Step 4: Build an ML experiment pipeline

Now that we have cleaned up our notebook, we can start building our machine learning pipeline. In essence we do so by creating a dedicated module for each stage in our pipeline and moving the code there as shown in the previous lesson. Additionally we should create a command-line interface (CLI) to run modules from our terminal.

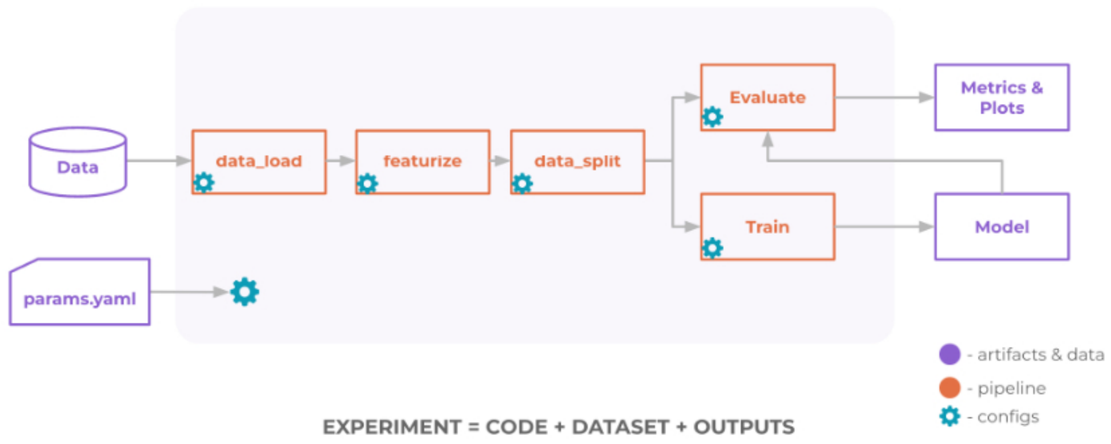
Creating a CLI enables us to run the module using a singular command. We will be using this command later on when defining how our pipeline should run.

Example

Let's consider a basic ML pipeline. We have the following stages:

- `data_load`: load the data from our data sources
- `featurize`: create features from the raw data on which we can train our model
- `data_split`: create a train/test split
- `train`: train a model based on the training data
- `evaluate`: evaluate the model based on the test data

The eventual outputs of this pipeline are the model itself and artifacts related to the evaluation, such as predictions, metrics, and plots.



In order to turn this flow into a proper pipeline, we create a module for each individual stage. Besides the code and dependencies (as shown in the previous lesson) we will also include a `if __name__ == "__main__"` statement in our module. This statement defines the default behavior when running our module and is what is executed when we call our function through the CLI.

Here's what the `data_load.py` module could look like with such a statement:

```
import yaml
```

```
def data_load(config_path: Text) -> None:
    config = yaml.safe_load(open(config_path))
    raw_data_path = config['data_load']['raw_data_path']
    ...
    data.to_csv(config['dataset_processed_path'])

if __name__ == '__main__':
    args_parser = argparse.ArgumentParser()
    args_parser.add_argument('--config', dest='config', required=True)
    args = args_parser.parse_args()

    data_load(config_path=args.config())
```

We could then run the following command from our CLI to execute the `data_load` stage: `$ python data_load.py --config=params.yaml`. This would execute the function and save the data to a `CSV` file in the location that is specified in our `params.yaml`.