During disaster events, emergency operation centres are usually overwhelmed with messages of various requests. Processing all of them manually could cause slow responses. Thus, Natural language processing (NLP) and machine learning are adopted to build a model for an API that classifies messages during disaster events so that appropriate disaster relief agencies could be informed earlier.

**Dataset:** The dataset is provided by [Figure Eight](https://www.figure-eight.com/data-for-everyone/). It contains more than 26k pre-labelled messages sent during real-life disaster incidents (either via social media or directly to disaster relief organizations). There are 36 pre-defined categories, and "Aid Related", "Shelter", and "Missing People" are some of the examples. In addition, the initial dataset includes the original messages, the English version of messages and the corresponding genres (direct, news, social).

**Deliverables:**

1. Extract Transform Load(ETL) Pipeline: Complete the ETL script for data extracting , data cleaning and SQLite database creation
2. Machine Learning(ML) Pipeline: Complete the ML script to creates the disaster message classifier
3. Flask Web App: Build a web application that displays visualizations regarding the dataset and outputs the classification results based on the user's input in real time.

**2. Project Details**

* ETL Pipeline:
  + Loads and merges the messages and categories datasets
  + Cleans the categories part of the dataset with pandas
  + Stores clean data in a SQLite database with the SQLAlchemy engine
* ML Pipeline:
  + Loads data from the SQLite database created by the ETL pipeline
  + Splits the dataset into training and test sets with the ratio of 8:2
  + By the package, nltk, builds a text processing pipeline which:
    - Cleans and tokenizes each message into seperated words
    - Lemmatizes them to further reduce the complexity of features
    - Vectorizes the text data by computing the Bag of Words and TF-IDF values for feature extraction
    - Extracts the text feature with the custom tranformer, StartingVerbExtractor
    - Performs feature union of the above feature extraction processes
  + Combine the feature extraction pipeline with the AdaBoostClassifier
  + Trains and tunes a model using scikit-learn's GridSearchCV
  + Outputs metrics(precision, recall, F1-score) of the test set
  + Exports the best model as a pickle file
* Flask Web App:
  + With the help of Pandas and Plotly, two data visualizations are created

**3. File Structure & Description**

|-- app

|-- templates

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

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

|-- utils

|-- custom\_scorer.py

|-- custom\_transformer.py

|-- plotting.py # to return figures for flask web app

|-- run.py # Flask file that runs app

|-- data

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

|-- disaster\_categories.csv # categories data to process

|-- disaster\_message.csv # message data to process

|-- process\_data.py # ETL script that takes .csv as input, cleans data and stores them into SQLite database

|-- images

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

|-- disaster\_categories.csv # categories data to process

|-- disaster\_message.csv # message data to process

|-- models

|-- classifier.pkl # saved model

|-- train\_classifier.py # machine learning script that creates and trains a classifier, and stores the classifier into a pickle file

|-- README.md

|-- requirements.txt # list of necessary python packages

### 4.2. Dependencies

The code is developed with Python 3.9.1 and is dependent on python packages listed in requirements.txt. To install required packages, run the following command in the project's root directory:

pip install -r requirements.txt

### 4.3. Executions

* Run the following command in the data directory to clean the data and load them as database:

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

* Run the following command in the models directory to run machine learning pipeline that trains classifier and saves model:

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

* Run the following command in the app directory to launch the web app:

python run.py

Go to <http://0.0.0.0:3001/> when the web app starts running.