

Reproducible Machine Learning

Why and how to accomplish reproducibility in ML pipelines

```
~/Timothy Dobbins base 12:06:34 PM
> whoami
timdobbins
~/Timothy Dobbins base 12:06:39 PM
> tree -t
.
├── 2012 Belmont University
│   ├── Economics
│   └── Statistics
├── 2015 Perception Health
│   ├── Software Engineer Intern
│   └── Software Engineer Project lead
├── 2017 The General
│   ├── Data Scientist I
│   ├── Data Scientist II
│   └── Lead Data Scientist
├── 2020 Unum
│   ├── Move To Chattanooga
│   ├── Senior NLP Data Scientist
│   └── Principal Data Scientist
├── 2020 Gridsearch Consulting
│   └── Builder
├── 2022 Trilliant Health
│   └── Principal Data Scientist
└── 18 directories, 0 files
~/Timothy Dobbins base 12:06:41 PM
> |
```

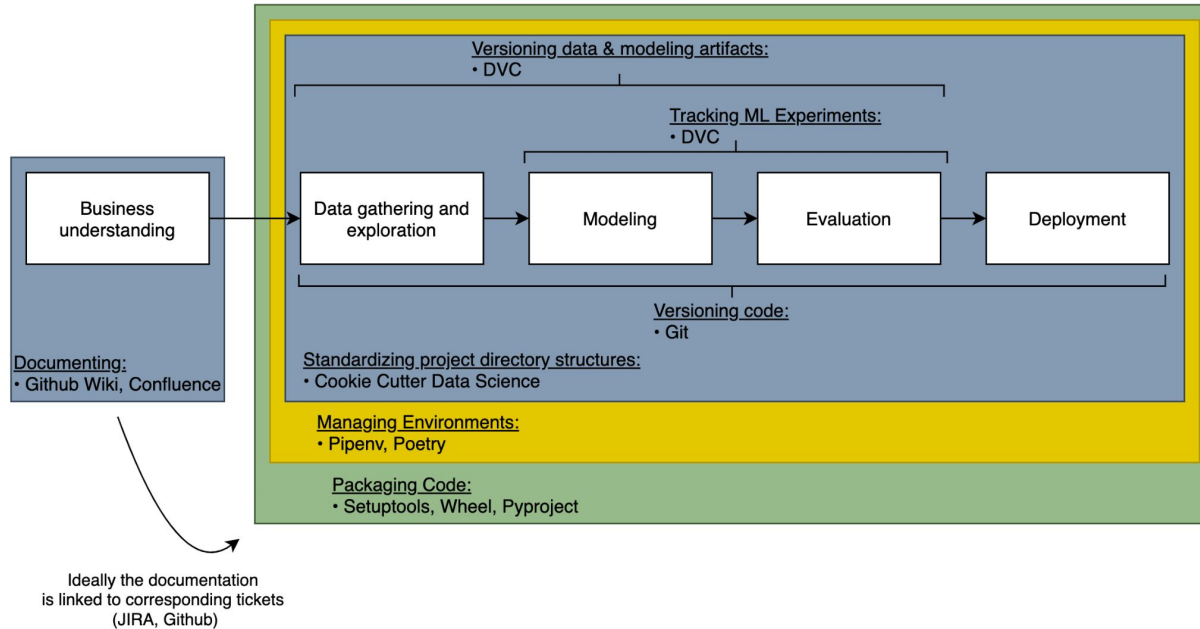
Who works with ML models that are in production?



Who works with reproducible ML models?



