DSND Term 2

Project 3:

Disaster Response Pipelines

2021.02.20.

1.\_ Project Introduction

- Robert Munra, CTO, Figure Eight

2.\_ Project Overview

- message + category data

- supervised ML model



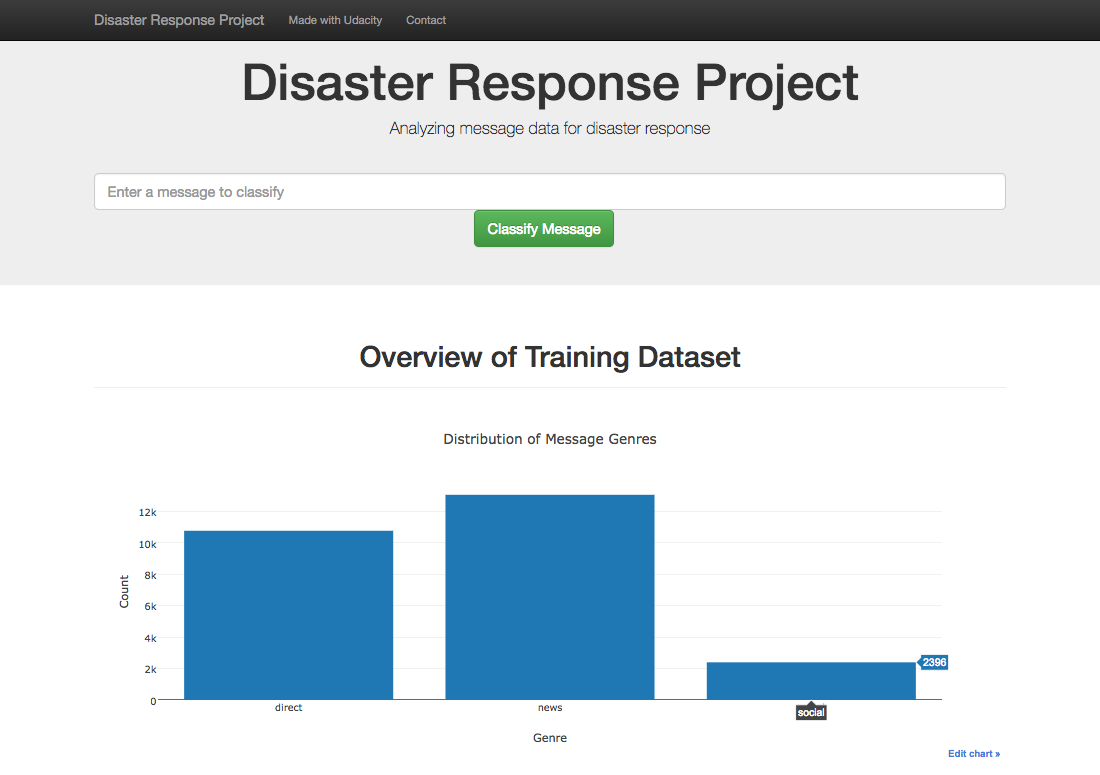
**Project Overview**

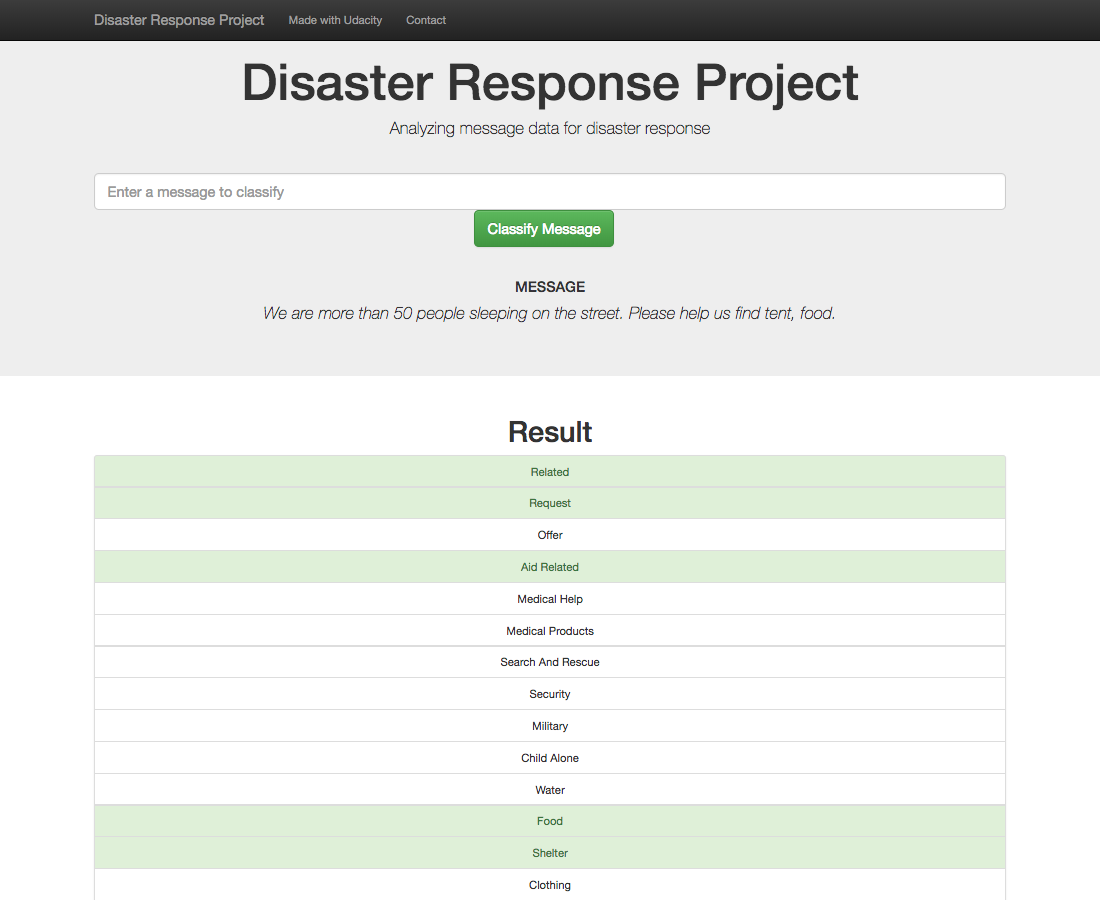
In this course, you've learned and built on your data engineering skills to expand your opportunities and potential as a data scientist. In this project, you'll apply these skills to analyze disaster data from [Figure Eight](https://www.figure-eight.com/) to build a model for an API that classifies disaster messages.

