Data pipelines. Sklearn API

Lecture #6, Features

Sklearn modules

- preprocessing
- impute
- pipeline (Pipeline, FeatureUnion)
- compose (ColumnTransformer)

Basic Pipeline API

Each preprocessing object (step) must implement fit and transform methods.

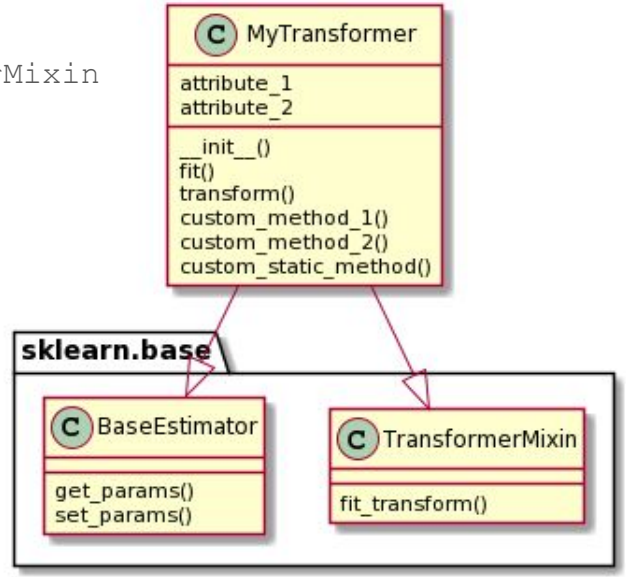
```
Pipeline(steps=[('name_of_preprocessor_1', preprocessor_1),
                ('name_of_preprocessor_2', preprocessor_2)
                ('name_of_ml_model', ml_model())])
```

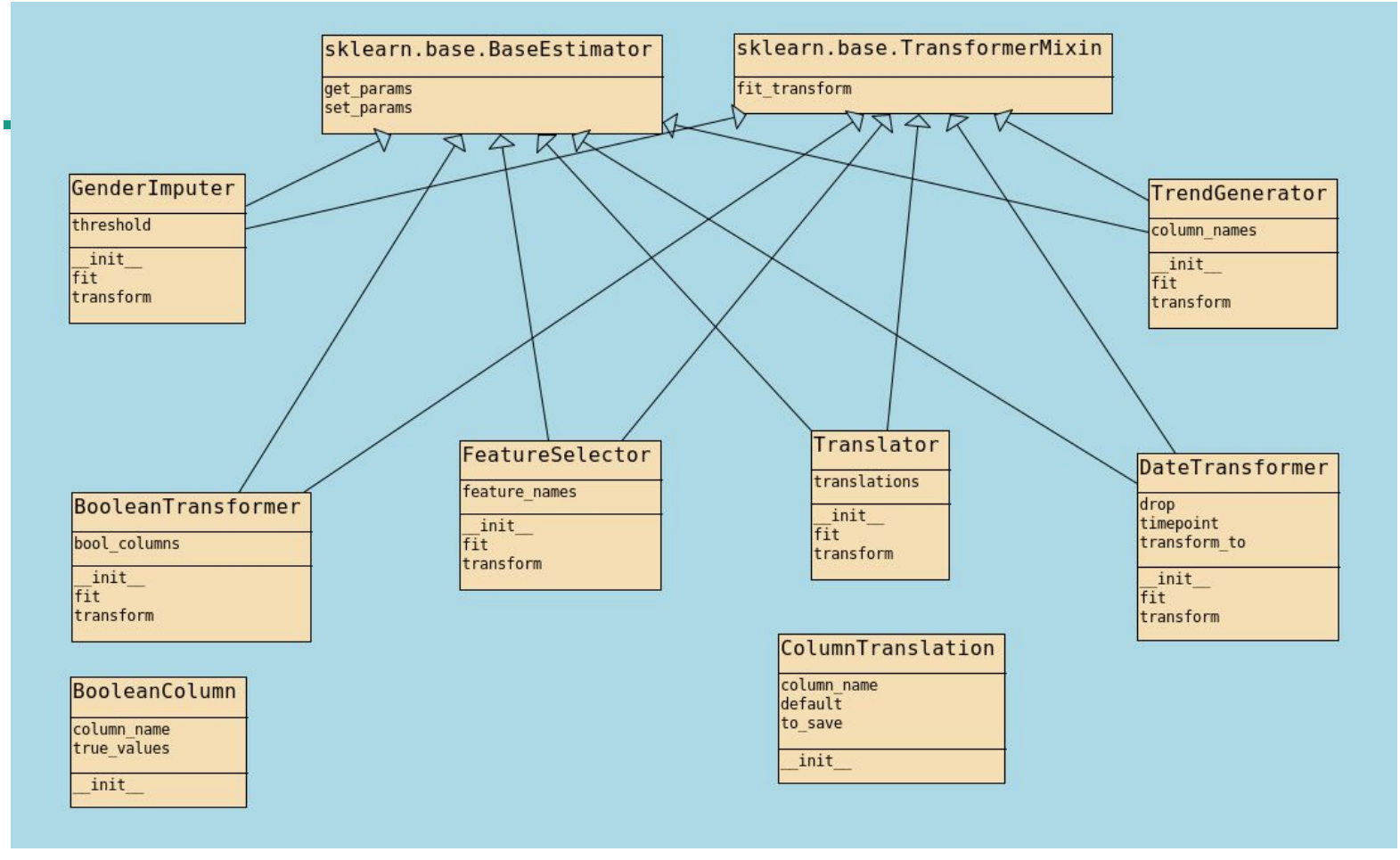
Preprocessor inheritance

```
from sklearn.base import BaseEstimator, TransformerMixin
```

