SENTIMENT ANALYSIS USING NLP

INTRODUCTION

Sentiment analysis, a prominent application of Natural Language Processing (NLP), aims to understand and interpret the emotions, attitudes, and opinions expressed in human language. With the explosive growth of online content, social media, and customer reviews, sentiment analysis has become a crucial tool for businesses and organizations seeking to extract valuable insights from vast amounts of textual data. In the context of NLP, sentiment analysis involves using computational methods to automatically classify text into various sentiment categories, typically as positive, negative, or neutral. The process goes beyond simple keyword matching, as it leverages advanced algorithms, machine learning, and deep learning techniques to comprehend the complex nuances and context of human language.

The main objective of sentiment analysis is to derive actionable intelligence from unstructured text data. By identifying sentiments from customer feedback, social media posts, product reviews, and more, businesses can gauge public opinion about their products or services, evaluate brand perception, and make data-driven decisions to improve customer satisfaction.

Sentiment analysis can also be applied to monitor public sentiment towards specific events, political figures, or social issues, providing valuable insights for various industries, including marketing, customer service, reputation management, and market research.

Despite its potential benefits, sentiment analysis faces challenges due to the inherent complexities of human language, including sarcasm, irony, and contextual ambiguity. Researchers and developers continually refine NLP models to improve accuracy and adapt to the evolving nature of language usage.

DATASET OVERVIEW

Data Source: https://www.kaggle.com/datasets/crisbam/imdb-dataset-of-65k-movie-reviews-and-translation

Basic Statistics of Data:

- IMDB review dataset contained four columns: Ratings, Reviews, Movies, Resenhas
- Number of Reviews:149780
- Number of Movies:14205

Attribute Information:

- Review: User review in English language
- Ratings: Rating between 1 to 10
- Movies: Movie names
- Resenhas: User review translation in Portuguese language

METHODOLOGY

1. DATA PREPROCESSIND AND DATA ANALYSIS:
    - First, I made custom adjustments to the stopwords, which are common words that often add noise to the data and are not relevant for analysis.
    - Next, I built a Data Cleaning and Preprocessing Pipeline to prepare the text data for sentiment analysis. In this pipeline, I implemented several functions to perform specific tasks.

      The first function, 'remove_special_character()', was designed to eliminate words containing special characters enclosed in square brackets, such as '[^&@#!]*'. These types of words were removed as they might not contribute meaningful information to sentiment analysis.

      The second function, 'remove_url()', focused on removing words that appeared to be URLs or contained URL links. URLs are not helpful for sentiment analysis, so it was essential to get rid of them.

      In the third function, 'remove_stop_words()', I removed stop words from the corpus. Stop words are common words like "the," "is," "and," etc., which usually occur frequently in language but don't carry much contextual significance.

      Lastly, I combined all these cleaning functions into a single 'data_cleaning()' function to streamline the entire process.
    - After completing the Data Cleaning and Preprocessing Pipeline, I applied it to the reviews dataset. The resulting clean reviews were stored in a new variable named 'Reviews_clean.' This new variable contains the processed text data that is now ready for sentiment analysis, with unnecessary information, special characters, URLs, and stop words removed, making the data more suitable for analysis and modeling.

- After performing the Data Cleaning and Preprocessing Pipeline, I proceeded with the data overview step, where I aimed to gain a better understanding of the dataset and its contents.

  To begin with, I used the command 'df.isna().sum()' to check for missing values in the dataset. This allowed me to identify if there were any entries with incomplete or empty data, which could impact the analysis. By examining the number of missing values for each column, I ensured that the dataset was reasonably complete and suitable for further exploration.

  Next, I used the command 'df['Ratings'].describe()' to obtain a statistical summary of the ratings column. The description provided information such as the count of ratings, which indicated how many ratings were present in the dataset. Additionally, the output contained the minimum and maximum rating values, giving me an idea of the rating range covered by the dataset. This allowed me to understand the distribution of ratings and identify any potential outliers or unusual patterns.

  Similarly, I used 'df['Reviews'].describe()' to gain insights into the 'Reviews' column. This description provided statistical information about the reviews, such as their count, mean, standard deviation, minimum, and maximum length. By examining these values, I could determine if the reviews varied significantly in length or if they were relatively consistent in size. This knowledge could be useful in understanding the diversity of the reviews and their potential impact on the sentiment analysis process.