An ML reproducibility framework*

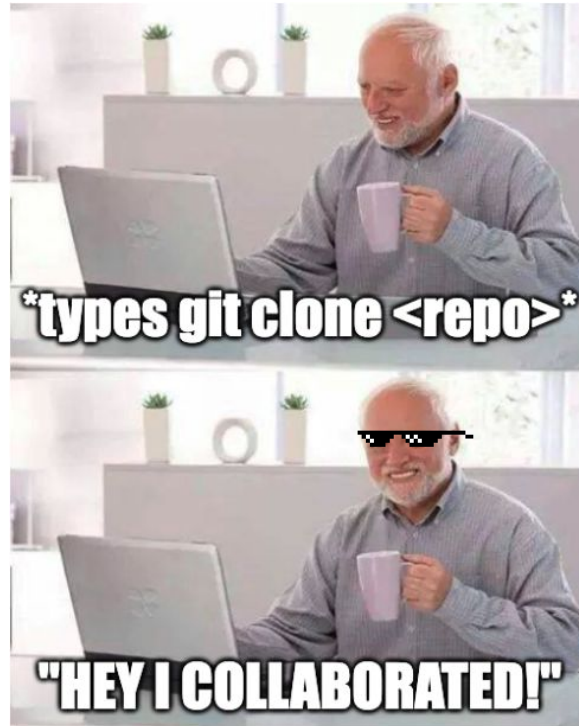


*Inspired by [this framework](#).

Why reproducible machine learning?

Why reproducible machine learning?

- Collaborating



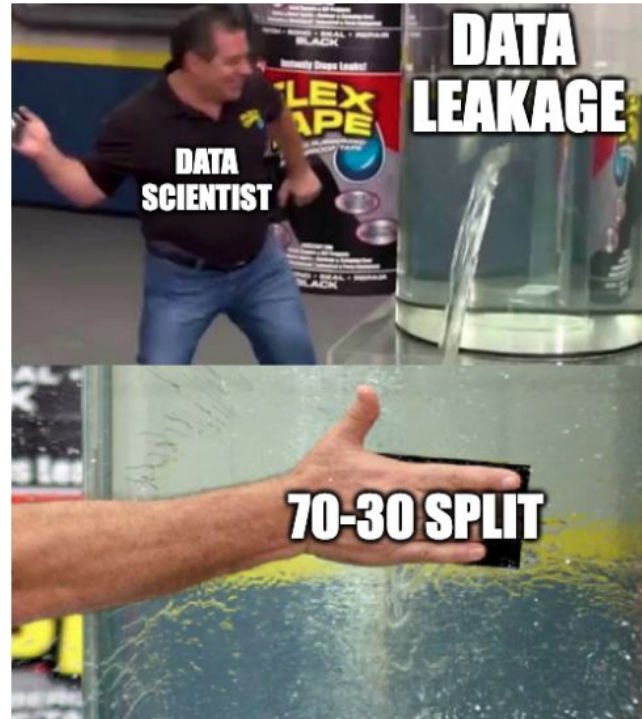
Why reproducible machine learning?

- Collaborating
- Debugging (remotely)



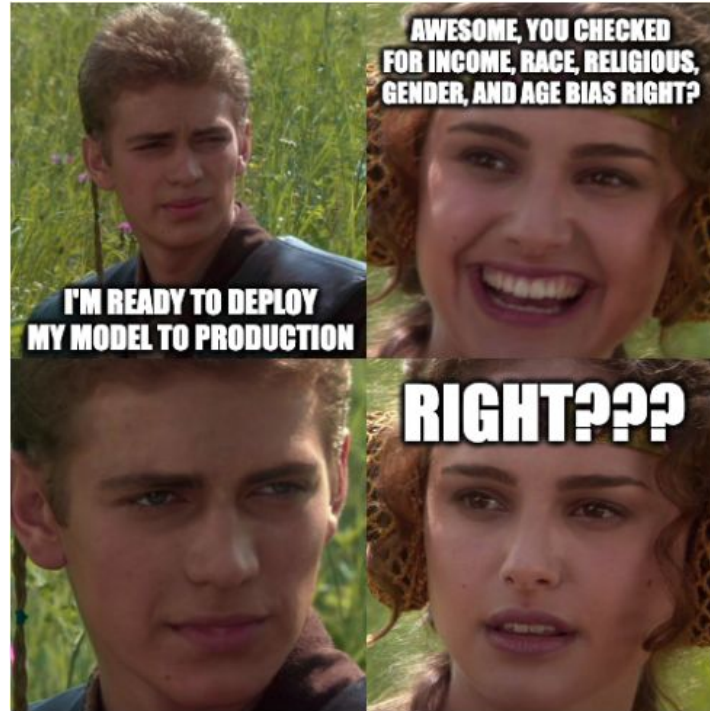
Why reproducible machine learning?

