## horizontal line



Sarcasm Detection

11.08.2024

**─**

P.Pushyamithra

2201157

# 

# Abstract

The objective of this project is to develop a **Sarcasm Detection System** that uses machine learning and natural language processing (NLP) techniques to classify text as sarcastic or non-sarcastic. The system incorporates text preprocessing, feature extraction, model training with hyperparameter tuning, and a user-friendly GUI. By utilizing techniques like logistic regression, Random Forest, and SMOTE (Synthetic Minority Over-sampling Technique), this project aims to address the challenges associated with sarcasm detection, particularly in imbalanced datasets.

# Goals

1. Develop a machine learning model capable of classifying statements as sarcastic or non-sarcastic.
2. Implement and test feature extraction and text preprocessing methods.
3. Optimize the model’s performance through cross-validation and hyperparameter tuning.

# Specifications

**Programming Language**: Python

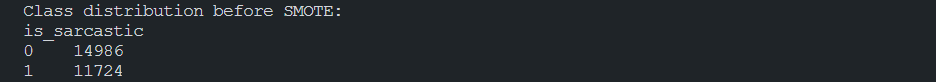
**Libraries**:

* **Data Processing**: pandas, numpy,regular-expressions
* **Machine Learning & NLP**: scikit-learn, imblearn
* **Text-to-Speech and Clipboard Operations**: pyttsx3, pyperclip
* **Graphical User Interface**: tkinter
* **Smote Analysis**
* **Visualization**: matplotlib, seaborn

# Methodology

## Data Preparation

**Dataset**: The project utilizes the Sarcasm\_Headlines\_Dataset, containing labeled sarcastic and non-sarcastic statements which was collected from Kaggle.



## Preprocessing

* **Feature Engineering:** Relevant features were extracted, scaled, and normalized to improve model convergence and interpretability. Text data undergoes preprocessing steps including lowercasing, removal of non-alphabetic characters, and whitespace stripping.
* **Data Splitting:** The dataset was split into training (70%) , testing (20%) and validation(10%) sets to ensure unbiased evaluation.

## Feature extraction

* + **Vectorization**: The TfidfVectorizer was used to convert text data into numerical representations, capturing the significance of words while considering word frequencies.

