**# # Cross-validation for parameter tuning, model selection, and feature selection**

# \*From the video series: [Introduction to machine learning with scikit-learn](https://github.com/justmarkham/scikit-learn-videos)\*

# ## Agenda

#

# - What is the drawback of using the \*\*train/test split\*\* procedure for model evaluation?

# - How does \*\*K-fold cross-validation\*\* overcome this limitation?

# - How can cross-validation be used for selecting \*\*tuning parameters\*\*, choosing between \*\*models\*\*, and selecting \*\*features\*\*?

# - What are some possible \*\*improvements\*\* to cross-validation?

# ## Review of model evaluation procedures

# \*\*Motivation:\*\* Need a way to choose between machine learning models

#

# - Goal is to estimate likely performance of a model on \*\*out-of-sample data\*\*

#

# \*\*Initial idea:\*\* Train and test on the same data

#

# - But, maximizing \*\*training accuracy\*\* rewards overly complex models which \*\*overfit\*\* the training data

#

# \*\*Alternative idea:\*\* Train/test split

#

**# - Split the dataset into two pieces, so that the model can be trained and tested on \*\*different data\*\***

# - \*\*Testing accuracy\*\* is a better estimate than training accuracy of out-of-sample performance

# - But, it provides a \*\*high variance\*\* estimate since changing which observations happen to be in the testing set can significantly change testing accuracy

**# - How KFold works an example form sklearn**

**# - KFolds implementation to linear regression**

from sklearn.datasets import load\_iris

from sklearn.cross\_validation import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn import metrics

# read in the iris data

iris = load\_iris()

# create X (features) and y (response)

X = iris.data

y = iris.target

# use train/test split with different random\_state values

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=4)

# check classification accuracy of KNN with K=5

knn = KNeighborsClassifier(n\_neighbors=5)

knn.fit(X\_train, y\_train)

y\_pred = knn.predict(X\_test)

print metrics.accuracy\_score(y\_test, y\_pred)

# \*\*Question:\*\* What if we created a bunch of train/test splits, calculated the testing accuracy for each, and averaged the results together?

#

# \*\*Answer:\*\* That's the essense of cross-validation!

# ## Steps for K-fold cross-validation

# 1. Split the dataset into K \*\*equal\*\* partitions (or "folds").

# 2. Use fold 1 as the \*\*testing set\*\* and the union of the other folds as the \*\*training set\*\*.

# 3. Calculate \*\*testing accuracy\*\*.

# 4. Repeat steps 2 and 3 K times, using a \*\*different fold\*\* as the testing set each time.

# 5. Use the \*\*average testing accuracy\*\* as the estimate of out-of-sample accuracy.

# Diagram of \*\*5-fold cross-validation:\*\*

#

# ![5-fold cross-validation](images/cross\_validation\_diagram.png)

# simulate splitting a dataset of 25 observations into 5 folds

from sklearn.cross\_validation import KFold

kf = KFold(25, n\_folds=5, shuffle=False)

# print the contents of each training and testing set

print '{} {:^61} {}'.format('Iteration', 'Training set observations', 'Testing set observations')

for iteration, data in enumerate(kf, start=1):

print '{:^9} {} {:^25}'.format(iteration, data[0], data[1])

# - Dataset contains \*\*25 observations\*\* (numbered 0 through 24)

# - 5-fold cross-validation, thus it runs for \*\*5 iterations\*\*

# - For each iteration, every observation is either in the training set or the testing set, \*\*but not both\*\*

# - Every observation is in the testing set \*\*exactly once\*\*

# ## Comparing cross-validation to train/test split

# Advantages of \*\*cross-validation:\*\*

#

# - More accurate estimate of out-of-sample accuracy

# - More "efficient" use of data (every observation is used for both training and testing)

#

# Advantages of \*\*train/test split:\*\*

#

# - Runs K times faster than K-fold cross-validation

# - Simpler to examine the detailed results of the testing process

# ## Cross-validation recommendations

# 1. K can be any number, but \*\*K=10\*\* is generally recommended

# 2. For classification problems, \*\*stratified sampling\*\* is recommended for creating the folds

# - Each response class should be represented with equal proportions in each of the K folds

# - scikit-learn's `cross\_val\_score` function does this by default

# ## Cross-validation example: parameter tuning

# \*\*Goal:\*\* Select the best tuning parameters (aka "hyperparameters") for KNN on the iris dataset

from sklearn.cross\_validation import cross\_val\_score

# 10-fold cross-validation with K=5 for KNN (the n\_neighbors parameter)

knn = KNeighborsClassifier(n\_neighbors=5)

scores = cross\_val\_score(knn, X, y, cv=10, scoring='accuracy')

print scores

# use average accuracy as an estimate of out-of-sample accuracy

print scores.mean()

# search for an optimal value of K for KNN

k\_range = range(1, 31)

k\_scores = []

for k in k\_range:

knn = KNeighborsClassifier(n\_neighbors=k)

scores = cross\_val\_score(knn, X, y, cv=10, scoring='accuracy')

k\_scores.append(scores.mean())

print k\_scores

import matplotlib.pyplot as plt

# plot the value of K for KNN (x-axis) versus the cross-validated accuracy (y-axis)

plt.plot(k\_range, k\_scores)

plt.xlabel('Value of K for KNN')

plt.ylabel('Cross-Validated Accuracy')

# ## Cross-validation example: model selection

# \*\*Goal:\*\* Compare the best KNN model with logistic regression on the iris dataset

# 10-fold cross-validation with the best KNN model

knn = KNeighborsClassifier(n\_neighbors=20)

print cross\_val\_score(knn, X, y, cv=10, scoring='accuracy').mean()

