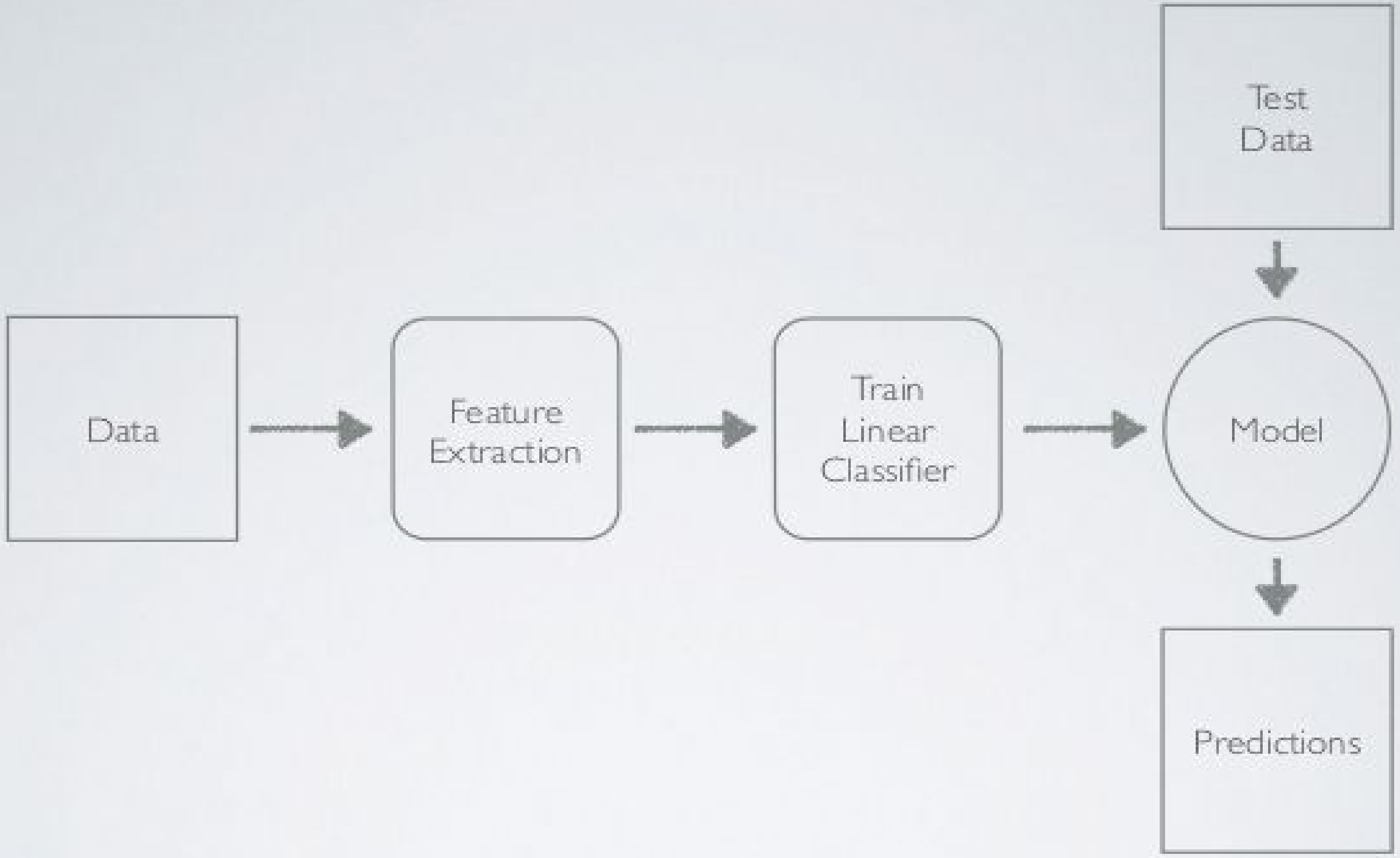
Pipelines

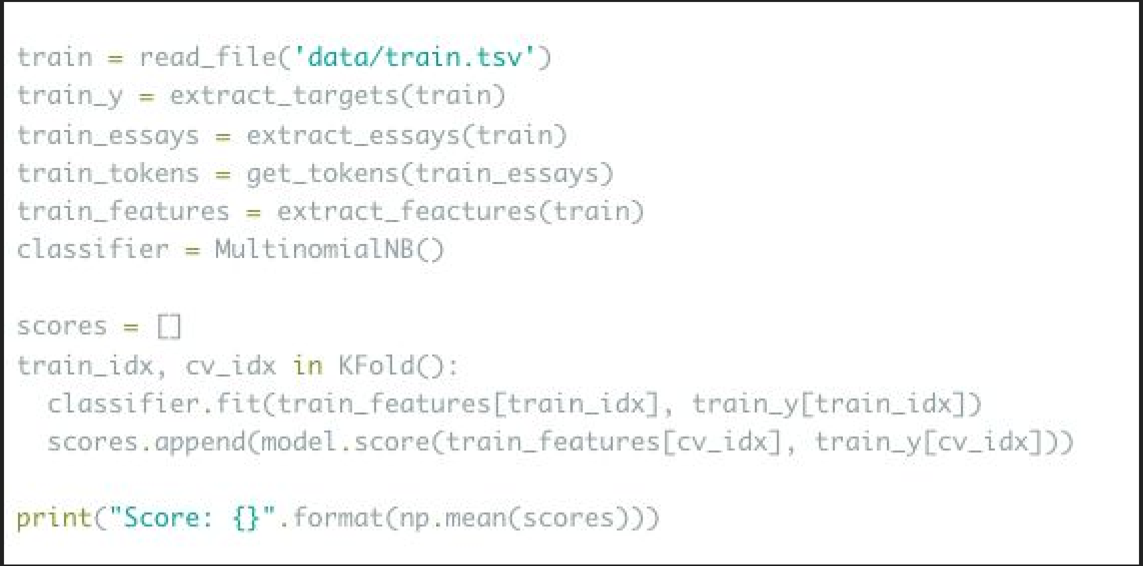
Justin Hooker
Sy Ahmed

Machine Learning Pipelines

- There are standard workflows in a machine learning project that can be automated
- Utility that allows a linear sequence of data transformations to chained together
- Great way to organize models

