

# Twitter Sentiment Analysis

NLP Model that rates  
sentiment in tweet  
depending on content

# Contents

01

Overview

02

Data Understanding

03

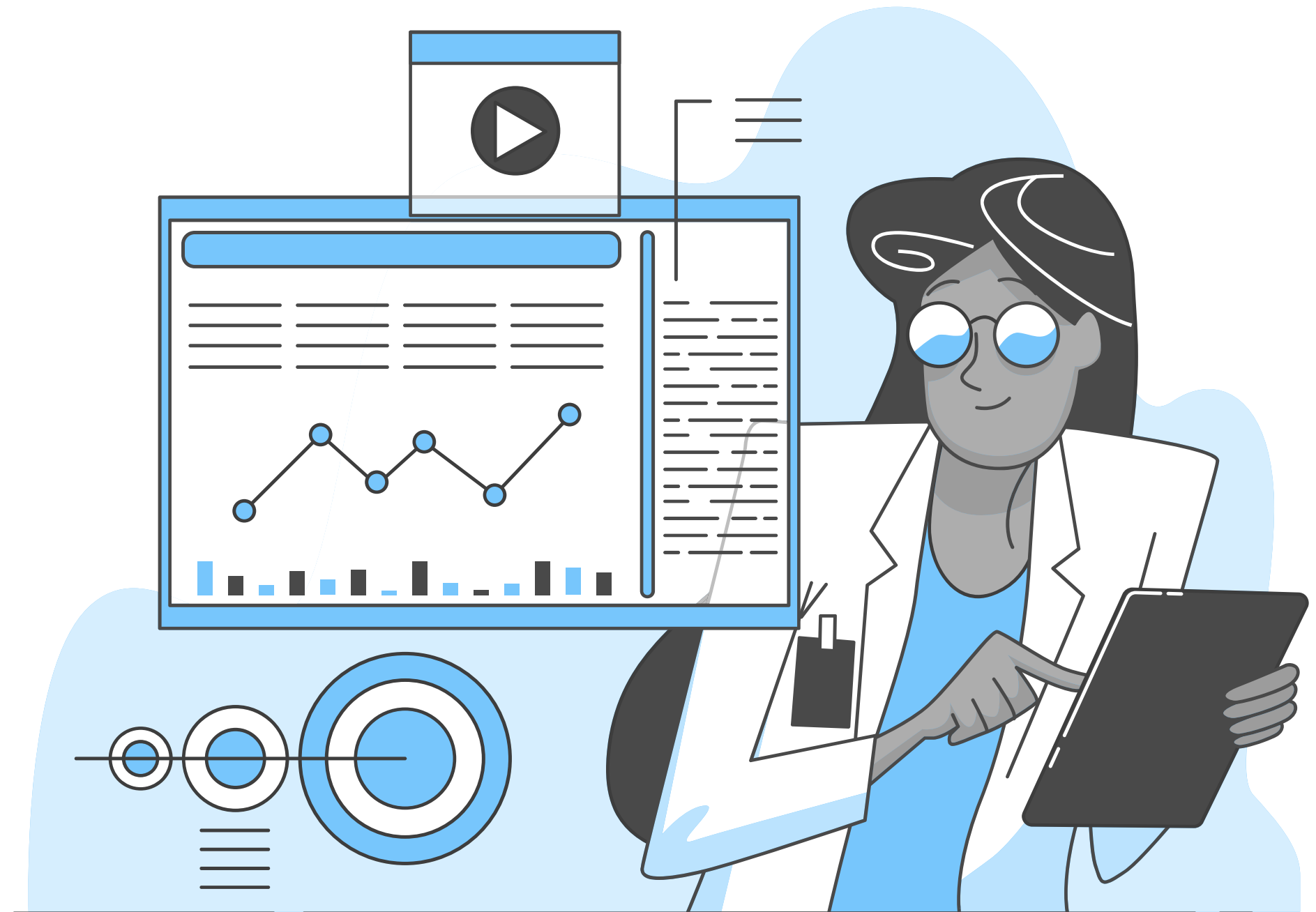
Modelling

04

Evaluation

05

Conclusion &  
Recommendation

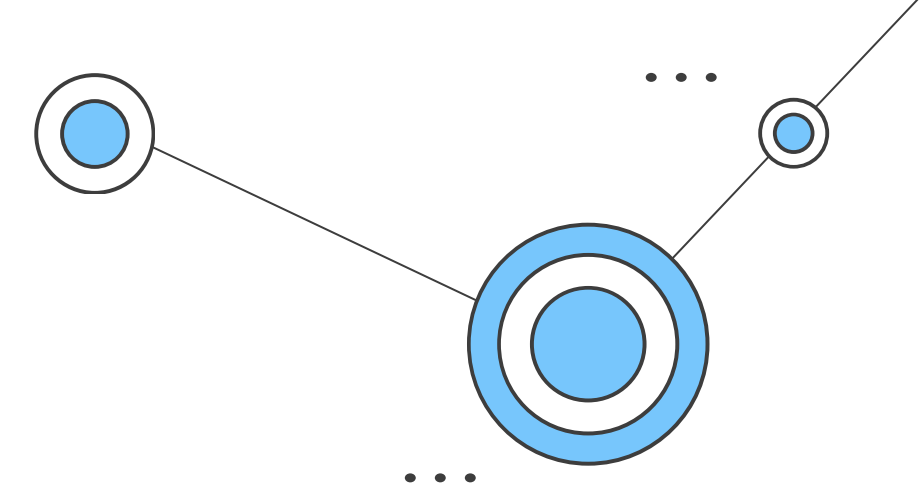


# Overview



- ❑ This project analyzes tweeter sentiments with the aim of building a classification model that rates text data into categories i.e Negative, Positive and Neutral
- ❑ Sentiment analysis is essential to comprehending public opinion in today's digital environment. Businesses employ this technique to extract valuable amounts of textual data including news articles, social media comments and consumer reviews.
- ❑ Support vector machines(SVM) and Naïve bayes are among NLP algorithms that will be used

# Data Understanding



The Twitter dataset contains multiple columns, including date information, but our analysis focuses solely on the text data and the sentiment column.

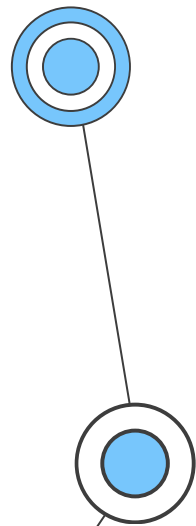