- Collaborating
- Debugging (remotely)
- Scientifically validating



Why reproducible machine learning?

- Collaborating
- Debugging (remotely)
- Scientifically validating
- Auditing



Why reproducible machine learning?

- Collaborating
- Debugging (remotely)
- Scientifically validating
- Auditing
- Legacy



How to accomplish reproducibility?

Four components to account for



1. Environment



common pipenv commands

pipenv install --dev # install all deps in Pipfile

pipenv shell # activate the env

pipenv run <command to run> # run without activating

pipenv sync --dev # sync local env with Pipfile.lock

2. Code



```
# common git commands
```

```
git clone https://github.com/tmthyjames/ds-meetup-ml-repro.git
```

```
git checkout -b <branch>
```

```
git add .
```

```
git commit -m "commit message"
```

```
git push origin <branch>
```

3. Data (data & model artifacts)



```
# common DVC commands
```

```
dvc remote add ...# link remote storage container to local code
```

```
dvc stage add ... # add stage for dvc to track
```

```
dvc dag          # view the pipeline's dag
```

```
dvc repro        # reproduce the entire pipeline
```

```
dvc push         # push all artifacts to remote storage
```


4. Results (performance metrics)



```
# common DVC commands
```

```
$ dvc metrics show
```

```
eval.json:
```

```
AUC: 0.66729
```

```
error: 0.16982
```

```
TP: 516
```

```
$ dvc metrics diff
```

| Path | Metric | HEAD | workspace | Change |
|-----------|--------|---------|-----------|---------|
| eval.json | ACU | 0.65115 | 0.66729 | 0.01614 |
| eval.json | error | 0.1666 | 0.16982 | 0.00322 |
| eval.json | TP | 528 | 516 | -12 |

Let's walkthrough an example

Predicting a song's genre given its lyrics

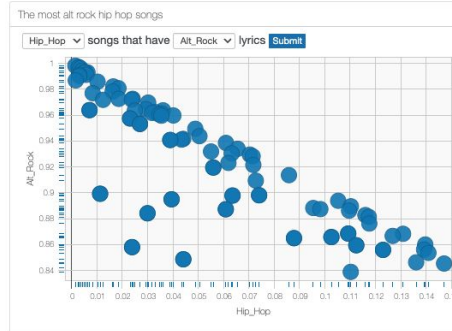


← Link to blog



← Link to code

■ Hip_Hop ■ Alt_Rock ■ Country



The top 100 alt rock hip hop songs

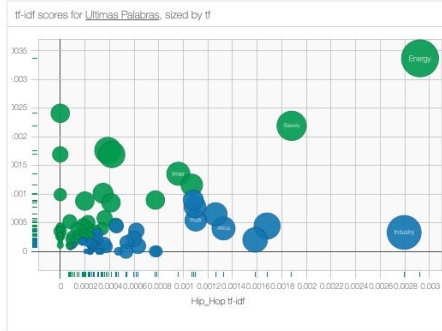
Search:

| Genre | Artist | Song | Hip_Hop | Alt_Rock | Country |
|---------|--------------------|------------------|--------------|--------------|----------|
| Hip_Hop | Immortal_Technique | Ultimas Palabras | 0.0028286947 | 0.9971708538 | 4.615e-7 |

Showing 1 of 1 entries (filtered from 100 total entries)

The most alt rock words weighted by tf-idf for Ultimas Palabras

audience shape a new american revolution has begun not against the forces of a colonial kingdom but a rebellion against an oppressor that has risen among us it is not a foreign invasion we have to fear rather the threat of a force within our nation that has usurped what was once of having the greatest democracy ever known to man we now live in a world where the population has grown exponentially and the planet is running out of resources to sustain us all we in the inner city and those struggling in the suburban ghettos may not realize it yet but make no mistake the people who control technology and run every enterprise that makes up our world have seen this coming for a long time the ideas of renewable energy global warming the idea of collectively working were purposefully bought out teralised demonised or corrupted in favor of an economic structure designed by a monetary cast system in a desperate attempt to convince us that we need to maintain that extravagant existence they've pretended we might share in their dream that we can justify any humanity in its name out of this blind ignorance was born the curse of slavery many of the founders of this country were themselves masons that is not a leftwing or a