# 10-fold cross-validation with logistic regression

from sklearn.linear\_model import LogisticRegression

logreg = LogisticRegression()

print cross\_val\_score(logreg, X, y, cv=10, scoring='accuracy').mean()

# ## Cross-validation example: feature selection

# \*\*Goal\*\*: Select whether the Newspaper feature should be included in the linear regression model on the advertising dataset

import pandas as pd

import numpy as np

from sklearn.linear\_model import LinearRegression

# read in the advertising dataset

data = pd.read\_csv('http://www-bcf.usc.edu/~gareth/ISL/Advertising.csv', index\_col=0)

# create a Python list of three feature names

feature\_cols = ['TV', 'Radio', 'Newspaper']

# use the list to select a subset of the DataFrame (X)

X = data[feature\_cols]

# select the Sales column as the response (y)

y = data.Sales

# 10-fold cross-validation with all three features

lm = LinearRegression()

scores = cross\_val\_score(lm, X, y, cv=10, scoring='mean\_squared\_error')

print scores

# fix the sign of MSE scores

mse\_scores = -scores

print mse\_scores

# convert from MSE to RMSE

rmse\_scores = np.sqrt(mse\_scores)

print rmse\_scores

# calculate the average RMSE

print rmse\_scores.mean()

# 10-fold cross-validation with two features (excluding Newspaper)

feature\_cols = ['TV', 'Radio']

X = data[feature\_cols]

print np.sqrt(-cross\_val\_score(lm, X, y, cv=10, scoring='mean\_squared\_error')).mean()

**# - How KFold works an example form sklearn**

>>> from sklearn.cross\_validation import KFold

>>> X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])

>>> y = np.array([1, 2, 3, 4])

>>> kf = KFold(4, n\_folds=2)

>>> len(kf)

2

>>> print(kf)

sklearn.cross\_validation.KFold(n=4, n\_folds=2, shuffle=False,

random\_state=None)

>>> for train\_index, test\_index in kf:

... print("TRAIN:", train\_index, "TEST:", test\_index)

... X\_train, X\_test = X[train\_index], X[test\_index]

... y\_train, y\_test = y[train\_index], y[test\_index]

TRAIN: [2 3] TEST: [0 1]

TRAIN: [0 1] TEST: [2 3]

.. automethod:: \_\_init\_\_

**# - KFolds implementation to linear regression**

import pandas as pd

import numpy as np

from sklearn.linear\_model import LinearRegression

data = pd.read\_csv('http://www-bcf.usc.edu/~gareth/ISL/Advertising.csv', index\_col=0)

feature\_cols = ['TV', 'Radio', 'Newspaper']

X = data[feature\_cols]

y = data.Sales

kf = KFold(200, n\_folds=4)

len(kf)

from sklearn import metrics

KFMSE=[]

for train\_index, test\_index in kf:

X\_train,x\_test = X.loc[train\_index,:],X.loc[test\_index,:]

Y\_train,y\_test = y.loc[train\_index],y.loc[test\_index]

X\_train=X\_train.dropna()

x\_test=x\_test.dropna()

Y\_train=Y\_train.dropna()

y\_test=y\_test.dropna()

linreg=LinearRegression()

linreg.fit(X\_train,Y\_train)

true=y\_test.dropna()

x\_test=x\_test.dropna()

pred=linreg.predict(x\_test)

KFMSE.append(metrics.mean\_squared\_error(true, pred))

# ## Improvements to cross-validation

# \*\*Repeated cross-validation\*\*

#

# - Repeat cross-validation multiple times (with \*\*different random splits\*\* of the data) and average the results

# - More reliable estimate of out-of-sample performance by \*\*reducing the variance\*\* associated with a single trial of cross-validation

#

# \*\*Creating a hold-out set\*\*

#

# - "Hold out" a portion of the data \*\*before\*\* beginning the model building process

# - Locate the best model using cross-validation on the remaining data, and test it \*\*using the hold-out set\*\*

# - More reliable estimate of out-of-sample performance since hold-out set is \*\*truly out-of-sample\*\*

#

# \*\*Feature engineering and selection within cross-validation iterations\*\*

#

# - Normally, feature engineering and selection occurs \*\*before\*\* cross-validation

# - Instead, perform all feature engineering and selection \*\*within each cross-validation iteration\*\*

# - More reliable estimate of out-of-sample performance since it \*\*better mimics\*\* the application of the model to out-of-sample data

# ## Resources

#

# - scikit-learn documentation: [Cross-validation](http://scikit-learn.org/stable/modules/cross\_validation.html), [Model evaluation](http://scikit-learn.org/stable/modules/model\_evaluation.html)

# - scikit-learn issue on GitHub: [MSE is negative when returned by cross\_val\_score](https://github.com/scikit-learn/scikit-learn/issues/2439)

# - Section 5.1 of [An Introduction to Statistical Learning](http://www-bcf.usc.edu/~gareth/ISL/) (11 pages) and related videos: [K-fold and leave-one-out cross-validation](https://www.youtube.com/watch?v=nZAM5OXrktY) (14 minutes), [Cross-validation the right and wrong ways](https://www.youtube.com/watch?v=S06JpVoNaA0) (10 minutes)

# - Scott Fortmann-Roe: [Accurately Measuring Model Prediction Error](http://scott.fortmann-roe.com/docs/MeasuringError.html)

# - Machine Learning Mastery: [An Introduction to Feature Selection](http://machinelearningmastery.com/an-introduction-to-feature-selection/)

# - Harvard CS109: [Cross-Validation: The Right and Wrong Way](https://github.com/cs109/content/blob/master/lec\_10\_cross\_val.ipynb)

# - Journal of Cheminformatics: [Cross-validation pitfalls when selecting and assessing regression and classification models](http://www.jcheminf.com/content/pdf/1758-2946-6-10.pdf)