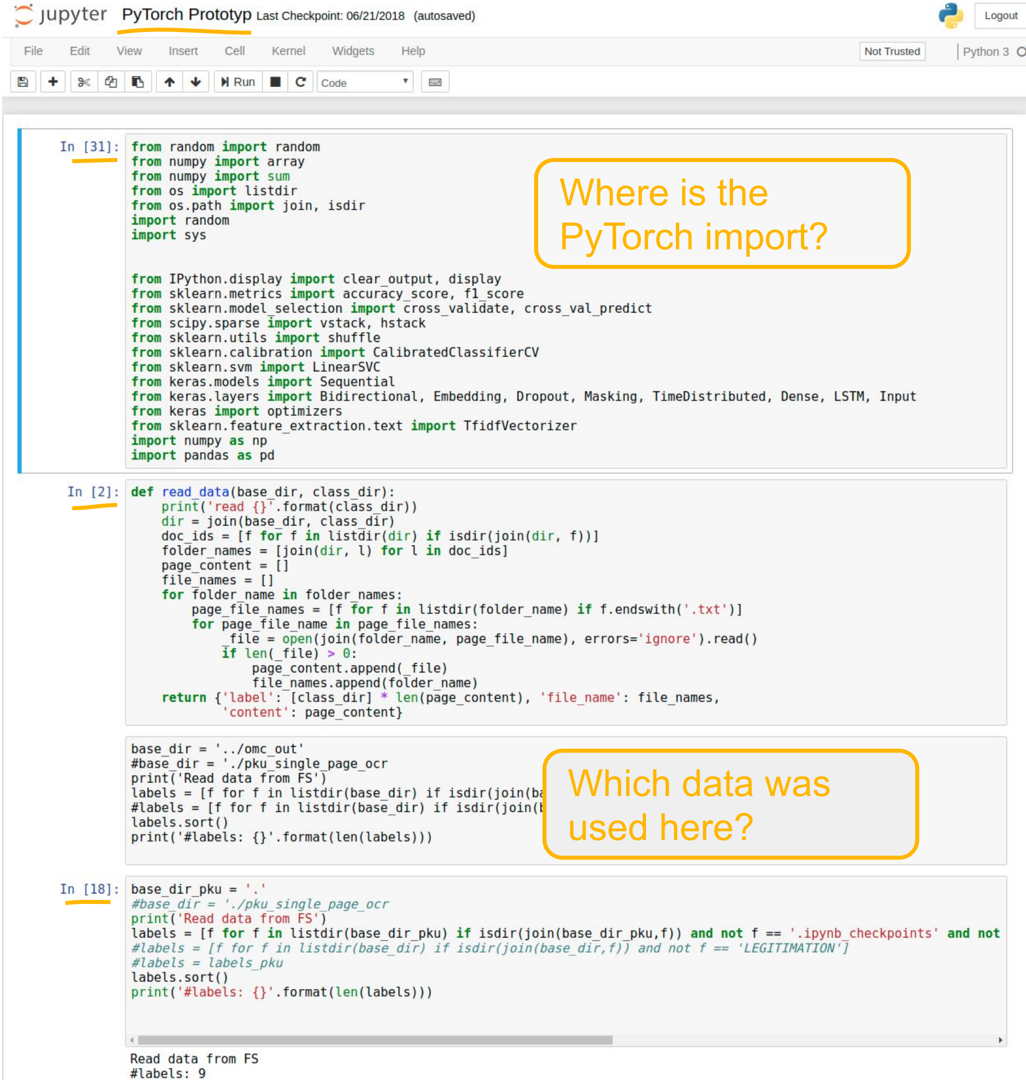
Improving your notebook workflow

By Corrie Bartelheimer

Messy Notebooks

Messy?

