RBAN 3263 BIG DATA ALGORITHMS PAIR ASSIGNMENT

Twitter Sentiment Analysis on Malaysia Election

Course Lecture :

Ts Dr Fadilla A'tyka Nor Rasyid

Project :

Choosing the Right Algorithm for Big Data analysis

Acknowledgements

Most reports conclude with a page acknowledging the contributions of the people who worked tirelessly on the projects mentioned within.

KHAIRUL NAZWAN BIN DZULQARNAIN

BBA in AI

(B22361205)

FARAH ADIBA BINTI FAHIMI

BBA in AI

(B23761190)

Introduction

Our project, **"Twitter Sentiment Analysis on Malaysia Election Period,"** classifies tweets into positive, negative, and neutral sentiments, providing insights into public opinion.

We deployed **Naive Bayes Classifier Multinomial** and **Support Vector Machine (SVM)** algorithms to efficiently process and analyze the labeled twitter dataset.

Algorithms Used:

- Naive Bayes Classifier Multinomial
- Support Vector Machine (SVM)

These algorithms efficiently process and analyze the labeled Twitter dataset.

Analysis on Malaysia Election Period

DataSet Description

"Twitter Sentiment Analysis Dataset" containing tweets related to the Malaysia election.

SIZE AND FORMAT

- The dataset is in CSV format with a semicolon (;) delimiter.
- 15,912 entries and 9 columns.
- It contains multiple columns including username, date, time, replies_count, retweets_count, likes_count, steaming data, Compound_Score, and Sentiments.

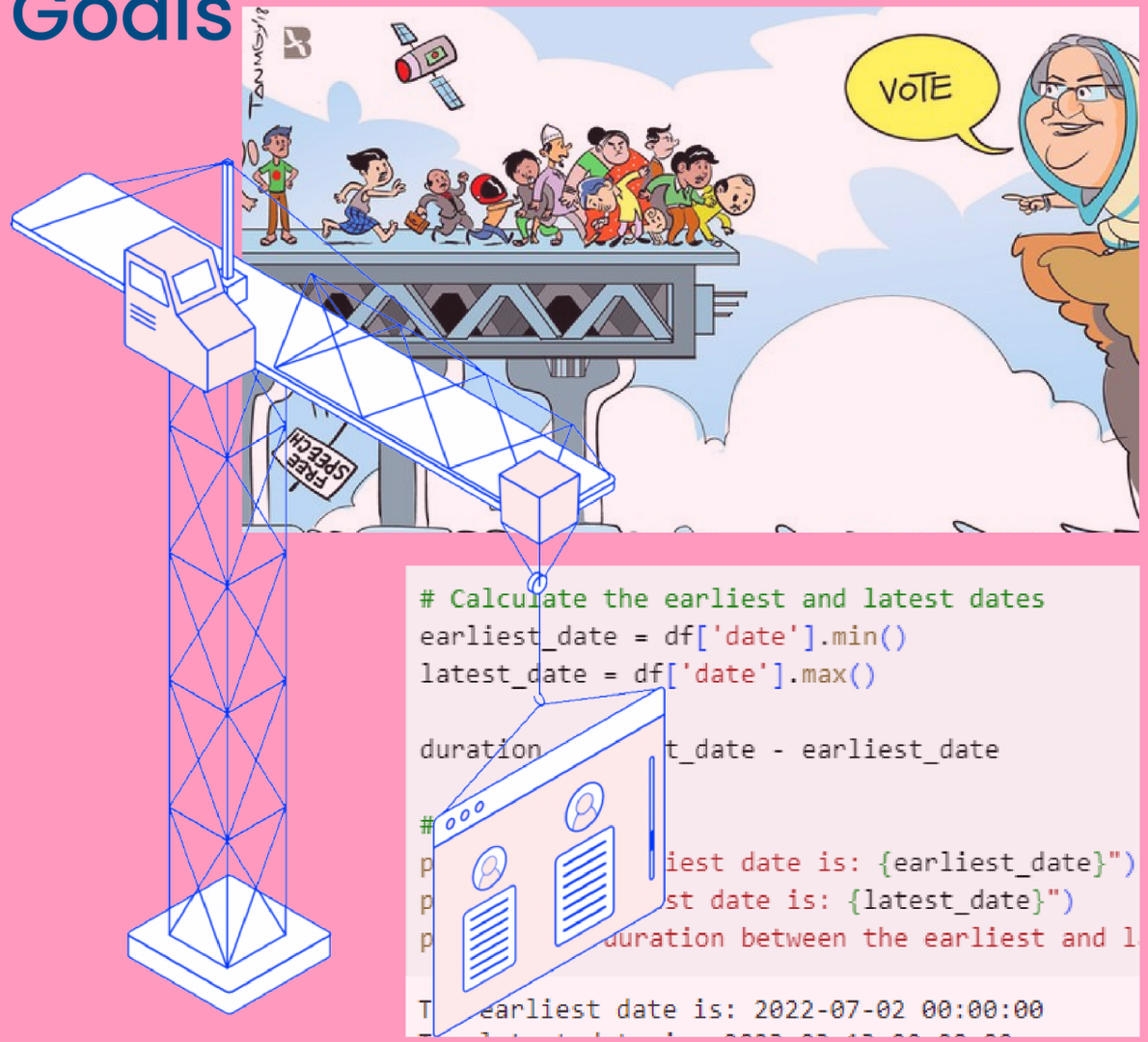
STRUCTURE

- username: Twitter account username.
- date: Date of the tweet.
- time: Time of the tweet.
- replies_count: Number of replies.
- retweets_count: Number of retweets.
- likes_count: Number of likes.
- steaming data: Tweet content.
- Compound_Score: Sentiment score.
- Sentiments: Sentiment label (positive, negative, neutral).

TYPES OF INFORMATION

- User Interaction Metrics: Engagement data (replies, retweets, likes).
- Content Data: Text of tweets.
- Temporal Information: Date and time of tweets.
- Sentiment Analysis: Sentiment scores and labels.

Data Analysis Goals



Objective

Classify tweets related to the Malaysia Election Period into positive, negative, and neutral sentiments using advanced machine learning algorithms.

Goal

Identify and track overall sentiment trends and patterns over the election period, providing a dynamic view of public opinion as events unfold.

Python Step 1-3

Importing Libraries, Loading the Dataset and Data Cleaning and Preprocessing

PANDAS, MATPLOTLIB, SEABORN, NUMPY, NLTK, SCIKIT-LEARN

Twitter Dataset

```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import numpy as np
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterSt
from sklearn.model_selection import t
from sklearn.linear_model import Logi
```

IMPORT NECESSARY LIBRARIES

DF.INFO()

```
df = pd.read_csv("TWSentiment.csv", sep=";")
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15912 entries, 0 to 15911
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   username        15912 non-null  object
 1   date            15907 non-null  object
 2   time            15907 non-null  object
 3   replies_count   15907 non-null  float64
 4   retweets_count  15911 non-null  float64
 5   likes_count     15911 non-null  float64
 6   steaming data   15912 non-null  object
 7   Compound_Score  15908 non-null  float64
 8   Sentiments      15912 non-null  object
dtypes: float64(4), object(5)
```

LOADING THE DATASET

DF['SENTIMENTS']

```
df.head()
```

	username	date	time	replies_count	retweets_count	like
0	tvonenews	30/08/2022	10:02:37	2964.0	4058.0	
1	idextratime	15/02/2023	12:38:10	2715.0	4492.0	
2	detikcom	21/09/2022	14:27:25	2360.0	1193.0	
3	burhanmuhtadi	30/08/2022	20:20:23	1569.0	11600.0	
4	ask rl	22/02/2023	18:40:03	1527.0	271.0	

```
df['Sentiments'] = df['Sentiments'].replace('Negatif', 'Negative')
df['Sentiments'] = df['Sentiments'].replace('Positif', 'Positive')
df['Sentiments'] = df['Sentiments'].replace('Netral', 'Neutral')
```

```
df.head()
```

	username	date	time	replies_count	retweets_count	like
0	tvonenews	30/08/2022	10:02:37	2964.0	4058.0	
1	idextratime	15/02/2023	12:38:10	2715.0	4492.0	

CLEANING AND PREPROCESSING

Python Step 4-6

Descriptive Statistics, Filter Data Related to Malaysia and Extract Specific Rows

DF.DESCRIBE()

```
df.describe()
```

	replies_count	retweets_count	like
count	15907.000000	15911.000000	1591
mean	7.628780	13.925775	6
std	54.868947	159.337774	71
min	1.000000	0.000000	
25%	1.000000	0.000000	
50%	1.000000	0.000000	
75%	2.000000	3.000000	1
max	2964.000000	11600.000000	4202

STATISTICS

DF[CONTAINS_MALAYSIA]

Exploration Data Analysis

Hot news about Malaysia

```
contains_malaysia =df["steaming data"].str.contains('Malaysia', case=Fa

new_malaysia = df[contains_malaysia]

print(new_malaysia)
```

```
             username        date      time  replies_count  retweets_c
9023         good_on_g  13/07/2022  21:56:24            1.0
9642   the_agastyfa_94  21/11/2022  13:38:45            1.0
9646   the_agastyfa_94  21/11/2022  08:56:01            1.0
11799            blood  28/12/2022  17:11:58            1.0
15864   hockeymalaysia  12/08/2022  13:09:32            1.0

       likes_count                                      steaming data
9023           0.0  male female buya yahya ms glow nguyen squidgam...
9642           0.0  Beritartm rtmmalaysia hangs the Malaysian parl...
9646           0.0  Malaysian parliament hanged Friday elect PM Se...
11799          0.0  suriyamaara aravosambo selenofyl chandersbc ma...
15864          3.0  womens indoor asia cup bangkok thailand pool b...

       Compound_Score Sentiments
9023           4588.0   Positive
9642          -1779.0   Negative
```

FILTER DATA

DF.LOC[9642, 'STEAMING DATA']

```
row_9642 = df.loc[9642, 'stea
row_9646 = df.loc[9646, 'stea

print(f'Election New: {row_96
print(f'Election New: {row_96
```

```
Election New: Beritartm rtmma
Election New: Malaysian parli
```

EXTRACT AND DISPLAY

Python Step 7-9

Calculate Data Collection Duration, Data Visualization and Correlation Analysis

DF['DATE']

