PROJECT SPECIFICATION

Disaster Response Pipeline

Github & Code Quality

CRITERIA	MEETS SPECIFICATIONS
The project demonstrates an understanding of Git and Github.	All project code is stored in a GitHub repository and a link to the repository has been provided for reviewers. The student made at least 3 commits to this repository.
The project shows proper use of documentation.	The README file includes a summary of the project, how to run the Python scripts and web app, and an explanation of the files in the repository. Comments are used effectively and each function has a docstring.
The project code is clean and modular.	Scripts have an intuitive, easy-to-follow structure with code separated into logical functions. Naming for variables and functions follows the PEP8 style guidelines.

ETL

CRITERIA	MEETS SPECIFICATIONS
ETL script runs without errors.	The ETL script, process_data.py, runs in the terminal without errors. The script takes the file paths of the two datasets and database, cleans the datasets, and stores the clean data into a SQLite database in the specified database file path.
ETL script properly cleans the data.	The script successfully follows steps to clean the dataset. It merges the messages and categories datasets, splits the categories column into separate, clearly named columns, converts values to binary, and drops duplicates.

Machine Learning

CRITERIA	MEETS SPECIFICATIONS
Machine learning script runs without errors.	The machine learning script, train_classifier.py, runs in the terminal without errors. The script takes the database file path and model file path, creates and trains a classifier, and stores the classifier into a pickle file to the specified model file path.
The project shows an understanding of NLP techniques to process text data.	The script uses a custom tokenize function using nltk to case normalize, lemmatize, and tokenize text. This function is used in the machine learning pipeline to vectorize and then apply TF-IDF to the text.
The project demonstrates proper use of pipelines and grid search.	The script builds a pipeline that processes text and then performs multi-output classification on the 36 categories in the dataset. GridSearchCV is used to find the best parameters for the model.
The project demonstrates an understanding of training vs. test data and model evaluation.	The TF-IDF pipeline is only trained with the training data. The f1 score, precision and recall for the test set is outputted for each category.

Deployment

CRITERIA	MEETS SPECIFICATIONS
The web app runs without errors and displays visualizations that describe the training data.	The web app, run.py, runs in the terminal without errors. The main page includes at least two visualizations using data from the SQLite database.
The web app successfully uses the trained model to input text and return classification results.	When a user inputs a message into the app, the app returns classification results for all 36 categories.

Suggestions to Make Your Project Stand Out!

- Go into more detail about the dataset and your data cleaning and modeling process in your README file, add screenshots of your web app and model results.
- Add more visualizations to the web app.
- Based on the categories that the ML algorithm classifies text into, advise some organizations to connect to.
- Customize the design of the web app.
- Deploy the web app to a cloud service provider.
- Improve the efficiency of the code in the ETL and ML pipeline.
- This dataset is imbalanced (ie some labels like water have few examples). In your README, discuss how this imbalance, how that affects training the model, and your thoughts about emphasizing precision or recall for the various categories.