

## 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
```

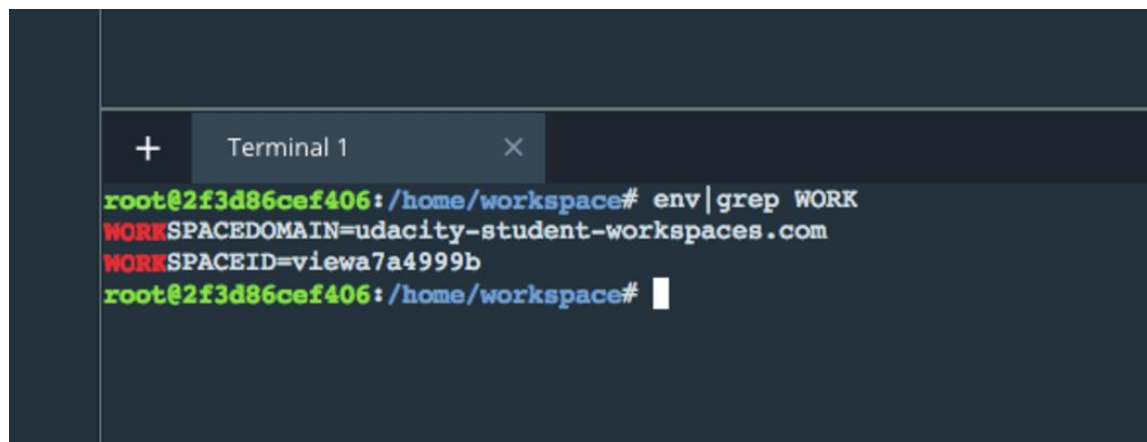
Your web app should now be running if there were no errors.

Now, open another Terminal Window.

Type

```
env|grep WORK
```

You'll see output that looks something like this:

A screenshot of a terminal window titled "Terminal 1". The prompt is "root@2f3d86cef406: /home/workspace#". The command "env|grep WORK" has been entered, and the output is displayed in two lines: "WORKSPACEDOMAIN=udacity-student-workspaces.com" and "WORKSPACEID=viewa7a4999b". The prompt is now ready for the next command.

```
root@2f3d86cef406: /home/workspace# env|grep WORK
WORKSPACEDOMAIN=udacity-student-workspaces.com
WORKSPACEID=viewa7a4999b
root@2f3d86cef406: /home/workspace#
```

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/>" (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](#) when you work on your project to assure you meet all of the necessary criteria for developing the pipelines and web app.

Supporting Materials

[Data Pipeline](#)