Source

Welcome to Week 7 Lecture 2!

Data Science in Python & Machine Learning

Announcements:

1. Belt Exams This Friday:
    a. Weeks 5-6 assignments and resubmits due by Friday 3/11 by 9AM PST
    b. Must have attended 80% of the live lectures
    c. Must have submitted 90% of assignments
    d. Set aside at least a day between 3/11 and 3/13 to complete exam.
    e. I have posted the solution notebook to the mock belt exam today on the discord channel.
    f. Try the mock belt exam as a classification problem.

2. Purvi will be joining the data science instructor team

3. Josh Johnson joining in on the lectures

Learning Goals

By the end of this lesson you will be able to:

- ❑ Define a hyperparameter
- ❑ Regularize a model by tuning hyperparameters
- ❑ Determine what hyperparameters a model has for tuning
- ❑ Explain cross-validation
- ❑ Efficiently determine the optimal hyperparameter settings for a model using GridSearchCV

Hyperparameters

Parameter: A value learned by a model. For example:

1. Coefficients for Linear and Logistic Regression
2. Rules for Splitting Nodes in a Decision Tree

Hyperparameter: A developer tuned value or argument in a model. For example:

1. K in K-Nearest Neighbors
2. Max Depth in a Decision Tree
3. C in Logistic Regression

Tuning Hyperparameters

1. **Regularization**

2. **Increase/decrease model complexity**

3. **Experiment with different model approaches**
    a. Solvers in logistic regression
    b. Distance metrics in KNN

GridSearchCV:

1. Tune many hyperparameters at once
2. Try all combinations

RandomForestClassifier:

1. n_estimators = [50, 100, 200]
2. max_depth = [4,7,10,20]
3. min_samples_split = [2, 10,100] (or can give a range; range(1,10))
4. max_features = [‘auto’, ‘sqrt’, ‘log2’]

Gridsearch will try all combinations and return the best!

Model Development Cycle with GridSearch CV

1. Prepare the best version of your data that you can. This affects model performance.

2. Model Type should be explored next

3. Tuning hyperparameters should be explored last.

GridSearchCV

- As you become more comfortable with machine learning, you will want to optimize the models you use by “tuning the hyperparameters.”
- Learning and understanding all of the hyperparameters associated with each model is beyond the scope of this course, but you can still try different combinations to see what produces the best results.
- GridSearchCV allows us to test multiple combinations of hyperparameters at a time to more easily determine the optimal values.
- The more combinations you try, the longer it will take to run the GridSearchCV!

Note: If you want to try many combinations and get faster output, you can also try RandomizedSearchCV which will only test a sampling of your hyperparameters. It will go faster to help you narrow down your values for GridSearchCV.

What does the CV in GridSearch CV mean?

Cross-Validation!

