Pipelines, feature & text preprocessing

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

Peter Bull
Co-founder of DrivenData

The pipeline workflow

- Repeatable way to go from raw data to trained model
- Pipeline object takes sequential list of steps
  - Output of one step is input to next step
- Each step is a tuple with two elements
  - Name: string
  - Transform: obj implementing `.fit()` and `.transform()`
- Flexible: a step can itself be another pipeline!

Instantiate simple pipeline with one step

```
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.multiclass import OneVsRestClassifier
```

```
pl = Pipeline([
        ('clf', OneVsRestClassifier(LogisticRegression()))
    ])
```

Train and test with sample numeric data

```
sample_df.head()
```

```
  label    numeric     text  with_missing
0     a  -4.167578      bar     -4.084883
1     a  -0.562668               2.043464
2     a -21.361961             -33.315334
3     b  16.402708  foo bar     30.884604
4     a -17.934356      foo    -27.488405
```

Train and test with sample numeric data

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
                                       sample_df[['numeric']],
                                       pd.get_dummies(sample_df['label']),
                                       random_state=2)

pl.fit(X_train, y_train)
```

```
Pipeline(steps=[('clf', OneVsRestClassifier(estimator=LogisticRegression(C=1.0,
class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False),
          n_jobs=1))])
```

Train and test with sample numeric data

```
accuracy = pl.score(X_test, y_test)
print('accuracy on numeric data, no nans: ', accuracy)
```

```
accuracy on numeric data, no nans:  0.44
```

Adding more steps to the pipeline

```
X_train, X_test, y_train, y_test = train_test_split(sample_df[['numeric',
                                   'with_missing']], pd.get_dummies(
                                   sample_df['label']), random_state=2)

pl.fit(X_train, y_train)
```

```
Traceback (most recent call last):
  ...
ValueError: Input contains NaN, infinity or a value too large for
dtype('float64').
```

Preprocessing numeric features with missing data

```
from sklearn.preprocessing import Imputer
X_train, X_test, y_train, y_test = train_test_split(
                                        sample_df[['numeric', 'with_missing']],
                                        pd.get_dummies(sample_df['label']),
                                        random_state=2)
pl = Pipeline([
        ('imp', Imputer()),
        ('clf', OneVsRestClassifier(LogisticRegression()))
    ])
```

Preprocessing numeric features with missing data

```
pipeline.fit(X_train, y_train)
accuracy = pl.score(X_test, y_test)
print('accuracy on all numeric, incl nans: ', accuracy)
```

```
accuracy on all numeric, incl nans:  0.48
```

- No errors!

Let's practice!

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

Text features and feature unions

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

Peter Bull
Co-founder of DrivenData

Preprocessing text features

```
from sklearn.feature_extraction.text import CountVectorizer
X_train, X_test, y_train, y_test = train_test_split(sample_df['text'],
                                                    pd.get_dummies(
                                                    sample_df['label']),
                                                    random_state=2)
pl = Pipeline([
        ('vec', CountVectorizer()),
        ('clf', OneVsRestClassifier(LogisticRegression()))
    ])
```

Preprocessing text features

```
pl.fit(X_train, y_train)
```

```
Pipeline(steps=[('vec', CountVectorizer(analyzer='word', binary=False,
decode_error='strict', dtype=<class 'numpy.int64'>, encoding='utf-8',
input='content', lowercase=True, max_df=1.0, max_features=None, min_df=1,
ngram_range=(1, 1), preprocessor=None, stop_words=None, strip_...=None,
solver='liblinear', tol=0.0001, verbose=0, warm_start=False), n_jobs=1))])
```

```
accuracy = pl.score(X_test, y_test)
print('accuracy on sample data: ', accuracy)
```

```
accuracy on sample data:  0.64
```

Preprocessing multiple dtypes

- Want to use **all** available features in one pipeline
- Problem
  - Pipeline steps for numeric and text preprocessing can't follow each other
  - e.g., output of `CountVectorizer` can't be input to `Imputer`
- Solution
  - `FunctionTransformer()` & `FeatureUnion()`

FunctionTransformer

- Turns a Python function into an object that a scikit-learn pipeline can understand
- Need to write two functions for pipeline preprocessing
  - Take entire DataFrame, return numeric columns
  - Take entire DataFrame, return text columns
- Can then preprocess numeric and text data in separate pipelines

Putting it all together

