Project 1: Sentiment Analysis on Social Media

📝 DESCRIPTION:

Analyze social media posts or reviews to determine the sentiment (positive, negative, neutral). Use natural language processing (NLP) techniques to preprocess the text and train a sentiment analysis model. Visualize sentiment trends over time or by topic.

INTRODUCTION:

This project dives into sentiment analysis, where we'll use social media posts to figure out if people are generally feeling positive, negative, or neutral. We'll use a step-by-step process, from cleaning the data to building and improving machine learning models. We'll also experiment with different techniques and visualizations to understand sentiment trends and hopefully build a tool that can automatically analyze public opinion.

🔧PROJECT STRUCTURE:

🧠 Define the Problem: Understand the objective and type of problem (classification, regression, etc.).

🗂️ Collect and Prepare Data: Obtain the dataset, handle missing values, and preprocess data.

📊 Exploratory Data Analysis (EDA): Visualize data to understand patterns and correlations.

📐 Feature Engineering: Select and create relevant features.

🔀 Split the Data: Divide the dataset into training and testing sets.

🤖 Choose a Model: Select a suitable machine learning algorithm.

🏋️ Train the Model: Train the model using the training set.

📈 Evaluate the Model: Use appropriate metrics to evaluate the model on the test set.

🔧 Improve the Model: Tune hyperparameters, try different algorithms, or enhance features.

🚀 Deploy the Model (optional): Create an application or API to make predictions using the trained model.
🧪 I'll experiment with:

Logistic Regression

Support Vector Machine (SVM)

Multinomial Naive Bayes

LSTM (for deep learning baseline)

BERT (transformer-based fine-tuning)

TF-IDF Vectorization

Word2Vec Embeddings

BERT Embeddings (for contextual understanding)

Text Preprocessing Techniques:

Lowercasing

Removing stop words

Lemmatization

Tokenization

SMOTE (to address class imbalance)

GridSearchCV (for hyperparameter tuning)

Visualization Tools:

Word clouds

Sentiment distribution histograms

Time series sentiment trends

Stacked bar charts (sentiment by topic or category)

Confusion matrix heatmaps

1. 🧠 Define the Problem

The problem is to classify the sentiment of social media posts into three categories: positive, negative, or neutral. This is a multi-class classification problem.

OBJECTIVE:

Figure out what you're trying to do. In this case, it's determining if social media posts are generally positive, negative, or neutral in tone.

🚀 **DEPLOY THE MODEL:**

This step involves putting the trained model into a real-world application where it can be used to make predictions on new data.
This might involve creating an API, deploying to a cloud platform, or embedding the model in a software application.
The code includes an example of how to save the trained model and scaler to files, which can then be loaded and used in a separate application.
Saving Both Model and TF-IDF:

It's essential to save the best_tfidf vectorizer along with the model. This is because when you deploy the model and get new text data, you need to transform that new text into the same numerical format that the model was trained on. You cannot just load the model; you must load the TF-IDF vectorizer used to create the numerical features.

joblib.dump(best_model_nb, 'best_model_nb.joblib')

joblib.dump(best_tfidf, 'best_tfidf.joblib')

2. Loading Both Model and TF-IDF:

The loading section now loads both the model and the TF-IDF vectorizer.

loaded_model_nb = joblib.load('best_model_nb.joblib')

loaded_tfidf = joblib.load('best_tfidf.joblib')

3. Example Usage with New Data:

I've added an example of how you would use the loaded model and TF-IDF vectorizer to predict the sentiment of new text. This is the most important part, as it shows how you would actually use the deployed model.

new_text = ["Absolutely fantastic experience — exceeded my expectations!"].

new_text_vectorized = loaded_tfidf.transform(new_text) Crucially, you use the loaded TF-IDF vectorizer to transform the new text. This ensures that the new data is in the same format as the data the model was trained on.

prediction = loaded_model_nb.predict(new_text_vectorized)

print(f"Predicted sentiment: {prediction}") This prints the predicted sentiment category.

CONCLUSION:

This project successfully implemented sentiment analysis on social media data, achieving promising results with Logistic Regression and SVM models. While the initial Naive Bayes model provided a baseline, hyperparameter tuning significantly improved the performance of Logistic Regression and SVM, demonstrating their effectiveness in capturing nuanced sentiment. The meticulous data preprocessing, feature engineering (TF-IDF), and EDA were crucial for model accuracy

RECOMMENDATION:

Focus on further refining the top-performing Logistic Regression and SVM models, potentially exploring more advanced techniques like BERT embeddings for even better contextual understanding. Also, prioritize deploying the chosen model via an API to facilitate real-time sentiment analysis for practical applications.