# NATURAL LANGUAGE PROCESSING MODEL FOR TWITTER SENTIMENT ANALYSIS

# CRISP DM DOCUMENTATION

**1.BUSINESS UNDERSTANDING**

As a business grows, it launches new products and features. In addition to existing products and features, they need to evaluate the performance of new products in the existing markets. It is necessary to consider customer feedback in order to improve the quality of the product and have an increased advantage on competitors in the market.  
A useful technique for analyzing the product is via social media reviews. Businesses can evaluate how customers feel regarding the product through analysis of posts, tweets and threads about the product.

**Stakeholder**

Apple is a global brand that receives millions of customer reviews on X/Twitter daily. These reviews are in the form of tweets and include complaints, product feedback, praise, feature requests and general discussions.

**Business Problem**

We are tasked with building a **Natural Language Processing Model (NLP model**) to monitor brand sentiments about Apple products and respond to customer feedback effectively

##### **Objective**

To build a model that classifies tweets as positive, negative or neutral to help automate sentiment analysis

**Success Metrics**

We aim for an Average AUC score of more than 70% which indicates that our model is doing well in classifying the sentiments

**Situation Assessment**

Apple, being a global brand, receives millions of customer mentions on Twitter daily. These tweets include product feedback, complaints, praise, feature requests, and general discussions. Manually analyzing these tweets is inefficient and time-consuming. An automated sentiment analysis system will help Apple:

* Understand public perception of its products and services.
* Detect early signs of dissatisfaction and prevent potential PR crises.
* Identify common issues with Apple devices, software updates, or services.
* Improve customer engagement by responding proactively to concerns.
* Track sentiment trends over time to measure the impact of marketing campaigns, product launches, or policy changes.

**Data Mining Goals**

To achieve the business objective, the data mining goals include:

***Data Collection***:

* Gathering Apple-related tweets using an Apple Twitter Sentiment dataset containing tweets by Twitter users.

***Data Preprocessing & Cleaning:***

* The dataset has no missing values, so no imputation is required.
* Removing duplicate tweets for instance, one tweet appearing 304 times.
* Drop irrelevant sentiment labels, such as ‘not\_relevant’, to retain only the core sentiments (negative, positive, and neutral).
* Converting sentiments to integer values to ensure compatibility with machine learning models.
* Removing hyperlinks, usernames, single-character words, and hashtags (including their values) using regular expressions, as they do not meaningfully contribute to sentiment analysis.
* Eliminating stopwords and punctuation using the NLTK corpus library for stopwords and the string library for punctuation, as these do not add significant meaning to sentences.
* Applying lemmatization using the WordNet Lemmatizer, converting words to their root

Form.

***Data Visualization:***

* Use of Seaborn’s countplot to visualize the distribution of target sentiment classes.
* Use of WordCloud to generate a visual representation of the most common words in the dataset.

***Modeling***

* Using Scikit-learn’s model\_selection library to split the dataset into training and testing sets.
* Implementing pipelines to streamline vectorization, SMOTE (Synthetic Minority Over-sampling Technique), and classification models.
* Using SMOTE to address class imbalance
* Evaluating various machine learning algorithms for classification to determine the best-performing model.

***Feature Engineering***:

* Converting text into numerical features using TF-IDF (Term Frequency-Inverse Document Frequency), text features (word count and sentiment score).

***Model Selection & Training:***

* Experiment with various NLP models, such as Logistic Regression, Naïve Bayes, Support Vector Machines, Decision Trees and Random Forest.
* Fine-tuning hyperparameters to improve performance. XGboost and Logistic Regression are used.

***Evaluation Metrics:***

* Use of accuracy, precision, recall, F1-score, and AUC-ROC to assess model performance.

***Model Evaluation & Predictions***

* The test set obtained from train\_test\_split (X\_test, y\_test) is used for model evaluation and making predictions.
* Developing a custom function that allows classification of sentiment, either positive, negative, or neutral, based on user input.

|  |
| --- |
| **PROJECT PLAN** |
| |  |  |  | | --- | --- | --- | | **Phase** | **Task** | **Tools/Technologies used** | | Business Understanding | Define objectives & goals | CRISP-DM framework | | Data Collection | Collect tweets using Apple Twitter Sentiment Dataset | Pandas. | | Data Cleaning | Remove noise, tokenize, normalize text | NLTK, regex | | Feature Engineering | Convert text to Numerical Representation | TF-IDF | | Modeling | Train ML Models | Scikit-learn, GridSearchCv, RandomSearchCv | | Evaluation | Assess model performance | Accuracy Score, Recall, Precision,F1-score | | Deployment | Deploy model for real-time analysis | Flask, FastAPI, Streamlit | |

**2. DATA UNDERSTANDING**

**2.1 Data Collection**

The dataset used in this project is the Apple-Twitter-Sentiment-DFE.csv, which contains tweets mentioning Apple. The dataset was loaded using Pandas and encoded in "latin1" to handle special characters.