```
X_train, X_test, y_train, y_test = train_test_split(sample_df[['numeric',
                                    'with_missing', 'text']], pd.get_dummies(
                                    sample_df['label']), random_state=2)
from sklearn.preprocessing import FunctionTransformer
from sklearn.pipeline import FeatureUnion
```

Putting it all together

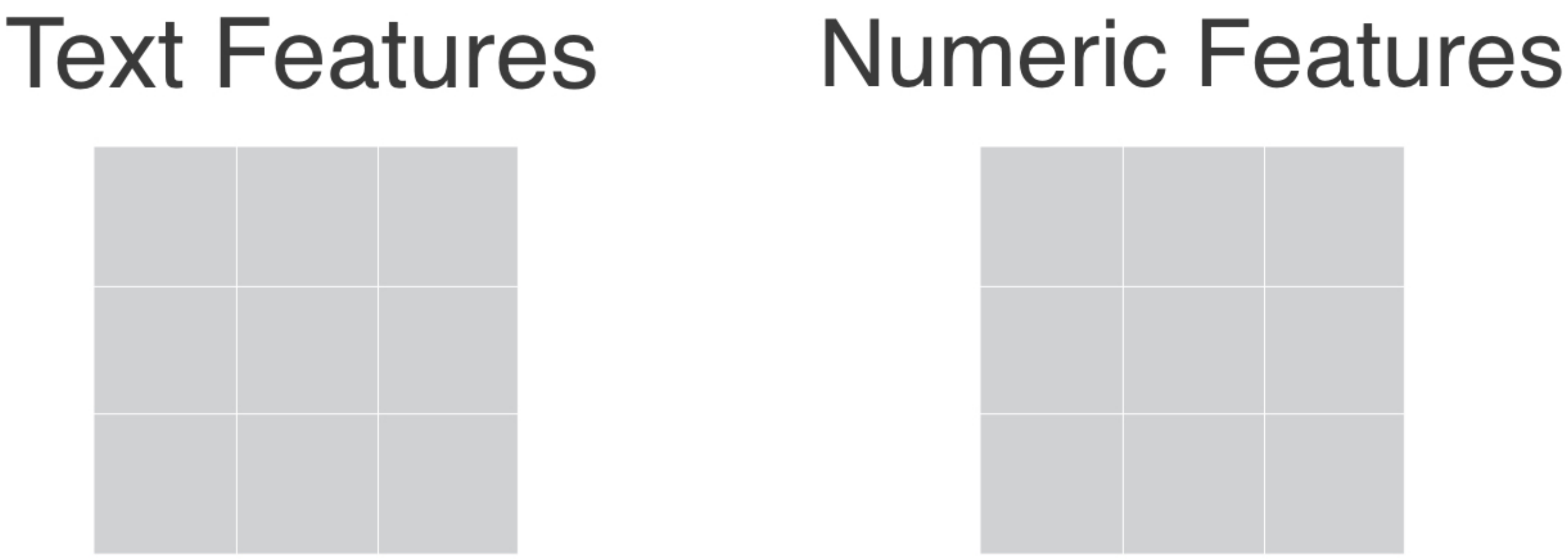
```
get_text_data = FunctionTransformer(lambda x: x['text'],
                                    validate=False)
get_numeric_data = FunctionTransformer(lambda x: x[['numeric',
                                       'with_missing']], validate=False)
```

FeatureUnion Text and Numeric Features

Text Features

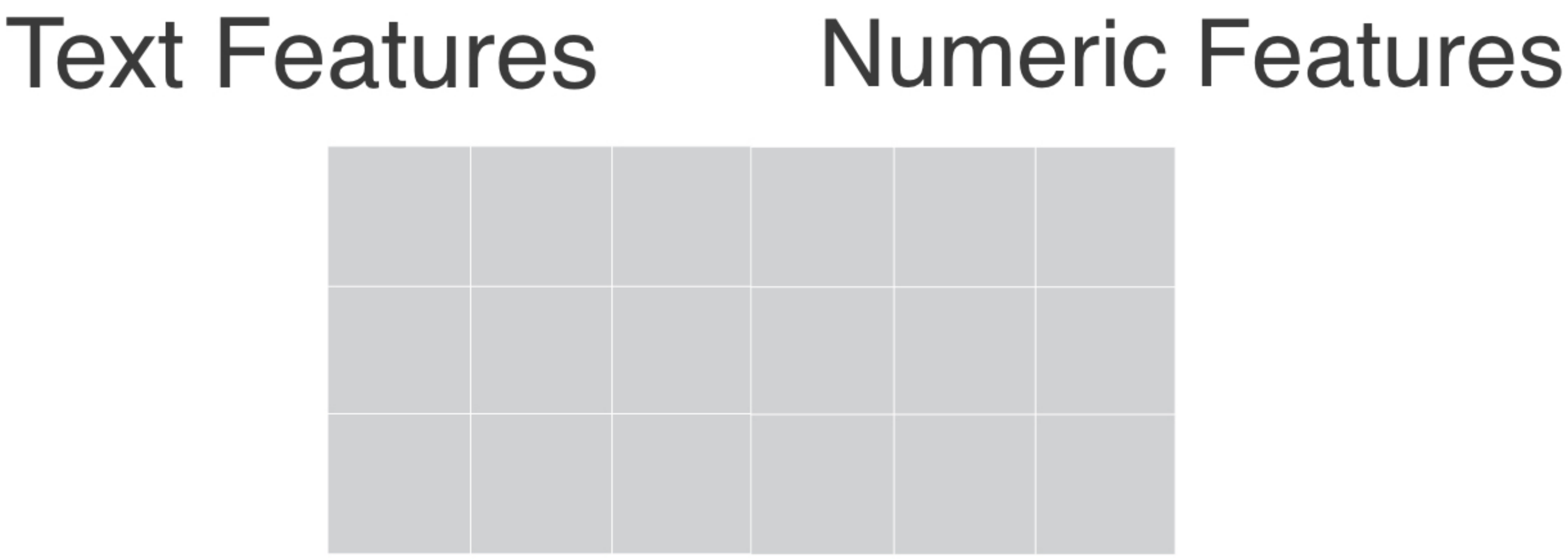