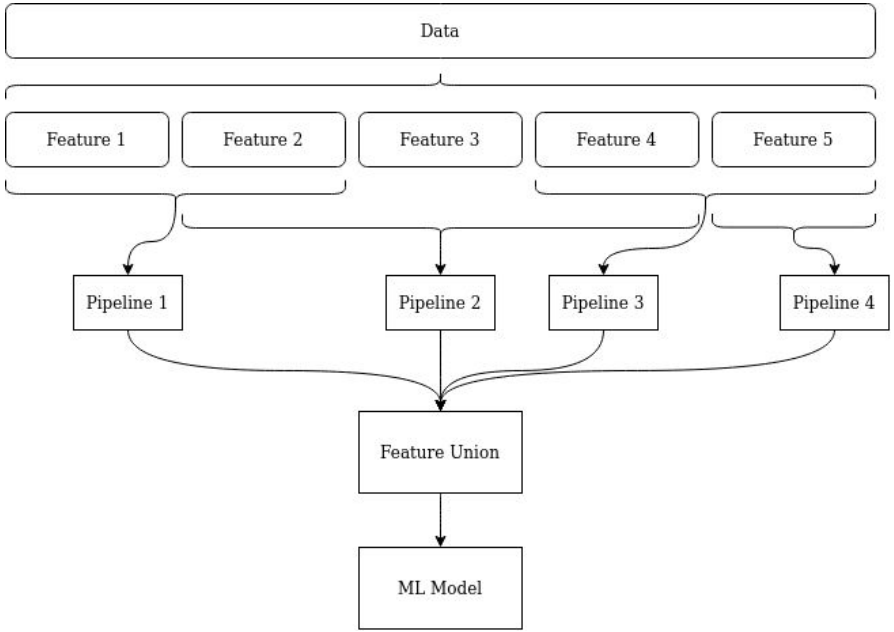
We can create any custom attributes and methods inside our transformers, imputers, etc.

They should all inherit from two classes (BaseEstimator, TransformerMixin) and implement fit and transform methods.




sklearn.base.BaseEstimator
get_params
set_params
sklearn.base.TransformerMixin
fit_transform
GenderImputer
threshold
__init__
fit
transform
TrendGenerator
column_names
__init__
fit
transform
FeatureSelector
feature_names
__init__
fit
transform
Translator
translations
__init__
fit
transform
DateTransformer
drop
timepoint
transform_to
__init__
fit
transform
BooleanTransformer
bool_columns
__init__
fit
transform
BooleanColumn
column_name
true_values
__init__
ColumnTranslation
column_name
default
to_save
__init__