Source: Twitter (An existing dataset from [Kaggle](https://www.kaggle.com/datasets/slythe/apple-twitter-sentiment-crowdflower))

Data Size: 3,886 tweets

This dataset contains features such as the text/tweet, sentiment of the text, sentiment strength, the date of publication of the text/tweet, among others.  
Our goal is to predict whether the text is negative, neutral or positive i.e. target variable is sentiment which is to be predicted based on the tweet/text.

We have 2 columns of interest:

* text – The actual tweet content
* sentiment – The sentiment label of the tweet (Positive, Negative, Neutral, Irrelevant)

**2.2 Dataset Limitations**

The dataset is relatively small and may not capture the full diversity of language for example slang and sarcasm which may lead to overfitting or biased predictions

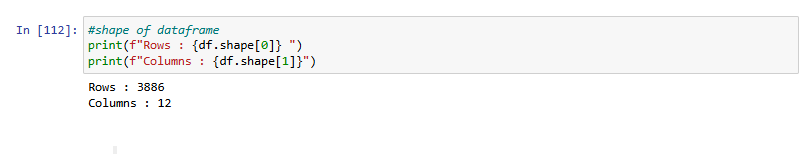
Our target class(sentiment) is imbalances as Neutral tweets dominate (53.1%), while positive tweets are rare (11.9%). This may skew the model towards the majority class.

**2.3 Data Description**

After loading the dataset, an initial examination was performed using df.info() and df.head(). The most relevant columns for this analysis are:

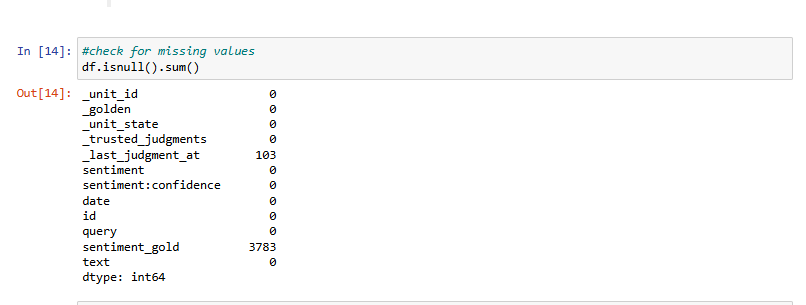
|  |  |
| --- | --- |
| **Column Name** | **Description** |
| text | The tweet text related to Apple |
| sentiment | The sentiment label (3 = Neutral, 1 = Negative, 5 = Positive, Irrelevant values removed) |

The dataset contains 3886 rows and 12 columns

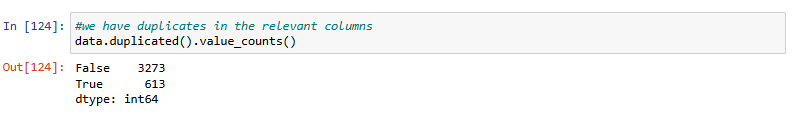


**2.4 Data Exploration**

* Checking for Missing Values, it was confirmed that no missing values exist in the relevant columns. However, 2 columns: \_last\_judgment\_at and sentiment\_gold do have missing values.



* Checking for Duplicates - Using data.duplicated(), 613 duplicate tweets were identified. The duplicates needed to be removed later on to ensure unbiased model training.



**2.4.1 Target (Sentiment) Distribution**

* To understand the class balance, the sentiment column was visualized later on in the EDA section.



***Observations***

* The majority class is Neutral (3).
* The dataset is imbalanced, which could affect model performance.
* SMOTE (Synthetic Minority Over-Sampling Technique) may be needed to balance classes.

**2.4.2Verifying Data Quality**

To assess the data’s quality, key factors such as noise, inconsistencies, and redundancy were checked.

Text Noise:

* Tweets contain URLs, mentions (@user), hashtags, punctuations and special characters.

Solution:

* Using regex and NLP techniques (NLTK) to clean text.

Sentiment Labels:

* The sentiment column contains irrelevant labels (NaN or Not Relevant), which should be removed.

Class Balance Issue:

* The dataset has more neutral tweets compared to positive and negative tweets.
* Solution: Oversample or under sample to balance classes.

**3.DATA PREPARATION**

The Data Preparation phase involves selecting, cleaning, transforming, and formatting the data to ensure it is ready for analysis and modeling. The steps taken include:

**3.1 Data Selection**

The dataset consists of tweets related to Apple and their corresponding sentiment labels. After the Data Understanding phase, we identified the relevant columns for analysis:

* text: Contains the raw tweet.
* sentiment: Represents the sentiment of the tweet (1 = Negative, 3 = Neutral, 5 = Positive).

We removed unnecessary columns and retained only those relevant to sentiment analysis.

* data = df[['text', 'sentiment']] <--- Used this line of code to retain the relevant columns for our sentiment analysis.

