonderful way to spend ti...	positive
3	basically theres a family where a little boy j...	negative
4	petter matteis love in the time of money is a ...	positive
...	...	...
49995	i thought this movie did a down right good job...	positive
49996	bad plot bad dialogue bad acting idiotic direc...	negative
49997	i am a catholic taught in parochial elementary...	negative
49998	im going to have to disagree with the previous...	negative
49999	no one expects the star trek movies to be high...	negative

The DataFrame has 50000 rows and 2 columns.

Visual Studio Code interface showing the continuation of the Jupyter Notebook. The Explorer pane on the left shows the project structure: NLP PROJECT, nlp project, .vscode, IMDB Dataset.csv, and movie\_sentiment.ipynb. The main editor displays the notebook content, which includes a cell with the following code:

```
imdb_reviews = imdb_reviews.sample(frac=0.3, replace=True, random_state=1)
```

The output of the cell shows a preview of the `x` variable, which is a sparse matrix of type `int64`:

```
x
```

```
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]], dtype=int64)
```

The next cell contains the following code:

```
y = imdb_reviews['sentiment']
```

The output of the cell shows a preview of the `y` variable, which is a 1D array of sentiment labels:

```
y
```

```
array(['positive', 'negative', 'negative', 'negative', 'negative',
       ..., 'positive', 'negative', 'negative', 'negative', 'negative'],
      dtype=object)
```

The final cell contains the following code:

```
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
y = label.fit_transform(y)
```

File Edit Selection View Go Run Terminal Help movie\_sentiment.ipynb - nlp project - Visual Studio Code

EXPLORER

- NLP PROJECT
  - nlp project
    - .vscode
    - IMDB Dataset.csv
    - movie\_sentiment.ipynb

movie\_sentiment.ipynb X

```
nlp project > movie_sentiment.ipynb > imbd_reviews['review'] = imbd_reviews['review'].str.replace('[^\w\s]','')
+ Code + Markdown | Run All | Clear Outputs of All Cells | Restart | Interrupt | Variables | Outline ...
```

```
from sklearn.preprocessing import LabelEncoder
table = LabelEncoder()
y = table.fit_transform(y)
```

[11] ✓ 0.3s

```
y
```

[12] ✓ 0.4s

```
... array([1, 0, 0, ..., 0, 1, 1])
```

```
from sklearn.model_selection import train_test_split
```

[13] ✓ 0.9s

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.10, stratify=y)
```

[14] ✓ 0.2s

```
from sklearn.preprocessing import StandardScaler
```

[15] ✓ 0.3s

```
scaler = StandardScaler()
scaler.fit(x_train)
x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
```

[16] ✓ 2.4s

> OUTLINE

> TIMELINE

File Edit Selection View Go Run Terminal Help movie\_sentiment.ipynb - nlp project - Visual Studio Code

EXPLORER

- NLP PROJECT
  - nlp project
    - .vscode
    - IMDB Dataset.csv
    - movie\_sentiment.ipynb

movie\_sentiment.ipynb X

```
nlp project > movie_sentiment.ipynb > imbd_reviews['review'] = imbd_reviews['review'].str.replace('[^\w\s]','')
+ Code + Markdown | Run All | Clear Outputs of All Cells | Restart | Interrupt | Variables | Outline ... Python 3.10.3 64-bit
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score, fbeta_score, confusion_matrix
```

[17] ✓ 0.7s

```
models = {
    'LR': LogisticRegression(),
    'KNN': KNeighborsClassifier(),
    'DT': DecisionTreeClassifier(),
    'RF': RandomForestClassifier(),
    'NB': GaussianNB()
}
```

[18] ✓ 0.6s

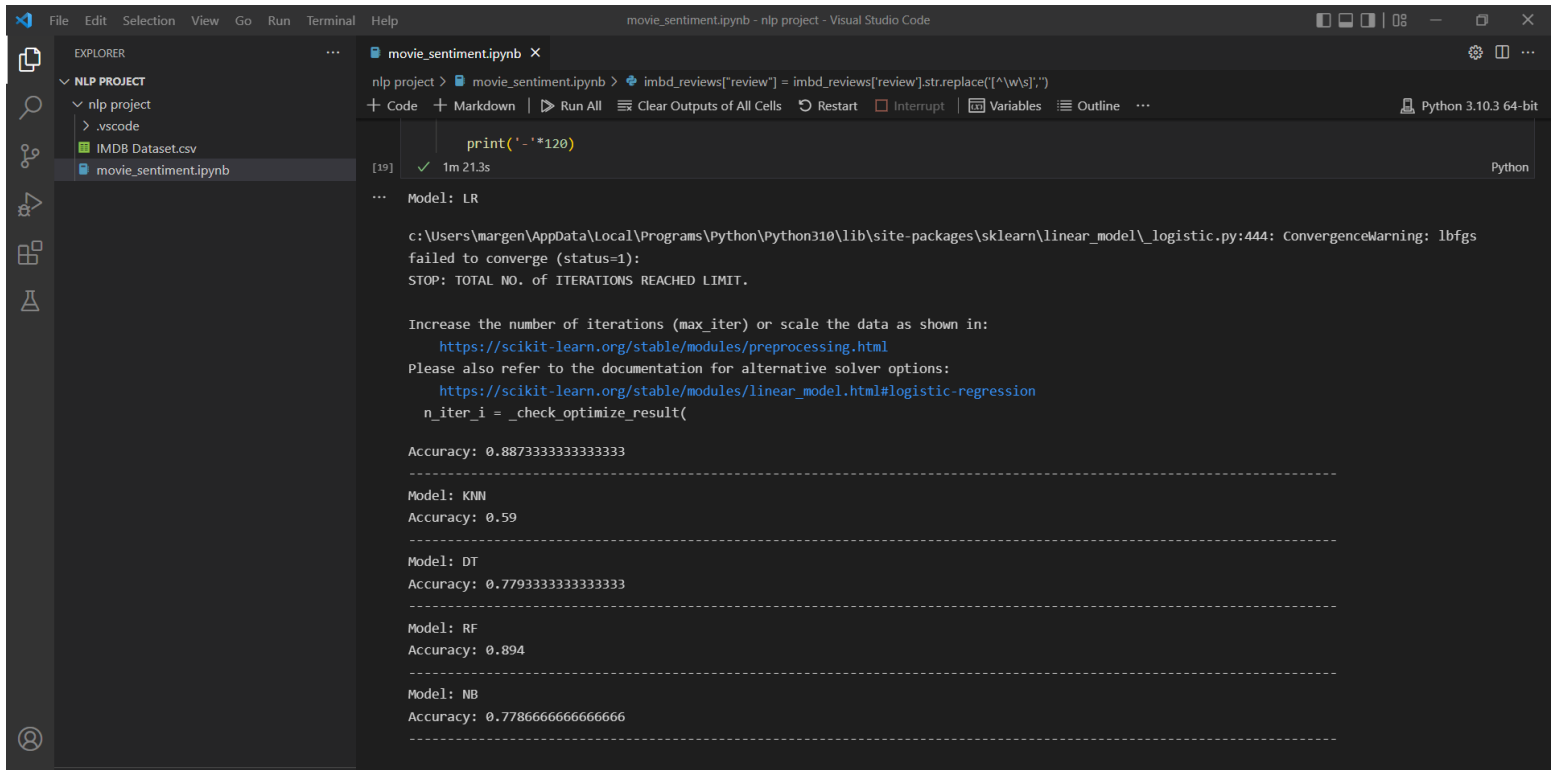
```
for name, model in models.items():
    print(f'Model: {name}')
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
    print('-'*120)
```

[19] ✓ 1m 21.3s

```
... Model: LR
```

> OUTLINE





movie\_sentiment.ipynb - nlp project - Visual Studio Code

Python 3.10.3 64-bit

```
imdb_reviews["review"] = imdb_reviews["review"].str.replace("[^\w\s]","")

print('-'*120)
```

[19] ✓ 1m 21.3s Python

Model: LR

c:\Users\margen\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\linear\_model\\_logistic.py:444: ConvergenceWarning: lbfgs failed to converge (status=1):  
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:  
<https://scikit-learn.org/stable/modules/preprocessing.html>  
Please also refer to the documentation for alternative solver options:  
[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)  
n\_iter\_i = \_check\_optimize\_result(

Accuracy: 0.8873333333333333

-----

Model: KNN  
Accuracy: 0.59

-----

Model: DT  
Accuracy: 0.7793333333333333

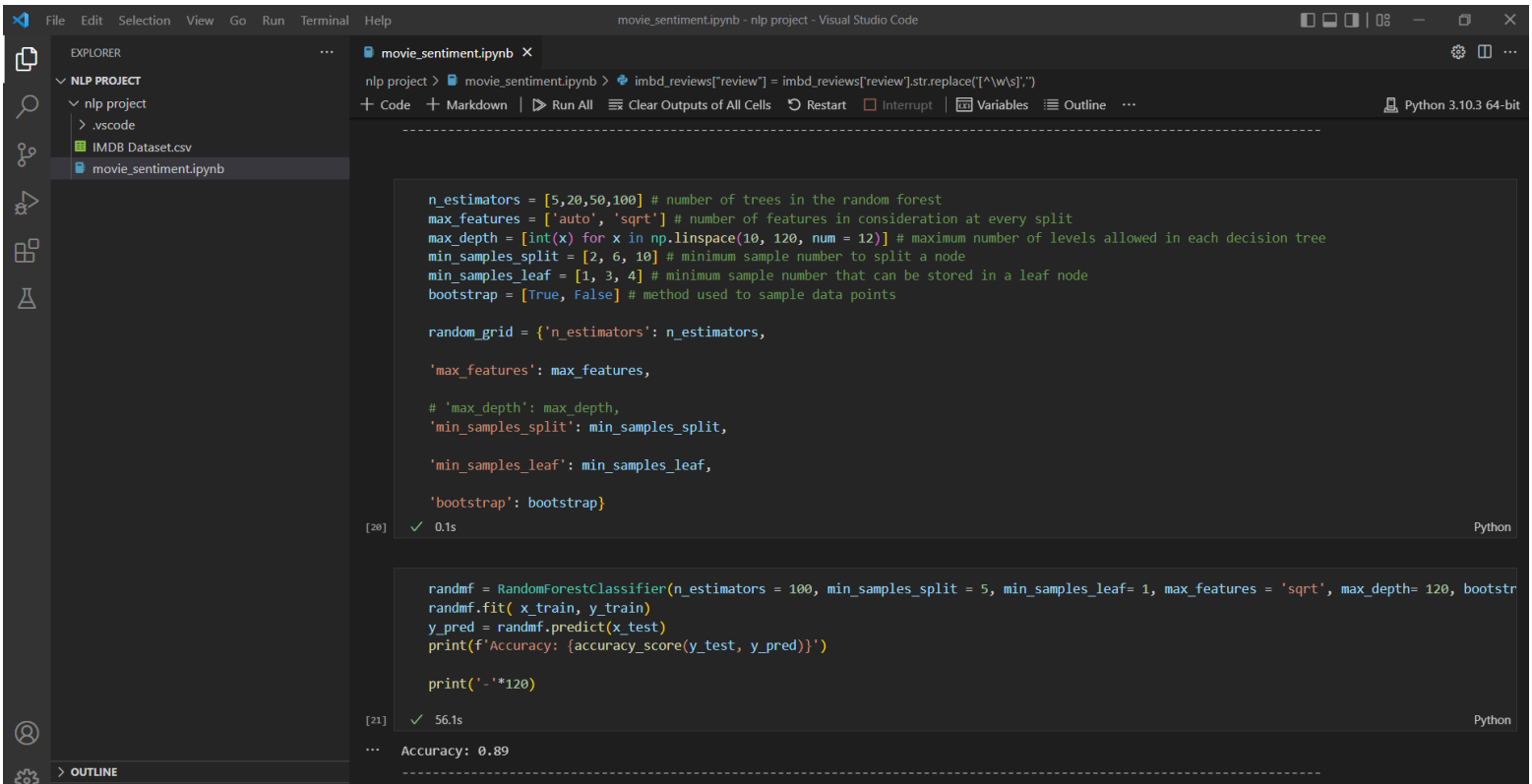
-----

Model: RF  
Accuracy: 0.894

-----

Model: NB  
Accuracy: 0.7786666666666666

-----



movie\_sentiment.ipynb - nlp project - Visual Studio Code

Python 3.10.3 64-bit

```
n_estimators = [5,20,50,100] # number of trees in the random forest
max_features = ['auto', 'sqrt'] # number of features in consideration at every split
max_depth = [int(x) for x in np.linspace(10, 120, num = 12)] # maximum number of levels allowed in each decision tree
min_samples_split = [2, 6, 10] # minimum sample number to split a node
min_samples_leaf = [1, 3, 4] # minimum sample number that can be stored in a leaf node
bootstrap = [True, False] # method used to sample data points

random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               # 'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
```

[20] ✓ 0.1s Python

```
randmf = RandomForestClassifier(n_estimators = 100, min_samples_split = 5, min_samples_leaf= 1, max_features = 'sqrt', max_depth= 120, bootstr
```

[21] ✓ 56.1s Python

Accuracy: 0.89

-----

## 8. References

1. <https://cseweb.ucsd.edu/classes/wi15/cse255-a/reports/fa15/003.pdf>
2. <https://www.slideshare.net/MahamFRajput1/sentiment-analysis-on-amazon-movie-reviews-dataset-61842962>
3. [https://www.researchgate.net/publication/321843804\\_Sentiment\\_Analysis\\_of\\_Movie\\_Reviews\\_using\\_Machine\\_Learning\\_Techniques](https://www.researchgate.net/publication/321843804_Sentiment_Analysis_of_Movie_Reviews_using_Machine_Learning_Techniques)
4. <https://www.javatpoint.com/data-preprocessing-machine-learning>
5. <https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews>
6. <https://towardsdatascience.com/sentiment-analysis-a-how-to-guide-with-movie-reviews-9ae335e6bcb2>
7. [https://www.researchgate.net/publication/318532057\\_Sentiment\\_Analysis\\_on\\_Movie\\_Reviews](https://www.researchgate.net/publication/318532057_Sentiment_Analysis_on_Movie_Reviews)