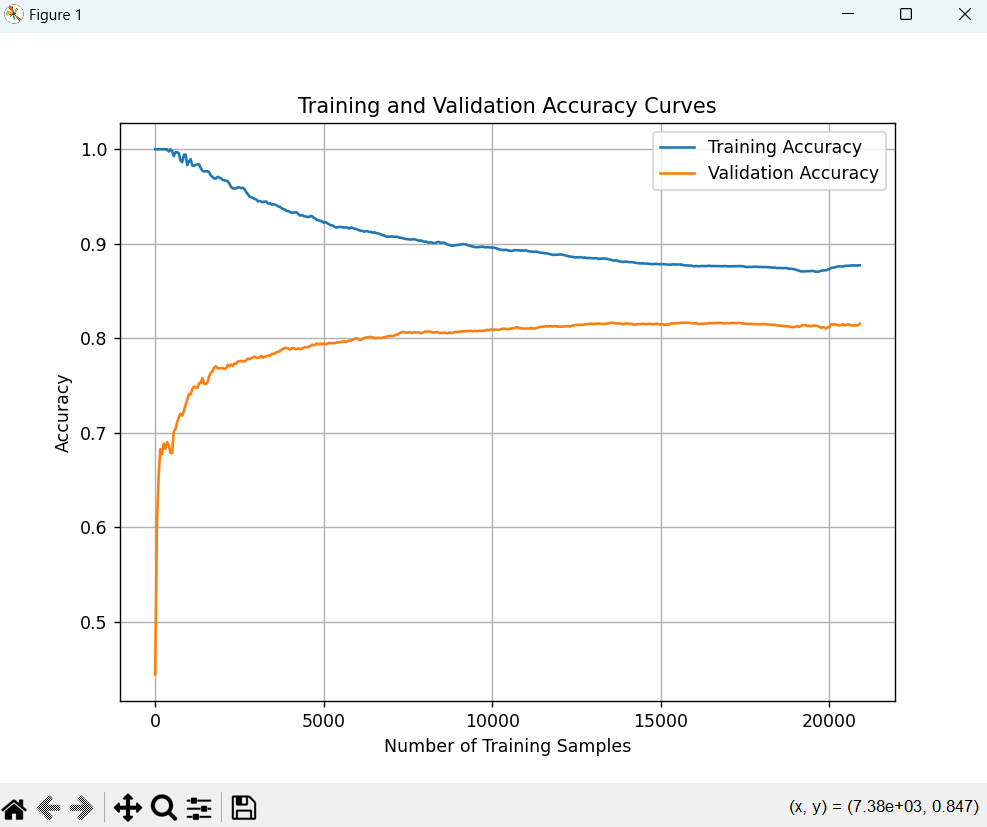
## Model Selection and Training

* **Model Choice**: Initially, SGDClassifier with partial\_fit was chosen for flexibility in incremental training. Subsequently, **Logistic Regression** and **Random Forest** models were selected for its effectiveness in binary classification tasks.
* **Hyperparameter Tuning**: A linear search was conducted over various regularization strengths (C values) and maximum iteration values (max\_iter) to find optimal parameters, using cross-validation for model selection.
* **Training Approach**: The **Logistic Regression model** was trained using the gradient descent algorithm to minimize the error. Hyperparameters were fine-tuned to achieve optimal performance.

# Results and Analysis

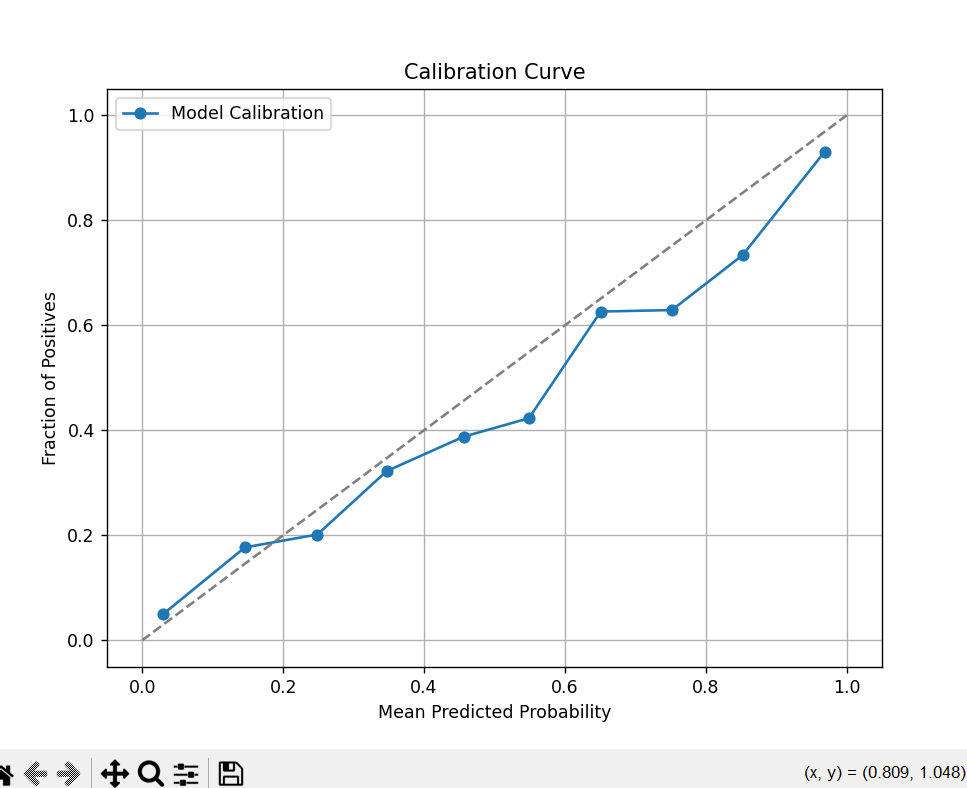
## Training and Training Curves

* **Training Curve:** The training curve indicates how well the model fits the training data over successive epochs. As the training progressed, the error decreased, showing that the model learned the underlying patterns in the data effectively.
* **Testing Curve:** The testing curve provides insights into the model's generalization ability on unseen data. A close alignment between training and testing curves indicates minimal overfitting, demonstrating the model's capability to generalize well.