**3.2 Data Cleaning and preprocessing**

***3.2.1 Handling Duplicate Values***

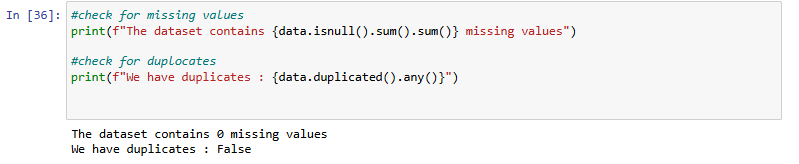
* We found 643 duplicate rows in the dataset, which were removed to ensure each tweet contributes uniquely to the model.

data.drop\_duplicates(inplace=True) <----- Used this line of code to drop the duplicated rows.



***3.2.2 Handling Missing Values***

* We checked for missing values in the dataset and confirmed there were none.
* data.isnull().sum() <----- Used this line of code to check for missing values.



***3.2.3 Handling Irrelevant Sentiment Labels***

* The dataset contained a **'not\_relevant'** sentiment label, which was removed as it does not contribute to the classification task



***3.2.4 Converting Sentiment Labels to Integer Format***

* The sentiment column was initially of type object. We converted it into an integer format for compatibility with machine learning models.
* data['sentiment'] = data['sentiment'].astype(int) <----- Used line of code to the sentiment column to an integer.

***3.2.5 Text Preprocessing***

Before applying NLP models, we processed the text to remove noise and standardize it.

Removing Noise (Stopwords, URLs, Mentions, Hashtags, Punctuation)

* Stopwords: Words like "the", "is", and "at" that do not contribute to sentiment.
* URLs & Mentions (@username): Removed as they do not add meaning.
* Hashtags & Special Characters: Removed to retain meaningful words.
* Punctuation & Extra Spaces: Removed for consistency.

***3.2.6 Tokenization***

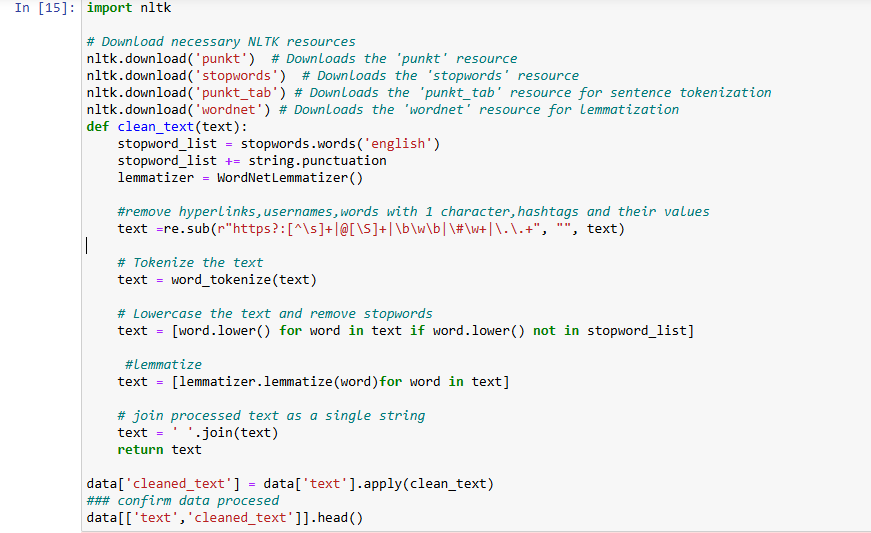
Tokenization breaks down the text into individual words, which helps in understanding the frequency of words in different sentiments.

* data['tokens'] = data['cleaned\_text'].apply(word\_tokenize) <----- Used this line of code to tokenize the cleaned text.

***3.2.7 Lemmatization***

* Lemmatization reduces words to their base form (e.g., "running" → "run") to standardize them.
* lemmatizer = WordNetLemmatizer()

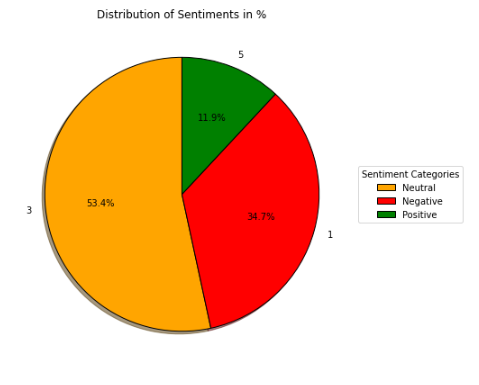
data['tokens'] = data['tokens'].apply(lambda words: [lemmatizer.lemmatize(word) for word in words]) 🡨------ Used this line of code to lemmatize the text



**3.3 Data Visualization**

***3.3.1 Target Distributions***

Our target variable is Sentiment and below are distributions for the target variable



***Handling Class Imbalance***

The dataset is imbalanced, with Neutral tweets (53%) being the majority, and Positive tweets (11%) being the minority. To address this imbalance, we can use:

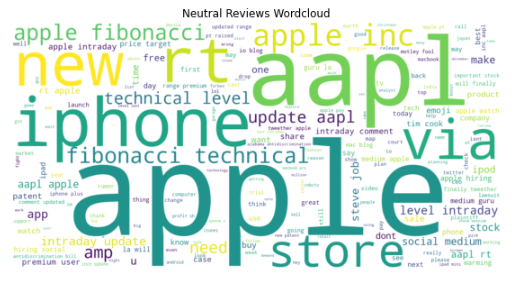
* Oversampling (SMOTE) to increase the number of positive tweets.
* Undersampling to reduce the number of neutral tweets.

