# Sentiment Analysis in Social Media Using Predictive Analysis

## Objective

Develop a machine learning model to analyze and predict the sentiment of social media posts (e.g., Twitter, Facebook, Instagram) to help businesses, policymakers, and researchers understand public opinion and trends in real-time.

## Project Workflow

## 1. Data Collection

- \*\*Sources\*\*:  
 - \*\*Twitter API\*\*: Use Tweepy for accessing live tweets.  
 - \*\*Facebook Graph API\*\*: Fetch public posts.  
 - \*\*Kaggle Datasets\*\*: Pre-labeled social media datasets.  
 - \*\*Reddit API (PRAW)\*\*: Extract user comments and posts.  
- \*\*Key Data\*\*:  
 - \*\*Textual Data\*\*: Posts, comments, hashtags.  
 - \*\*Metadata\*\*: User location, post time, likes, shares, and retweets.

## 2. Data Preprocessing

- \*\*Steps\*\*:  
 - \*\*Text Cleaning\*\*: Remove URLs, mentions (@), hashtags (#), emojis, and special characters.  
 - \*\*Tokenization\*\*: Split text into individual words.  
 - \*\*Stopword Removal\*\*: Remove common words (e.g., 'the,' 'is') that do not contribute to sentiment.  
 - \*\*Lemmatization/Stemming\*\*: Convert words to their root forms.  
 - \*\*Vectorization\*\*: Use techniques like TF-IDF or Word Embeddings (Word2Vec, GloVe) for numerical representation of text.

## Example Code

```python  
import re  
import nltk  
from nltk.corpus import stopwords  
from sklearn.feature\_extraction.text import TfidfVectorizer  
  
# Cleaning function  
def clean\_text(text):  
 text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)  
 text = re.sub(r'\@\w+|\#','', text)  
 text = re.sub(r'[^\w\s]', '', text)  
 text = text.lower().strip()  
 return text  
  
# Apply cleaning  
df['cleaned\_text'] = df['text'].apply(clean\_text)  
  
# Vectorization using TF-IDF  
tfidf = TfidfVectorizer(max\_features=5000)  
X = tfidf.fit\_transform(df['cleaned\_text'])  
```

## 3. Exploratory Data Analysis (EDA)

- \*\*Techniques\*\*:  
 - Word Clouds: Visualize the most frequent words.  
 - Sentiment Distribution: Analyze positive, negative, and neutral sentiment distribution.  
 - Correlation Analysis: Check if metadata (e.g., hashtags or user location) influences sentiment.

## Example Visualization

```python  
from wordcloud import WordCloud  
import matplotlib.pyplot as plt  
  
wordcloud = WordCloud(width=800, height=400).generate(' '.join(df['cleaned\_text']))  
plt.figure(figsize=(10, 5))  
plt.imshow(wordcloud, interpolation='bilinear')  
plt.axis('off')  
plt.show()  
```

## 4. Model Selection

- \*\*Machine Learning Models\*\*:  
 - Logistic Regression  
 - Support Vector Machine (SVM)  
 - Naive Bayes  
 - Ensemble Models (Random Forest, XGBoost)  
- \*\*Deep Learning Models\*\*:  
 - LSTM (Long Short-Term Memory)  
 - Bidirectional LSTM or GRU  
 - Transformers (BERT, RoBERTa)

## 5. Model Implementation Example (Using Logistic Regression)

```python  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import classification\_report, confusion\_matrix  
  
# Splitting the dataset  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, df['sentiment'], test\_size=0.2, random\_state=42)  
  
# Model Training  
model = LogisticRegression()  
model.fit(X\_train, y\_train)  
  
# Model Prediction  
y\_pred = model.predict(X\_test)  
print(classification\_report(y\_test, y\_pred))  
```

## 6. Model Evaluation

- \*\*Metrics\*\*:  
 - Accuracy  
 - Precision, Recall, F1-Score  
 - ROC-AUC Score

## Example Visualization

```python  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
# Confusion Matrix  
conf\_matrix = confusion\_matrix(y\_test, y\_pred)  
sns.heatmap(conf\_matrix, annot=True, cmap='Blues', fmt='d')  
plt.xlabel('Predicted')  
plt.ylabel('Actual')  
plt.title('Confusion Matrix')  
plt.show()  
```

## 7. Model Deployment

- \*\*Tools\*\*:  
 - Streamlit or Flask: Build a real-time sentiment analysis dashboard.  
 - FastAPI: Create an API endpoint to serve predictions.

## 8. Possible Enhancements

- Sentiment Lexicon Integration: Combine machine learning with sentiment lexicons like VADER or TextBlob.  
- Multi-Language Support: Train models for multiple languages.  
- Real-Time Monitoring: Use streaming data sources for live sentiment analysis.  
- Explainability: Use SHAP or LIME to interpret model predictions.