Machine learning in Python: scikit-learn

Romain Tavenard (Université de Rennes 2)

Installing scikit-learn

- On Deepnote:
  - Already installed
    (if a specific version is needed, use requirements.txt)
- On your machine:
  - pip install scikit-learn
    or
    conda install scikit-learn
- Loading the module in Python:
  - `import sklearn`
- Most importantly
  - The doc is awesome, use it!
    https://scikit-learn.org/stable/index.html

Preparing your data

- In scikit-learn:
  - X is a (n, p) numpy array
    n input observations in dimension p
  - y is a (n,) or (n, p_out) numpy array
    expected outputs

- Pre-processing
  - https://scikit-learn.org/stable/modules/preprocessing.html
  - Pre-processing methods are Transformers (cf. next slide)

Transformers

- Objects with following methods
  - `fit(X)`
  - `transform(X)`
  - `fit_transform(X)`

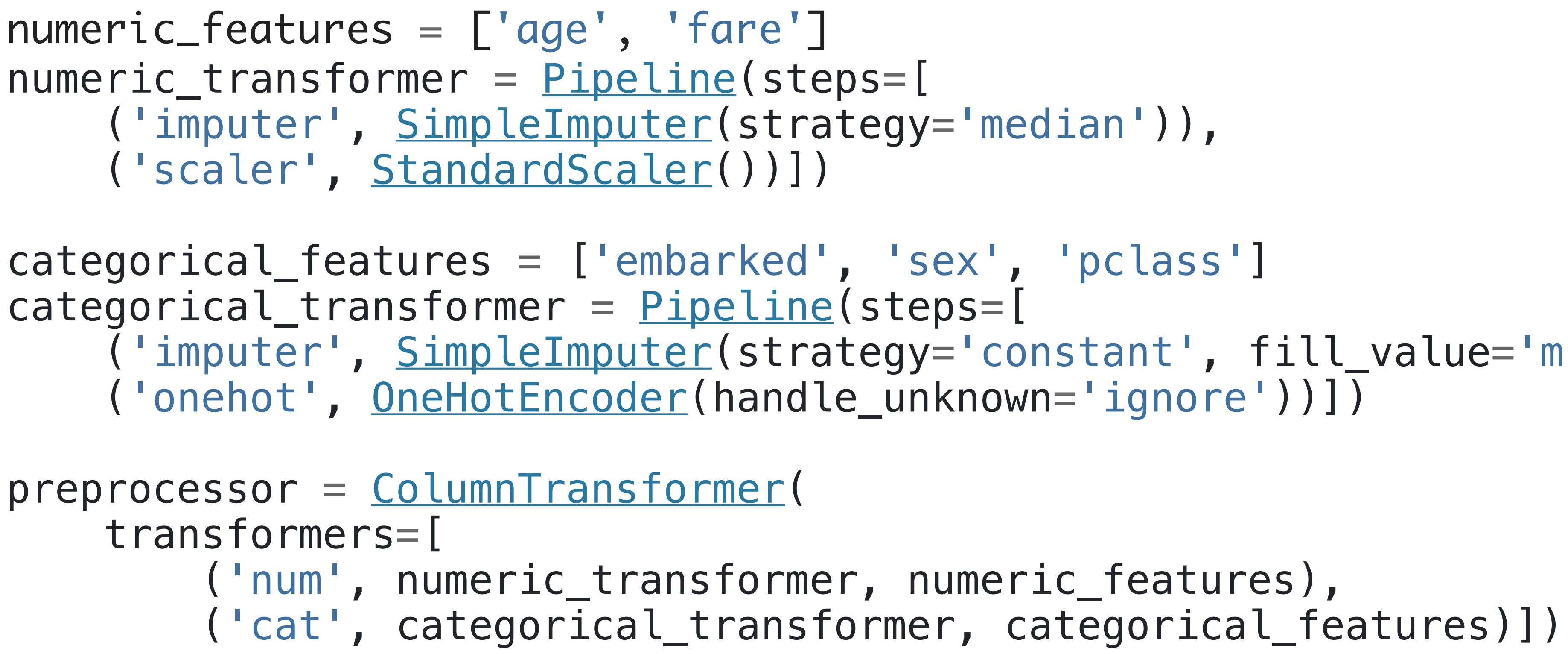
- Example (from scikit-learn docs)

```
>>> from sklearn.preprocessing import StandardScaler
>>> data = [[0, 0], [0, 0], [1, 1], [1, 1]]
>>> scaler = StandardScaler()
>>> scaler.fit(data)
>>> print(scaler.mean_)
[0.5 0.5]
>>> print(scaler.transform(data))
[[-1. -1.]
 [-1. -1.]
 [ 1.  1.]
 [ 1.  1.]]
>>> print(scaler.transform([[2, 2]]))
[[3. 3.]]
```