To prepare the data for machine learning, we:

- ❑ Removed duplicates to eliminate redundant information.
- ❑ Converted sentiment labels to integers for compatibility with ML models.
- ❑ Preprocessed text by:

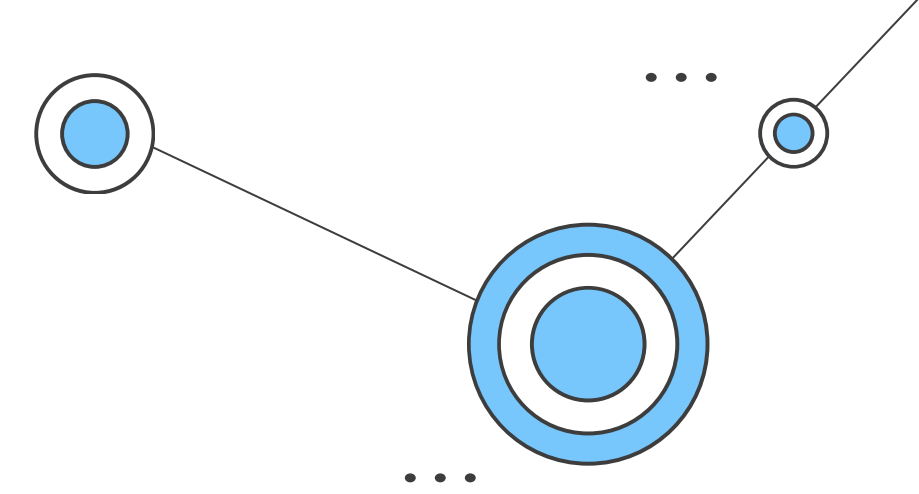
Removing stopwords and punctuation

Applying lemmatization to normalize words

We developed and evaluated multiple models to determine the best-performing approach for sentiment classification.



# Modelling



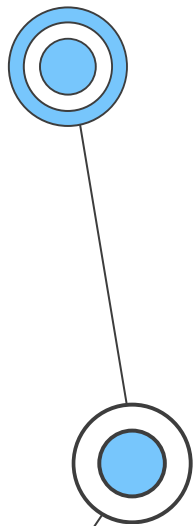
We trained five classification algorithms on the dataset, including Support Vector Machines (SVM), Naïve Bayes, and XGBoost, among others.

The dataset was split into training and test sets, and we utilized pipelines to streamline preprocessing, which included:

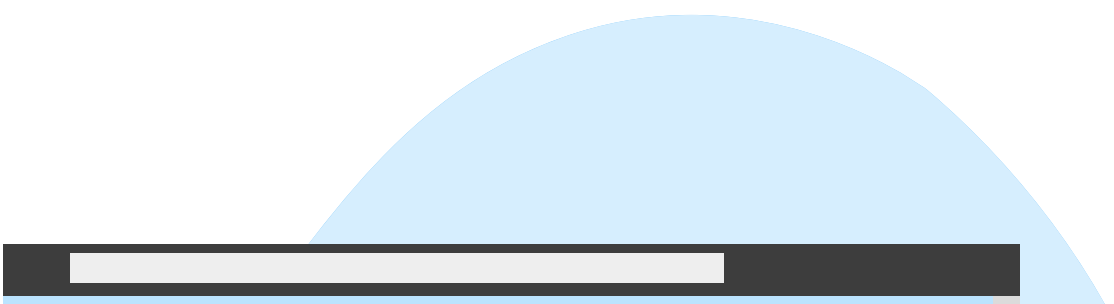
- ❑ Converting text data into numerical representations
- ❑ Using SMOTE to address class imbalance
- ❑ Training models on the processed data

A custom process was implemented to train models, make predictions, and compute evaluation metrics for all classifiers.