Duration the data was Scrapped

```
# Convert the 'date' column to datetime format
df['date'] = pd.to_datetime(df['date'], errors='coerce')

# Calculate the earliest and latest dates
earliest_date = df['date'].min()
latest_date = df['date'].max()

duration = latest_date - earliest_date

# Print results
print(f"The earliest date is: {earliest_date}")
print(f"The latest date is: {latest_date}")
print(f"The duration between the earliest and latest dates
```

```
The earliest date is: 2022-07-02 00:00:00
The latest date is: 2023-03-13 00:00:00
The duration between the earliest and latest dates is: 254
<ipython-input-10-6e0b644e5fe9>:2: UserWarning: Parsing da
  df['date'] = pd.to_datetime(df['date'], errors='coerce')
```

DATA COLLECTION

SNS.HISTPLOT(DF[])

Exploration Data Analysis

Hot news about Malaysia

```
contains_malaysia =df["steaming data"].str.contains('Malaysia', case=Fa

new_malaysia = df[contains_malaysia]

print(new_malaysia)
```

	username	date	time	replies_count	retweets_c
9023	good_on_g	13/07/2022	21:56:24	1.0	
9642	the_agastyfa_94	21/11/2022	13:38:45	1.0	
9646	the_agastyfa_94	21/11/2022	08:56:01	1.0	
11799	blood	28/12/2022	17:11:58	1.0	
15864	hockeymalaysia	12/08/2022	13:09:32	1.0	

	likes_count	steaming data
9023	0.0	male female buya yahya ms glow nguyen squidgam...
9642	0.0	Beritartm rtmmalaysia hangs the Malaysian parl...
9646	0.0	Malaysian parliament hanged Friday elect PM Se...
11799	0.0	suriyamaara aravosambo selenofyl chandersbc ma...
15864	3.0	womens indoor asia cup bangkok thailand pool b...

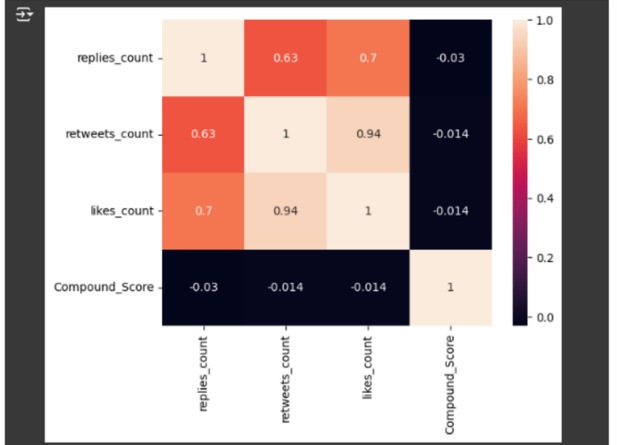
	Compound_Score	Sentiments
9023	4588.0	Positive
9642	-1779.0	Negative

DATA VISUALIZATION

CORRELATION_METRIC = DF.CORR()

Correlation on sentiment

```
correlation_metric =  df.corr
sns.heatmap(df[['replies_count', 'retweets_count',
       'likes_count','Compound_Score']].corr(), annot=True)
plt.show()
```



CORRELATION ANALYSIS

Data Visualization

Visualize the distribution of compound scores and sentiments.

This helps to understand the overall sentiment distribution and polarity of the tweets.

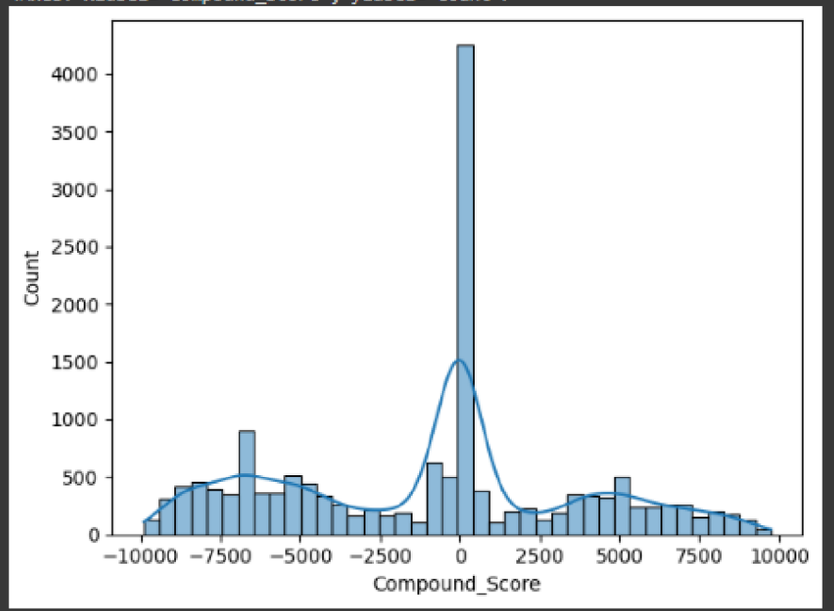
Visualised data

```
df.columns
```

```
Index(['username', 'date', 'time', 'replies_count', 'retweets_count',
       'likes_count', 'steaming data', 'Compound_Score', 'Sentiments'],
      dtype='object')
```

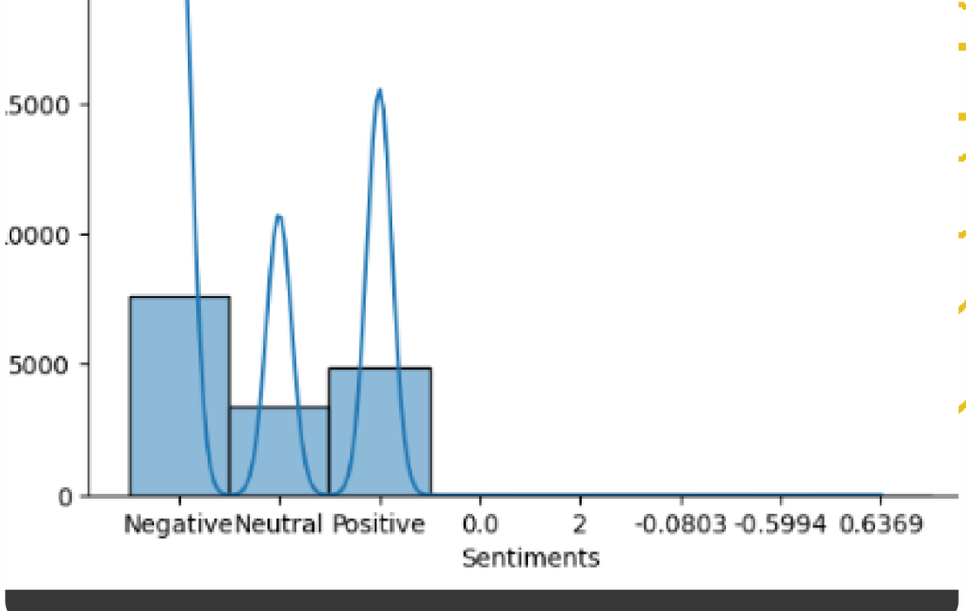
```
[ ] sns.histplot(df['Compound_Score'], kde=True)
```

```
<Axes: xlabel='Compound_Score', ylabel='Count'>
```




```
istplot(df['Sentiments'], kde=True)
```

```
: xlabel='Sentiments', ylabel='Count'>
```

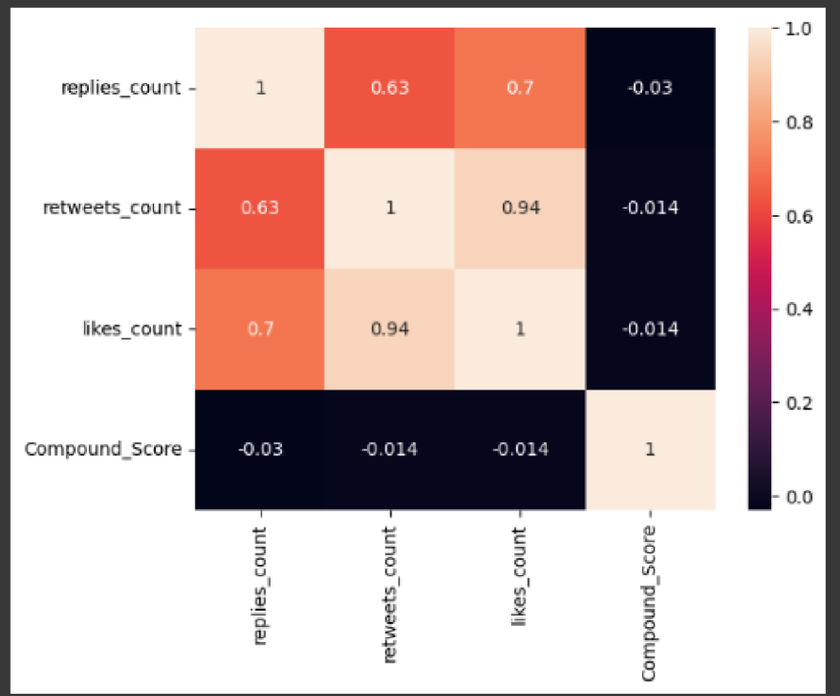


Python Step 8

Python Step 9

Correlation on sentiment

```
[ ]  correlation_metric =  df.corr
     sns.heatmap(df[['replies_count', 'retweets_count',
           'likes_count','Compound_Score']].corr(), annot=True)
     plt.show()
```



Correlation Analysis

Calculate and visualize the correlation between numerical features.

This identifies potential relationships between variables.