In the Project Workspace, you'll find a data set containing real messages that were sent during disaster events. You will be creating a machine learning pipeline to categorize these events so that you can send the messages to an appropriate disaster relief agency.

Your project will include a web app where an emergency worker can input a new message and get classification results in several categories. The web app will also display visualizations of the data. This project will show off your software skills, including your ability to create basic data pipelines and write clean, organized code!

Below are a few screenshots of the web app.





**Project Components**

There are three components you'll need to complete for this project.

**1. ETL Pipeline**

In a Python script, process\_data.py, write a data cleaning pipeline that:

* Loads the messages and categories datasets
* Merges the two datasets
* Cleans the data
* Stores it in a SQLite database

**2. ML Pipeline**

In a Python script, train\_classifier.py, write a machine learning pipeline that:

* Loads data from the SQLite database
* Splits the dataset into training and test sets
* Builds a text processing and machine learning pipeline
* Trains and tunes a model using GridSearchCV
* Outputs results on the test set
* Exports the final model as a pickle file

**3. Flask Web App**

We are providing much of the flask web app for you, but feel free to add extra features depending on your knowledge of flask, html, css and javascript. For this part, you'll need to:

* Modify file paths for database and model as needed
* Add data visualizations using Plotly in the web app. One example is provided for you

**Github and Code Quality**

Your project will also be graded based on the following:

* Use of Git and Github
* Strong documentation
* Clean and modular code

Follow the [RUBRIC](https://review.udacity.com/#!/rubrics/1565/view) when you work on your project to assure you meet all of the necessary criteria for developing the pipelines and web app.

3.\_ Project Details

**Project Details**

Below are additional details about each component and tips to get you started.

**Data Pipelines: Jupyter Notebooks**

We've provided Jupyter notebooks in Project Workspaces with instructions to get you started with both data pipelines. The Jupyter notebook is not required for submission, but highly recommended to complete before getting started on the Python script.

**Project Workspace - ETL**

The first part of your data pipeline is the Extract, Transform, and Load process. Here, you will read the dataset, clean the data, and then store it in a SQLite database. We expect you to do the data cleaning with pandas. To load the data into an SQLite database, you can use the pandas dataframe .to\_sql() method, which you can use with an SQLAlchemy engine.

Feel free to do some exploratory data analysis in order to figure out how you want to clean the data set. Though you do not need to submit this exploratory data analysis as part of your project, you'll need to include your cleaning code in the final ETL script, process\_data.py.

**Project Workspace - Machine Learning Pipeline**

For the machine learning portion, you will split the data into a training set and a test set. Then, you will create a machine learning pipeline that uses NLTK, as well as scikit-learn's Pipeline and GridSearchCV to output a final model that uses the message column to predict classifications for 36 categories (multi-output classification). Finally, you will export your model to a pickle file. After completing the notebook, you'll need to include your final machine learning code in train\_classifier.py.

**Data Pipelines: Python Scripts**

After you complete the notebooks for the ETL and machine learning pipeline, you'll need to transfer your work into Python scripts, process\_data.py and train\_classifier.py. If someone in the future comes with a revised or new dataset of messages, they should be able to easily create a new model just by running your code. These Python scripts should be able to run with additional arguments specifying the files used for the data and model.