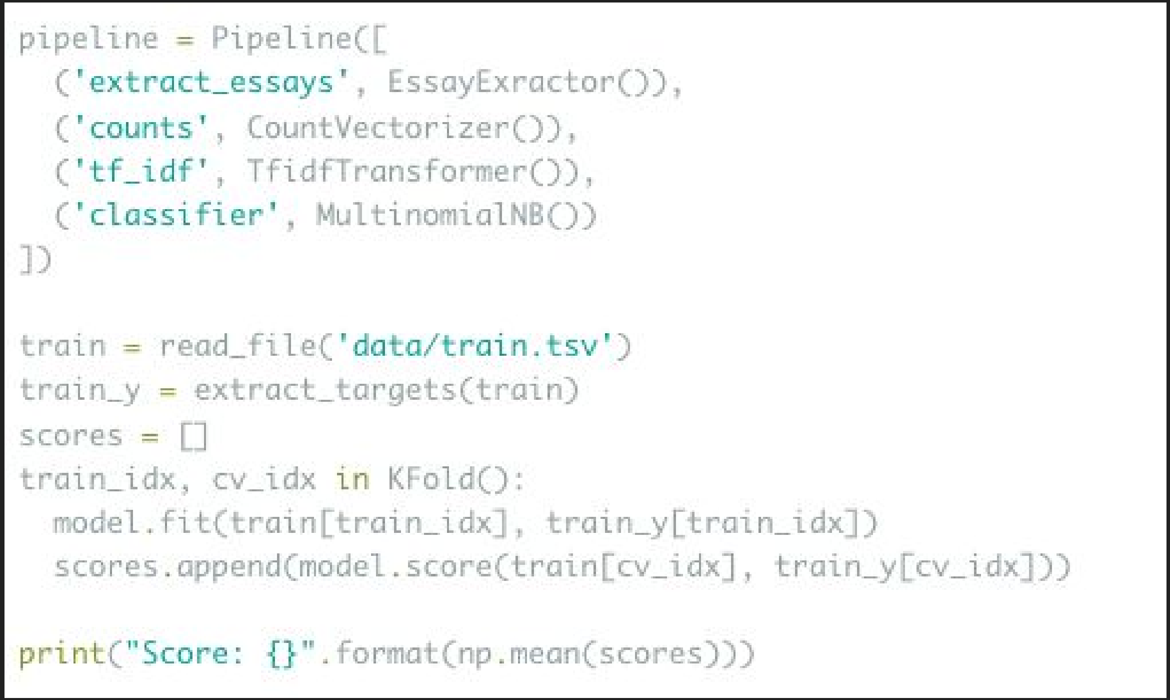

Data
Feature Extraction
Train Linear Classifier
Model
Test Data
Predictions

Implementing Multinomial NB on text data

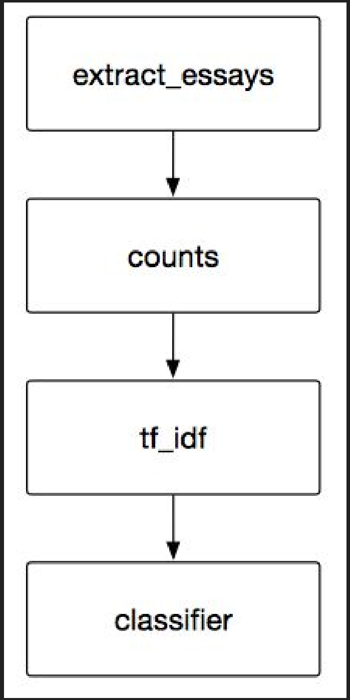


```
train = read_file('data/train.tsv')
train_y = extract_targets(train)
train_essays = extract_essays(train)
train_tokens = get_tokens(train_essays)
train_features = extract_feactures(train)
classifier = MultinomialNB()

scores = []
train_idx, cv_idx in KFold():
  classifier.fit(train_features[train_idx], train_y[train_idx])
  scores.append(model.score(train_features[cv_idx], train_y[cv_idx]))

print("Score: {}".format(np.mean(scores)))
```

Implementing Multinomial NB on text data with scikit-learn Pipeline



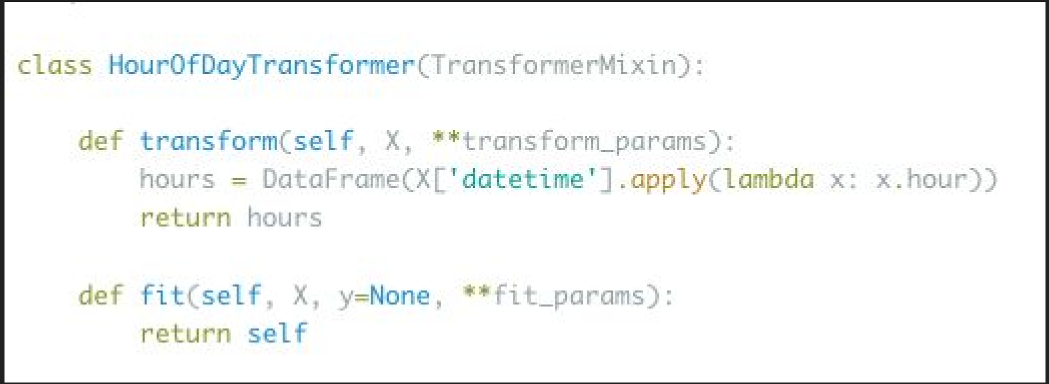
```
pipeline = Pipeline([
  ('extract_essays', EssayExractor()),
  ('counts', CountVectorizer()),
  ('tf_idf', TfidfTransformer()),
  ('classifier', MultinomialNB())
])

train = read_file('data/train.tsv')
train_y = extract_targets(train)
scores = []
train_idx, cv_idx in KFold():
  model.fit(train[train_idx], train_y[train_idx])
  scores.append(model.score(train[cv_idx], train_y[cv_idx]))

print("Score: {}".format(np.mean(scores)))
```