Python Step 10-12

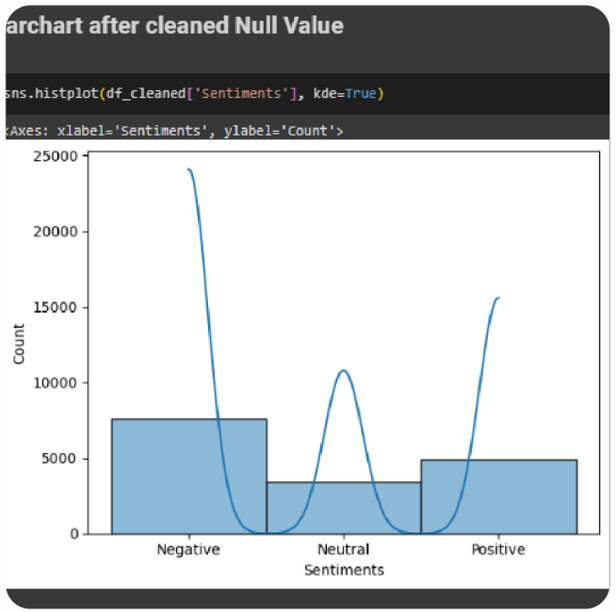
Data Cleaning (Dropping Null Values), Visualization after Cleaning, and Sentiment Count Visualization

DF_CLEANED.INFO()

```
Data cleaning (Drop all NULL values)

[ ]  df_cleaned = df.dropna()

     df_cleaned.info()

<class 'pandas.core.frame.DataFrame'>
Index: 15907 entries, 0 to 15911
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   username        15907 non-null  object
 1   date            15907 non-null  datetime64[ns]
 2   time            15907 non-null  object
 3   replies_count   15907 non-null  float64
 4   retweets_count  15907 non-null  float64
 5   likes_count     15907 non-null  float64
 6   steaming data   15907 non-null  object
 7   Compound_Score  15907 non-null  float64
 8   Sentiments      15907 non-null  object
dtypes: datetime64[ns](1), float64(4), object(4)
memory usage: 1.2+ MB
```

REMOVE ROWS WITH NULL VALUES

SNS.HISTPLOT(DF_CLEANED['SENTIMENTS']

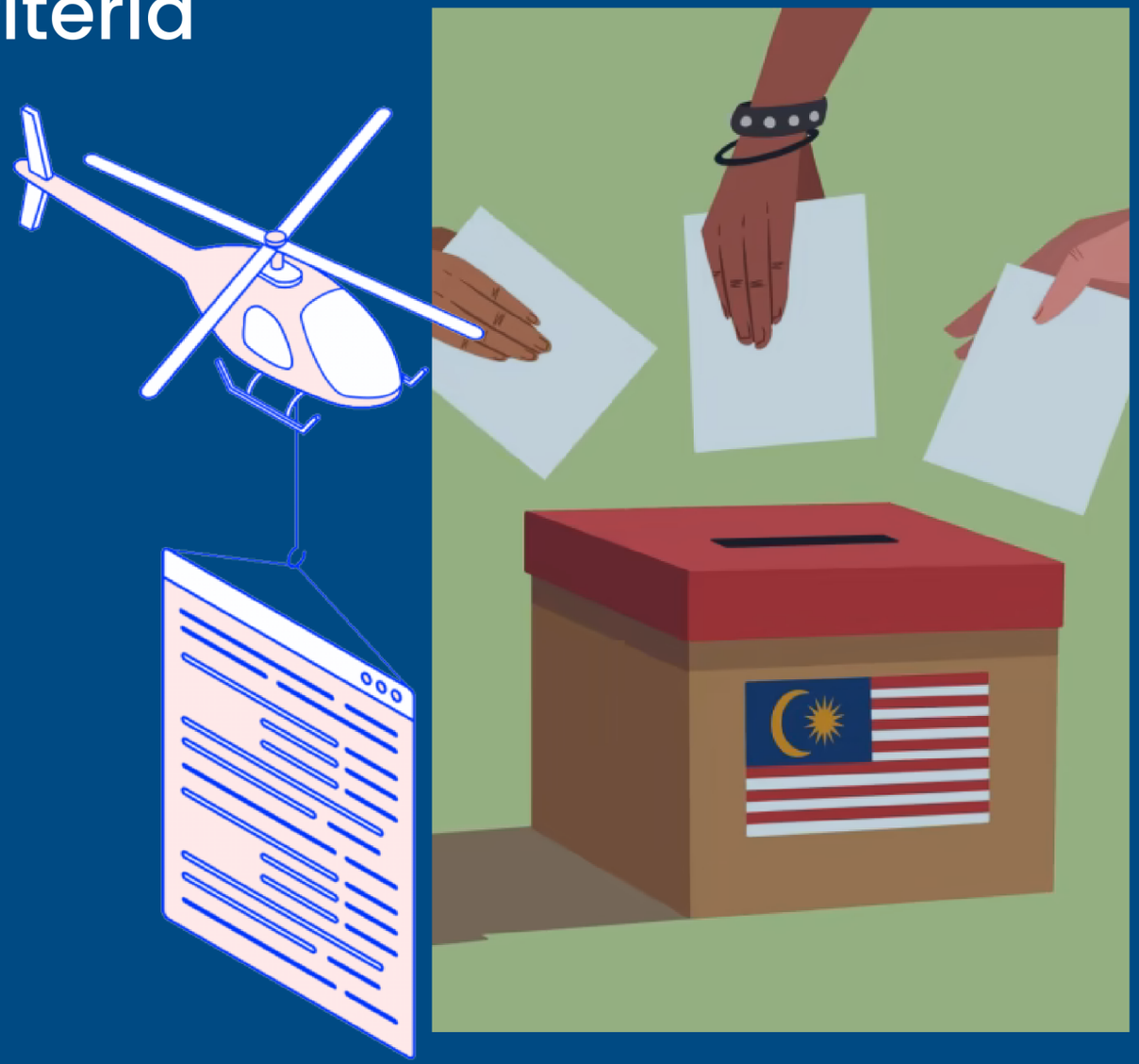


AFTER CLEANING

SENTIMENT_COUNT = DF_CLEANED['SENTIMENTS'].VALUE_COUNTS()

BAR PLOT OF SENTIMENT COUNTS

Algorithm Selection Criteria



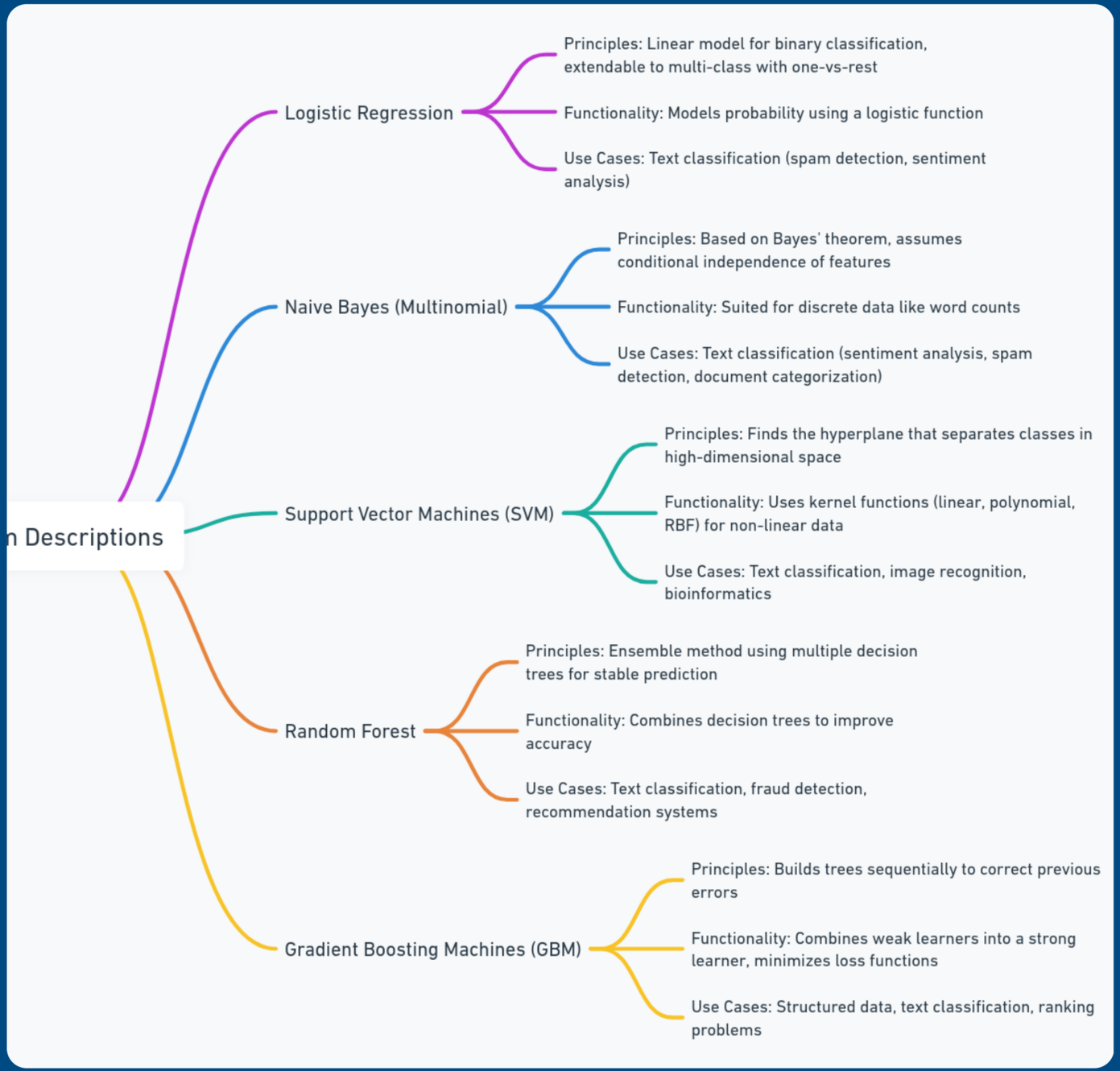
Types of Insights and Significance

- Quantitative analysis of sentiment distribution across different phases of the election.
- The ability to track changes in public opinion in real-time can help stakeholders adapt strategies and responses effectively, potentially influencing election outcomes.

Research and Analyze Relevant Algorithms

1. **Logistic Regression**
2. **Naive Bayes (Multinomial)**
3. **Support Vector Machines (SVM)**
4. **Random Forest**
5. **Gradient Boosting Machines (GBM)**

For Twitter Sentiment Analysis, several algorithms can be considered based on their suitability for text classification tasks.