  Overall, performing these data overview steps helped me ensure that the dataset was well-prepared and gave me valuable insights into the characteristics of the ratings and reviews data, which would be beneficial for conducting sentiment analysis and subsequent analyses.
- Once I completed all the previous data preprocessing steps, the next phase involved visualizing the dataset to perform various analyses:

  Class Imbalance Check:

  To assess class imbalance in the dataset, I plotted a graph or chart that displayed the distribution of positive and negative reviews. Class imbalance refers to an unequal distribution of classes, where one class significantly outweighs the other. In sentiment analysis, having a class imbalance could lead to biased results. By visualizing the class distribution, I could identify whether there was an imbalance and take

appropriate steps to address it, such as using techniques like oversampling, undersampling, or class weighting to balance the dataset.

Important Words from Positive Reviews:

To identify important words from positive reviews, I used techniques like word clouds or bar charts. A word cloud visually represents words from the positive reviews, with the size of each word indicating its frequency. This helped me spot the most commonly occurring words in positive sentiments, which might include words like "excellent," "amazing," "fantastic," etc. These words give insights into what customers liked about the product or service, and they could be essential for understanding the overall positive sentiment.

Important Words from Negative Reviews:

Similarly, I used word clouds or bar charts to identify important words from negative reviews. This allowed me to visualize the most frequently mentioned words associated with negative sentiments, such as "poor," "disappointing," "bad," etc. Understanding these negative indicators helped identify potential issues or areas for improvement in the product or service. Analyzing such words could assist in addressing specific pain points and enhancing customer satisfaction.

By incorporating data visualization techniques into the analysis, I could effectively explore and understand the dataset, gain insights into class distribution, and identify key sentiments and themes expressed in the reviews. These visualizations provided a clearer picture of the dataset, making it easier to draw meaningful conclusions and make informed decisions based on the sentiment analysis results.

- After performing the data preprocessing and visualization steps mentioned earlier, I further analyzed the review data using basic statistics and the CountVectorizer technique to extract common words and n-grams from positive reviews.

  Visualization of Number of Characters in Reviews:

  I created a visualization, such as a histogram or box plot, to understand the distribution of review lengths in terms of the number of characters. This helped me gain insights into the length of reviews and whether they were generally short or long. Understanding the distribution of review lengths can be valuable in gauging how customers express their opinions and whether there are any patterns or trends in the reviews' length.

CountVectorizer Technique for Common Words, Bigram, Trigram, 4-gram, and 5-gram in Positive Reviews:

I applied the CountVectorizer technique to extract common words and n-grams (groups of n adjacent words) from the positive reviews. CountVectorizer is a text processing technique that converts text data into numerical vectors, representing the frequency of words and n-grams.

For common words, I used CountVectorizer to identify the most frequently occurring words in positive reviews. These words give valuable insights into what customers mention most often when expressing positive sentiments about the product or service.

For bigrams, trigrams, 4-grams, and 5-grams, I used CountVectorizer to capture combinations of two, three, four, and five words that appeared frequently in positive reviews. These n-grams help identify meaningful phrases or expressions that customers use when writing positive feedback. Analyzing n-grams can reveal specific patterns and provide a deeper understanding of the reasons behind positive sentiments.

By employing basic statistics and the CountVectorizer technique, I gained valuable insights into the review data, such as review length distribution and the most common words and phrases associated with positive sentiments. These analyses enriched the sentiment analysis process, offering more detailed information for businesses to improve their products or services based on customer feedback.

2. FEATURE ENGINEERING AND FEATURE SELECTION:

We all know that feature engineering is an important step that used to extract meaningful information from the data and represent it in a way that make easier for the ML Algorithms to learn and make accurate predictions.

- In the initial step of the analysis, I converted the rating data into binary format, representing positive and negative reviews with numerical values 0 and 1, respectively.

  To achieve this, I assigned the value 0 to reviews with a positive sentiment, indicating that customers expressed satisfaction or appreciation for the product or service. On the other hand, I assigned

the value 1 to reviews with a negative sentiment, indicating that customers expressed dissatisfaction or criticism.

By transforming the ratings into a binary format, I simplified the sentiment analysis task for the machine learning algorithms, as they could now focus on distinguishing between positive and negative sentiments without the need to handle multiple rating classes.

- After completing the above steps, I proceeded with vectorization using Count Vectorizer and TF-IDF Vectorizer with unigram, bigram, and trigram representations.