- Easy to end up with a messy notebook pile
- Rapid prototyping leads to less documentation
- Reproducibility of results


jupyter
PyTorch Prototyp Last Checkpoint: 06/21/2018 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
Where is the PyTorch import?
Which data was used here?

```
In [31]: from random import random
         from numpy import array
         from numpy import sum
         from os import listdir
         from os.path import join, isdir
         import random
         import sys


         from IPython.display import clear_output, display
         from sklearn.metrics import accuracy_score, f1_score
         from sklearn.model_selection import cross_validate, cross_val_predict
         from scipy.sparse import vstack, hstack
         from sklearn.utils import shuffle
         from sklearn.calibration import CalibratedClassifierCV
         from sklearn.svm import LinearSVC
         from keras.models import Sequential
         from keras.layers import Bidirectional, Embedding, Dropout, Masking, TimeDistributed, Dense, LSTM, Input
         from keras import optimizers
         from sklearn.feature_extraction.text import TfidfVectorizer
         import numpy as np
         import pandas as pd
```

```
In [2]:  def read_data(base_dir, class_dir):
             print('read {}'.format(class_dir))
             dir = join(base_dir, class_dir)
             doc_ids = [f for f in listdir(dir) if isdir(join(dir, f))]
             folder_names = [join(dir, l) for l in doc_ids]
             page_content = []
             file_names = []
             for folder_name in folder_names:
                 page_file_names = [f for f in listdir(folder_name) if f.endswith('.txt')]
                 for page_file_name in page_file_names:
                     _file = open(join(folder_name, page_file_name), errors='ignore').read()
                     if len(_file) > 0:
                         page_content.append(_file)
                         file_names.append(folder_name)
             return {'label': [class_dir] * len(page_content), 'file_name': file_names,
                     'content': page_content}
```

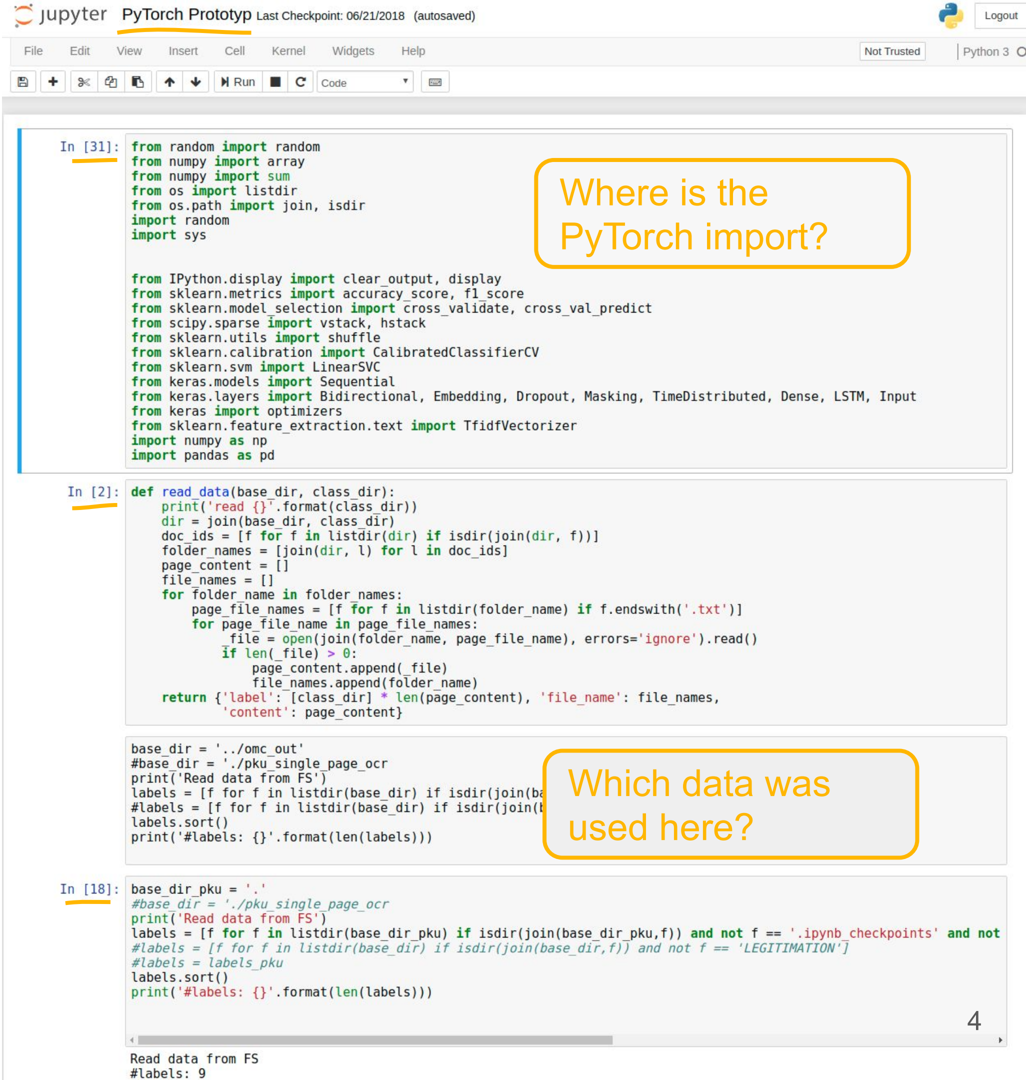
```
base_dir = '../omc_out'
#base_dir = './pku_single_page_ocr
print('Read data from FS')
labels = [f for f in listdir(base_dir) if isdir(join(ba
#labels = [f for f in listdir(base_dir) if isdir(join(b
labels.sort()
print('#labels: {}'.format(len(labels)))
```

```
In [18]: base_dir_pku = '.'
         #base_dir = './pku_single_page_ocr
         print('Read data from FS')
         labels = [f for f in listdir(base_dir_pku) if isdir(join(base_dir_pku,f)) and not f == '.ipynb_checkpoints' and not
         #labels = [f for f in listdir(base_dir) if isdir(join(base_dir,f)) and not f == 'LEGITIMATION']
         #labels = labels_pku
         labels.sort()
         print('#labels: {}'.format(len(labels)))

Read data from FS
#labels: 9
```

