Disaster Response Pipeline

REVIEW | HISTORY

Meets Specifications

Congratulations on completing the project! You should be very proud of your accomplishments in building ETL and machine learning pipelines, in addition to a web app that includes data visualizations and a message classification tool!

> You deserve true appreciation for completing the project in the second attempt. I am completely impressed with this submission. Might nominate your project for excellence.

Github & Code Quality

✓ **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.**

Great work here, Have a look at gitlab for an end to end to understand the flow of how applications work in general.

https://about.gitlab.com/

✓ **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.**

Good job in adding docstrings.

✓ **Scripts have an intuitive, easy-to-follow structure with code separated into logical functions. Naming for variables and functions follows the PEP8 style guidelines.**

Overall, your code structure and naming is great! One thing to improve on would be using more descriptive, explicit names for your variables

ETL

✓ **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.**

Nice job getting this to work!

✓ **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.**

Appropriate and thorough steps were taken to clean this dataset.

Machine Learning

✓ **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.**

Good work getting this to run smoothly!

✓ **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 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 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.**

Nice work using pipeline and grid search to create your model!

Deployment

✓ **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.**

✓ **When a user inputs a message into the app, the app returns classification results for all 36 categories.**

RETURN TO PATH