```python
from sklearn.pipeline import FeatureUnion
```

FeatureUnion Text and Numeric Features




```python
from sklearn.pipeline import FeatureUnion
```

FeatureUnion Text and Numeric Features




```python
from sklearn.pipeline import FeatureUnion
union = FeatureUnion([
              ('numeric', numeric_pipeline),
              ('text', text_pipeline)
          ])
```

Putting it all together

```
numeric_pipeline = Pipeline([
                        ('selector', get_numeric_data),
                        ('imputer', Imputer())
                    ])
text_pipeline = Pipeline([
                        ('selector', get_text_data),
                        ('vectorizer', CountVectorizer())
                     ])
pl = Pipeline([
         ('union', FeatureUnion([
             ('numeric', numeric_pipeline),
             ('text', text_pipeline)
         ])),
         ('clf', OneVsRestClassifier(LogisticRegression()))
          ])
```

Let's practice!

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

Choosing a classification model

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

Peter Bull
Co-founder of DrivenData

Main dataset: lots of text

```
LABELS = ['Function', 'Use', 'Sharing', 'Reporting', 'Student_Type',
          'Position_Type', 'Object_Type',  'Pre_K', 'Operating_Status']
NON_LABELS = [c for c in df.columns if c not in LABELS]
len(NON_LABELS) - len(NUMERIC_COLUMNS)
```

```
14
```

Using pipeline with the main dataset

```
import numpy as np
import pandas as pd
df = pd.read_csv('TrainingSetSample.csv', index_col=0)
dummy_labels = pd.get_dummies(df[LABELS])
X_train, X_test, y_train, y_test = multilabel_train_test_split(
                                       df[NON_LABELS], dummy_labels,
                                       0.2)
```

Using pipeline with the main dataset

```
get_text_data = FunctionTransformer(combine_text_columns,
                                     validate=False)
get_numeric_data = FunctionTransformer(lambda x:
                        x[NUMERIC_COLUMNS], validate=False)
pl = Pipeline([
             ('union', FeatureUnion([
                     ('numeric_features', Pipeline([
                         ('selector', get_numeric_data),
                         ('imputer', Imputer())
                     ])),
                     ('text_features', Pipeline([
                         ('selector', get_text_data),
                         ('vectorizer', CountVectorizer())
                     ]))
                  ])
             ),
             ('clf', OneVsRestClassifier(LogisticRegression()))
         ])
```

Performance using main dataset

```
pl.fit(X_train, y_train)
```

```
Pipeline(steps=[('union', FeatureUnion(n_jobs=1,
  transformer_list=[('numeric_features', Pipeline(steps=
  [('selector', FunctionTransformer(accept_sparse=False,
  func=<function <lambda> at 0x11415ec80>, pass_y=False,
  validate=False)), ('imputer', Imputer(axis=0, copy=True,
  missing_valu...=None, solver='liblinear', tol=0.0001,
  verbose=0, warm_start=False),n_jobs=1))])
```

Flexibility of model step

- Is current model the best?
- Can quickly try different models with pipelines
  - Pipeline preprocessing steps unchanged
  - Edit the model step in your pipeline
  - Random Forest, Naïve Bayes, k-NN

Easily try new models using pipeline

```
from sklearn.ensemble import RandomForestClassifier
pl = Pipeline([
             ('union', FeatureUnion(
                 transformer_list = [
                     ('numeric_features', Pipeline([
                         ('selector', get_numeric_data),
                         ('imputer', Imputer())
                     ])),
                     ('text_features', Pipeline([
                         ('selector', get_text_data),
                         ('vectorizer', CountVectorizer())
                     ]))
                  ]
             )),
             ('clf', OneVsRest(RandomForestClassifier()))
         ])
```

Let's practice!

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON