Scikit-learn

COMSW4705 Fall 2019
Elsbeth Turcan

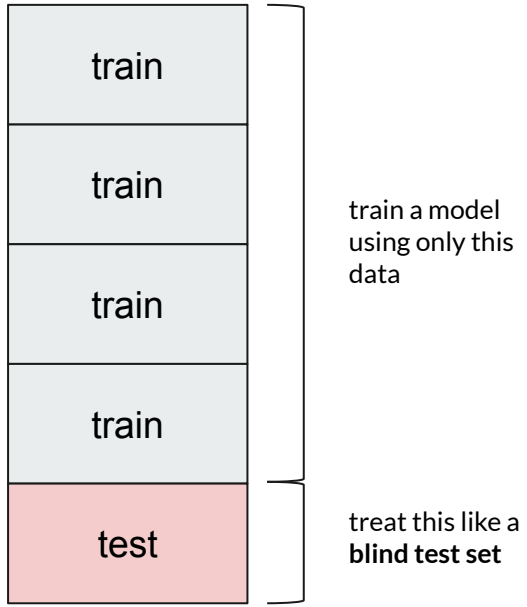
ML Pipeline

- Data gathering/preprocessing
- Vectorization
- Training
- Prediction

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
import sklearn.metrics
import sklearn.neighbors

print("Loading 20 newsgroups dataset for categories:")
data_train = fetch_20newsgroups(subset='train', shuffle=True, random_state=42)
data_test = fetch_20newsgroups(subset='test', shuffle=True, random_state=42)
print('data loaded')

'''Create tf-idf vectors for the input'''
vectorizer = TfidfVectorizer(sublinear_tf=True, max_df=0.9,
                                    stop_words='english')
X_train = vectorizer.fit_transform(data_train.data)
X_test = vectorizer.transform(data_test.data)
y_train = data_train.target
y_test = data_test.target

'''Train a K-Neighbors Classifier on the data'''
n_neighbors = 2
weights = 'uniform'
clf = sklearn.neighbors.KNeighborsClassifier(n_neighbors, weights=weights)
clf.fit(X_train, y_train)

'''Make predictions on the test data using the trained classifier'''
y_predicted = clf.predict(X_test)
print ('Classification report:')
print sklearn.metrics.classification_report(y_test, y_predicted,

target_names=data_test.target_names)
```

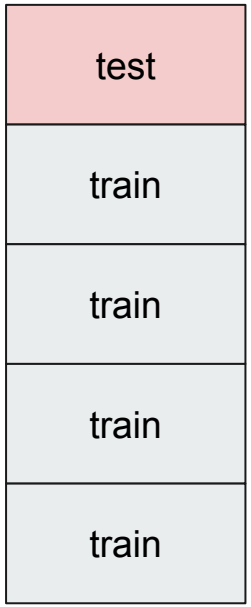
Cross-validation

- N-fold cross-validation splits the training data into N sections, or “folds”, and iterates over them, treating each fold as a miniature test set in one iteration and training on all other data
- Useful for analyzing the robustness of your model, or training on small data
- Be mindful that you **do not** train on features that only appear in test!
    - Sklearn’s built-in cross validation functions **DO NOT DO THIS CORRECTLY**!



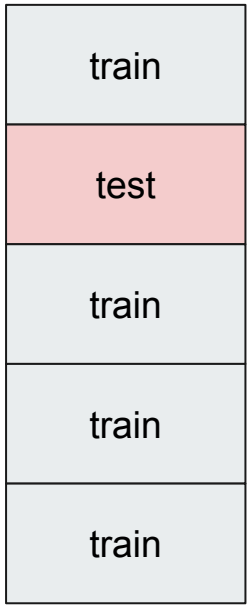
Cross-validation

- N-fold cross-validation splits the training data into N sections, or “folds”, and iterates over them, treating each fold as a miniature test set in one iteration and training on all other data
- Useful for analyzing the robustness of your model, or training on small data
- Be mindful that you **do not** train on features that only appear in test!
    - Sklearn’s built-in cross validation functions **DO NOT DO THIS CORRECTLY**!



Cross-validation

- N-fold cross-validation splits the training data into N sections, or “folds”, and iterates over them, treating each fold as a miniature test set in one iteration and training on all other data
- Useful for analyzing the robustness of your model, or training on small data
- Be mindful that you **do not** train on features that only appear in test!
    - Sklearn’s built-in cross validation functions **DO NOT DO THIS CORRECTLY**!



...and so on;
average the accuracies of all 5 iterations to get the model accuracy

Cross-validation

- N-fold cross-validation splits the training data into N sections, or “folds”, and iterates over them, treating each fold as a miniature test set in one iteration and training on all other data
- Useful for analyzing the robustness of your model, or training on small data
- Be mindful that you **do not** train on features that only appear in test!
    - Sklearn’s built-in cross validation functions **DO NOT DO THIS CORRECTLY**!
- Sklearn has useful built-in iterators you can use to split your data into the right folds

Tuning

- Models have various parameters and certain parameter settings are more appropriate for your problem
- The documentation will list them and their possible values
- To get 3/5 or even 4/5 points for HW1, you shouldn't need to worry too much about parameters

Parameters:

penalty : str, 'l1' or 'l2', default: 'l2'

Used to specify the norm used in the penalization. The 'newton-cg', 'sag' and 'lbfgs' solvers support only l2 penalties.

New in version 0.19: l1 penalty with SAGA solver (allowing 'multinomial' + L1)

dual : bool, default: False

Dual or primal formulation. Dual formulation is only implemented for l2 penalty with liblinear solver. Prefer dual=False when n_samples > n_features.

tol : float, default: 1e-4

Tolerance for stopping criteria.

C : float, default: 1.0

Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.

Saving models

- Scikit-learn saves models to file using the built-in library **pickle**

```
pickle.dump(model, open('model.pkl', 'w+'))
```

- Models can be loaded in new files (without knowing what they originally were)

```
model = pickle.load(open('model.pkl', 'r'))

model.predict(...)
```

- Good idea to save your best-performing models while you try different model settings

Tips and Tricks

- Try simple things first
- Make educated guesses to narrow down the search space
  - Look at the features given in the data .csv
  - Think why certain models or feature combinations might be good
- Don't tune your parameters and features individually and exhaustively
  - i.e., don't write a single classifier and keep changing individual numbers -- automate the search!
- Sklearn vectorizers are your friends for n-grams additional features such as LIWC
  - They have options too - e.g., n-grams have a range and vocabulary size
- HW1: try first to improve your plain n-gram model -- then your feature model has a good foundation
- Come to office hours if you need help with the basics of machine learning