Python for Data Science Bootcamp - Day 2

By Craig Sakuma

Schedule - Day 2

Time	Topic
10:00 – 12:00	Pandas
12:00 – 1:00	Lunch
1:00 – 2:00	Overview of Machine Learning
2:00 – 5:00	Random Forest Algorithm

Importing and Exporting Data

- Pandas has easy to use functions for importing and exporting different data types:
    - CSV Files
    - Excel Worksheets
    - Queries from Databases

Reading CSV Files

- Syntax:

File name
(include path if not located
in active directory)

Pandas
DataFrame
function

```
pd.read_csv(<file_name>)
```

Writing CSV Files

- Syntax:

Pandas function

File name (include path if save in different directory)

DataFrame name

```
df.to_csv(‘<file_name>’, header = True,
      index=False))
```

Optional Parameters

Read and Write CSV Files

Open the test_pandas.csv file as a dataframe

```
data = pd.read_csv(‘test_pandas.csv’,
                    header=True)
print data
```

Write the DataFrame to a CSV file

```
data.write_csv( ‘test_data.csv’, header=True,
                index=False)
```

Read Excel Files

Reading Excel files is similar. Just add worksheet you want to read.

```
data = pd.read_excel('test_pandas.xlsm', 'Sheet1')

print data
```

Write Excel Files

For writing Excel files, you need to use a writer object. This enables you to save multiple dataframes as separate sheets in the same file.

```
writer = pd.ExcelWriter('test_sheets.xlsx')
data.to_excel(writer,’Original’)
data.to_excel(writer,'Copy')
writer.save()
```

You can only write data files with .xlsx and not macro enabled files with .xlsm

Connecting to SQL Database

To read and write to SQL you need to establish a connection to a database. Creating connections are different for each type of database, but here is an example for sqlite.

```
import sqlite3
conn = sqlite3.connect(‘test_pandas.db')
```

Create DataFrame from Query

Use the connection for importing a query as a DataFrame

```
sql_query = "SELECT * FROM test"
data = pd.read_sql(sql_query,conn)
print data
```

Don’t forget to close your connection when done

```
conn.close()
```

Exploring Titanic Data

- Import data from CSV Files
- Investigate data
  - View samples of the data
  - Evaluate summary statistics
- Filter and slice the data for analysis

Load and Investigate the File

Load the train.csv file as DataFrame

```
data = pd.read_csv('train.csv')
```

View the first 5 rows of the data set

```
data.head()
```

Analyze the Data

View the last 10 rows of the data set

```
data.tail(10)
```

Investigate the data types in the DataFrame

```
data.info()
```

Get some summary statistics

```
data.describe()
```

Filter the Data

Filter the data for men

```
data[data.Sex=='male']
```

Filter just the ages of the men

```
data.Age[data.Sex==‘male']
```

Count the men and women

```
print data.Sex[data.Sex=='male'].count()
print data.Sex[data.Sex=='female'].count()
```

Filter with Multiple Criteria

Calc the survival rate for adult men (age>=18)

```
data.Survived[(data.Sex==‘male')&
                     (data.Age>=18)].mean()
```

What was the survival rate for women and children?

```
data.Survived[(data.Sex==‘female’)|
                     (data.Age<18)].mean()
```

Group By

Compare the survival rates by sex

```
data.groupby(‘Sex’)[‘Survived’].mean()
```

Create a DataFrame with groupby and view the index

```
new = data.groupby(‘Sex')[['Survived','Age']].mean()
print new.index
```

Reset Index

Reset index for new DataFrame

```
new.reset_index(inplace=True)
```

View the index and the DataFrame

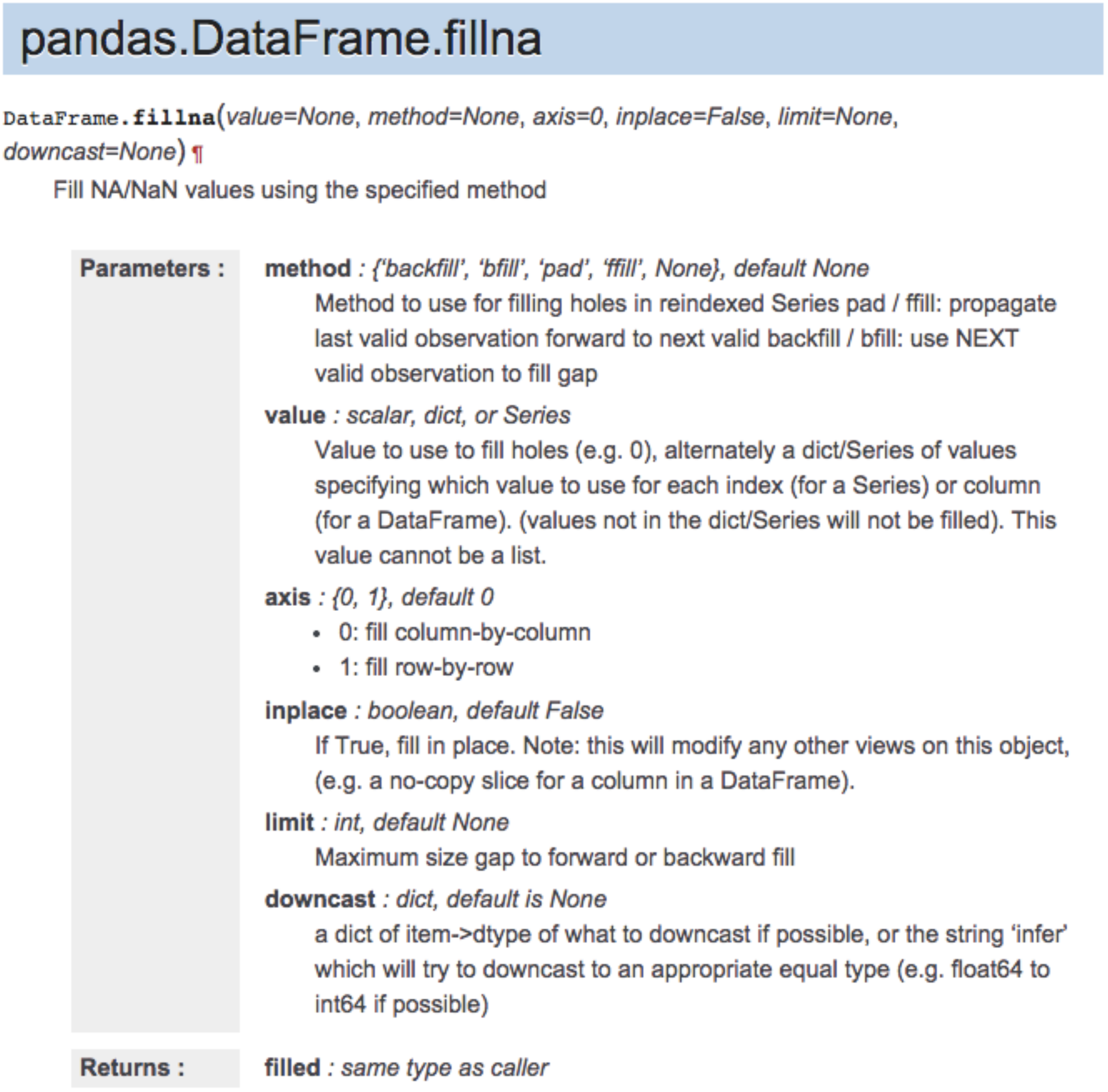
```
print new.index
print new
```

Exercises

- What was the average age of the survivors?
- What was the survival rate of the children (age less than 18) and seniors (age greater than 60)?
- Group by Pclass and investigate average survival rate, age and fare
- Create an Excel file with the names and ages of the surivors on one tab and the names and ages of the deceased in another tab
- Investigate the data and discover an interesting insight

Data Cleaning

- Missing Values
    - Identify Missing Values
    - Drop Values
    - Impute Values
        - Zero
        - Mean
- Categorical Data
    - Convert Text to Numbers
    - Encode Categories as Boolean Values



pandas.DataFrame.fillna

`DataFrame.fillna(value=None, method=None, axis=0, inplace=False, limit=None, downcast=None)` ¶

Fill NA/NaN values using the specified method

Parameters :

method : *{'backfill', 'bfill', 'pad', 'ffill', None}, default None*

Method to use for filling holes in reindexed Series pad / ffill: propagate last valid observation forward to next valid backfill / bfill: use NEXT valid observation to fill gap

value : *scalar, dict, or Series*

Value to use to fill holes (e.g. 0), alternately a dict/Series of values specifying which value to use for each index (for a Series) or column (for a DataFrame). (values not in the dict/Series will not be filled). This value cannot be a list.

axis : *{0, 1}, default 0*

- 0: fill column-by-column
- 1: fill row-by-row

inplace : *boolean, default False*

If True, fill in place. Note: this will modify any other views on this object, (e.g. a no-copy slice for a column in a DataFrame).

limit : *int, default None*

Maximum size gap to forward or backward fill

downcast : *dict, default is None*

a dict of item->dtype of what to downcast if possible, or the string 'infer' which will try to downcast to an appropriate equal type (e.g. float64 to int64 if possible)

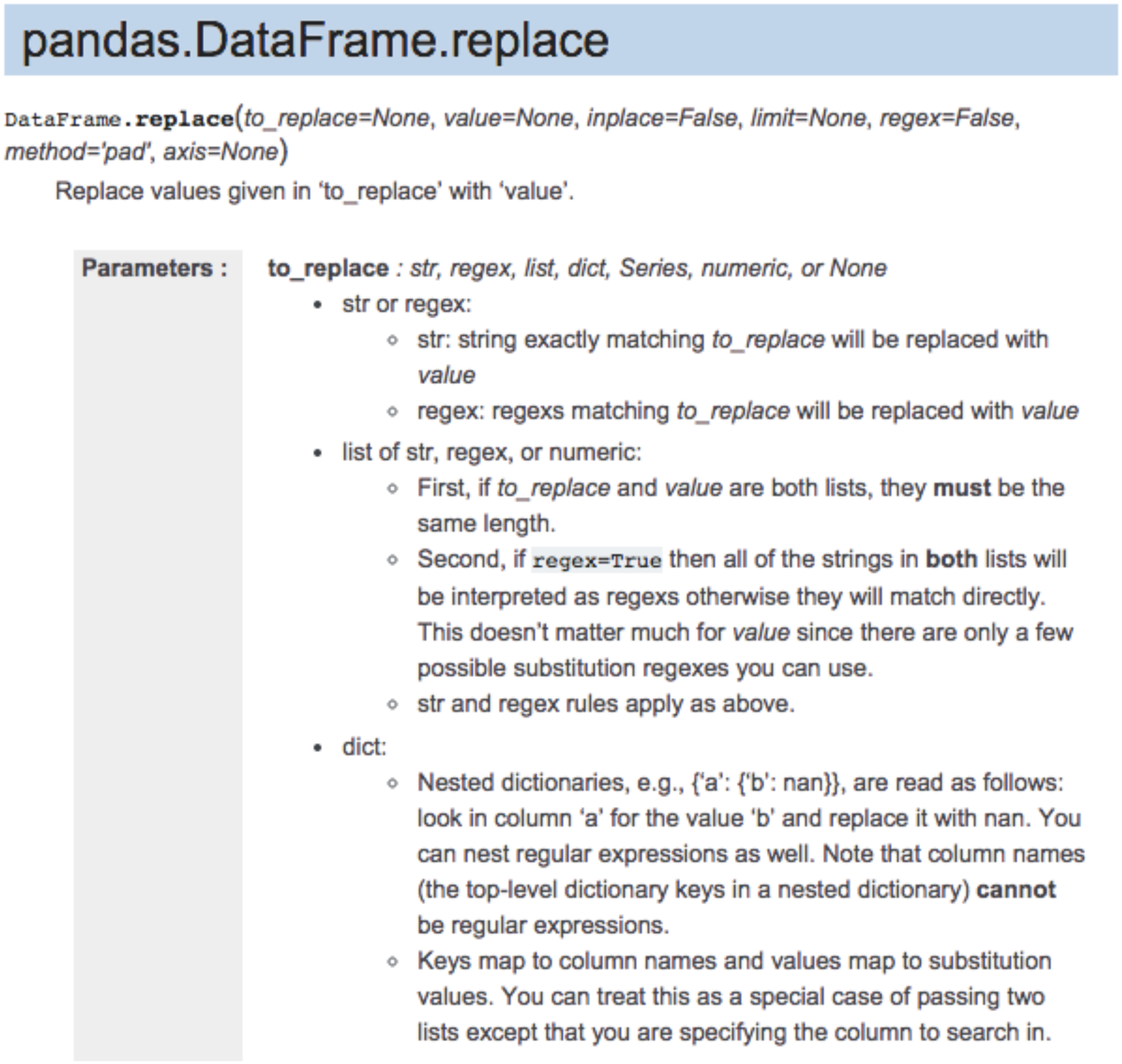
Returns :

filled : *same type as caller*

Fill Missing Age Values

Fill missing values for age

```
print data.info()
avg_age = data.Age.mean()
data.Age = data.Age.fillna(avg_age)
print data.info()
```



pandas.DataFrame.replace

`DataFrame.replace`(*to_replace=None*, *value=None*, *inplace=False*, *limit=None*, *regex=False*, *method='pad'*, *axis=None*)

Replace values given in 'to_replace' with 'value'.

