Sentiment Analysis Project

### **Phase 1: Data Exploration**

* Load and examine the dataset to understand its attributes.
* Identify and describe the key variables, including text content and sentiment labels.
* Use statistical methods to summarize the dataset's characteristics, such as shape, descriptive statistics, and class distribution.
* # Load the dataset
* data\_set = pd.read\_csv("/content/testdata.manual.2009.06.14.csv")
* # Rename columns for clarity
* data\_set.columns = ['sentiment','id','date','query','user','tweet\_text']
* # Display dataset information
* data\_set.info()
* # Display dataset shape
* data\_set.shape
* # Display descriptive statistics
* data\_set.describe()
* # Count the occurrences of sentiment labels
* data\_set["sentiment"].value\_counts()
* # Plot the distribution of sentiment labels
* data\_set["sentiment"].value\_counts().plot(kind = "bar")

### **Phase 2: Data Preprocessing**

* Lowercase the text to ensure consistency.
* Remove stop words to eliminate noise from the text.
* Handle special characters and punctuation marks.
* Tokenize the text into individual words or tokens.
* Lemmatize words to normalize variations and reduce dimensionality.
* # Lowercase text and apply other cleaning steps

### **Phase 3: Exploratory Data Analysis (EDA)**

* Visualize the distribution of sentiment labels using histograms or pie charts.
* Analyze the balance of sentiment classes to understand potential biases.

# Visualize distribution of sentiment labels

data\_set["sentiment"].value\_counts().plot(kind = "bar")

### **Phase 4: Text Vectorization**

* Convert preprocessed text into numerical vectors to facilitate machine learning model training.

#### Tasks:

* Chooses text vectorization techniques such as TF-IDF or word embeddings.
* Transform the preprocessed text into numerical representations compatible with machine learning algorithms.
* # Convert preprocessed text into numerical vectors using TF-IDF Vectorizer
* X = data\_set["cleaned\_tweets\_with\_SW"]
* vectorizer = TfidfVectorizer()
* vectorizer.fit(X)
* X = vectorizer.transform(X)

### **Phase 5: Model Selection**

# Train Support Vector Machine (SVM) model

model = svm.SVC(kernel='linear')

model.fit(X\_train, Y\_train)

### **Phase 6: Hyperparameter Tuning**

* Optimize model performance by fine-tuning hyperparameters.

#### Tasks:

* Utilize techniques like grid search or random search to identify optimal hyperparameters.
* Fine-tune hyperparameters to enhance model effectiveness and efficiency.

# Perform hyperparameter tuning using GridSearchCV

parameters = {'kernel': ['linear', 'poly', 'rbf', 'sigmoid'], 'C': [1, 5, 10, 20]}

grid\_search = GridSearchCV(model, parameters, cv=5)

grid\_search.fit(X, Y)

### **Phase 7: Cross-Validation**

implement cross-validation techniques to evaluate model performance on multiple subsets of the dataset.

# Perform cross-validation

cv\_score\_svc = cross\_val\_score(SVC(kernel='linear'), X, Y, cv=5)