Train In Data

Streamlining Machine Learning Pipelines with Open Source

open source

imbalanced learn

Category Encoders

MLXTEND

K

Featuretools

scikit learn

Scikit-Learn

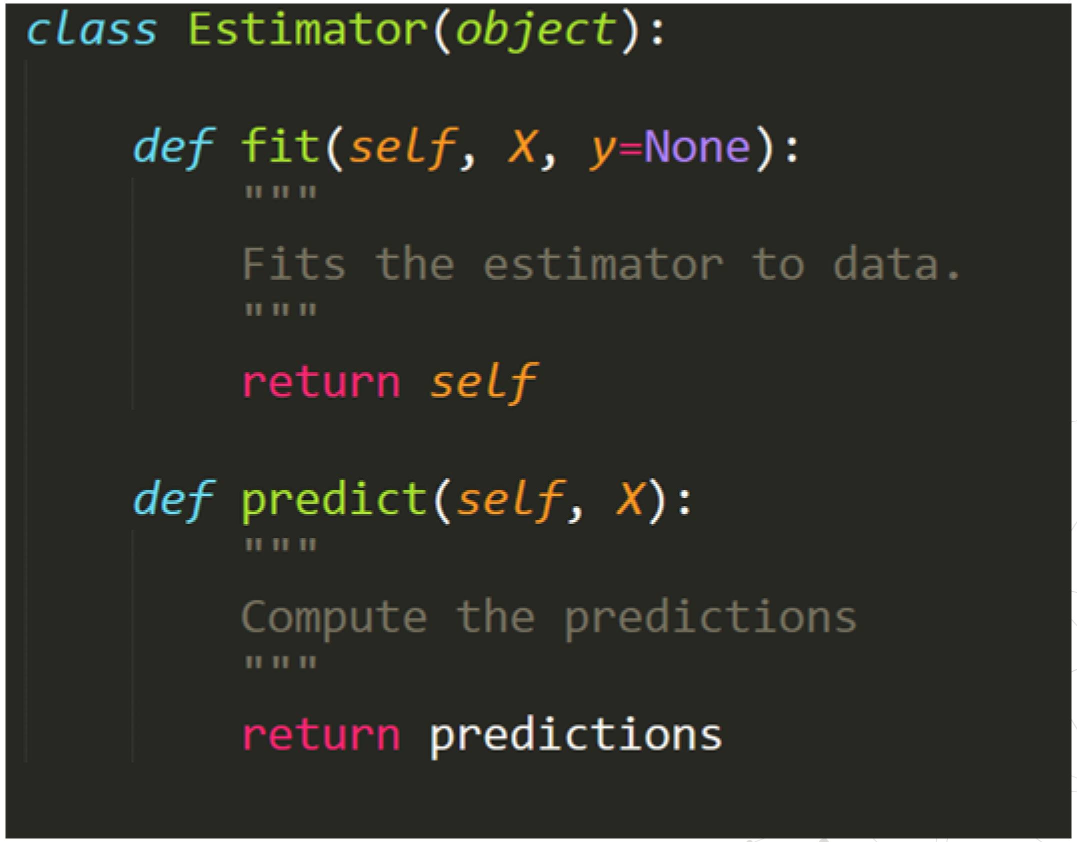
- Solid implementation of a wide range of machine learning algorithms and data transformations
- Clean, uniform, and streamlined API.
- Most algorithms follow the same functionality ➔ implementing new algos is super easy
  - Transformers
  - Estimators
  - Pipeline
- Complete online documentation, with some theory and examples
- Well established in the community ➔ new packages follow Scikit-learn functionality to be quickly adopted by end users, e.g., Keras, MLXtend, category-encoders, Feature-engine

Scikit-Learn Estimators

Estimator - A class with fit() and predict() methods.

It fits and predicts.

Any ML algorithm like Lasso, Decision trees, SVMs, are coded as estimators within Scikit-Learn.