Finally, the best-performing model was fine-tuned and used for final predictions.



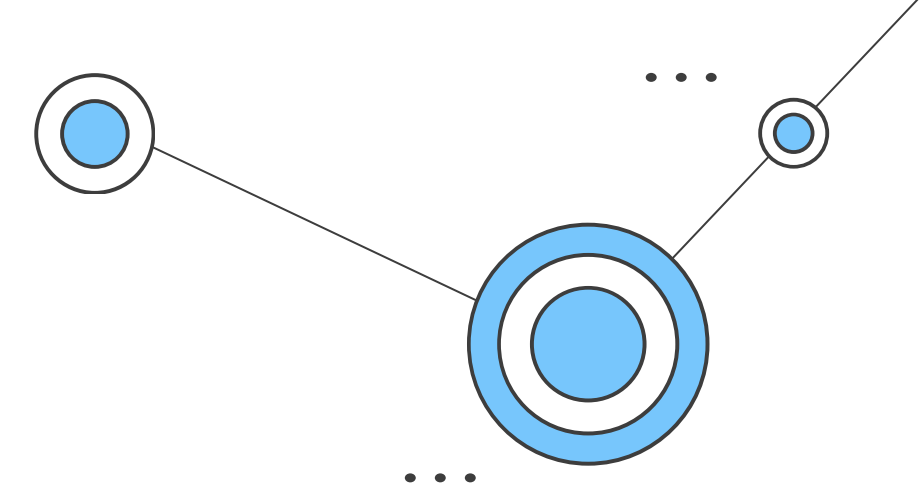
# Model Evaluation



|                     | Training Accuracy | Test Accuracy | Test precision | Test recall | Test f1_score | Average Train AUC | Average Test AUC |
|---------------------|-------------------|---------------|----------------|-------------|---------------|-------------------|------------------|
| Logistic Regression | 0.963622          | 0.707741      | 0.703716       | 0.707741    | 0.705475      | 0.993268          | 0.826946         |
| Random Forest       | 0.990510          | 0.704581      | 0.719630       | 0.704581    | 0.681571      | 0.999546          | 0.809192         |
| Naïve Bayes         | 0.941874          | 0.682464      | 0.713306       | 0.682464    | 0.691371      | 0.988934          | 0.837347         |
| Decision Tree       | 0.990510          | 0.647709      | 0.639782       | 0.647709    | 0.639702      | 0.999876          | 0.668403         |
| SVM                 | 0.980625          | 0.706161      | 0.693318       | 0.706161    | 0.685062      | 0.995330          | 0.821985         |
| XGBoost             | 0.891261          | 0.691943      | 0.702068       | 0.691943    | 0.672825      | 0.973329          | 0.792919         |

- ❑ From the models tested we see varying degrees of overfitting and generalization ability across different models. Decision trees and random Forest performance might fail in real word due to overfitting
- ❑ Higher AUC is seen in the models indicating models performs well in separating the classes
- ❑ XGBoost has the best generalization as it has the smallest gap between the train and test accuracy
- ❑ on simple models logistic regression performs well with a good balance between precision and recall
- ❑ On tuning XGBoost very slight increases as compared to tuned Logistic regression model. We will use Logistic regression to make our predictions

# Conclusion and Recommendation

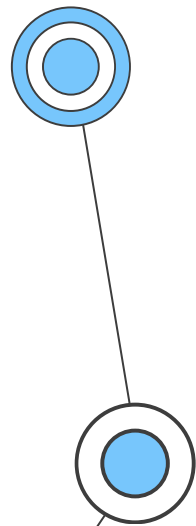


## Conclusion

- Logistic Regression outperformed XGBoost, achieving better accuracy and F1 score, making it the preferred model for sentiment classification.
- Hyperparameter tuning for XGBoost led to only marginal improvements (accuracy & F1 score increased by  $\leq 1\%$ , AUC by 0.0004).
- Tuning helped reduce overfitting slightly, meaning XGBoost generalized better than before, but still did not surpass Logistic Regression.

## Recommendations

- Use Logistic Regression for final predictions since it performs better than XGBoost.
- Consider feature engineering (e.g., word embeddings like Word2Vec or BERT) to improve model performance further.
- Explore deep learning models (e.g., LSTMs or Transformers) if higher accuracy is required.
- Continue tuning XGBoost or test alternative ensemble methods if needed for comparison.







...

# Thanks

**Email:** [gracemwendemicheni@gmail.com](mailto:gracemwendemicheni@gmail.com)

**Github:** <https://github.com/GraceMwende>

**Linkedin:** <https://www.linkedin.com/in/gracemwende/>

**medium:** <https://medium.com/@gracemwendemicheni>