Transformers

- To apply different preprocessing for different columns: ColumnTransformer

- Example (from scikit-learn docs): https://scikit-learn.org/stable/auto_examples/compose/plot_column_transformer_mixed_types.html



```
numeric_features = ['age', 'fare']
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())])

categorical_features = ['embarked', 'sex', 'pclass']
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)])
```

Estimators

- Objects with following methods
  - `fit(X[, y])`
  - `predict(X)`
  - `predict_proba(X) (but not always…)`

- Example (from scikit-learn docs)

```
>>> X = [[0], [1], [2], [3]]
>>> y = [0, 0, 1, 1]
>>> from sklearn.neighbors import KNeighborsClassifier
>>> neigh = KNeighborsClassifier(n_neighbors=3)
>>> neigh.fit(X, y)
KNeighborsClassifier(...)
>>> print(neigh.predict([[1.1]]))
[0]
>>> print(neigh.predict_proba([[0.9]]))
[[0.66666667 0.33333333]]
```

Pipeline

- Objects that represent successive Transformers / Estimators

- Example (adapted from scikit-learn docs)

```
from sklearn import datasets
from sklearn.decomposition import PCA
from sklearn.linear_model import SGDClassifier
from sklearn.pipeline import Pipeline

logistic = SGDClassifier(loss='log', penalty='l2', early_stopping=True,
                         max_iter=10000, tol=1e-5, random_state=0)
pca = PCA(n_components=30)
pipe = Pipeline(steps=[('pca', pca), ('logistic', logistic)])

digits = datasets.load_digits()
X_digits = digits.data
y_digits = digits.target

pipe.fit(X_digits, y_digits)
```

Model selection

- Goal: compare several models / hyper-parameter sets
- Base object in scikit-learn: `GridSearchCV`

- Example (from scikit-learn docs)

```
>>> from sklearn import svm, datasets
>>> from sklearn.model_selection import GridSearchCV
>>> iris = datasets.load_iris()
>>> parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
>>> svc = svm.SVC(gamma="scale")
>>> clf = GridSearchCV(svc, parameters, cv=5)
>>> clf.fit(iris.data, iris.target)
...
GridSearchCV(cv=5, error_score=...,
       estimator=SVC(C=1.0, cache_size=..., class_weight=..., coef0=...,
                     decision_function_shape='ovr', degree=..., gamma=...,
                     kernel='rbf', max_iter=-1, probability=False,
                     random_state=None, shrinking=True, tol=...,
                     verbose=False),
       iid=..., n_jobs=None,
       param_grid=..., pre_dispatch=..., refit=..., return_train_score=...,
       scoring=..., verbose=...)
```

Conclusion

- What you should know about scikit-learn
  - Data formats
  - Basic objects
  - **How to read the docs**

- On the course page
  - Tutorials for basic scikit-learn usage

Project

- Data: https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data
- Goal
  - What is the best model (from sklearn-implemented ones) for this data, and what is its performance?
- Format
  - Upload your notebook on CURSUS
- Deadline
  - December 16th, 2022, 23:59