***3.3.2 Word clouds***

***Negative Sentiment***



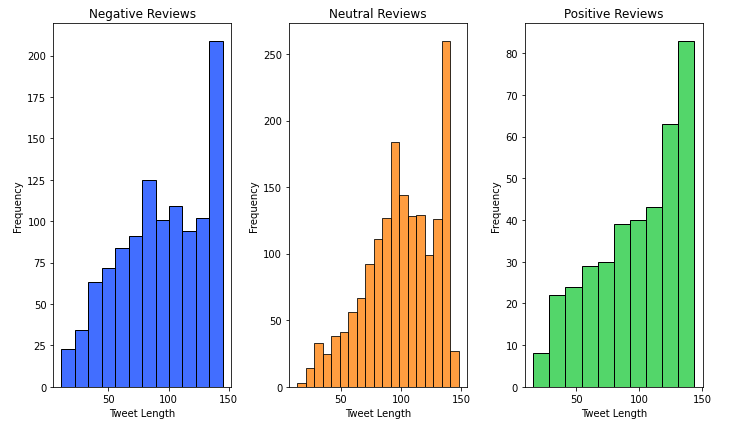
***Neutral Sentiment***



***Positive Sentiment***

******

***3.3.3 Text Length***

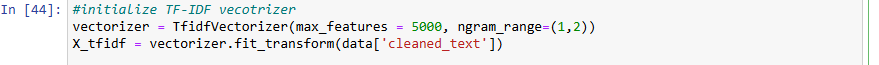
******

***Key Observations of text length***

* Both negative and positive sentiment tweets tend to be longer on average compared to neutral tweets. This might be because people feel more compelled to provide context or elaborate on their experiences (good or bad).
* The length of tweets might be correlated with the emotional intensity of the sentiment. People might use more words to emphasize their feelings.
* The peak at the longer end (around 140-150 characters) might be related to Twitter's original character limit. People might be using the full allotted space to express their strong opinions.

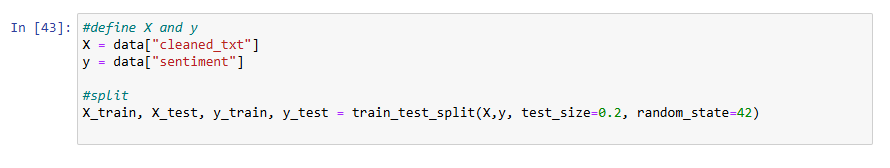
**3.4 Feature Engineering**

* We converted textual data into numerical vectors using TF-IDF (Term Frequency - Inverse Document Frequency) to improve model performance.



**3.5 Data Formatting**

Before training models, we define our target and features and split the dataset into training and testing sets (80-20 split).



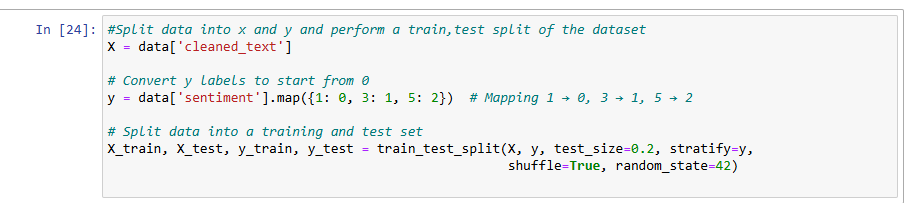
**4.MODELLING**

In this phase, we implement different machine learning models to classify sentiment based on textual data. We perform a train-test split, apply vectorization, and handle class imbalance using SMOTE. Finally, we evaluate models using key performance metrics.

**4.1Train-Test Split**

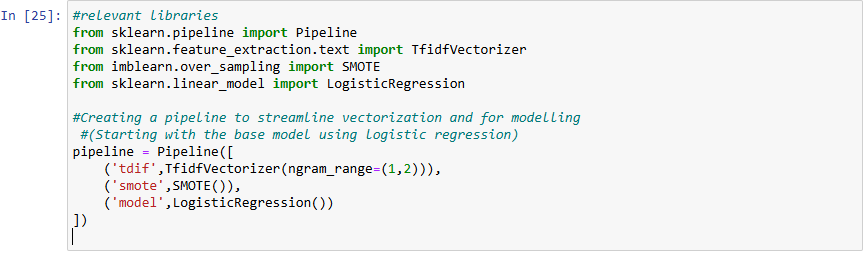
Before training models, we divide the dataset into training and test sets to evaluate performance fairly.

Mapping Converts labels to 0, 1, 2 which allows models to process sentiment labels.



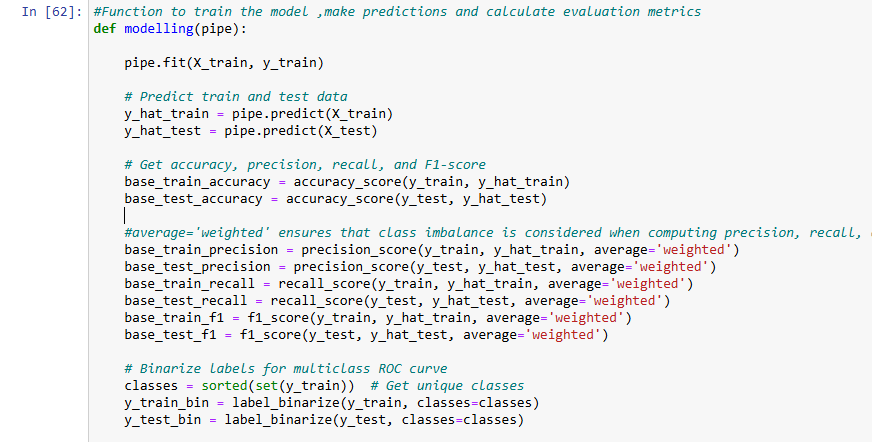
**4.2 Model Pipeline**

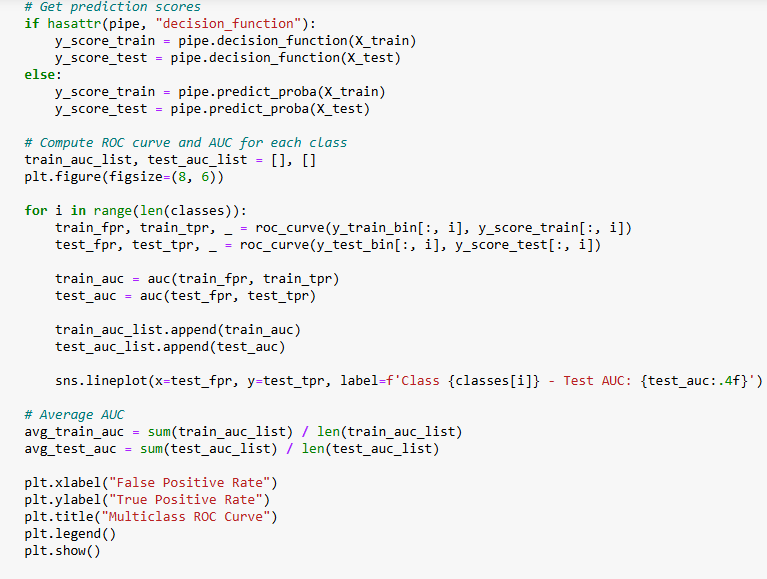
We use pipelines to streamline vectorization, handling class imbalance, and training models.

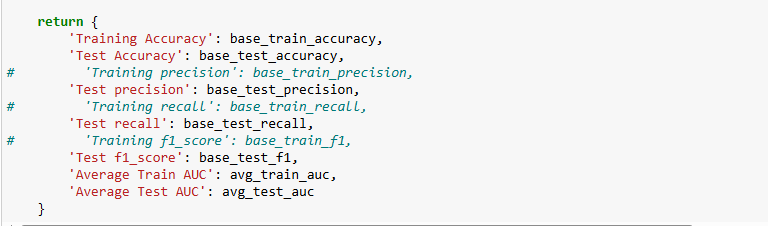


**4.3Model Training & Evaluation**

We created a function to train models, make predictions, and compute evaluation metrics.







The above function trains the model, obtains relevant metrics and plots ROC/AUC curves

**4.4Model Comparisons**

We tested multiple models and compare performance using the above function.

The models included;

Logistic Regression

Random forest

Multinomial Naïve Bayes

Support Vector Machines

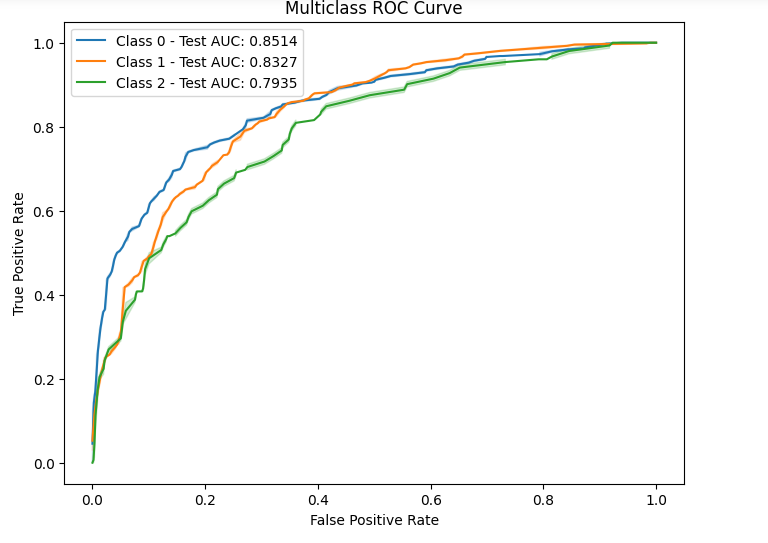
XGBoost

This included changing the parameters of our pipeline and applying modelling using our function

***4.4.1 Logistic Regression***

logreg = modelling(pipeline)

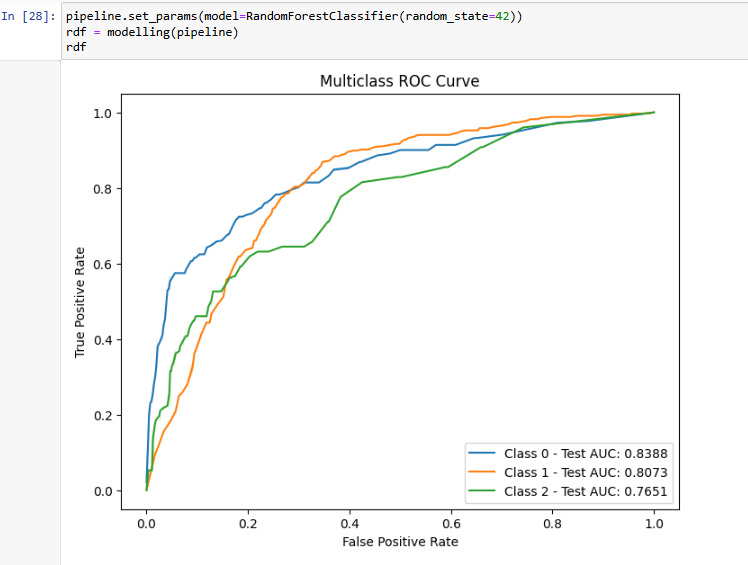
Using the above line of code returned the following output



***Key Findings***

* The model is generally effective at classifying data across all three classes, as evidenced by the AUC scores being above 0.79
* The model performs well in classifying data into three classes, with Class 0 showing the best performance and Class 2 the relatively weakest.

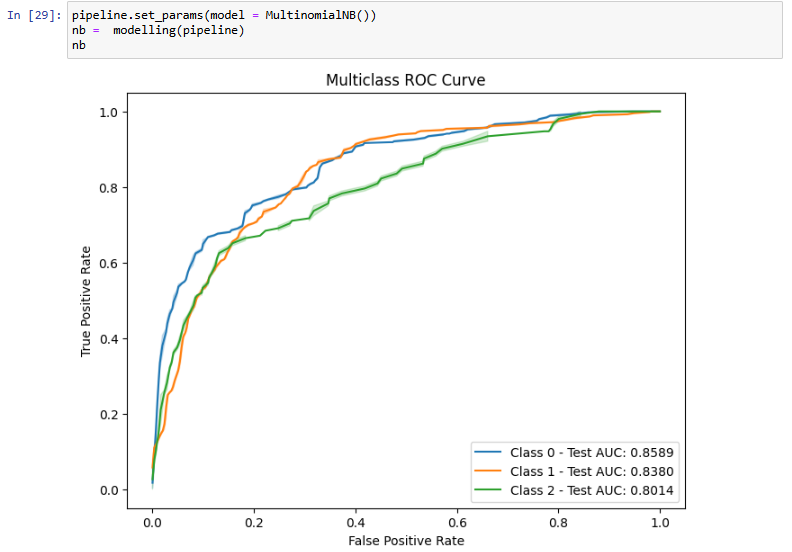
***4.4.2 Random Forest***



***Key Findings***

* The model demonstrates strong overall performance in classifying data across all three classes, with AUC scores consistently above 0.76
* Class 0 achieves the highest predictive accuracy (AUC = 0.8388), followed by Class 1(AUC = 0.8073) Class 2 (AUC = 0.7651) comparatively lower but still acceptable performance
* These results confirm the model’s effectiveness in distinguishing between the classes, with potential for further refinement, particularly for Class 2