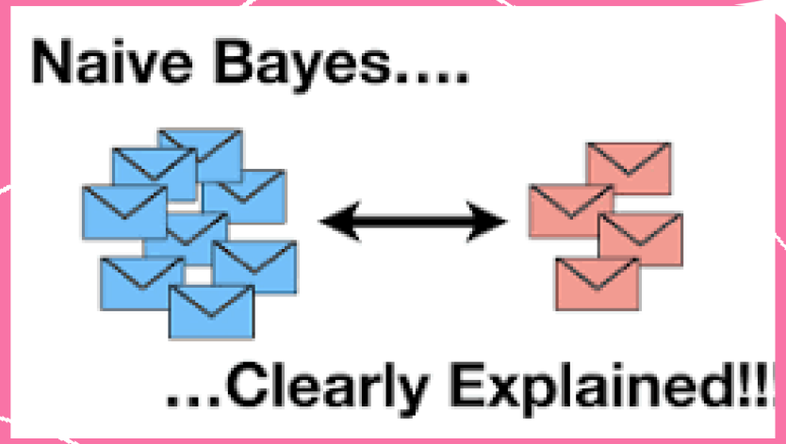
Algorithm Descriptions

Logistic Regression	Naive Bayes (Multinomial)	Support Vector Machines (SVM)
• Principles: Logistic Regression is a linear model used for binary classification, extended to multi-class classification using techniques like one-vs-rest.	• Principles: Based on Bayes' theorem, it assumes that the features (words) are conditionally independent given the class.	• Principles: SVM aims to find the hyperplane that best separates the classes in a high-dimensional space.
• Functionality: It models the probability of class membership using a logistic function.	• Functionality: Multinomial Naive Bayes is particularly suited for discrete data like word counts.	• Functionality: Can use different kernel functions (linear, polynomial, RBF) to handle non-linear data.
• Use Cases: Frequently used for text classification tasks such as spam detection and sentiment analysis.	• Use Cases: Ideal for text classification tasks like sentiment analysis, spam detection, and document categorization.	• Use Cases: Used in text classification, image recognition, and bioinformatics.

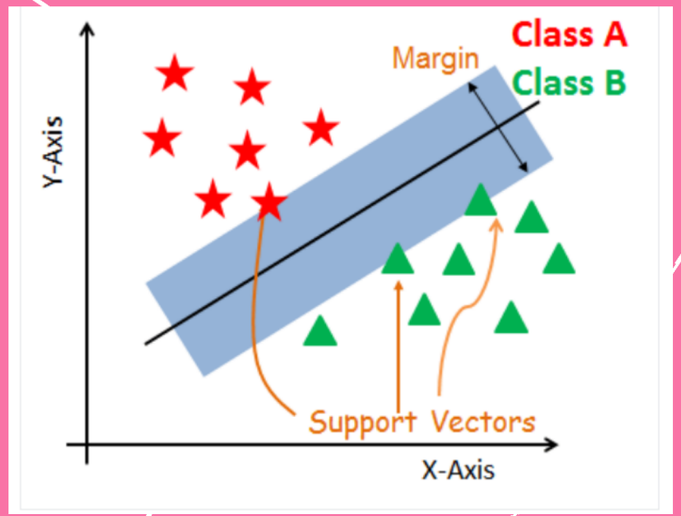
Evaluation of Algorithms

Logistic Regression	**Naive Bayes (Multinomial)**	**Support Vector Machines (SVM)**
Strengths: • Simple and easy to implement. • Good for binary and multi-class classification. • Interpretable results.	Strengths: • Fast and efficient. • Works well with small datasets. • Handles high-dimensional data well.	Strengths: • Effective in high-dimensional spaces. • Works well with a clear margin of separation.
Weaknesses: • Assumes a linear relationship between features and the log-odds of the outcome. • Not suitable for very complex relationships.	Weaknesses: • Assumes feature independence, which is often not true in practice. • Can be outperformed by other algorithms with more complex relationships.	Weaknesses: • Computationally intensive for large datasets. • Requires careful tuning of parameters and choice of kernel.
Trade-offs: • Can be less accurate than more complex models but is faster and interpretable.	Trade-offs: • Simple and efficient but relies heavily on the independence assumption.	Trade-offs: • Can provide high accuracy but at the cost of higher computational resources.

Selected Algorithms: Naive Bayes Classifier Multinomial and Support Vector Machine (Linear).

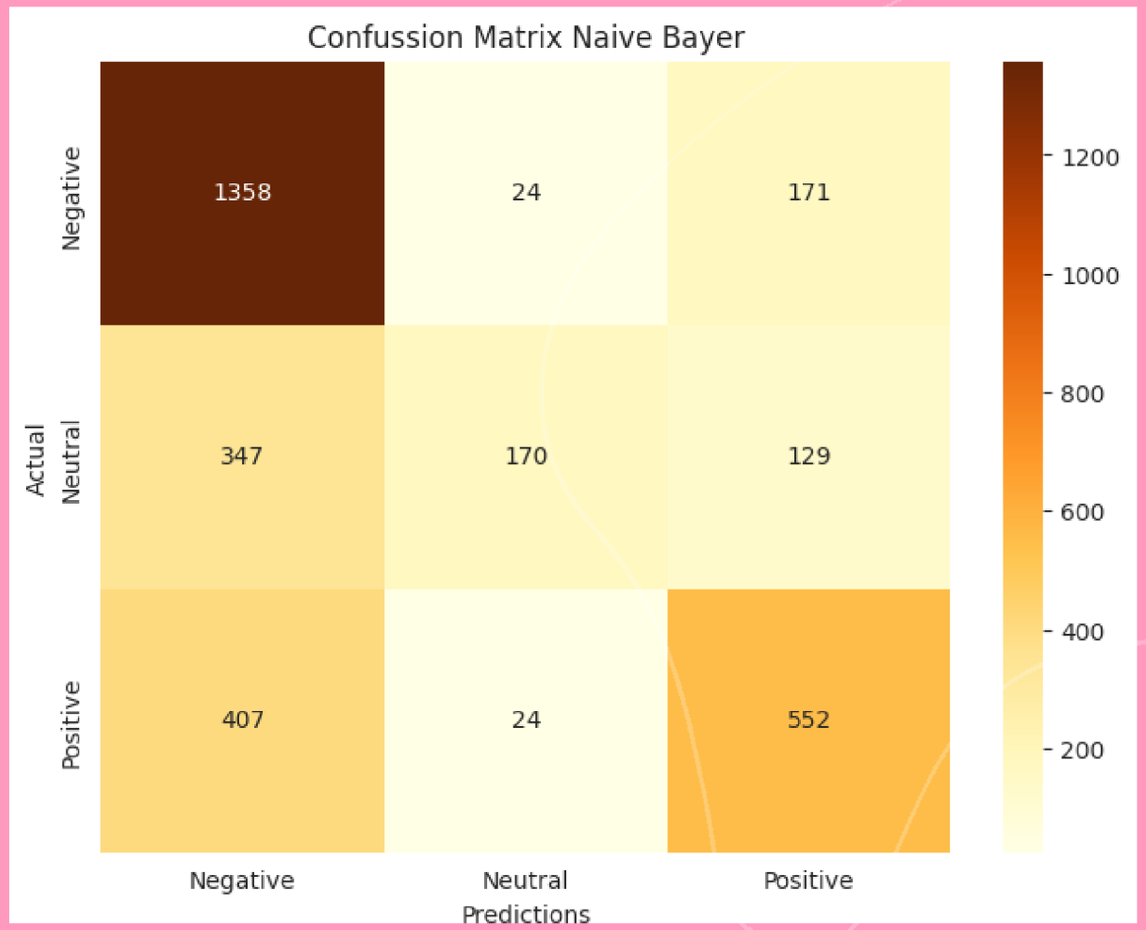


Naive Bayes provides probabilistic, assumes features are conditionally independent, and works well with text data.

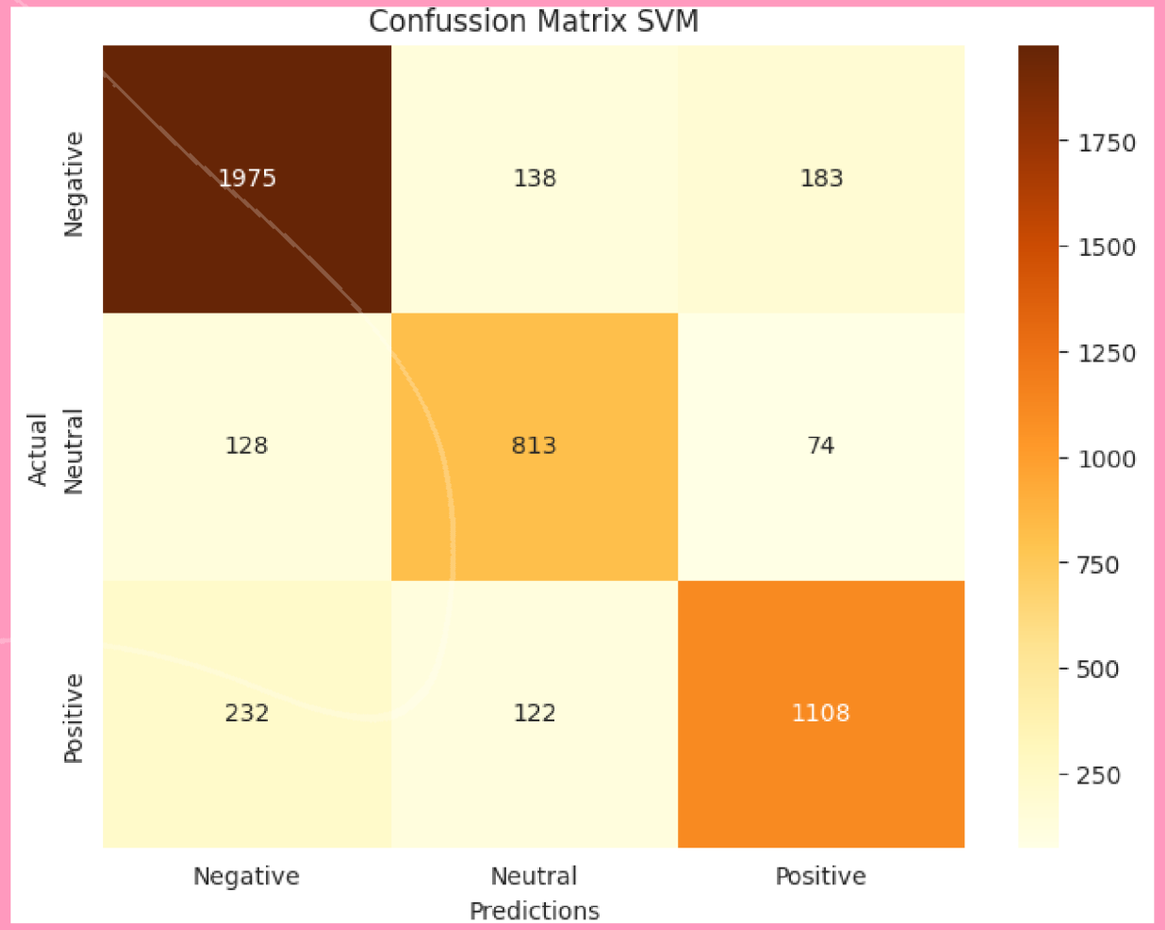


SVM works by trying to find the best hyperplane that separates data points into different classes.