Custom transformers

Transformer - data in data out black box

Stateless Transformer



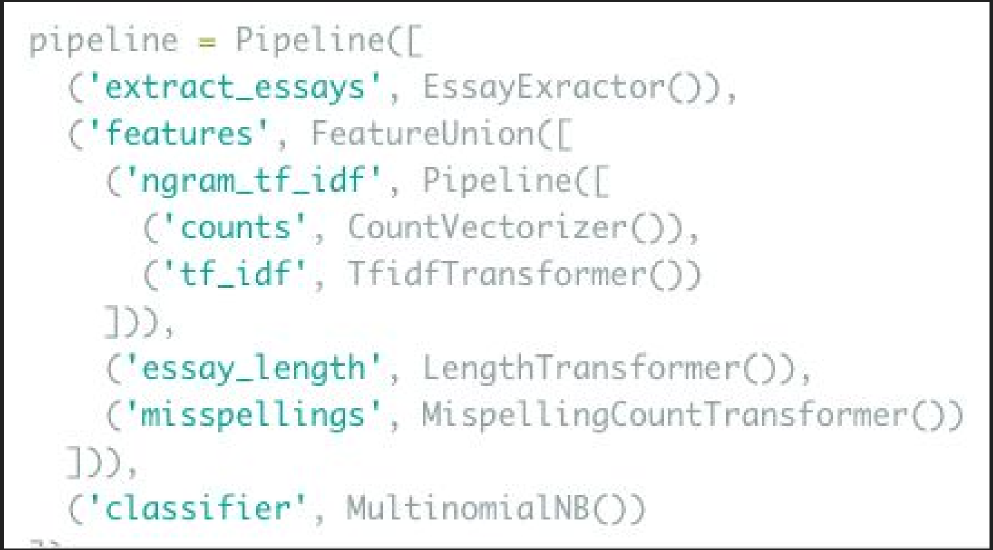
```python
class HourOfDayTransformer(TransformerMixin):

    def transform(self, X, **transform_params):
        hours = DataFrame(X['datetime'].apply(lambda x: x.hour))
        return hours

    def fit(self, X, y=None, **fit_params):
        return self
```

Data Leakage

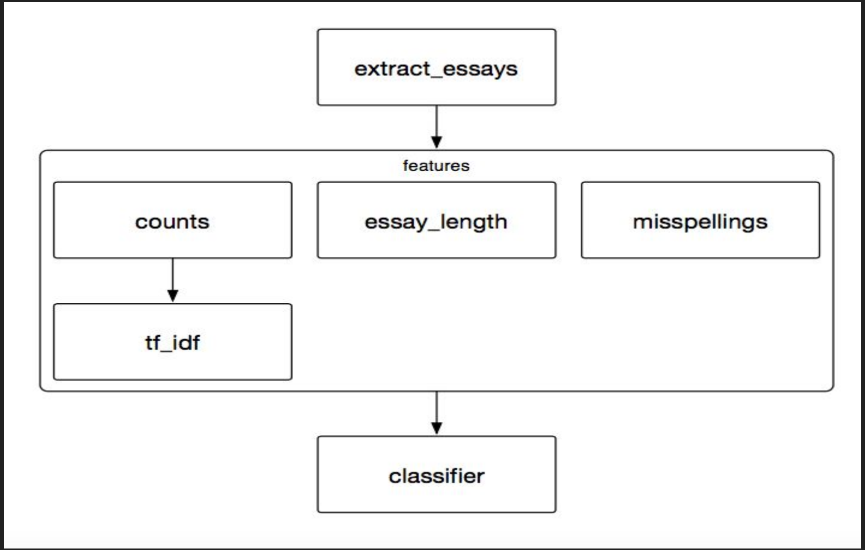
Leaking knowledge of of the whole dataset to the algorithm

Pipelines help you prevent data leakage in your test harness by ensuring that data preparation like standardization is constrained to each fold of your cross validation procedure.

Feature Unions



```
pipeline = Pipeline([
  ('extract_essays', EssayExractor()),
  ('features', FeatureUnion([
    ('ngram_tf_idf', Pipeline([
      ('counts', CountVectorizer()),
      ('tf_idf', TfidfTransformer())
    ])),
    ('essay_length', LengthTransformer()),
    ('misspellings', MispellingCountTransformer())
  ])),
  ('classifier', MultinomialNB())
```



Final Thoughts

- The last step of a pipeline is assumed to be the final estimator, so predict method is called instead of transform. For this reason any post processing is done outside of the pipeline.
- Pipelines do not support fit_partial API, so out-of-core training is not possible.
- 2 Important use cases:
  - Data preparation and modeling
  - Feature extraction and feature union
- Adapting, improving models and hyperparameter gridsearch

Demo

Resources

http://zacstewart.com/2014/08/05/pipelines-of-featureunions-of-pipelines.html

https://machinelearningmastery.com/automate-machine-learning-workflows-pipelines-python-scikit-learn/