Collaboration?

- Will future-you still understand your notebooks?
- Or your colleague?
- Will they be able to run the notebooks?


jupyter
PyTorch Prototyp Last Checkpoint: 06/21/2018 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
Where is the PyTorch import?
Which data was used here?

```
In [31]: from random import random
         from numpy import array
         from numpy import sum
         from os import listdir
         from os.path import join, isdir
         import random
         import sys


         from IPython.display import clear_output, display
         from sklearn.metrics import accuracy_score, f1_score
         from sklearn.model_selection import cross_validate, cross_val_predict
         from scipy.sparse import vstack, hstack
         from sklearn.utils import shuffle
         from sklearn.calibration import CalibratedClassifierCV
         from sklearn.svm import LinearSVC
         from keras.models import Sequential
         from keras.layers import Bidirectional, Embedding, Dropout, Masking, TimeDistributed, Dense, LSTM, Input
         from keras import optimizers
         from sklearn.feature_extraction.text import TfidfVectorizer
         import numpy as np
         import pandas as pd
```

```
In [2]: def read_data(base_dir, class_dir):
            print('read {}'.format(class_dir))
            dir = join(base_dir, class_dir)
            doc_ids = [f for f in listdir(dir) if isdir(join(dir, f))]
            folder_names = [join(dir, l) for l in doc_ids]
            page_content = []
            file_names = []
            for folder_name in folder_names:
                page_file_names = [f for f in listdir(folder_name) if f.endswith('.txt')]
                for page_file_name in page_file_names:
                    _file = open(join(folder_name, page_file_name), errors='ignore').read()
                    if len(_file) > 0:
                        page_content.append(_file)
                        file_names.append(folder_name)
            return {'label': [class_dir] * len(page_content), 'file_name': file_names,
                    'content': page_content}
```