Naive Bayes Classifier Multinomial



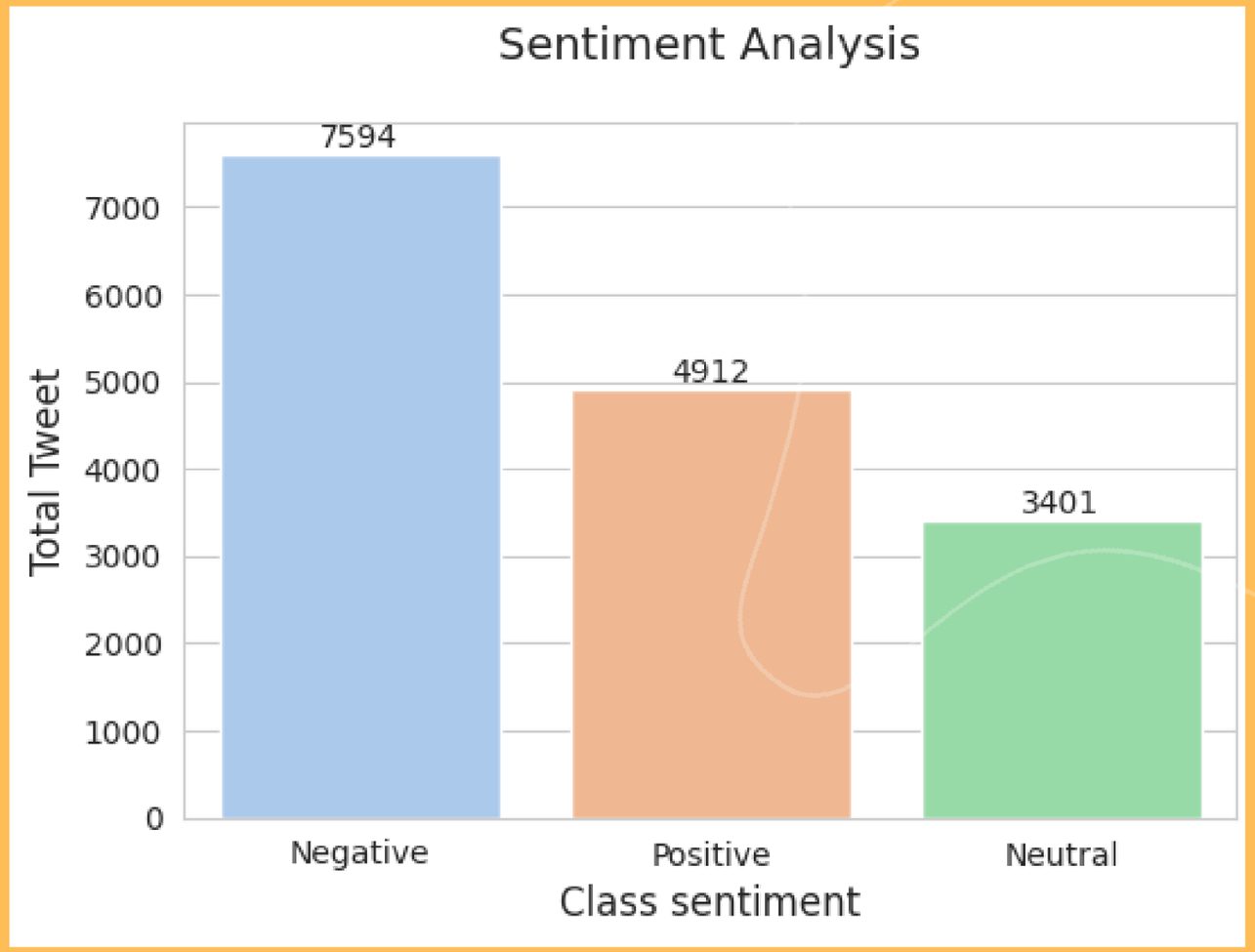
Support Vector Machine (Linear)

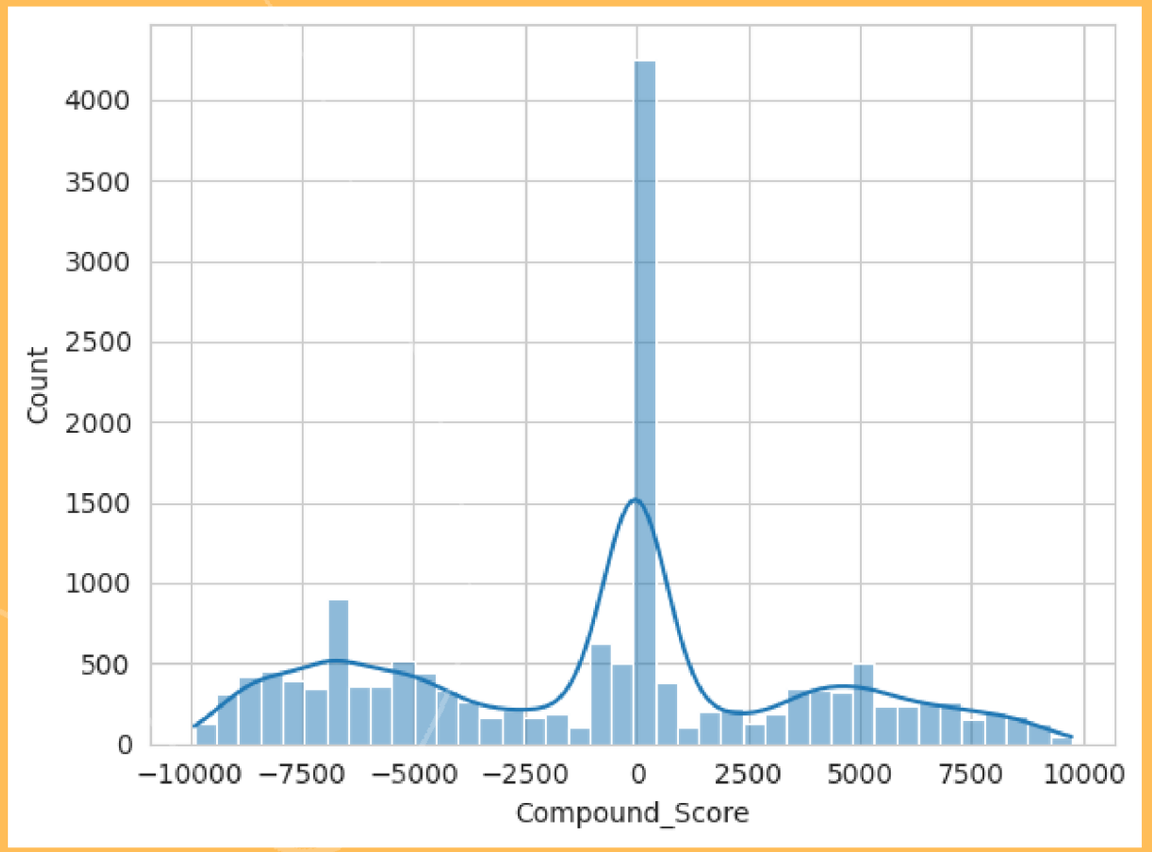


Research and Analysis

Naive Bayes Classifier

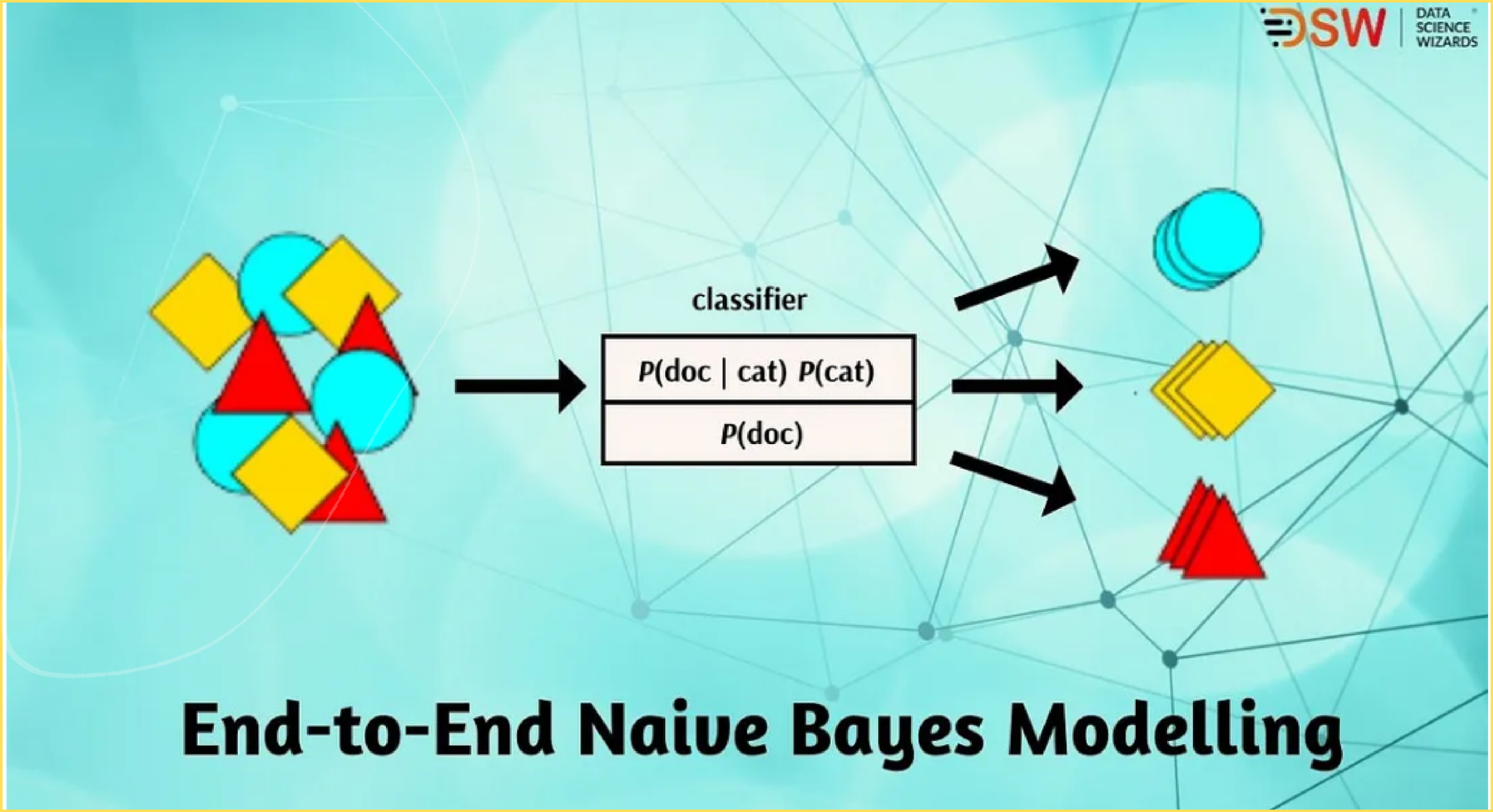
- **Principles:** Naive Bayes assumes feature independence within the dataset.
- **Functionality:** This classifier calculates the probabilities of each category and classifies the data based on the highest probability. It is particularly efficient with discrete data such as words in text.
- **Simplicity:** Easy to implement and requires a small amount of training data to estimate necessary parameters.
- **Weaknesses :** Assumption of Independence May lead to errors in presence of feature dependency.