Setting up the environment

Setting up the environment




```
# getting started with our example
git clone https://github.com/tmthyjames/ds-meetup-ml-repro.git

# Install pipenv environment
pipenv install --dev

# Create IPython Kernel for the virtual environment
pipenv run ipython kernel install --user --name=reproml

pipenv run python setup.py develop # to get access to the CLI
entry point defined in setup.py
```

Creating the ETL pipeline using DVC



```
# create the ETL pipeline
$ dvc stage add -n get-lyrics \
  -d reproml/etl/lyrics/_lyrics.py \
  -o data/raw/lyrics/ \
  reproml etl get-lyrics

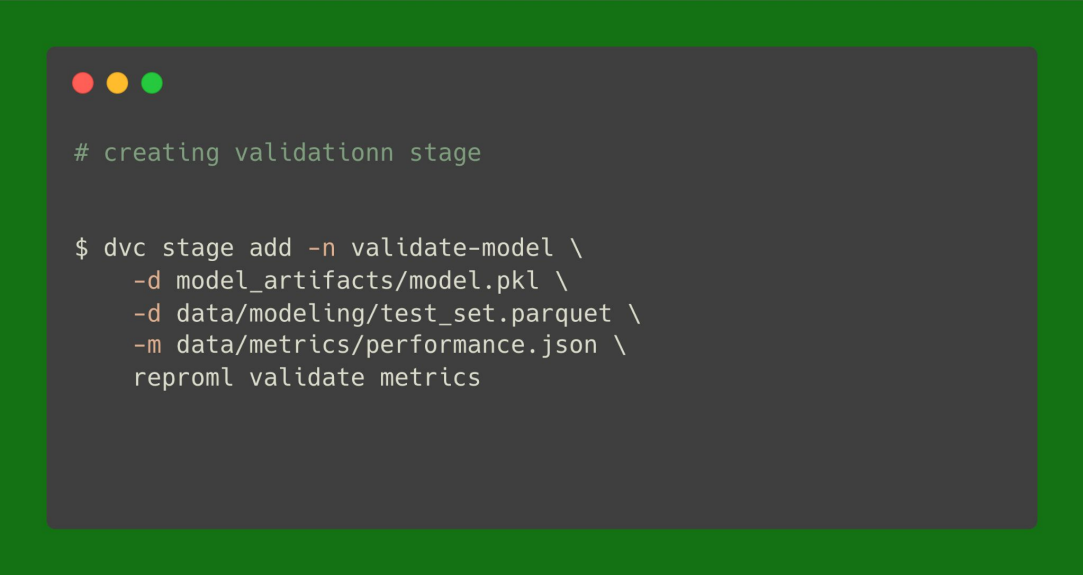
$ dvc stage add -n prepro-lyrics \
  -d reproml/preprocess/_preprocess.py \
  -d data/raw/lyrics/ \
  -o data/processed/lyrics \
  reproml prepro process
```

Creating the **modeling** pipeline using DVC

```
# creating the modeling pipeline
$ dvc stage add -n split-lyrics \
  -d reproml/preprocess/_preprocess.py \
  -d data/raw/lyrics/ \
  -o data/modeling/train_set.parquet \
  -o data/modeling/test_set.parquet \
  reproml prepro split

$ dvc stage add -n train-model \
  -d data/modeling/train_set.parquet \
  -d reproml/ml/_ml.py \
  -o model_artifacts/model.pkl \
  reproml ml train
```

Creating the **validation** stage using DVC



```
# creating validation stage

$ dvc stage add -n validate-model \
  -d model_artifacts/model.pkl \
  -d data/modeling/test_set.parquet \
  -m data/metrics/performance.json \
  reproml validate metrics
```


