# German Text Phrase Classifier

## Overview:

This project aims to develop and deploy a machine learning model to classify short German text phrases into predefined categories. The project encompasses the entire pipeline from data preprocessing and model training to deployment using a REST API, with the deployment process dockerized for consistency and ease of use.

## Components

### Model Training (train_model.py):

- The script train_model.py handles data loading, preprocessing, model training, and saving.
- Data preprocessing includes converting text to lowercase, removing non-alphabetic characters, and excluding German stop words.
- The model of choice is LinearSVC due to its efficiency with high-dimensional text data.
- The TF-IDF approach is used for feature extraction to convert text data into a format that is usable by the model.

### REST API (app.py):

- Developed using FastAPI, the API serves the trained model.
- The API includes endpoints for receiving text input and returning the classification result.
- Error handling is implemented to manage scenarios like empty input and unavailable model files.

### Testing (test_app.py):

- Tests are written using pytest to ensure the API functions correctly.
- Test cases include checking the availability of the API form, the response of the prediction endpoint to valid input, and validation handling for empty input.

### Docker (Dockerfile):

- The Dockerfile defines the environment to run the application, ensuring that it behaves consistently across various setups.
- It sets up a Python environment, installs necessary dependencies, and exposes the appropriate port for FastAPI.

### Frontend (form.html):

- Located in the static directory, form.html provides a basic user interface for interacting with the model.
- The form allows users to input a German text phrase and view the classification result.

## Machine Learning Pipeline

### LinearSVC and TF-IDF Vectorizer:

- LinearSVC was selected for its effectiveness in handling high-dimensional data, typical in text classification tasks.
- TF-IDF vectorization was chosen to convert text into a numerical format, emphasizing important words while reducing the impact of frequently occurring but less informative words.

### FastAPI for REST API:

Chosen for its ease of use, performance, and ability to handle asynchronous requests, FastAPI provides a robust framework for deploying the model.

### Docker for Deployment:

Using Docker standardizes the deployment process, avoiding the "works on my machine" problem and ensuring the application runs consistently in different environments.

## Testing and Validation

### Unit Tests:

Comprehensive unit tests are written to ensure the API endpoints function as expected.
Tests cover various scenarios including valid inputs, invalid inputs, and system error conditions.

### Model Evaluation:

- The model is evaluated using accuracy and other metrics (commented out in the code but can be enabled for a detailed analysis).
- Emphasis is placed on understanding and interpreting these metrics in the context of the application rather than just achieving high accuracy.

## Future Enhancements

### Advanced NLP Models:

Exploring advanced NLP models like BERT, specifically versions fine-tuned for the German language, could potentially improve classification accuracy.

## Conclusion

This documentation covers the major components and decisions in the German Text Phrase Classifier project. The project demonstrates a complete pipeline from model training to deployment, with a focus on practical implementation and thoughtful problem-solving.