Compound_Score Means

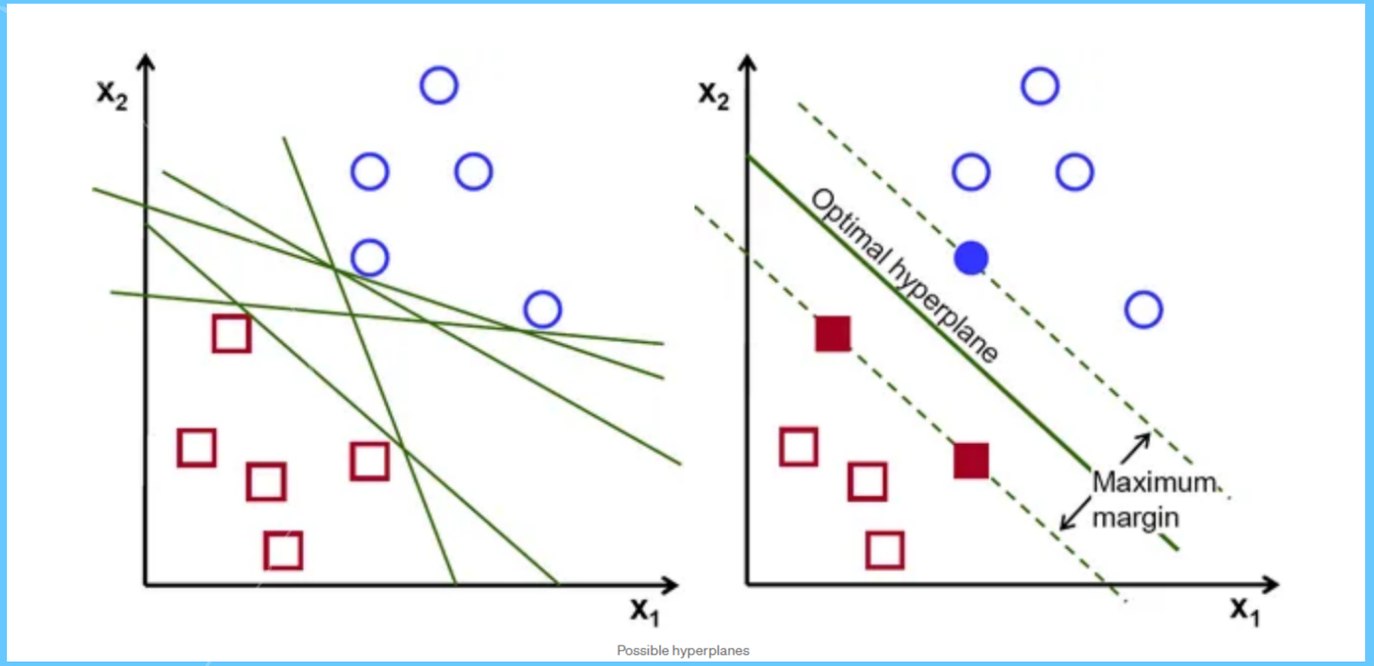
<-1 Negative, 0 Neutral, >1 Positive


DSW
DATA SCIENCE WIZARDS
classifier
P(doc | cat) P(cat)
P(doc)
End-to-End Naive Bayes Modelling

Research and Analysis

Support Vector Machine

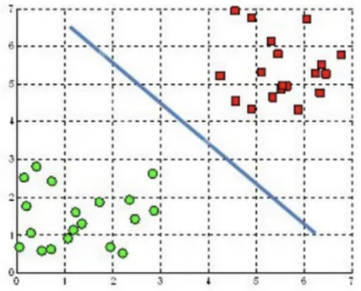
- **Principles: SVM constructs a hyperplane or set of hyperplanes in a high-dimensional space, which can be used for classification.**
- **Functionality: Best suited for binary classification problems, SVM finds the optimal separating hyperplane that maximizes the margin between two classes. It uses kernel tricks to handle linearly inseparable data.**
- **Accuracy: Highly accurate in high-dimensional spaces, even with a limited amount of data.**
- **Complexity: Requires careful tuning of parameters such as the kernel type, regularization, and Degree.**



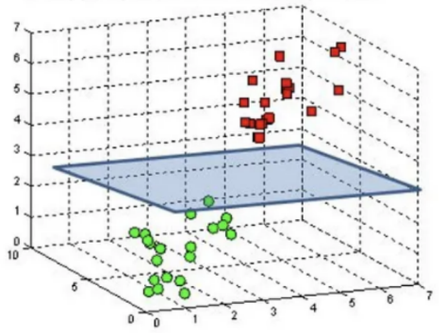
Possible hyperplanes

Hyperplanes and Support Vectors

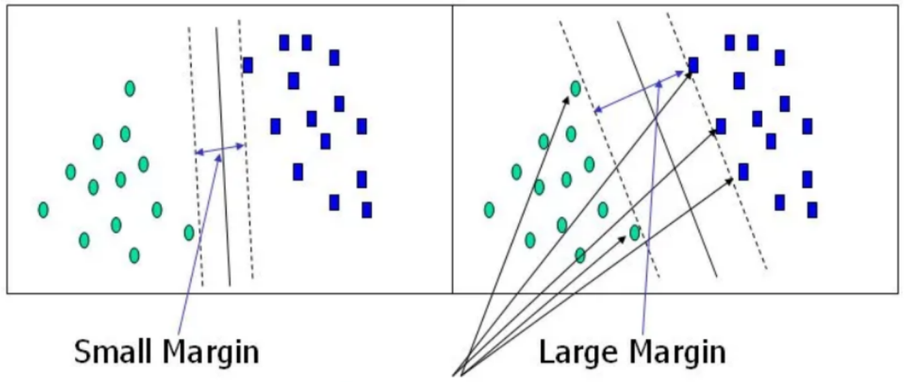
A hyperplane in $\mathbb{R}^2$ is a line



A hyperplane in $\mathbb{R}^3$ is a plane



Hyperplanes in 2D and 3D feature space

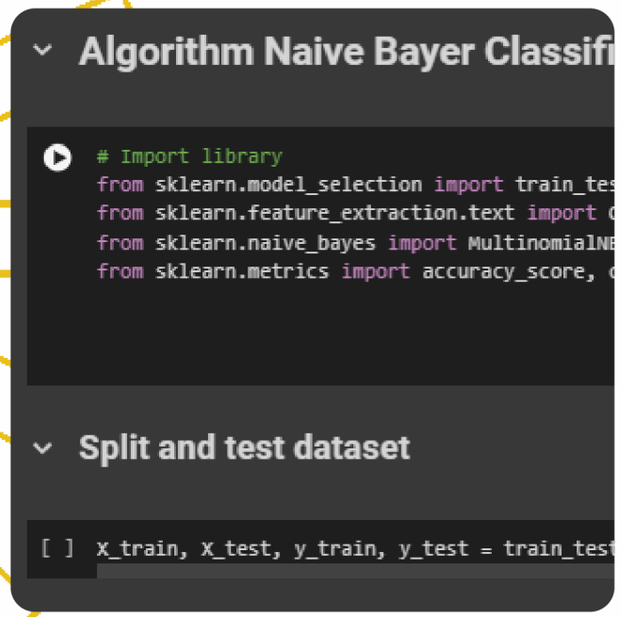


Support Vectors

Python Step 13-15

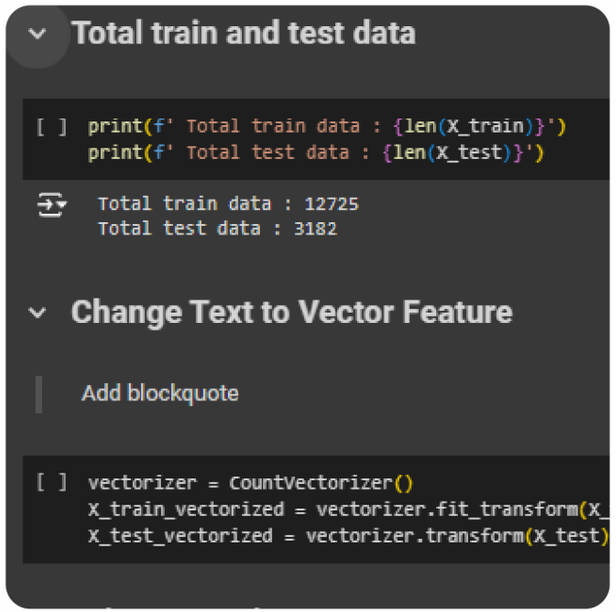
Train-Test Split for Naive Bayes Classifier,Text Vectorization and Naive Bayes Model Training and Evaluation

X_TRAIN, X_TEST, Y_TRAIN, Y_TEST = TRAIN_TEST_SPLIT(DF_CLEANED[]



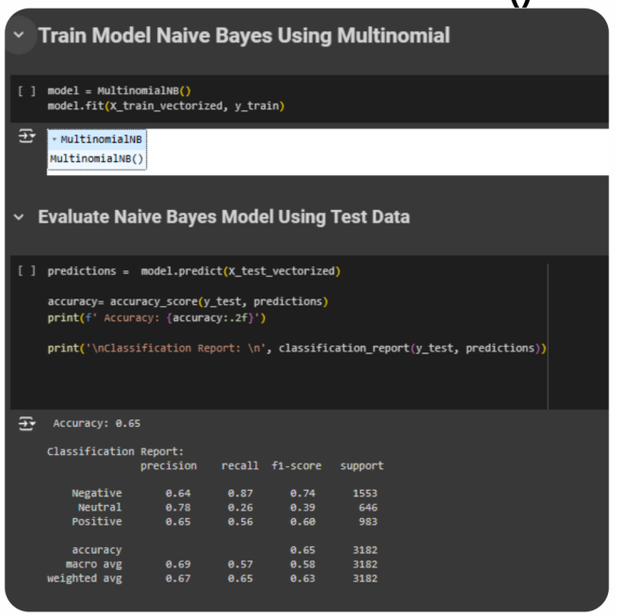
SPLIT THE DATASET

VECTORIZER = COUNTVECTORIZER()



CONVERT TEXT DATA

MODEL = MULTINOMIALNB()

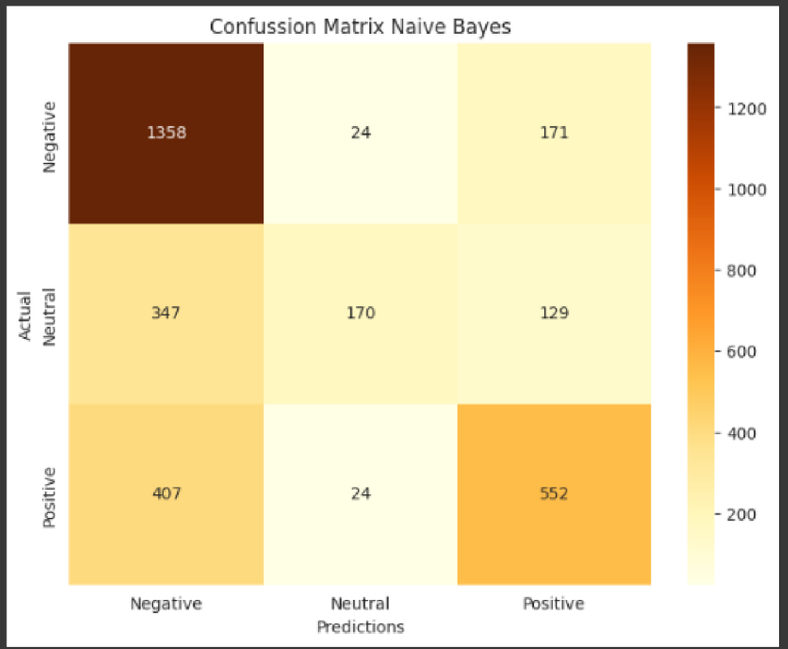


TRAIN A NAIVE BAYES CLASSIFIER

Python Step 16

Create Plot Confussion Matrix Naive Bayes Test Model

```
[ ]  conf_matrix = confusion_matrix(y_test, predictions)

     # plot confussion matrix
     plt.figure(figsize=(8,6))
     sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='YlOrBr', xticklabels=['Negative', 'Neutral',
     plt.title('Confussion Matrix Naive Bayes')
     plt.xlabel('Predictions')
     plt.ylabel('Actual')
     plt.show()
```



Confusion Matrix for Naive Bayes Model

Visualize the confusion matrix to understand the model's performance in classifying each sentiment category.

Python Step 17-19

Training Data Evaluation, Confusion Matrix for Training Data and Support Vector Machine (SVM) Model

PREDICTIONS_TRAIN = MODEL.PREDICT()

Evaluate Naive Bayes Model Using Train D

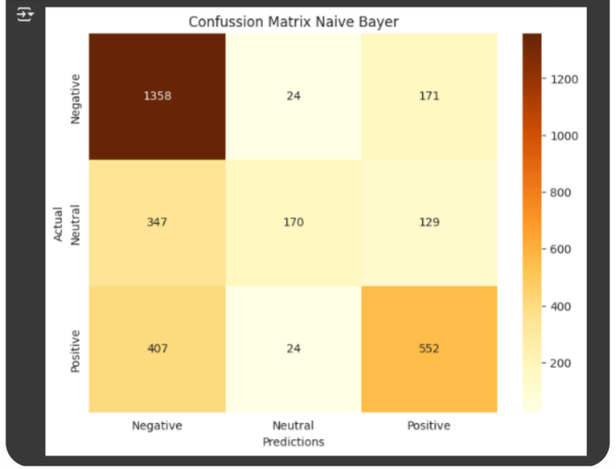
```
predictions_train =  model.predict(X_train_vectorized)

accuracy_train= accuracy_score(y_train, predictions_tr
print(f' Accuracy on Training Data: {accuracy_train:.2

print('\nClassification Report on Training data: \n',
```

```
 Accuracy on Training Data: 0.80

Classification Report on Training data:
              precision    recall  f1-score   support

    Negative       0.75      0.94      0.83      6041
     Neutral       0.93      0.59      0.72      2755
    Positive       0.83      0.73      0.78      3929

    accuracy                           0.80     12725
   macro avg       0.84      0.75      0.78     12725
weighted avg       0.81      0.80      0.79     12725
```

EVALUATE THE MODEL'S PERFORMANCE

CONF_MATRIX_TRAIN = CONFUSION_MATRIX()

```
# plot confussion matrix
plt.figure(figsize=(8,6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='YlOrBr', xticklabels=['Negative', 'Ne
plt.title('Confussion Matrix Naive Bayer')
plt.xlabel('Predictions')
plt.ylabel('Actual')
plt.show()
```



VISUALIZE THE CONFUSION MATRIX

FROM SKLEARN.SVM IMPORT SVC

Algorithm Support Vector Machine (SVM)

```
# import Library
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
X_train, X_test, y_train, y_test = train_test_split(df_cleaned['steaming data'], df_cle
```

```
print(f' Total train data; {len(X_train)}')
print(f' Total test data; {len(X_test)}')
```

```
 Total train data; 11134
 Total test data; 4773
```

Transform text data into vector TF-IDF

```
tfidf_Vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_Vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_Vectorizer.transform(X_test)
```

```
features_names = tfidf_Vectorizer.get_feature_names_out()

print("Transformation from train data result:")
print(X_train_tfidf)

print("\n feature from vector TF-IDF:")
print(features_names)
```

```
Transformation from train data result:
```

TRAIN AN SVM MODEL

SVM Model Evaluation

Evaluate the SVM model's performance on the test data and generate a classification report.

Model prediction on test data after vectoris

```
[ ] y_pred= svm_model.predict(X_test_tfidf)

    print("Example Prediction on test data:")
    print(y_pred[:10])
```

```
Example Prediction on test data:
['Negative' 'Neutral' 'Negative' 'Positive' 'Positive'
 'Positive' 'Neutral' 'Positive' 'Neutral']
```

Evaluate SVM Model Performance

```
[ ] accuracy= accuracy_score(y_test, y_pred)
    print(f'Accuracy; {accuracy:.2f}')
```

```
Accuracy; 0.82
```

```
] accuracy= accuracy_score(y_test, y_pred)
  print(f'Accuracy; {accuracy:.2f}')
```

```
Accuracy; 0.82
```

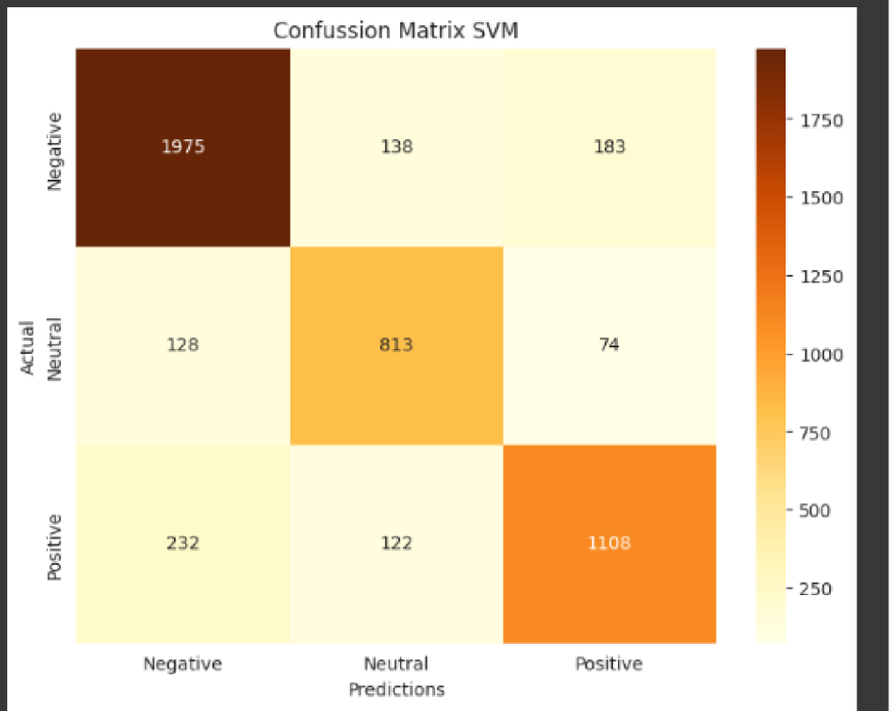
Classification Report On SVM Model

```
] print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
Negative	0.85	0.86	0.85	2296
Neutral	0.76	0.80	0.78	1015
Positive	0.81	0.76	0.78	1462
accuracy			0.82	4773
macro avg	0.81	0.81	0.81	4773
weighted avg	0.82	0.82	0.82	4773

Python Step 21