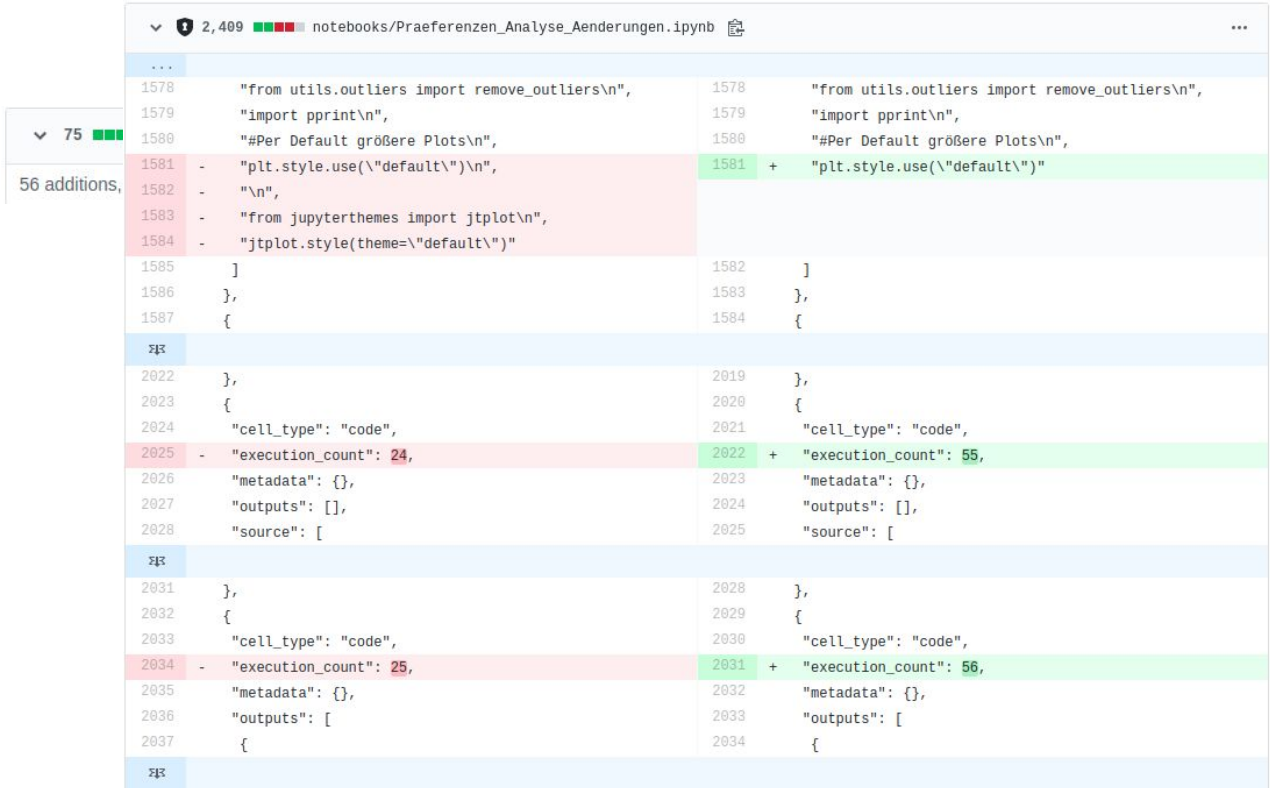
```
base_dir = '../omc_out'
#base_dir = './pku_single_page_ocr
print('Read data from FS')
labels = [f for f in listdir(base_dir) if isdir(join(ba
#labels = [f for f in listdir(base_dir) if isdir(join(b
labels.sort()
print('#labels: {}'.format(len(labels)))
```

```
In [18]: base_dir_pku = '.'
         #base_dir = './pku_single_page_ocr
         print('Read data from FS')
         labels = [f for f in listdir(base_dir_pku) if isdir(join(base_dir_pku,f)) and not f == '.ipynb_checkpoints' and not
         #labels = [f for f in listdir(base_dir) if isdir(join(base_dir,f)) and not f == 'LEGITIMATION']
         #labels = labels_pku
         labels.sort()
         print('#labels: {}'.format(len(labels)))

Read data from FS
#labels: 9
```

Version Control?

- Diff too large
- Or too cryptic



```
75
56 additions,
```

```
2,409  notebooks/Praeferenzen_Analyse_Aenderungen.ipynb

...
     "from utils.outliers import remove_outliers\n",        "from utils.outliers import remove_outliers\n",
     "import pprint\n",                                     "import pprint\n",
     "#Per Default größere Plots\n",                        "#Per Default größere Plots\n",
-    "plt.style.use(\"default\")\n",                    +   "plt.style.use(\"default\")"
-    "\n",
-    "from jupyterthemes import jtplot\n",
-    "jtplot.style(theme=\"default\")"
    ]                                                      ]
   },                                                     },
   {                                                      {

   },                                                     },
   {                                                      {
    "cell_type": "code",                                   "cell_type": "code",
-   "execution_count": 24,                             +   "execution_count": 55,
    "metadata": {},                                        "metadata": {},
    "outputs": [],                                         "outputs": [],
    "source": [                                            "source": [

   },                                                     },
   {                                                      {
    "cell_type": "code",                                   "cell_type": "code",
-   "execution_count": 25,                             +   "execution_count": 56,
    "metadata": {},                                        "metadata": {},
    "outputs": [                                           "outputs": [
     {                                                      {
```

Tidy Notebooks

Version Control!

