**Classification of Disaster Response Messages using machine learning pipelines**

Contents

[**Classification of Disaster Response Messages using machine learning pipelines** 1](#_Toc69077969)

[**Installation** 1](#_Toc69077970)

[**Quick Start** 2](#_Toc69077971)

[**Project Overview** 2](#_Toc69077972)

[**File Descriptions** 3](#_Toc69077973)

[**Results** 4](#_Toc69077974)

[**Acknowledgements** 4](#_Toc69077975)

**Installation**

The code should run using Python versions 3.\*. The necessary libraries are:

* pandas
* re
* sys
* sklearn
* nltk
* sqlalchemy
* pickle

**Quick Start**

1. Run the following commands in the project's root directory to set up your database and model.

- To run ETL pipeline that cleans data and stores in database

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

- To run ML pipeline that trains classifier and saves

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

2. Run the following command in the app's directory to run your web app.

`python run.py`

3. open another Terminal Window and type

env|grep WORK

4. In a new web browser window, type in the following. SPACEID and SPACEDOMAIN values would be known from above step

https:*//SPACEID-3001.SPACEDOMAIN*

**Project Overview**

As part of this project , I have analyzed disaster data from source [Figure Eight](https://appen.com/) and built a model for an API that classifies disaster messages using a ML pipeline into 36 categories. categorized these events , so that any future incoming messages can be sent to an appropriate disaster relief agency.

The dataset contains 26,248 pre-labelled text messages taken from real life disaster

In the web app that is built with this training data one could enter any disaster related message and a classification to related categories will be given to the user as output. The model built will help to respond to future disaster events

**File Descriptions**

- 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

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

- models

|- train\_classifier.py

|- classifier.pkl # saved model

- README.md

The file `process\_data.py` contains an ETL pipeline that:

* Loads data for messages and categories from csv
* Merges two datasets
* Cleans the data
* Saves it in a SQLite database `DisasterResponse.db`

The file `train\_classifier.py` contains a NLP and ML pipeline that:

* Loads data from database `DisasterResponse.db`
* Splits data into training and test sets
* Tokenize , Normalize text and build machine learning pipeline
* Fit the model to Training set and tunes a model using GridSearchCV
* Outputs results on the test set by using the best performance parameters from GridSearchCV
* Stores the model in `classifier.pkl` so that it can be used by the Flask app

The file `run.py` contains a Flask web app that enables the user to enter a disaster message, and then view the categories of the message. The web app also contains some visualizations that describe the data used to train the model.

**Results**

When a disaster message is submitted and the Classify Message button is clicked, the app shows how the message is classified by highlighting the categories in green.

**Acknowledgements**

Udacity.com , Figure Eight