Some Horseshoe Nails

Cass Petrus
Lead Data Scientist
@mathcass
MailChimp

Dependent workflows

My typical working tree

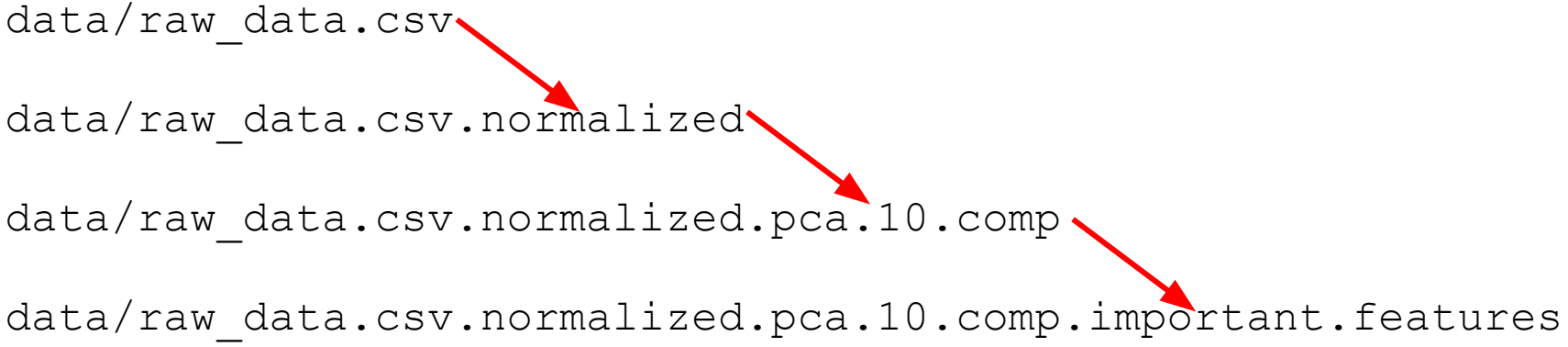
```
data/raw_data.csv

data/raw_data.csv.normalized

data/raw_data.csv.normalized.pca.10.comp

data/raw_data.csv.normalized.pca.10.comp.important.features
```

My typical working tree



Store intermediate data

Multiple datasets

```
data/raw_data.csv

data/raw_data.csv.normalized

data/raw_data.csv.normalized.pca.10.comp

data/raw_data.csv.normalized.pca.10.comp.
important.features
```

Multiple datasets

data/**train**.csv

data/**train**.csv.normalized

data/**train**.csv.normalized.pca.10.comp

data/**train**.csv.normalized.pca.10.comp.
important.features

Multiple datasets

data/**train**.csv

data/**train**.csv.normalized

data/**train**.csv.normalized.pca.10.comp

data/**train**.csv.normalized.pca.10.comp.
important.features

data/**test**.csv

data/**test**.csv.normalized

data/**test**.csv.normalized.pca.10.comp

data/**test**.csv.normalized.pca.10.comp.
important.features

Multiple datasets

data/**train**.csv

data/**train**.csv.normalized

data/**train**.csv.normalized.pca.10.comp

data/**train**.csv.normalized.pca.10.comp.important.features

data/**test**.csv

data/**test**.csv.normalized

data/**test**.csv.normalized.pca.10.comp

data/**test**.csv.normalized.pca.10.comp.important.features

data/**validate**.csv

data/**validate**.csv.normalized

data/**validate**.csv.normalized.pca.10.comp

data/**validate**.csv.normalized.pca.10.comp.important.features

data/**all**.csv

data/**all**.csv.normalized

data/**all**.csv.normalized.pca.10.comp

data/**all**.csv.normalized.pca.10.comp.important.features

Use CLI scripts where possible

Python CLI

```
script.py input.csv output.csv
```

Option parsing is fun

In the beginning

```
import sys


if len(sys.argv) == 3:
    do_stuff(sys.argv[1], sys.argv[2])
else:
    exit("Usage script.py <input> <output>")
```

Building more

```
script.py input.csv \
       output.csv \
       10 \
       1000
```

This would be better

```
script.py -i input.csv \
       -o output.csv \
       --split-size=10 \
       --n-trees=1000
```

Enter Argparse

```
parser = argparse.ArgumentParser()

parser.add_argument("-i", help="Input filename",
                    required=True)

parser.add_argument("-o", help="Output filename",
                    required=True)

parser.add_argument("--vars-per-split", action="store",...
```

Docopt

```
"""Docopt

Usage:

  docopt.py (--input=<input>) (--output=<output>) [--split-size=<split_size>] [--n-
trees=<n_trees>]

Options:

  -i <input> --input=<input>                    Input filename

  -o <output> --output=<output>                Output filename

  -s <split_size> --split-size=<split_size>   Total variables to use per tree [default: 10]

  -n <n_trees> --n-trees=<n_trees>             Total trees to train [default: 1000]

"""
```

Docopt

```
{'--input': 'input.csv',
 '--n-trees': '1000',
 '--output': 'output.csv',
 '--split-size': '10'}
```

click

```
@click.command()

@click.option('--inputf', required=True, help="Input filename")

@click.option('--outputf', required=True, help="Output filename")

@click.option('--split-size', default=10, help="Total variables to use per tree")

@click.option('--n-trees', default=1000, help="Total trees to train")

def run(inputf, outputf, split_size, n_trees):

    """Runs a training on the input file

    """

    pass
```

Resources

- http://zmjones.com/make/
- http://blog.kaggle.com/2012/10/15/make-for-data-scientists/
- https://www.factual.com/blog/introducing-drake-a-kind-of-make-for-data
- https://github.com/docopt/docopt
- http://click.pocoo.org/
- http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html