```
[ ]  # plot confussion matrix SVM
     plt.figure(figsize=(8,6))
     sns.heatmap(conf_matrix_svm, annot=True, fmt='d', cmap='YlOrBr', xticklabels=['Negative',
     plt.title('Confussion Matrix SVM')
     plt.xlabel('Predictions')
     plt.ylabel('Actual')
     plt.show()
```



Confusion Matrix for SVM

Visualize the confusion matrix for the SVM model to assess its classification performance.

Recommendation

For the Twitter Sentiment Analysis dataset, Naive Bayes and SVM are particularly suitable due to their **effectiveness in text classification tasks.**

Naive Bayes is simple and efficient for this type of data, while **SVM provides high accuracy** and handles high-dimensional data well.

The choice between these algorithms depends on the specific requirements of the analysis, such as computational resources and the need for interpretability.

Naive Bayes (Multinomial)

Justification	Challenges and Limitations
• Simplicity and Efficiency: Naive Bayes is straightforward to implement and computationally efficient, making it ideal for large datasets.	• Independence Assumption: Naive Bayes assumes that the features are conditionally independent given the class, which is often not true in practice. This assumption can sometimes lead to suboptimal performance.
• Performance on Text Data: Despite its simplicity, Naive Bayes performs well on text classification tasks, as it directly models the probability distributions of the features.	• Sensitivity to Imbalanced Data: Naive Bayes can be sensitive to class imbalances. If the dataset has significantly more instances of one class, the model may be biased towards that class.
• Fast Training and Prediction: The training and prediction times are significantly lower compared to more complex models, making it suitable for real-time applications.	• Less Accurate than Complex Models: While it performs well on many tasks, it may be outperformed by more sophisticated models like SVM or ensemble methods on complex datasets.

Support Vector Machines (SVM)

Justification	Challenges and Limitations
• Effectiveness in High-dimensional Spaces: SVM is particularly effective in handling high-dimensional text data, making it suitable for the large feature set generated by the TF-IDF vectorizer.	• Computational Intensity: Training an SVM model, especially with a large dataset, can be computationally expensive and time-consuming.
• Performance: SVM has shown to provide high accuracy in text classification tasks, as demonstrated by the accuracy of 82% on the test data.	• Parameter Tuning: SVM requires careful tuning of hyperparameters (such as the regularization parameter and the kernel type) to achieve optimal performance.
• Kernel Trick: The use of different kernels allows SVM to model non-linear relationships, making it versatile for various types of data distributions.	• Interpretability: SVM models are less interpretable compared to simpler models like Logistic Regression or Naive Bayes.

Discussion and Reflection

Challenges and Considerations in Selecting Algorithms for Big Data Analysis

1

CHALLENGES

- Challenge: The initial dataset often contains missing values, inconsistencies, and irrelevant information that must be addressed before analysis. This requires a significant amount of time and effort.
- Consideration: Effective data preprocessing, including handling missing values, standardizing sentiment labels, and filtering relevant data, is critical to ensure the accuracy and reliability of the analysis.

2

FUTURE DIRECTIONS

- Challenge: Text data needs to be transformed into a numerical format that algorithms can process. The choice between methods like Count Vectorizer and TF-IDF Vectorizer can significantly impact the performance of the models.
- Consideration: It's important to experiment with different feature extraction methods and select the one that best captures the underlying patterns in the data.

3

ALGORITHM SELECTION

- Challenge: Different algorithms have different strengths, weaknesses, and computational requirements. Selecting the appropriate algorithm involves balancing these factors to meet the specific needs of the dataset and analysis goals.
- Consideration: Factors such as accuracy, interpretability, computational efficiency, and scalability should be considered when selecting algorithms. Additionally, the nature of the data (e.g., high-dimensional text data) plays a crucial role in this decision.

Challenges and Considerations in Selecting Algorithms for Big Data Analysis

4

MODEL EVALUATION AND TUNING

- Challenge: Evaluating model performance and tuning hyperparameters to optimize results can be complex and time-consuming. Overfitting and underfitting are common issues that need to be addressed.
- Consideration: Cross-validation and careful tuning of hyperparameters are essential to develop robust models. It’s important to use appropriate metrics and visualization tools (like confusion matrices) to evaluate and improve model performance.

5

COMPUTATIONAL RESOURCES

- Challenge: Some algorithms, such as SVM, can be computationally intensive, especially with large datasets. This can lead to longer training times and require significant computational resources.
- Consideration: Balancing the trade-off between computational efficiency and model performance is crucial. Choosing efficient algorithms or optimizing existing ones can help manage computational constraints.

Discussion and Reflection

Future Directions for Research or Analysis

1

ADVANCED TEXT PROCESSING TECHNIQUES

- Future Research: Investigating advanced text processing techniques such as word embeddings (e.g., Word2Vec, GloVe) and transformers (e.g., BERT) could enhance feature extraction and improve model performance.
- Application: These techniques can capture semantic meanings and contextual information, leading to more accurate sentiment analysis.

2

ENSEMBLE METHODS

- Future Research: Exploring ensemble methods that combine the strengths of multiple algorithms could lead to better predictive performance.
- Application: Techniques like Random Forest, Gradient Boosting, or voting classifiers can be employed to enhance robustness and accuracy.

3

HANDLING IMBALANCED DATA

- Future Research: Developing methods to handle class imbalances more effectively, such as resampling techniques, synthetic data generation, or cost-sensitive learning.
- Application: Addressing class imbalances can lead to more accurate and fair models, particularly in sentiment analysis where negative sentiments may be more prevalent.

Future Directions for Research or Analysis

4

REAL-TIME SENTIMENT ANALYSIS

- Future Research: Implementing models that can perform real-time sentiment analysis to provide instant insights into public opinion.
- Application: Real-time analysis can be valuable for monitoring ongoing events, such as elections or marketing campaigns, allowing for timely responses and adjustments.

5

DEEP LEARNING MODELS

- Future Research: Investigating deep learning models like Convolutional Neural Networks (CNNs) for text and Recurrent Neural Networks (RNNs) for sequence modeling.
- Application: These models can capture complex patterns and dependencies in the data, potentially improving the accuracy of sentiment classification.

6

EXPLAINABLE AI

- Future Research: Focusing on explainable AI to improve the interpretability of complex models.
- Application: Developing techniques that make models like SVM and deep learning more interpretable can help stakeholders understand and trust the model predictions.

Summary

Selecting the right algorithms for big data analysis involves addressing multiple challenges related to data preprocessing, feature extraction, algorithm selection, model evaluation, and computational resources.

The recommendations provided in this analysis, particularly **the use of SVM and Naive Bayes for the Twitter Sentiment Analysis dataset,** are based on careful consideration of these factors.

Future research directions, including advanced text processing techniques, ensemble methods, and deep learning models, hold promise for further enhancing the accuracy and applicability of sentiment analysis.

Addressing these challenges and exploring these directions can lead to more robust, efficient, and interpretable models, ultimately providing deeper insights and more actionable intelligence from big data.

References 01

1. Seman, N., & Razmi, N. A. (2020). Machine learning-based technique for big data sentiments extraction. Retrieved from https://ijai.iaescore.com/index.php/IJAI/article/download/20469/pdf
2. Mohamad, A. T., & Sia Abdullah, N. A. (2021). A Case Study on Social Media Analytics for Malaysia Budget. Retrieved from chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://thesai.org/Downloads/Volume12No10/Paper 64-A Case Study on Social Media Analytics.pdf
3. Nugraha Putra, F. D., Pranowo, & Setyohadi, B. (2020). Sentiment analysis of Indonesian presidential election 2019 on the twitter with lexicon-based and support vector machine (SVM). Retrieved from https://pubs.aip.org/aip/acp/article-abstract/2217/1/030136/1025097/Sentiment-analysis-of-Indonesian-presisential?redirectedFrom=fulltext
4. Atmajaya, D., Febrianti, A., & Darwis, H. (2023). Metode SVM dan Naive Bayes untuk Analisis Sentimen ChatGPT di Twitter. Retrieved from http://ijcs.stmikindonesia.ac.id/ijcs/index.php/ijcs/article/download/3341/231
5. Ismail, N. F., Sia Abdullah, N. A., & Idrus, Z. (2021). Multistage Sentiment Classification Model using Malaysia Political Ontology. Retrieved from chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://thesai.org/Downloads/Volume12No10/Paper_48-Multistage_Sentiment_Classification_Model.pdf
6. Buntoro, G. A., Arifin, R., Syaifuddiin, G. N., Selamat, A., Krejcar, O., & Fujita, H. (2021). IMPLEMENTATION OF A MACHINE LEARNING ALGORITHM FOR SENTIMENT ANALYSIS OF INDONESIA'S 2019 PRESIDENTIAL ELECTION. Retrieved from https://journals.iium.edu.my/ejournal/index.php/iiumej/article/download/1532/790

References 02

- 7. Fitriana, D. N., & Sibaroni, Y. (2020). Sentiment Analysis on KAI Twitter Post Using Multiclass Support Vector Machine (SVM). Retrieved from http://jurnal.iaii.or.id/index.php/RESTI/article/download/2231/314
- 8. Muzaki, A., & Witanti, A. (2021). SENTIMENT ANALYSIS OF THE COMMUNITY IN THE TWITTER TO THE 2020 ELECTION IN PANDEMIC COVID-19 BY METHOD NAIVE BAYES CLASSIFIER. Retrieved from http://jutif.if.unsoed.ac.id/index.php/jurnal/article/download/51/32
- 9. Silitonga, W. H., & Sihotang, J. I. (2019). Analisis Sentimen Pemilihan Presiden Indonesia Tahun 2019 Di Twitter Berdasarkan Geolocation Menggunakan Metode Naïve Bayesian Classification. Retrieved from https://jurnal.unai.edu/index.php/teika/article/download/2199/1425/
- 10. Tineges, R., Triayudi, A., & Sholihati, I. D. (2020). Analisis Sentimen Terhadap Layanan Indihome Berdasarkan Twitter Dengan Metode Klasifikasi Support Vector Machine (SVM). Retrieved from https://ejurnal.stmik-budidarma.ac.id/index.php/mib/article/download/2181/1650
- 11. Holbrook, A., Nishimura, A., Ji, X., & Suchard, M. A. (2022). Computational Statistics and Data Science in the Twenty-First Century. Retrieved from https://arxiv.org/pdf/2204.05530v1.pdf
- 12. Shanmuk, P., Ravikiran, P., Tarun, P., Preethi, K., & Divya, C. (2021). Sentiment Analysis of Twitter Data. Retrieved from https://www.aclweb.org/anthology/W11-0705.pdf