Report on Nike Dunk Low Sentiment Analysis

1. Introduction
The objective of this project is to analyze user sentiment from YouTube comments related to **Nike Dunk Low** shoes.
The analysis aims to determine whether comments express **positive, negative, or neutral** sentiments.
This project is beneficial for understanding customer opinions, improving marketing strategies, and enhancing product perception.

The project consists of two main components:
1. A **Jupyter Notebook** (nike_lowdunk_review.ipynb) used for **data preprocessing, exploratory data analysis (EDA), and model training**.
2. A **Streamlit application** (testNikeReview.py) that deploys the trained model for **real-time sentiment prediction** based on user input.

2. Data Analysis (Jupyter Notebook)
2.1 Data Collection
- The dataset consists of YouTube comments extracted from videos reviewing **Nike Dunk Low** shoes.
- Comments include raw textual feedback, which needs cleaning and processing before analysis.

2.2 Data Cleaning and Preprocessing
To ensure high-quality input for the model, the following steps are performed:
- **Removing Special Characters**: Eliminates symbols, punctuation, and numbers.
- **Converting to Lowercase**: Standardizes text to ensure uniformity.
- **Removing Stopwords**: Removes frequently used words that do not contribute to sentiment.
- **Lemmatization/Stemming**: Reduces words to their root forms.

2.3 Exploratory Data Analysis (EDA)
EDA provides insights into the dataset through visualization and statistical analysis:
- **Sentiment Distribution**: Plots showing how many comments are positive, negative, or neutral.
- **Word Cloud Analysis**: Identifies frequently used words in each sentiment category.
- **Most Common Phrases**: Highlights commonly occurring phrases in positive and negative reviews.
- **Sentiment Trend Over Time**: Analyzes sentiment evolution based on comment timestamps.

3. Model Training (Streamlit Application)

3.1 Model Pipeline Setup
The model is built using **Scikit-learn** components:
1. **TF-IDF Vectorizer**: Converts textual data into numerical format.
2. **Naïve Bayes Classifier (MultinomialNB)**: A probabilistic algorithm well-suited for text classification.

3.2 Model Training and Persistence
- The dataset is split into **80% training and 20% testing**.
- The trained pipeline is saved using **joblib** as `sentiment_pipeline.joblib`.

3.3 Model Evaluation
- The trained model is evaluated on the test set using **accuracy, precision, recall, and F1-score**.

4. Deployment and User Interaction
4.1 Streamlit Web Application
The **testNikeReview.py** script provides a web interface using **Streamlit**:
- Users can **enter a comment** into a text box.
- The trained model predicts whether the comment is **positive, negative, or neutral**.
- The result is displayed instantly to the user.

4.2 Loading and Using the Model
- If `sentiment_pipeline.joblib` exists, it is loaded for predictions.
- If not, the model is trained again from scratch.

5. Conclusion
5.1 Key Takeaways
- **TF-IDF + Naïve Bayes** provides an effective sentiment classification model.
- **Streamlit enhances usability** by allowing real-time sentiment analysis.
- **Model persistence via joblib** ensures efficient deployment.

5.2 Potential Improvements
1. **Use Advanced NLP Models**: Implement **BERT, RoBERTa** for improved accuracy.
2. **Expand the Dataset**: Gather more comments for better generalization.
3. **Deploy Online for Wider Accessibility**: Host the app on **Streamlit Sharing, Hugging Face Spaces, or Heroku**.
4. **Integrate Additional Features**: Add **graphical visualizations** for sentiment trends.

By implementing these improvements, the project can evolve into a **more robust sentiment analysis tool**.