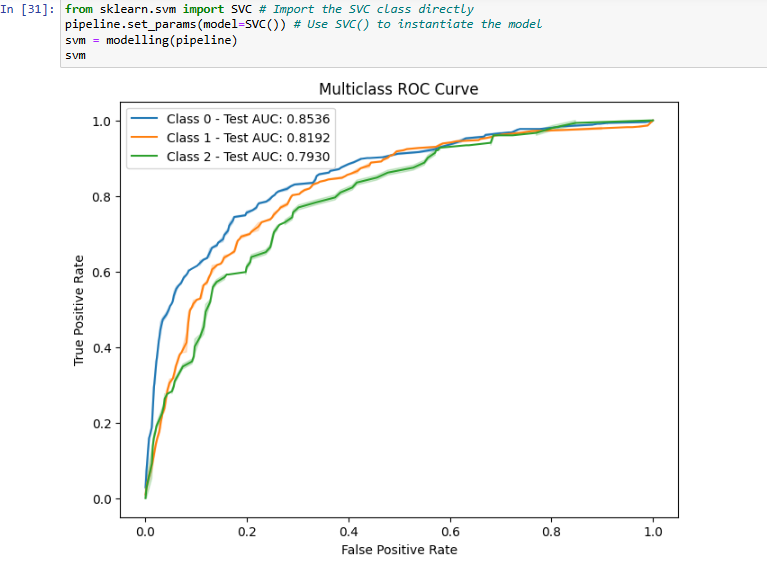
***4.4.3Multinomial Naïve Bayes***



***Key Findings***

* Strong overall performance, with all AUCs > 0.80
* Class 0 is the MOST distinguishable, Class 2 is the least but is still a good score.

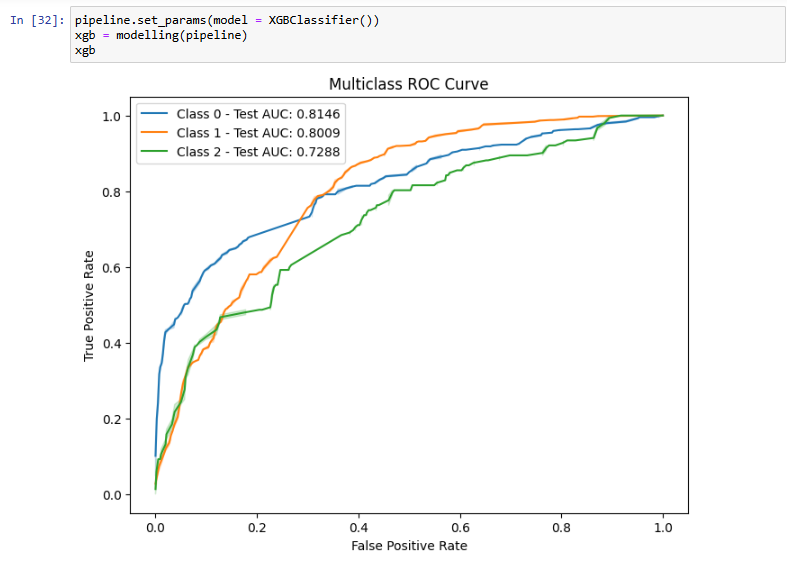
***4.4.4Support Vector Machine***



***Key Findings***

* Strong overall performance with Class 0 leading and class 2 as the least distinguishable yet still robust.
* SVM almost matches MultinomialNB but slightly under performs MultinomialNB
* SVM outperforms the Random Forest model particularly for Class 2

***4.4.5 XGBoost***



***Key findings***

* All classes are still above random chance (0.5)
* The model is still robust but with notable class imbalance challenges

**4.5 MODEL RESULTS**



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

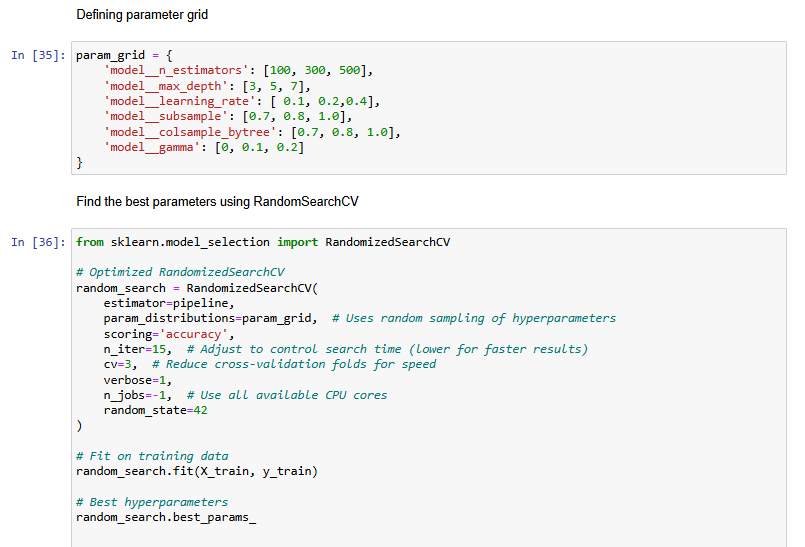
Higher AUC is seen in the models indicating models performs well in separating the classes XGBoost has the best generalization as it has the smallest gap between the train and test accuracy

On simple models logistic regression performs well with a good balance between precision and recall

**5.HYPERPARAMETER TUNING**

***XGBoost***

We optimize XGBoost using RandomizedSearchCV.



