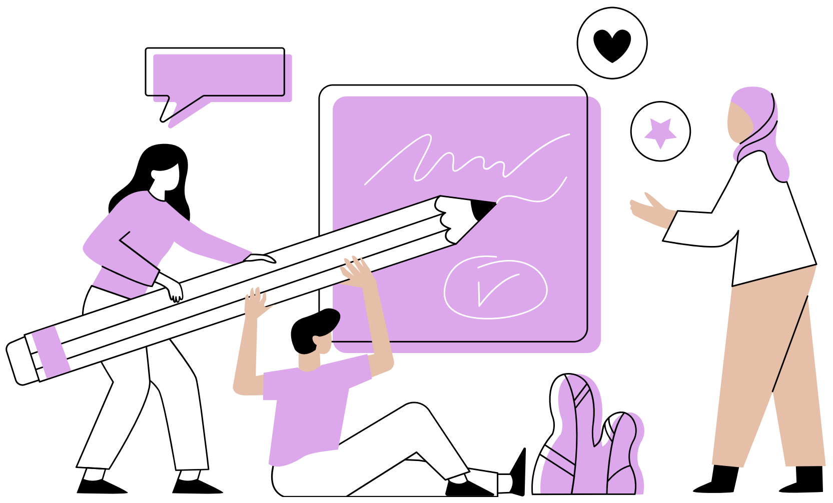
# Report

MACHINE
LEARNING
ASSIGNMENT

NAVDEEP

# TABLE OF CONTENTS



## Exploratory Data Analysis (EDA)

Here are the findings, observations, and patterns from the exploratory data analysis (EDA) performed on the given data:

**1. Data Overview:**
- The dataset contains 8,589 entries with three columns: 'text', 'brand', and 'sentiment'.
- The 'text' column represents the tweet text, 'brand' indicates the brand or product mentioned in the tweet, and 'sentiment' denotes the sentiment expressed in the tweet.

**2. Data Cleaning:**
- The column names have been standardized for easier analysis by renaming them to 'text', 'brand', and 'sentiment'.

**3. Missing Values:**
- There is one missing value in the 'text' column and a significant number of missing values in the 'brand' column (approximately 52% missing).
- Addressing missing values in 'text' may be necessary for further analysis.

**4. Sentiment Distribution:**
- The 'sentiment' column has four categories: 'No emotion toward brand or product', 'Positive emotion', 'Negative emotion', and 'I can't tell'.
- The majority of tweets (5,388) express 'No emotion toward brand or product', followed by 'Positive emotion' (2,672) and 'Negative emotion' (519). There are only a few tweets where sentiment is hard to determine ('I can't tell': 9).

**5. Brand Distribution:**
- The 'brand' column has nine unique values, indicating different brands or products mentioned in the tweets.
- The most mentioned brand is 'iPad', with 946 mentions.

**6. Sentiment Visualization:**
- A bar chart illustrates the distribution of sentiments across all tweets. 'No emotion toward brand or product' dominates the dataset, followed by 'Positive emotion' and 'Negative emotion'.

**7. Brand-Sentiment Relationship:**
- A stacked bar chart shows the distribution of sentiments across different brands. It provides insights into how sentiments vary across brands.
- Some brands may have more positive or negative sentiments, while others may have a higher proportion of neutral sentiments.

**8. Patterns:**
- The dataset seems to be imbalanced, with a substantial number of tweets expressing 'No emotion toward brand or product'. This imbalance may impact the performance of machine learning models, and addressing it may be necessary.
- Certain brands may be more polarizing, eliciting more positive or negative sentiments.

.

## Ways to improve the dataset further and pre-processing

o further improve the dataset for analysis and potential machine learning model development, following steps can be considered:

- **Handle Missing Values:**
  - Address the missing value in the 'text' and ‘brand’ column by either imputing missing values or removing the corresponding rows.
  - Explore techniques like imputation based on surrounding text or utilizing natural language processing (NLP) methods to infer missing text data.

- **Address Class Imbalance:**
  - The dataset is imbalanced, with a significant number of tweets expressing 'No emotion toward brand or product'. Techniques to handle class imbalance include oversampling the minority classes, undersampling the majority class, or using techniques like SMOTE (Synthetic Minority Over-sampling Technique).

- **Text Preprocessing:**
  - Apply text preprocessing techniques such as lowercasing, removing stop words, punctuation, and special characters, and stemming or lemmatization to clean and standardize the text data. This enhances the quality of the text features for machine learning models.

- **Advanced Text Vectorization:**
  - Experiment with advanced text vectorization techniques such as TF-IDF (Term Frequency-Inverse Document Frequency) or word embeddings (Word2Vec, GloVe) to represent text data in a more meaningful way for machine learning algorithms.

- **Explore Additional Features:**
  - Consider incorporating external data sources or additional features that might enhance the predictive power of the model. For example, sentiment analysis models might benefit from including features related to user engagement, retweet counts, or user followers.