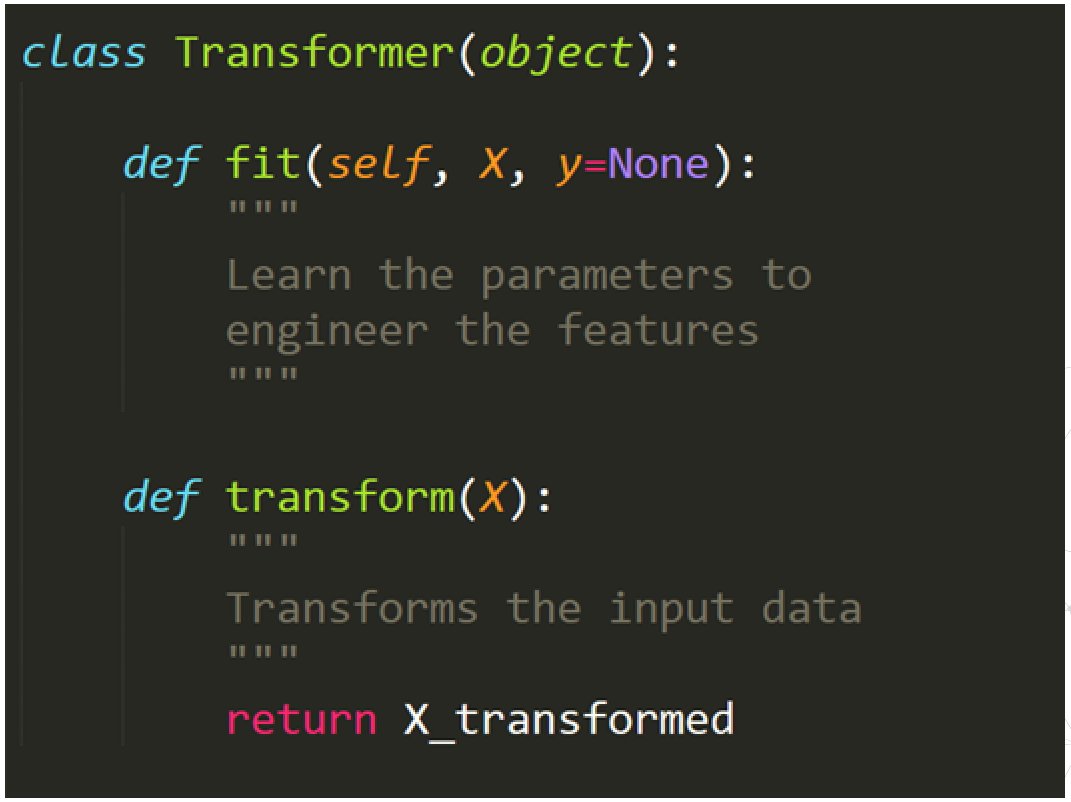
```python
class Estimator(object):

    def fit(self, X, y=None):
        """
        Fits the estimator to data.
        """
        return self

    def predict(self, X):
        """
        Compute the predictions
        """
        return predictions
```

Scikit-Learn Transformers

Transformers - class that have fit() and transform() methods.

It transforms data.

- Scalers
- Feature selectors
- Encoders
- Imputers
- Discretizers
- Transformers