  Vectorization with Count Vectorizer and TF-IDF Vectorizer (Unigram, Bigram, and Trigram):

  I applied both Count Vectorizer and TF-IDF Vectorizer techniques to convert the text data into numerical representations. For unigram vectorization, each word was treated as an individual feature. For bigram vectorization, consecutive pairs of words were treated as features, and for trigram vectorization, consecutive triplets of words were considered as features. This allowed us to capture not only single words but also sequences of words, which might carry more context and meaning.

  Feature Importance with Logistic Regression and Count Vectorizer (Unigram, Bigram, and Trigram):

  To determine the importance of the features generated by the Count Vectorizer, I used logistic regression. Logistic regression is a binary classification algorithm that can identify the importance and influence of each feature in predicting the positive or negative sentiment of the reviews. By examining the coefficients of the logistic regression model, I could assess which words or n-grams had a stronger impact on sentiment prediction.

  Feature Importance with TF-IDF Vectorizer and Logistic Regression (Unigram, Bigram, and Trigram):

  Similarly, I applied logistic regression to the features generated by the TF-IDF Vectorizer. The TF-IDF scores indicate the importance of each word or n-gram relative to the entire document collection. By combining the TF-IDF Vectorizer with logistic regression, I could identify the most influential words or n-grams in predicting sentiment, considering their uniqueness and significance across the entire dataset.

By performing these analyses, I gained valuable insights into the importance of different features extracted from the text data using various vectorization techniques. This allowed me to identify which words and combinations of words were most indicative of positive or negative sentiments in the reviews. These findings were instrumental in building more accurate and interpretable sentiment analysis models, helping businesses and organizations better understand customer feedback and make data-driven decisions for improving products, services, or customer experiences.

- After performing vectorization with Count Vectorizer and TF-IDF Vectorizer for unigram, bigram, and trigram representations, I proceeded with the feature selection technique using the Chi-Squared method.

  The Chi-Squared method is a statistical test that measures the dependency between two categorical variables. In the context of feature selection for sentiment analysis, it helps us identify which words or n-grams (unigram, bigram, and trigram) are most relevant and influential in predicting sentiment.

  To apply the Chi-Squared method, I calculated the Chi-Squared statistic for each feature (word or n-gram) in relation to the target variable (sentiment labels: positive or negative). The Chi-Squared statistic quantifies the difference between the expected frequency and the observed frequency of each feature in the positive and negative sentiment classes.

  Higher Chi-Squared values indicate a stronger association between a feature and the sentiment labels, suggesting that the feature is more discriminatory in determining positive or negative sentiment. Consequently, features with higher Chi-Squared scores are considered more important for sentiment analysis.

  By using the Chi-Squared method for feature selection on unigram, bigram, and trigram representations, I was able to identify the most informative and discriminating words and n-grams. This process helped to reduce the dimensionality of the feature space, selecting the most relevant features while disregarding less informative ones.

  Feature selection using the Chi-Squared method is essential for improving the model's efficiency, reducing computational overhead, and enhancing its performance by focusing on the most relevant and

significant features. This step ensures that the sentiment analysis model is better equipped to make accurate predictions and capture the key factors influencing sentiment in the text data.

3. MODEL SELECTION AND HYPERMETER TUNING:
    - In the model selection phase, I explored various machine learning algorithms to identify the best model for sentiment analysis. The models I considered were:
      Logistic Regression Model: Logistic regression is a simple and effective algorithm for binary classification tasks like sentiment analysis. It estimates the probability of an instance belonging to a particular class and makes predictions based on a threshold.
      Decision Tree Classifier: Decision trees are versatile algorithms that can handle both classification and regression tasks. They create a tree-like structure, where each node represents a decision based on a feature, leading to different branches representing possible outcomes.
      Decision Tree Classifier with Depth 11: To mitigate the risk of overfitting, I employed a decision tree classifier with a specific depth of 11. By limiting the depth of the tree, I aimed to prevent it from becoming too complex and overly specialized to the training data.
    - To make an informed choice, I evaluated the performance of each model using various metrics, such as precision, recall, F1 score, and accuracy. These metrics allowed me to assess how well each model predicted sentiment and handled positive and negative instances.
    - After selecting the most promising model, I proceeded with hyperparameter tuning to optimize its performance further. For this purpose, I used the logistic regression model and performed hyperparameter tuning using grid search.
    - Grid search involves specifying a range of hyperparameter values and systematically evaluating the model's performance with each combination of these values. This exhaustive search allows us to find the best combination of hyperparameters that yields the highest performance.
      By tuning the hyperparameters of the logistic regression model using grid search, I aimed to improve its ability to generalize to new data and achieve better predictive accuracy. This iterative process ensured that

the model's hyperparameters were carefully chosen, leading to a more robust and effective sentiment analysis model.