Now we can view our DAG

```
# viewing the DAG
$ dvc dag
+-----+
| get-lyrics |
+-----+
  *
  *
  *
+-----+
| prepro-lyrics |
+-----+
  *
  *
  *
+-----+
| split-lyrics |
+-----+
  *
  *
  *
+-----+
| train-model |
+-----+
  *
  *
  *
+-----+
| validate-model |
+-----+
```

Now DVC knows how to track our pipeline

- DVC will track changes
- DVC will add all tracked files to a remote storage
- DVC knows how to handle branches

Let's run the pipeline

```
$ dvc repro --force
Running stage 'get-lyrics':
> reproml etl get-lyrics

Running stage 'prepro-lyrics':
> reproml prepro process
Updating lock file 'dvc.lock'

Running stage 'split-lyrics':
> reproml prepro split
Updating lock file 'dvc.lock'

Running stage 'train-model':
> reproml ml train
Updating lock file 'dvc.lock'

Running stage 'validate-model':
> reproml validate metrics
Updating lock file 'dvc.lock'

To track the changes with git, run:

    git add dvc.lock

To enable auto staging, run:

    dvc config core.autostage true
Use 'dvc push' to send your updates to remote storage.
```

Here's our model training code

```
import joblib
import pandas as pd
from sklearn.feature_extraction.text import (
    CountVectorizer,
    TfidfVectorizer
)
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
from reproml.config import ml_conf as mc

def train(
    srcpath: str = (
        mc.root_path /
        mc.modeling_data_path /
        mc.train_set.filename
    ),
    dstpath: str = mc.root_path / mc.model_artifact_path,
    **kwargs
):
    train_df = pd.read_parquet(srcpath)

    # define our model
    text_clf = Pipeline(
        [
            ("vect", CountVectorizer()),
            ("clf", MultinomialNB(alpha=0.1))
        ]
    )

    # train our model on training data
    text_clf.fit(train_df["lyric"], train_df["ranker_genre"])

    joblib.dump(text_clf, dstpath)

    return text_clf
```

We're going to change our training code

```
##### Old Code #####  
text_clf = Pipeline(  
    [  
        ("vect", CountVectorizer()),  
        ("clf", MultinomialNB(alpha=0.1))  
    ]  
)  
#####  
|  
V  
##### New Code #####  
text_clf = Pipeline(  
    [  
        ("vect", TfidfVectorizer()),  
        ("clf", MultinomialNB(alpha=0.1))  
    ]  
)  
#####
```

And rerun dvc repro

```
$ dvc repro
Stage 'get-lyrics' didn't change, skipping # notice these do not run again
Stage 'prepro-lyrics' didn't change, skipping # notice these do not run again
Stage 'split-lyrics' didn't change, skipping # notice these do not run again
Running stage 'train-model':
> reproml ml train
Updating lock file 'dvc.lock'

Running stage 'validate-model':
> reproml validate metrics
Updating lock file 'dvc.lock'

To track the changes with git, run:

    git add dvc.lock

To enable auto staging, run:

    dvc config core.autostage true
Use `dvc push` to send your updates to remote storage.
```

Now let's compare metrics

```
$ dvc metrics diff -- count-model tfidf-model
```

| Path | Metric | count-model | tfidf-model | Change |
|-------------------------------|-----------|-------------|-------------|----------|
| data/metrics/performance.json | f1 | 0.6016 | 0.59422 | -0.00737 |
| data/metrics/performance.json | fns | 9753 | 9294 | -459 |
| data/metrics/performance.json | fps | 9753 | 9294 | -459 |
| data/metrics/performance.json | precision | 0.6177 | 0.64136 | 0.02366 |
| data/metrics/performance.json | recall | 0.60166 | 0.62041 | 0.01875 |
| data/metrics/performance.json | tps | 14731 | 15190 | 459 |

To close...

- 5 reasons to care about reproducibility in ML
 - Collaboration, Debugging, Validation, Auditing, Legacy
- 4 components to account for
 - Environment, Code, Data, Metrics

Questions?



Code and slides on Github



Let's connect!

**Timothy
Dobbins**



DATA SCIENTIST

Helping visionaries bring
data products to market



GRIDSEARCH
CONSULTING