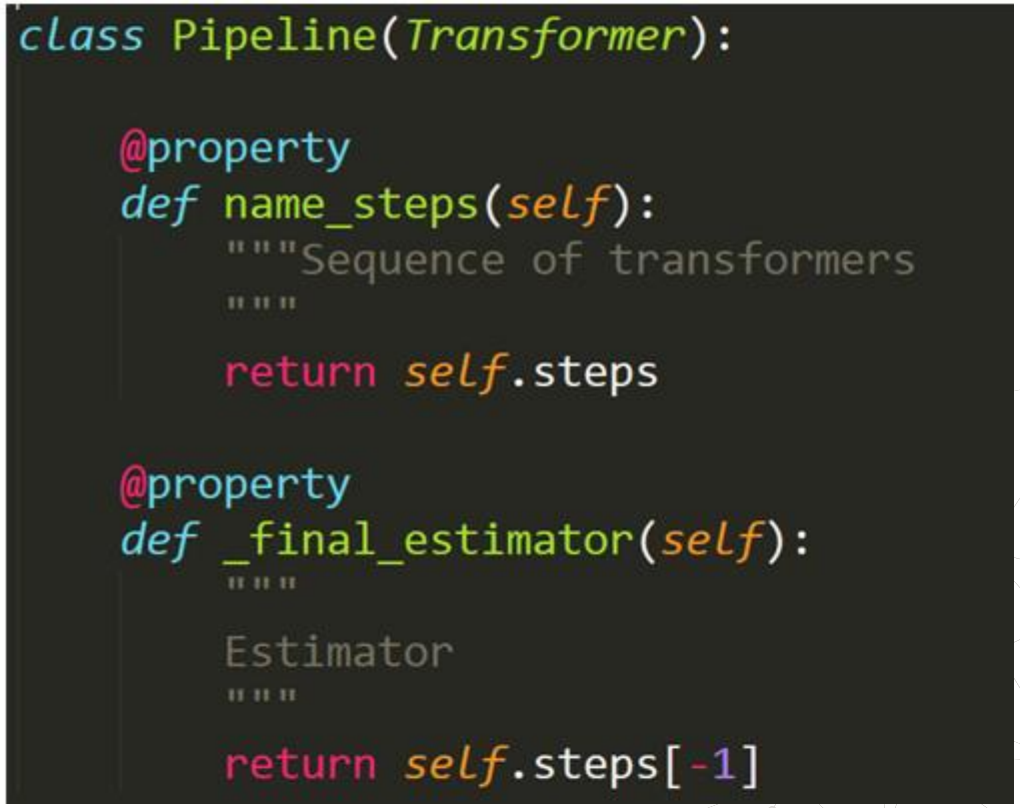


```python
class Transformer(object):

    def fit(self, X, y=None):
        """
        Learn the parameters to
        engineer the features
        """

    def transform(X):
        """
        Transforms the input data
        """
        return X_transformed
```