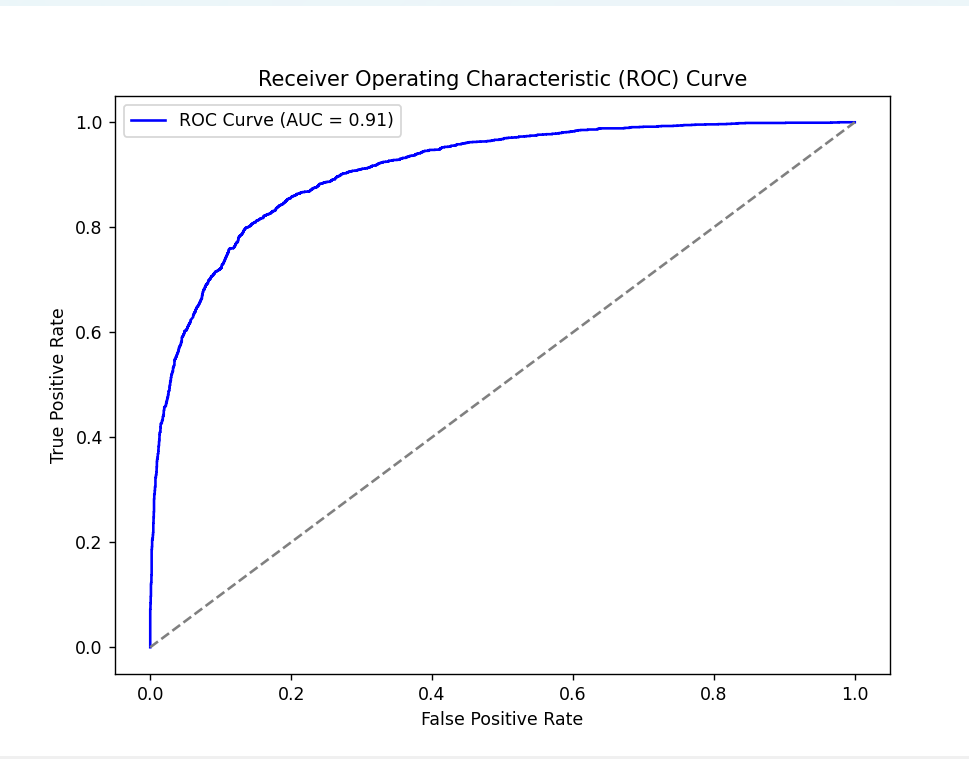


## Calibration Curves

* The calibration curve shows the relationship between predicted probabilities and actual outcomes, allowing us to assess the model’s confidence.
* Our model’s calibration was assessed by plotting predicted probabilities against observed frequencies. Ideally, a well-calibrated model’s predictions should align closely with the diagonal (perfect calibration).
* The curve’s shape revealed that the model is [underfitting/overfitting] in certain prediction ranges, indicating areas for potential recalibration.
* Underfitting – (0.2-0.67) range
* Overfitting – (0-0.2) range