General workflow



Step 1. Split data into separate pipelines

ACT_DATE	STATUS	TP_CURRENT	TP_CHANGES_NUM	START_PACK	OFFER_GROUP	BIRTHDAY	GENDER	DEVICE_TYPE_BUS	USAGE_AREA	REFILL_OCT_16	REFILL_NOV_16	OUTGOING_OCT_16	OUTGOING_NOV_16	GPRS_OCT_16	GPRS_NOV_16
2016-04-19	Q	TP_3GM	0	Commercial	Promo	1983-08-26	M	Modem/Router	Regional Cities	0	0	0	0	0	0
2014-11-22	D	TP_XS	0	Commercial	Standard	1981-03-19	F	Other	Local Towns	0	0	0	0	0	0
2009-04-02	D	TP_FREE	0				F	Smartphone	Countryside	150000	0	59.4833333333333	46.85	8	3
2011-12-15	D	TP_ANDR	0			1984-11-28	M	Undefined				0	0	0	0
2013-06-17	D	TP_DL012	0	Commercial	Standard	1992-08-31	F	Smartphone	Minsk	60000	90000	260.2	266.066666666667	4205	4727
2009-07-12	D	TP_CRTBL	0				M	Undefined	Local Towns	10000	10000	1.58333333333333	2.63333333333333	0	0
2015-10-15	F	TP_INTER	0	Promo	Standard	1982-03-11	M	Undefined		0	0	0	0	0	0
2015-12-21	D	TP_ANDR	0	Commercial	Standard	1974-10-28	M	Smartphone	Local Towns	200000	210000	477.45	431.083333333333	940	2403
2012-11-06	D	TP_XS	1			1994-06-16	F	Smartphone	Minsk	490000	280000	200.233333333333	130.1	20753	25719

Step 2. Create pipeline for each data subset

```
translate_ohe = OneHotEncoder(sparse=False)
translate_pipeline = Pipeline(
    steps=[
        ('translate_selector', FeatureSelector(['STATUS', 'ASSET_TYPE_LAST',
                                                'DEVICE_TYPE_BUS', 'USAGE_AREA'])),
        ('translate_transformer', Translator(
            [ColumnTranslation(column_name='STATUS',
                               to_save=['D', 'F', 'R', 'W'],
                               default='U'),
             ColumnTranslation(column_name='ASSET_TYPE_LAST',
                               to_save=['Smartphone', 'Tablet']),
             ColumnTranslation(column_name='DEVICE_TYPE_BUS',
                               to_save=['Smartphone', 'Tablet', 'Undefined']),
             ColumnTranslation(column_name='USAGE_AREA',
                               to_save=['Minsk', 'Undefined'])])),
        ('translate_encoder', translate_ohe)
    ]
)
```

Step 3. Combine all pipelines

```
full_pipeline = FeatureUnion(transformer_list=[
    ('num', numeric_pipeline),
    ('translate', translate_pipeline),
    ('trend', trend_pipeline),
    ('date', date_pipeline),
    ('gender', gender_pipeline),
    ('bool', bool_pipeline)
])
```

Step 4. Add model as last step of pipeline

```
pip = Pipeline(
    steps=[
        ('preparation', full_pipeline),
        ('gc', GridSearchCV(method(random_state=29, **ctor_params), params, n_jobs=-1,
                            scoring='accuracy', cv=5, refit=True, verbose=2))
    ]
)

pip.fit(X_train, y_train)
y_true, y_pred = y_test, pip.predict(X_test)
```

Another approach: ColumnTransformer

```
numeric_transformer = Pipeline(steps=[
       ('imputer', SimpleImputer(strategy='mean'))
      ,('scaler', StandardScaler())
])


categorical_transformer = Pipeline(steps=[
       ('imputer', SimpleImputer(strategy='constant'))
      ,('encoder', OrdinalEncoder())
])
```

```
numeric_features = ['temp', 'atemp', 'hum', 'windspeed']


categorical_features = ['season', 'mnth', 'holiday', 'weekday',
'workingday', 'weathersit']


preprocessor = ColumnTransformer(
   transformers=[
     ('numeric', numeric_transformer, numeric_features)
   ,('categorical', categorical_transformer, categorical_features)
])
```