**Example:**

python process\_data.py disaster\_messages.csv disaster\_categories.csv DisasterResponse.db

python train\_classifier.py ../data/DisasterResponse.db classifier.pkl

Templates for these scripts are provided in the Resources section, as well as the **Project Workspace IDE**. The code for handling these arguments on the command line is given to you in the templates.

**Flask App**

In the last step, you'll display your results in a Flask web app. We have provided a workspace for you with starter files. You will need to upload your database file and pkl file with your model.

This is the part of the project that allows for the most creativity. So if you are comfortable with html, css, and javascript, feel free to make the web app as elaborate as you would like.

In the starter files, you will see that the web app already works and displays a visualization. You'll just have to modify the file paths to your database and pickled model file as needed.

There is one other change that you are required to make. We've provided code for a simple data visualization. Your job will be to create two additional data visualizations in your web app based on data you extract from the SQLite database. You can modify and copy the code we provided in the starter files to make the visualizations.

**Github and Code Quality**

Throughout the process, make sure to push your code and comments to Github so that you will not repeat your work and you can keep track of the changes you've made. This will also help you keep your code modular and well documented. Make sure to include effective comments and docstrings. These software engineering practices will improve your communication and collaboration in the future when you work within a team.

**Starter Code**

The coding for this project can be completed using the Project Workspace IDE provided. Here's the file structure of the project:

- app

| - template

| |- master.html # main page of web app

| |- go.html # classification result page of web app

|- run.py # Flask file that runs app

- data

|- disaster\_categories.csv # data to process

|- disaster\_messages.csv # data to process

|- process\_data.py

|- InsertDatabaseName.db # database to save clean data to

- models

|- train\_classifier.py

|- classifier.pkl # saved model

- README.md

**Running the Web App from the Project Workspace IDE**

When working in the Project Workspace IDE, here is how to see your Flask app.

Open a new terminal window. You should already be in the workspace folder, but if not, then use terminal commands to navigate inside the folder with the run.py file.

Type in the command line:

python run.py

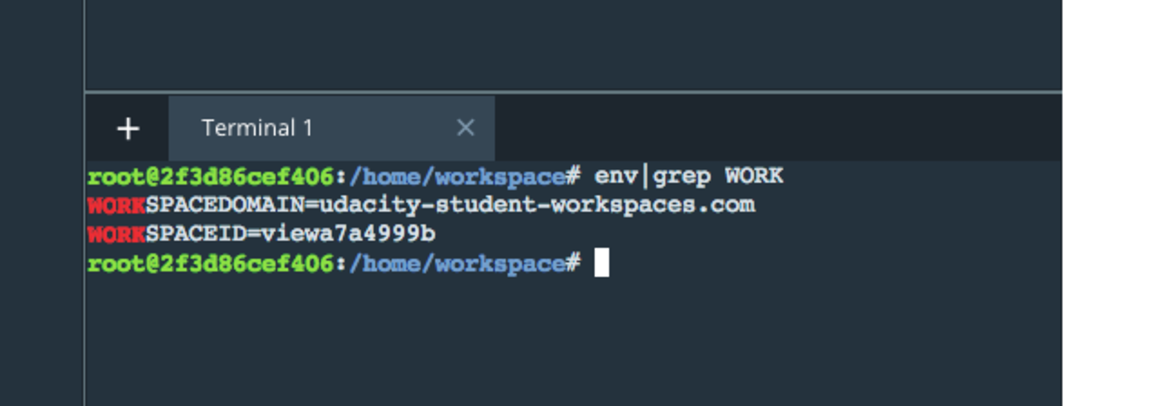
Your web app should now be running if there were no errors.

Now, open another Terminal Window.

Type

env|grep WORK

You'll see output that looks something like this:



In a new web browser window, type in the following:

https:*//SPACEID-3001.SPACEDOMAIN*

In this example, that would be: "[https://viewa7a4999b-3001.udacity-student-workspaces.com/"](https://viewa7a4999b-3001.udacity-student-workspaces.com/%22) (Don't follow this link now, this is just an example.)

Your SPACEID might be different.

You should be able to see the web app. The number 3001 represents the port where your web app will show up. Make sure that the 3001 is part of the web address you type in.

**Rubric**

Follow the [RUBRIC](https://review.udacity.com/#!/rubrics/1565/view) when you work on your project to assure you meet all of the necessary criteria for developing the pipelines and web app.

Supporting Materials

7.\_ Project: Disaster Response Pipelines

Project Submission

Your **deliverables** will be a **Github repo** containing your Python file and files for a working web app. Remember that reviewers will run your Python script with the original data set and then check that the script outputs a SQLite database file and a trained scikit-learn model.

**Before you submit:**

Ensure you meet specifications for all items in the [Project Rubric](https://review.udacity.com/#!/rubrics/1565/view). Your project must **Meet Specifications** in each category in order for your submission to pass.

It can take us up to a week to grade the project, but in most cases it is much faster. You will get an email once your submission has been reviewed. In the meantime, you should feel free to proceed with your learning journey by continuing on to the next module in the program.

Rubric

**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.