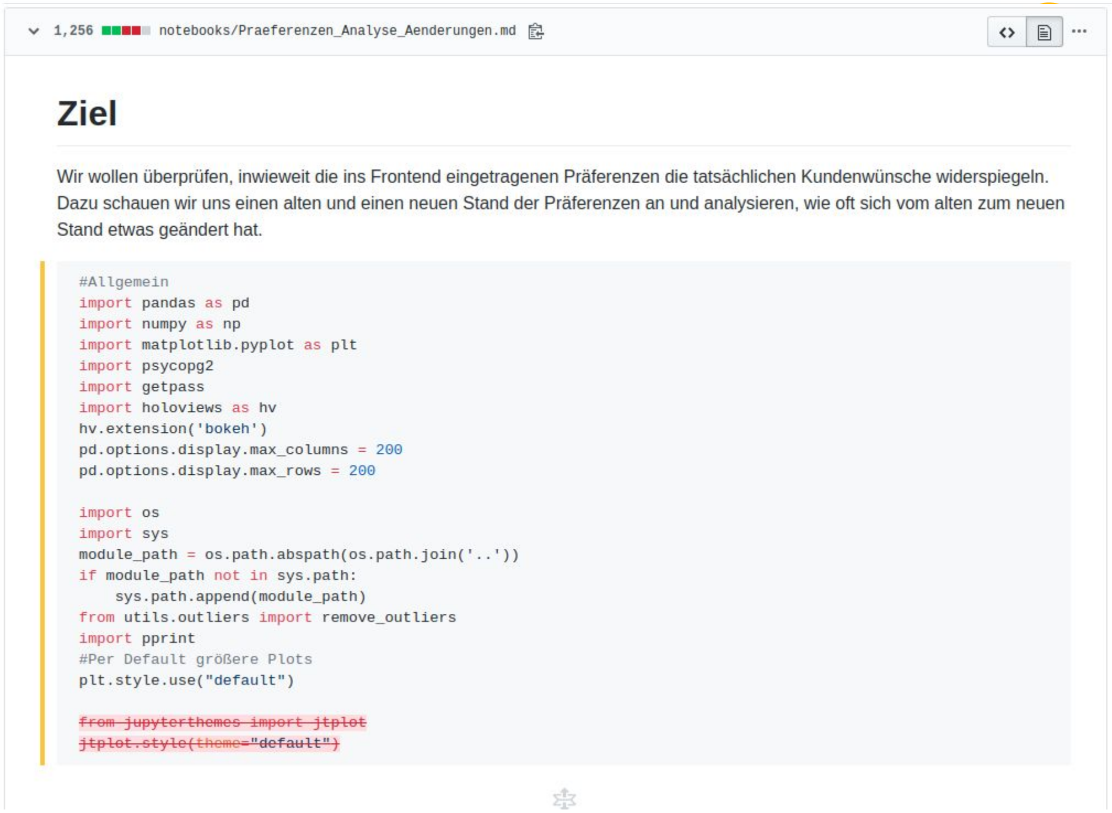
- Convert notebooks to markdown
- Using nbconvert*
- Can be automated via Jupyter SaveHook

```
jupyter nbconvert --to md notebook.ipynb
```

*Other options are for example: jupytext, reviewnb

Version Control!

- Diff of both code and results
- Rich Markdown diff is also possible



Ziel

Wir wollen überprüfen, inwieweit die ins Frontend eingetragenen Präferenzen die tatsächlichen Kundenwünsche widerspiegeln. Dazu schauen wir uns einen alten und einen neuen Stand der Präferenzen an und analysieren, wie oft sich vom alten zum neuen Stand etwas geändert hat.

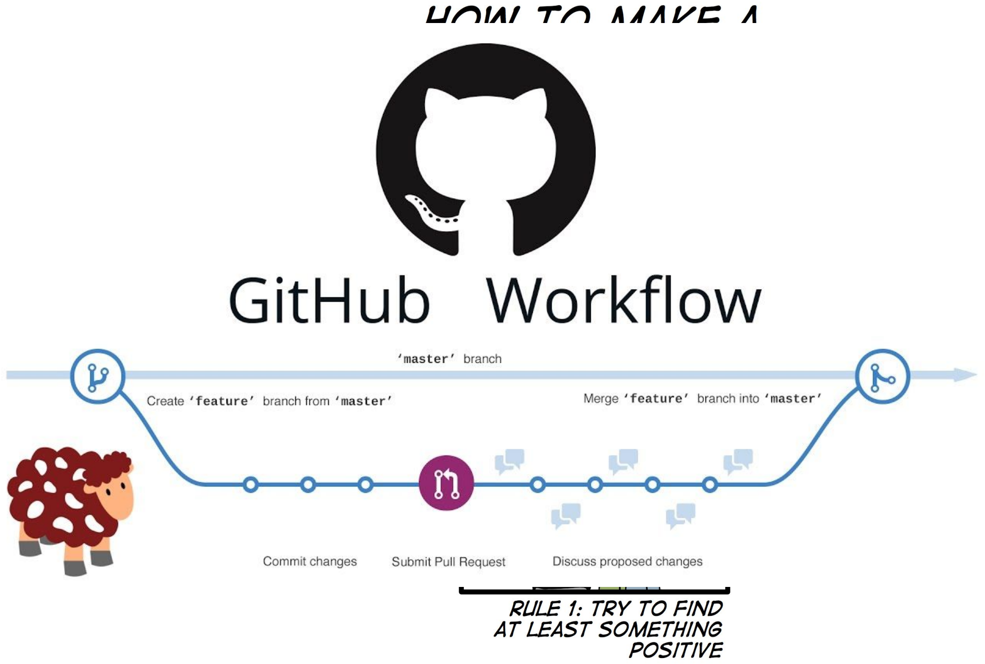
```
#Allgemein
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import psycopg2
import getpass
import holoviews as hv
hv.extension('bokeh')
pd.options.display.max_columns = 200
pd.options.display.max_rows = 200

import os
import sys
module_path = os.path.abspath(os.path.join('..'))
if module_path not in sys.path:
    sys.path.append(module_path)
from utils.outliers import remove_outliers
import pprint
#Per Default größere Plots
plt.style.use("default")

~~from jupyterthemes import jtplot~~
~~jtplot.style(theme="default")~~
```

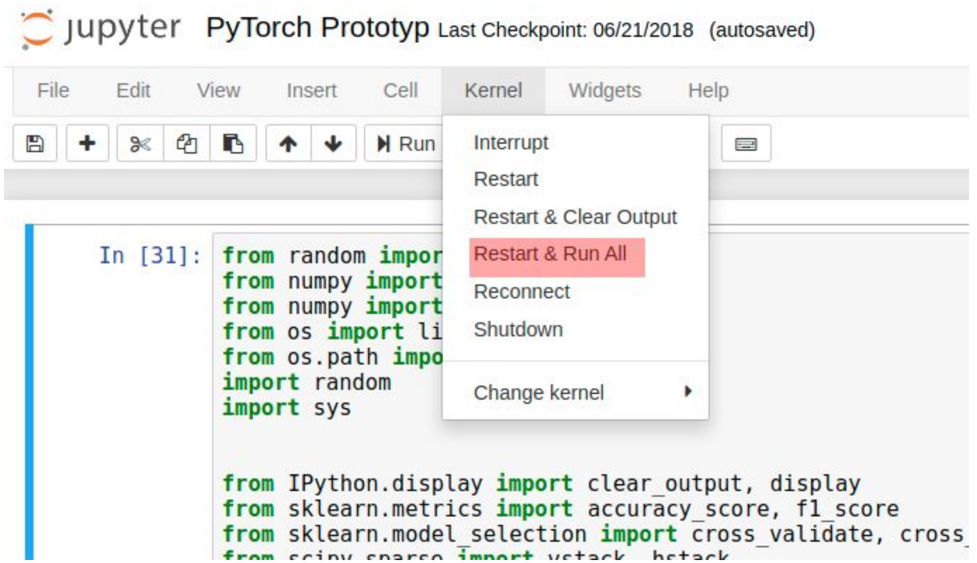
Collaboration!

- Use a code (and analysis) review process
- In our team, we use the GitHub workflow
- A review encourages clean-ups and documentation


GitHub Workflow
'master' branch
Create 'feature' branch from 'master'
Merge 'feature' branch into 'master'
Commit changes
Submit Pull Request
Discuss proposed changes
RULE 1: TRY TO FIND AT LEAST SOMETHING POSITIVE

Reproducibility!