Scikit-Learn Pipeline

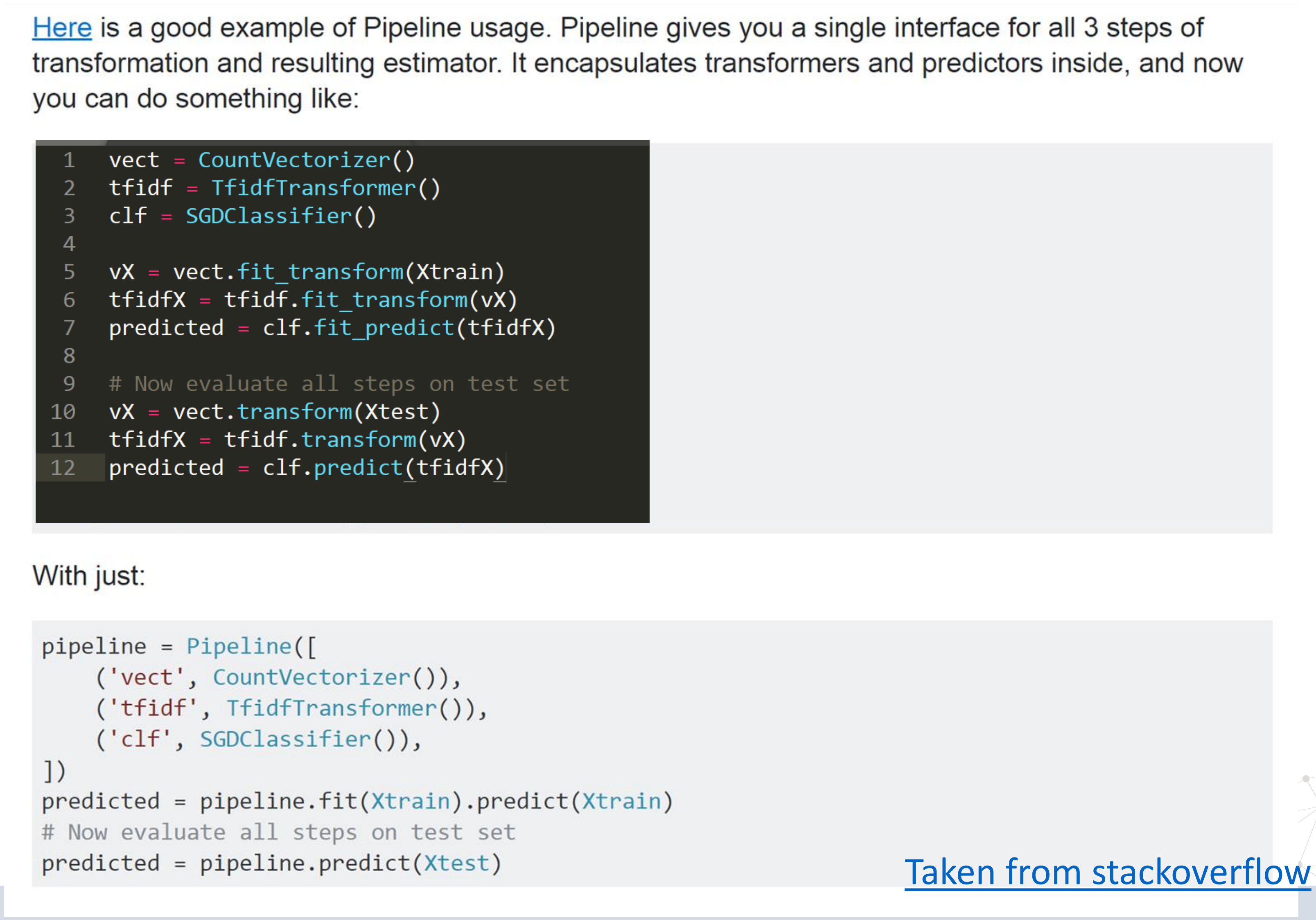
Pipeline - class that allows to run transformers and estimators in sequence.

- Most steps are Transformers
- Last step can be an Estimator



```python
class Pipeline(Transformer):

    @property
    def name_steps(self):
        """Sequence of transformers
        """
        return self.steps

    @property
    def _final_estimator(self):
        """
        Estimator
        """
        return self.steps[-1]
```

Scikit-Learn Pipeline in action



Here is a good example of Pipeline usage. Pipeline gives you a single interface for all 3 steps of transformation and resulting estimator. It encapsulates transformers and predictors inside, and now you can do something like:

```
vect = CountVectorizer()
tfidf = TfidfTransformer()
clf = SGDClassifier()

vX = vect.fit_transform(Xtrain)
tfidfX = tfidf.fit_transform(vX)
predicted = clf.fit_predict(tfidfX)

# Now evaluate all steps on test set
vX = vect.transform(Xtest)
tfidfX = tfidf.transform(vX)
predicted = clf.predict(tfidfX)
```

With just:

```
pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', SGDClassifier()),
])
predicted = pipeline.fit(Xtrain).predict(Xtrain)
# Now evaluate all steps on test set
predicted = pipeline.predict(Xtest)
```

Taken from stackoverflow

Open-source for Feature Engineering

scikit learn

Category Encoders

Feature-engine

Featuretools

imbalanced learn

Open-source for Feature Selection

Feature-engine

Open-source for Model Training

- Py-earth
- xgboost
- Lasagne
- Many others

Train In Data

Thank you

www.trainindata.com