Overall, the model selection technique involved comparing different machine learning algorithms and hyperparameter tuning to find the optimal configuration, resulting in a sentiment analysis model with enhanced performance and generalization capabilities.

4. MODEL EVALUATION:
    - In the model evaluation phase, I utilized the logistic regression model to train and evaluate the sentiment analysis model. First, I split the data into training and testing sets, using the training data to train the logistic regression model. Once the model was trained, I evaluated its performance using various metrics to gauge its effectiveness in predicting sentiment.
    - The evaluation metrics I calculated were as follows:

      Precision Score: Precision is the ratio of true positive predictions to the total positive predictions made by the model. It measures how many of the positive predictions are correct.

      Recall Score: Recall, also known as sensitivity or true positive rate, is the ratio of true positive predictions to the total actual positive instances in the dataset. It measures the model's ability to correctly identify positive instances.

      AUC Score (Area Under the Receiver Operating Characteristic Curve): AUC is a metric that evaluates the model's ability to distinguish between positive and negative instances. It provides an aggregate measure of the model's performance across various classification thresholds.

      F1 Score: The F1 score is the harmonic mean of precision and recall. It balances precision and recall and is useful when there is an uneven class distribution.

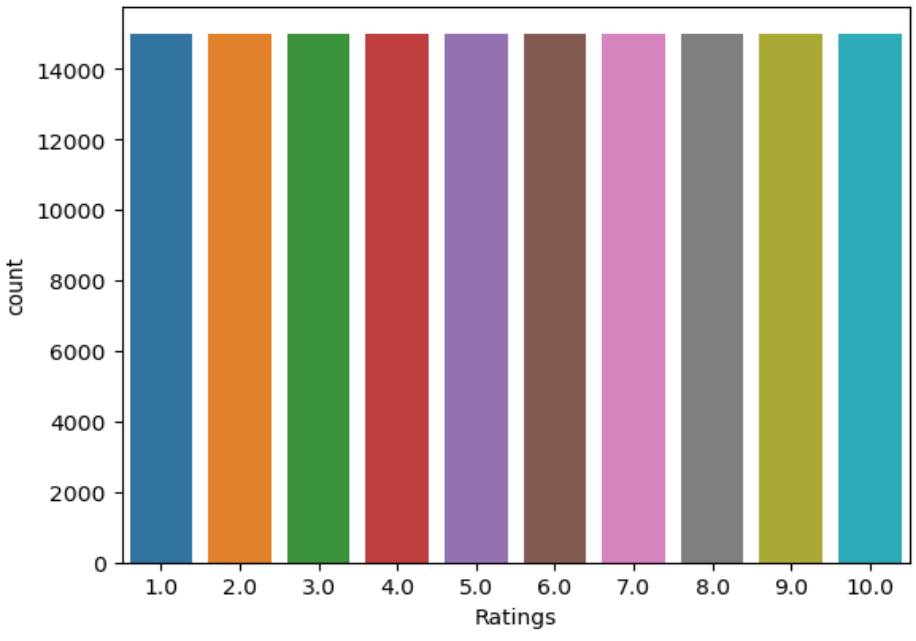
      Accuracy Score: Accuracy is the ratio of correctly classified instances to the total instances in the dataset. It provides an overall measure of the model's correctness.
    - In addition to evaluating the logistic regression model directly, I also used a logistic regression pipeline, which combined feature extraction, feature selection, and logistic regression into a single pipeline. I calculated the same evaluation metrics for the logistic regression

pipeline to compare its performance with the standalone logistic regression model.

- After evaluating the models, I used the trained logistic regression model or the logistic regression pipeline to predict the sentiment of the test data. This allowed me to assess how well the model generalized to new, unseen data.
- To gain a more detailed understanding of the model's predictions, I calculated a confusion matrix. The confusion matrix provided a breakdown of the model's predictions, showing the number of true positives, false positives, true negatives, and false negatives. This helped me assess the model's accuracy and potential misclassifications. Overall, this thorough model evaluation process enabled me to assess the performance and effectiveness of the sentiment analysis model, providing valuable insights into its predictive capabilities and generalization to new data.