***Retraining with Best Parameters***

XGBoost model is retrained with the best parameters which included:

Subsample = 1.0

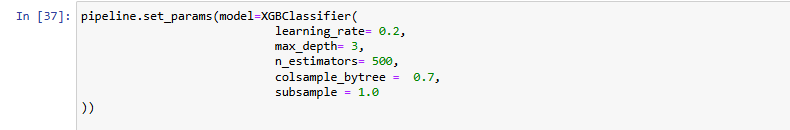
N\_estimators = 100

Max\_depth = 7

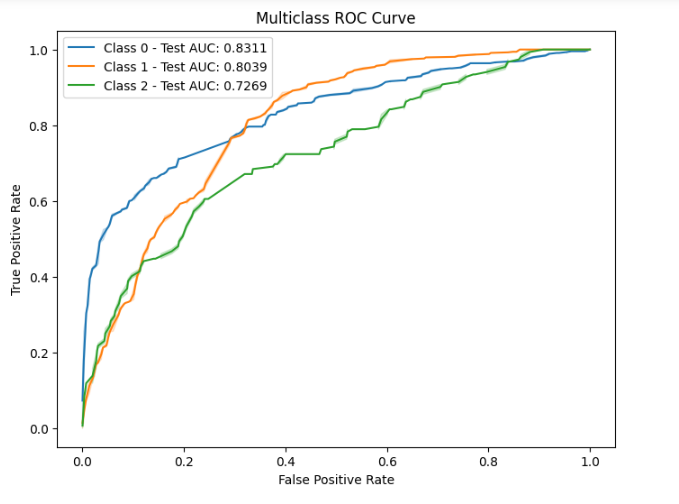
Learning\_rate = 0.1

Gamma = 0.1,

Colsample\_bytree = 0.8



An ROC/AUC curve for tuned Xgboost:



***Key Insights***

* Logistic Regression performed well as a simple model.
* XGBoost had the best generalization ability with optimized hyperparameters.
* Random Forest and Decision Trees were prone to overfitting.
* SVM provided strong performance but was computationally expensive.
* Hyperparameter tuning improved XGBoost marginally but reduced overfitting.

**6.EVALUATION**

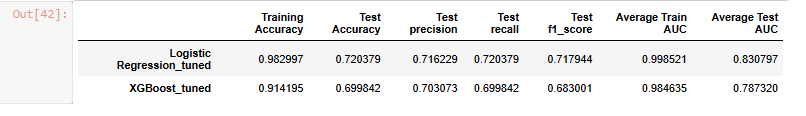
Evaluate Results

Business Success Criteria

The goal of this NLP project was to develop a sentiment classification model that can accurately classify customer sentiments (Positive, Neutral, and Negative) based on textual input. The ideal model should:

* Achieve high accuracy and F1-score to ensure reliable predictions.
* Generalize well to new data with minimal overfitting
* Be computationally efficient for practical deployment.
* Be interpretable and explainable to ensure trust in business applications.

***Model performance comparison***

******

**Key Observations**

* Logistic Regression outperformed XGBoost in terms of test accuracy (71.09% vs. 69.19%) and F1-score.
* XGBoost showed better precision but lower recall, meaning it was more conservative in classifying positive sentiment.
* AUC scores suggest strong class separation, with Logistic Regression achieving an AUC of 0.828, indicating better discrimination between sentiment classes.
* Tuning improved Logistic Regression significantly, boosting test accuracy by 2% and F1-score.
* Overfitting was reduced in XGBoost, but its generalization ability still fell short of Logistic Regression.
* Based on these findings, Logistic Regression is the best model for sentiment classification and should be used for business applications.

**Review Process**

1. **Workflow Assessment**

The modeling phase followed a structured approach:

* Preprocessing: Data cleaning (removal of stopwords, lemmatization) ensured text standardization.
* Feature Engineering: TF-IDF vectorization captured textual features.
* Handling Class Imbalance: SMOTE addressed the imbalance in sentiment classes.
* Model Selection: Various models were tested, including Logistic Regression, SVM, Naïve Bayes, Decision Trees, Random Forest, and XGBoost.
* Hyperparameter Tuning: GridSearchCV/ RandomSearchCV were used to find the best model configurations.
* Evaluation Metrics: Accuracy, precision, recall, F1-score, and ROC-AUC were used for assessment.

1. **Possible Enhancements**

* Feature Engineering Improvements:
* Consider using word embeddings (e.g., Word2Vec, GloVe, or BERT) instead of TF-IDF.
* Explore n-gram combinations for better phrase detection.

1. **More Advanced Models:**

* Implement deep learning (LSTMs or Transformers) for improved contextual understanding.
* Ensemble different models to leverage their strengths.

1. **Data Expansion:**

* Gather more labeled data to further improve generalization.
* Use semi-supervised learning to leverage unlabeled text data.

**7.NEXT STEPS**

Deployment Readiness

✅ Logistic Regression meets business objectives and is ready for deployment.

**Future Enhancements**

* Refine feature engineering: Experiment with embeddings and linguistic features.
* Explore deep learning: Transformers (e.g., BERT) could improve accuracy.
* Monitor model performance: Continuous evaluation using real-world data.
* User feedback integration: Enhance sentiment analysis with real-time user input.