## Hyper parameter Tuning and Model Training

**1. Models Selected:**

- Four distinct models were chosen for sentiment analysis:
  - **Multinomial Naive Bayes (MultinomialNB):**
    - Naive Bayes models, including Multinomial Naive Bayes, are well-suited for text classification tasks. They work particularly well with high-dimensional sparse data, such as bag-of-words representations commonly used in natural language processing tasks.
  - **Support Vector Machine (SVM):**
    - SVMs are powerful classifiers that are effective in high-dimensional spaces, making them suitable for text classification. They aim to find a hyperplane that maximally separates different classes in the feature space.
  - **Logistic Regression:**
    - Logistic Regression is a versatile and interpretable model commonly used for binary and multi-class classification tasks. It performs well in scenarios where the relationship between features and the target variable is approximately linear.
  - **Random Forest:**
    - Random Forests are ensemble models that consist of multiple decision trees. They are robust and can capture complex relationships in the data. Random Forests are often chosen for their ability to handle non-linearity and capture interactions between features.

**2. Hyperparameter Grids:**

- A set of hyperparameter grids was defined for each model. These grids span various hyperparameter values, covering both the TF-IDF vectorizer and the specific machine learning model. The hyperparameters include parameters like n-gram range, max features, alpha, C, kernel, number of estimators, and max depth.

**3. Training Process:**

- For each model, a scikit-learn pipeline was created, consisting of a TF-IDF vectorizer followed by the corresponding machine learning model.
- Hyperparameter tuning was performed using Grid Search Cross Validation (GridSearchCV).
- The training process utilized the training data (train_data['text'] and train_labels).
- The best-performing model, as determined by grid search, was selected for further evaluation.

**4. Model Evaluation:**

- The best model obtained from hyperparameter tuning was applied to the validation data (val_data['text']).
- Predictions were generated for the validation data using the best model.
- A classification report was produced, summarizing key metrics such as precision, recall, and F1-score for each sentiment class.

## Evaluation

Let's analyze the classification reports for each model to compare their performance and identify the best model for sentiment analysis:

**Multinomial Naive Bayes:**
- Accuracy: 68%
- Macro F1-Score: 30%
- Weighted F1-Score: 61%

**Support Vector Machine (SVM):**
- Accuracy: 70%
- Macro F1-Score: 33%
- Weighted F1-Score: 65%

**Logistic Regression:**
- Accuracy: 69%
- Macro F1-Score: 31%
- Weighted F1-Score: 63%

**Random Forest:**
- Accuracy: 66%
- Macro F1-Score: 28%
- Weighted F1-Score: 59%

**Comparison:**
- All models show a noticeable challenge in correctly classifying "Negative emotion" and "Positive emotion."
- SVM outperforms other models in accuracy, macro F1-score, and weighted F1-score.
- Logistic Regression and Multinomial Naive Bayes exhibit similar performance, with Logistic Regression having a slightly higher macro and weighted F1-score.
- Random Forest performs the least well among the models in terms of accuracy and F1-scores.

**Best Model:**
- Based on the overall comparison, the **Support Vector Machine (SVM)** appears to be the best-performing model for this sentiment analysis task. It achieves the highest accuracy and F1-scores, indicating better generalization to the validation dataset.

## Steps to Deploy a Machine Learning Model:

**Model Serialization:**
Serialize the trained model to a format that can be easily loaded during inference. Common serialization formats include pickle, ONNX, or TensorFlow SavedModel.

**Containerization:**
Package the model, its dependencies, and any required preprocessing steps into a container (e.g., Docker container). This containerization simplifies deployment across various platforms.

**API Development:**
Develop an API that exposes the model's functionality. This API should accept input data, perform inference using the deployed model, and return predictions.

**Integration with Deployment Platform:**
Integrate the containerized model with a deployment platform or infrastructure (e.g., Kubernetes, AWS Lambda, or Azure ML). This allows for seamless scaling and management.

**Testing:**
Conduct thorough testing of the deployed model, including unit tests, integration tests, and end-to-end tests. Verify that the model performs as expected in the production environment.

**Rollout Strategy:**
Plan a rollout strategy to gradually deploy the model, monitor its performance, and address any issues incrementally. This may involve deploying to a subset of users initially.

**Continuous Integration/Continuous Deployment (CI/CD):**
Implement CI/CD pipelines to automate the deployment process and ensure that changes can be pushed to production rapidly and reliably.

## Post-Deployment Monitoring Metrics:

After deploying a machine learning model, ongoing monitoring is essential. Key metrics to monitor include:

**Accuracy and Precision:**
Monitor the accuracy and precision of the model to ensure it maintains high predictive performance on new data.

**Latency:**
Track the inference latency to ensure predictions are delivered within acceptable time frames.

**Resource Utilization:**
Monitor the resource utilization of the deployed model, such as CPU and memory usage. Optimize resource allocation for efficiency.

**Error Rates:**
Monitor error rates to identify any degradation in model performance. Track false positives, false negatives, and other error types.

**Data Drift:**
Check for data drift to detect changes in the distribution of incoming data. Adapt the model if the data distribution shifts significantly.

**Security Metrics:**
Monitor security-related metrics, such as access logs and potential security threats, to ensure the model remains secure.

**Feedback Loops:**
Implement feedback loops to collect user feedback and continuously improve the model based on real-world performance.

**Model Versioning:**
Keep track of model versions and monitor the transition between different versions to identify any unexpected behavior or performance changes.

## Repo Link

**Link**