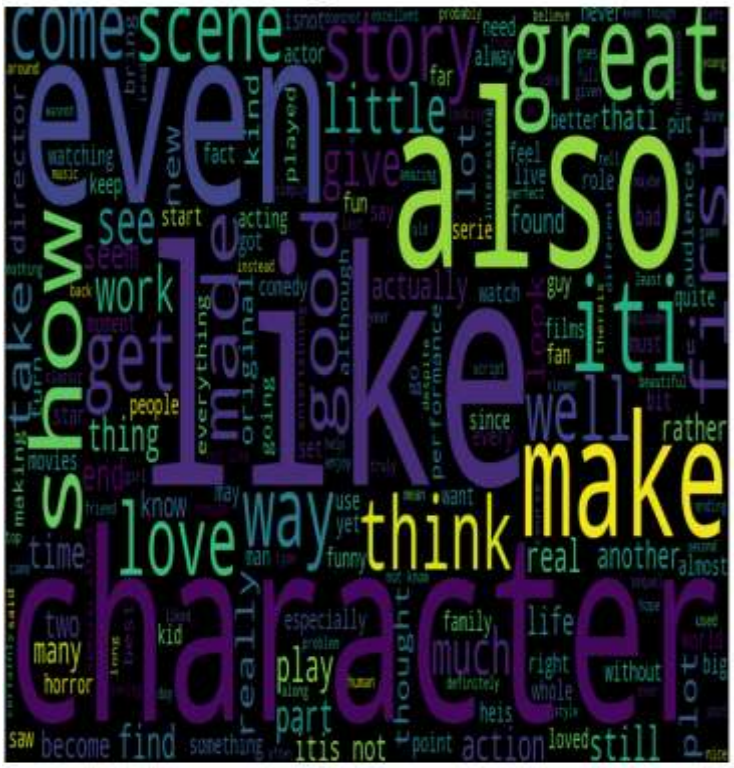
RESULT

```
#CHECKING CLASS IMBALANCE
sns.countplot(x=df['Ratings'])
plt.show()
print(df['Ratings'].value_counts())
```