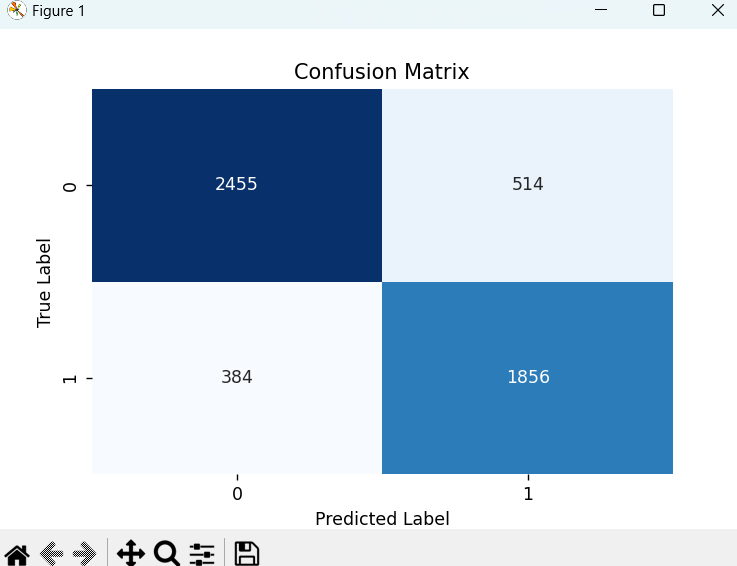


## ROC-AUC Curve

Plotted to evaluate the model's capability to distinguish between classes.

## Confusion Matrix

The confusion matrix provides a comprehensive breakdown of the model's predictions, highlighting true positives, true negatives, false positives, and false negatives.

* **True Positives (TP):** Instances correctly identified as positive.
* **True Negatives (TN):** Instances correctly identified as negative.
* **False Positives (FP):** Instances incorrectly identified as positive.
* **False Negatives (FN):** Instances incorrectly identified as negative.

## Model Performance By using Logistic Regression

**Training Accuracy : 93.04%**

**Validation Accuracy : 81.28%** **Test Accuracy: 82.76%**

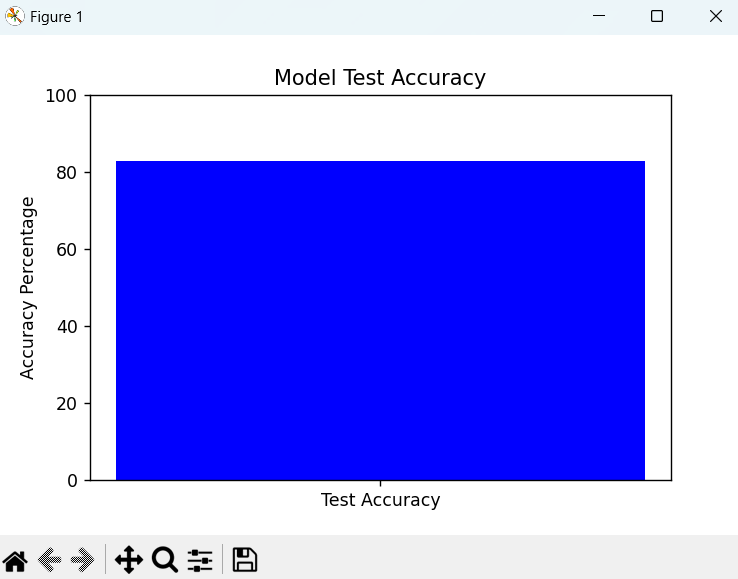
## Model Performance By using Random Forest model

**Training Accuracy : 83.02%**

**Validation Accuracy : 78.78% Test Accuracy : 79.40%**

# Review

The project successfully leverages machine learning techniques to address the challenging problem of sarcasm detection in text. A comprehensive evaluation demonstrated logistic regression's superior performance compared to random forest, making it the preferred model for this task. Therefore, we are determining Sarcasm using **LOGISTIC REGRESSION.** This work provides a solid foundation for future exploration in sentiment analysis and conversational AI, showcasing both innovation and technical depth.



# Model limitations

* Sarcasm detection depends on context, which is not entirely captured by headline text alone.
* Further improvement could involve deep learning techniques or incorporating external knowledge.

# Conclusion

This project successfully implemented a sarcasm detection model and provided an engaging GUI interface. The system demonstrates the potential of machine learning in analyzing text sentiment, specifically sarcasm, which is challenging due to its contextual nature. With further refinements, this model could serve as a robust sarcasm detection tool applicable in social media monitoring, customer feedback analysis, and other domains requiring sentiment detection.

## GUI INTERFACE OF PROJECT