- Document where your data comes from
- Or better: access data programmatically
- Run the whole notebook before commiting



```
In [31]: from random impor
         from numpy import
         from numpy import
         from os import li
         from os.path impo
         import random
         import sys


         from IPython.display import clear_output, display
         from sklearn.metrics import accuracy_score, f1_score
         from sklearn.model_selection import cross_validate, cross_
         from scipy.sparse import vstack, hstack
```

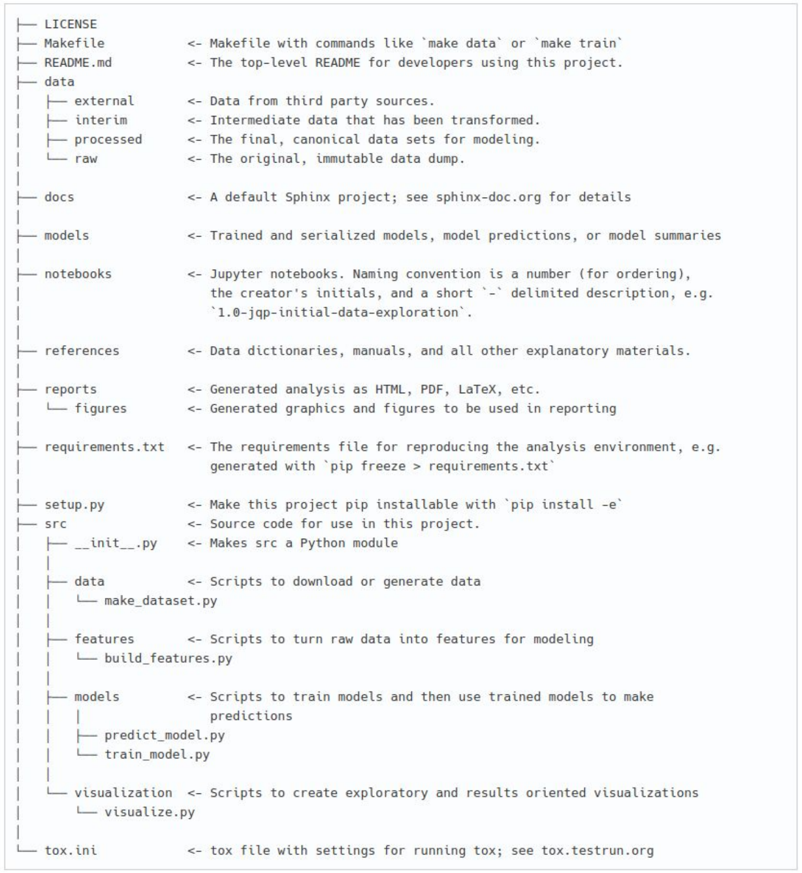
Some more tidying

- Introduce a naming convention for notebooks, e.g. `1.0-cba-initial-data-exploration`

- Include a TL;DR summary of the question you're trying to solve and your conclusion

- Use a default folder structure

Cookiecutter Data Science:



```
├── LICENSE
├── Makefile           <- Makefile with commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default Sphinx project; see sphinx-doc.org for details
│
├── models             <- Trained and serialized models, model predictions, or model summaries
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
│                         the creator's initials, and a short `-` delimited description, e.g.
│                         `1.0-jqp-initial-data-exploration`.
│
├── references         <- Data dictionaries, manuals, and all other explanatory materials.
│
├── reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reporting
│
├── requirements.txt   <- The requirements file for reproducing the analysis environment, e.g.
│                         generated with `pip freeze > requirements.txt`
│
├── setup.py           <- Make this project pip installable with `pip install -e`
├── src                <- Source code for use in this project.
│   ├── __init__.py    <- Makes src a Python module
│   │
│   ├── data           <- Scripts to download or generate data
│   │   └── make_dataset.py
│   │
│   ├── features       <- Scripts to turn raw data into features for modeling
│   │   └── build_features.py
│   │
│   ├── models         <- Scripts to train models and then use trained models to make
│   │   │                 predictions
│   │   ├── predict_model.py
│   │   └── train_model.py
│   │
│   └── visualization  <- Scripts to create exploratory and results oriented visualizations
│       └── visualize.py
│
└── tox.ini            <- tox file with settings for running tox; see tox.testrun.org
```

Summary

- Version Control with converted notebooks
- Collaborate and use Code Reviews
- Make notebooks reproducible
- Clean up your notebooks

Thanks!

Any questions?

@corrieaar corriebar