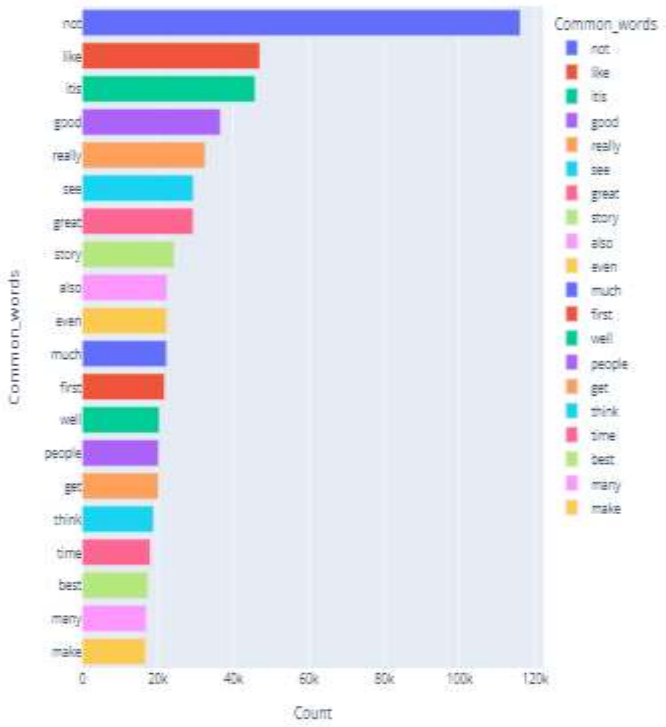


Positive Reviews

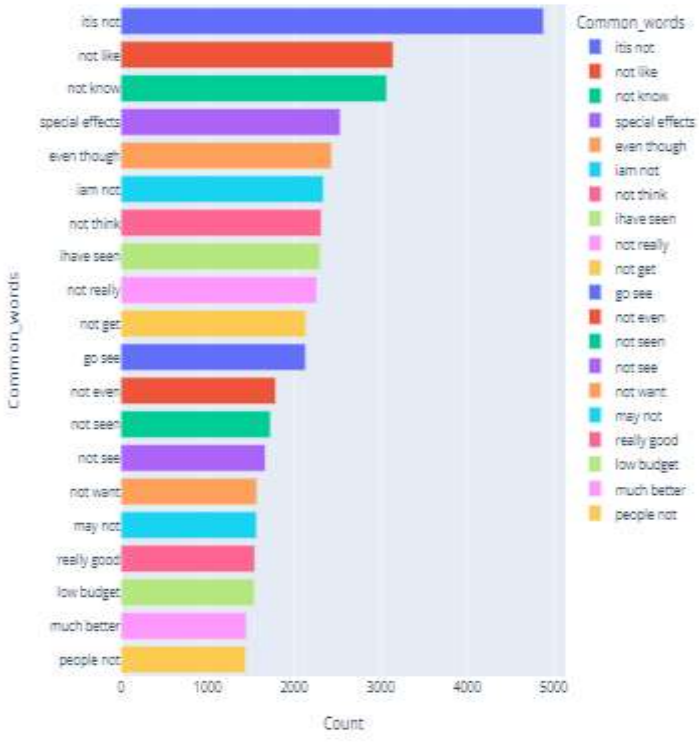
Negative Reviews



Commmon Words in Positive Reviews



Commmon bigram in Positive Reviews



Feature	Score
able	0.3855763334671161
absolutely	0.3241495564157434
across	-0.1852367803395897
act	-0.8964407752420137
acting	-1.4019696475626178
action	2.2073924389131543
actor	-1.0910723106099058
actress	-0.7554724820160148
actually	-0.3167104553976524
add	0.2876891530476976
alien	-0.36006659308783706
almost	-0.2297937819636975
along	0.8679735208490931
already	-0.9798291130343953
also	1.2452934390084898
although	1.341825978160732
always	1.6205478949265952
amazing	4.250174862244916
american	0.45043226777744527
amount	0.29027880446799503
another	-1.109177218352053
anyone	0.25932910491539535
anything	-1.812458435016667
arenot	-0.4581620907155794
around	-0.8480478223290929
art	0.4023488886154831
attempt	-3.880314246611631
audience	0.496468534586501

Feature	Score
absolutely nothing	-2.72518257523347
academy award	1.1447846021047434
act like	-1.105371859006979
acting good	1.21844487946788
acting not	-0.8568447355032708
action movie	1.4293126701263865
action not	0.9455505366978866
action scene	0.6327705951055316
action sequence	0.563084494608206
actor actress	-0.172197878481703
actor not	-0.9207112445030657
actually not	-0.26985848059591055
actually pretty	0.16450967791333904
actually quite	0.03344316083528684
almost every	-0.2757162565576098
also good	1.9780101479696983
also great	2.713680814076092
also not	-0.3833103781535998
although not	0.9539308070865992
anything else	-0.813408575716249
bad acting	-2.335095696712584
bad bad	-2.5256936020199054
bad guy	0.13592075403370305
bad itis	-1.8365595410678315
bad movie	-1.8654334198863245
bad not	-1.166075829362868
bad review	2.5171453346418797
bad thing	0.47103354834119016
best friend	0.45818678493932385

```
Precision Score for Logistic Regression: 0.8705
Recall Score for Logistic Regression: 0.8705
AUC Score for Logistic Regression: 0.9436600021483178
F1 Score for Logistic Regression: 0.8705024138091288
Accuracy Score for Logistic Regression: 0.8705
Precision Score for Logistic Regression Pipeline: 0.8721944444444445
Recall Score for Logistic Regression Pipeline: 0.8721944444444445
AUC Score for Logistic Regression Pipeline: 0.9446373463458841
F1 Score for Logistic Regression Pipeline: 0.8721966021320475
Accuracy Score for Logistic Regression Pipeline: 0.8721944444444445
CPU times: total: 3min 1s
Wall time: 3min 18s
```

```
y_predict=model_1.predict(x_test_tfidf)
y_predict_prob=model_1.predict_proba(x_test_tfidf)[:,1]
print(y_predict)
print(y_predict_prob)
```

```
['0' '1' '0' ... '0' '1' '1']
[0.30623328 0.87714766 0.04000807 ... 0.13194534 0.68159395 0.88398997]
```

```
confusion_matrix_plot(y_test,y_predict)
```