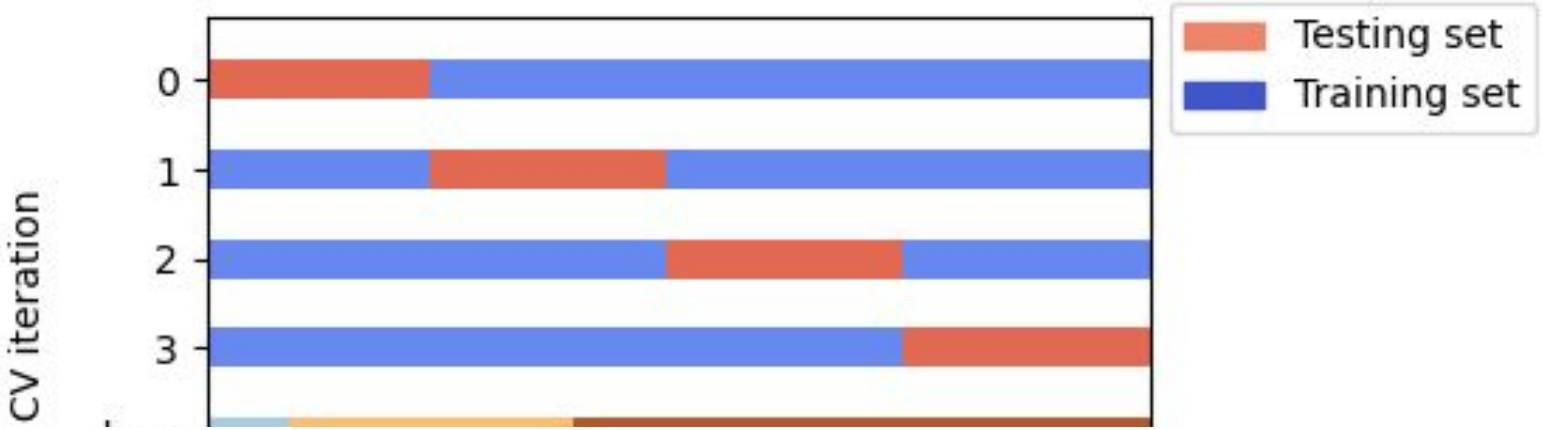

Image Source and Learn More Here!

Steps to follow

1. Make the pipeline

```
from sklearn.pipeline import make_pipeline

# create a pipeline
knn_pipe = make_pipeline(StandardScaler(), KNeighborsClassifier())
```

2. **Param Grid Dictionary:**
3. Define the hyperparameters you want to try. Note that since you are using make_pipeline, the lowercase name of the model followed by __ (dunder) will precede the name of the hyperparameter.

```
# Dictionary of hyperparamters to test.
# Use pipeline naming conventions
param_grid = {'kneighborsclassifier__n_neighbors': range(1, 10),
              'kneighborsclassifier__weights' : ['uniform', 'distance']}
```

How did I know the prefix in the param grid?

```
1   pipe
```

```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('kneighborsclassifier', KNeighborsClassifier())])
```

1. Examine the pipe object to see the names of the steps
2. The KNN step is named ‘kneighborsclassifier
3. In the param grid, access the step to tune with:
   <name of step>__<name of parameter> (2 underscores, called ‘dunder’)
4. So tuning the number of neighbors to query in a KNN model would be:

```
'kneighborsclassifier__n_neighbors'
```

Cool note, you can tune hyperparameters of preprocessing steps, too!

```
params = {'simpleimputer__strategy':['mean','median'],
          'kneighborsclassifier__n_neighbors':range(1,20)}
```

How to view the hyperparameters for a model

```
1 knn_pipe.get_params()
```

```
{'kneighborsclassifier': KNeighborsClassifier(),
 'kneighborsclassifier__algorithm': 'auto',
 'kneighborsclassifier__leaf_size': 30,
 'kneighborsclassifier__metric': 'minkowski',
 'kneighborsclassifier__metric_params': None,
 'kneighborsclassifier__n_jobs': None,
 'kneighborsclassifier__n_neighbors': 5,
 'kneighborsclassifier__p': 2,
 'kneighborsclassifier__weights': 'uniform',
 'memory': None,
 'standardscaler': StandardScaler(),
 'standardscaler__copy': True,
 'standardscaler__with_mean': True,
 'standardscaler__with_std': True,
 'steps': [('standardscaler', StandardScaler()),
  ('kneighborsclassifier', KNeighborsClassifier())],
 'verbose': False}
```

Steps to follow

1. Instantiate your grid search. Be sure to use your pipeline for the model

```
# Instantiate grid-search model using your pipeline and hyperparateter dictionary
grid = GridSearchCV(knn_pipe, param_grid)
```

2. Fit the gridsearch model, identify the best hyperparameters

```
# fit the gridsearch, identify the best hyperparams, and evaluate model
grid.fit(X_train, y_train)
print(grid.best_params_)
```

3. Get the best version of the model and evaluate as usual!

```
best_knn = grid.best_estimator_
```

CodeAlong Notebook

CodeAlong Data

Challenge Notebook

Challenge Data