Parameters : **to_replace** *: str, regex, list, dict, Series, numeric, or None*

- str or regex:
  - str: string exactly matching *to_replace* will be replaced with *value*
  - regex: regexs matching *to_replace* will be replaced with *value*
- list of str, regex, or numeric:
  - First, if *to_replace* and *value* are both lists, they **must** be the same length.
  - Second, if `regex=True` then all of the strings in **both** lists will be interpreted as regexs otherwise they will match directly. This doesn't matter much for *value* since there are only a few possible substitution regexes you can use.
  - str and regex rules apply as above.
- dict:
  - Nested dictionaries, e.g., {'a': {'b': nan}}, are read as follows: look in column 'a' for the value 'b' and replace it with nan. You can nest regular expressions as well. Note that column names (the top-level dictionary keys in a nested dictionary) **cannot** be regular expressions.
  - Keys map to column names and values map to substitution values. You can treat this as a special case of passing two lists except that you are specifying the column to search in.

Replace Text

Replace male and female with True and False

```
data.Sex = data.Sex.replace([‘male','female'],
                                              [True, False])
data.head(20)
```

Save data as clean_data.csv

```
data.to_csv(‘clean_data.csv’, header=True, index=False)
```

What is Machine Learning?

“A field of study that gives computers the ability to learn without being explicitly programmed.” (1959)

- Arthur Samuel, AI Pioneer

Netflix Prize

- Challenge to make 10% improvement in Netflix's recommendation system
- Grand prize was $1 million, with annual $50k progress prize to the leader at the end of the year
- 50k teams participated from over 180 countries
- Ratings matrix contained over 100 million numerical entries from 500k users across 17k movies
- Competition began in 2006 and the grand prize was awarded in 2009
- Winning entry was an ensemble of 100's of models

Supervised vs. Unsupervised

Supervised

- Requires truth set of data for training algorithms
- Examples:
  - Forecasting sales
  - Classifying spam

Unsupervised

- Autonomous algorithm that requires no training
- Examples:
  - Cluster analysis
  - Anomaly detection

Machine Learning Categories

	Continuous	Categorical
Supervised	Regression	Classification
Unsupervised	Dimension Reduction	Clustering

Supervised Training

- Training Set
  - Data used to create coefficients for model
  - For example, data set used to create a linear regression
- Test Set
  - Data used to measure performance of trained model

Training Set

Age	*Sex*	*Pclass*	*Survived?*
25	*Male*	*3*	*FALSE*
17	*Female*	*1*	*TRUE*
40	*Male*	*2*	*FALSE*
9	*Female*	*2*	*TRUE*

Classes (target)

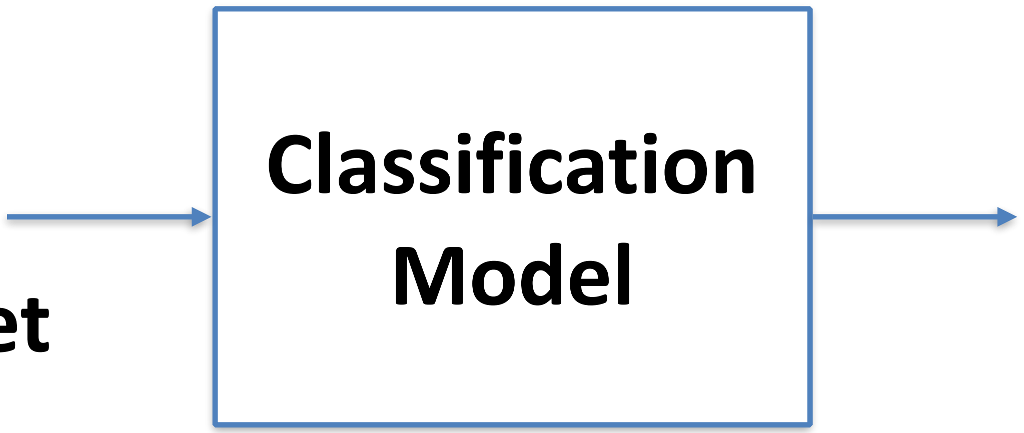
Independent Variables (features)

Classification Algorithm

<u>Input</u>

Attribute Set

- **Age**
- **Pclass**
- **Sex**

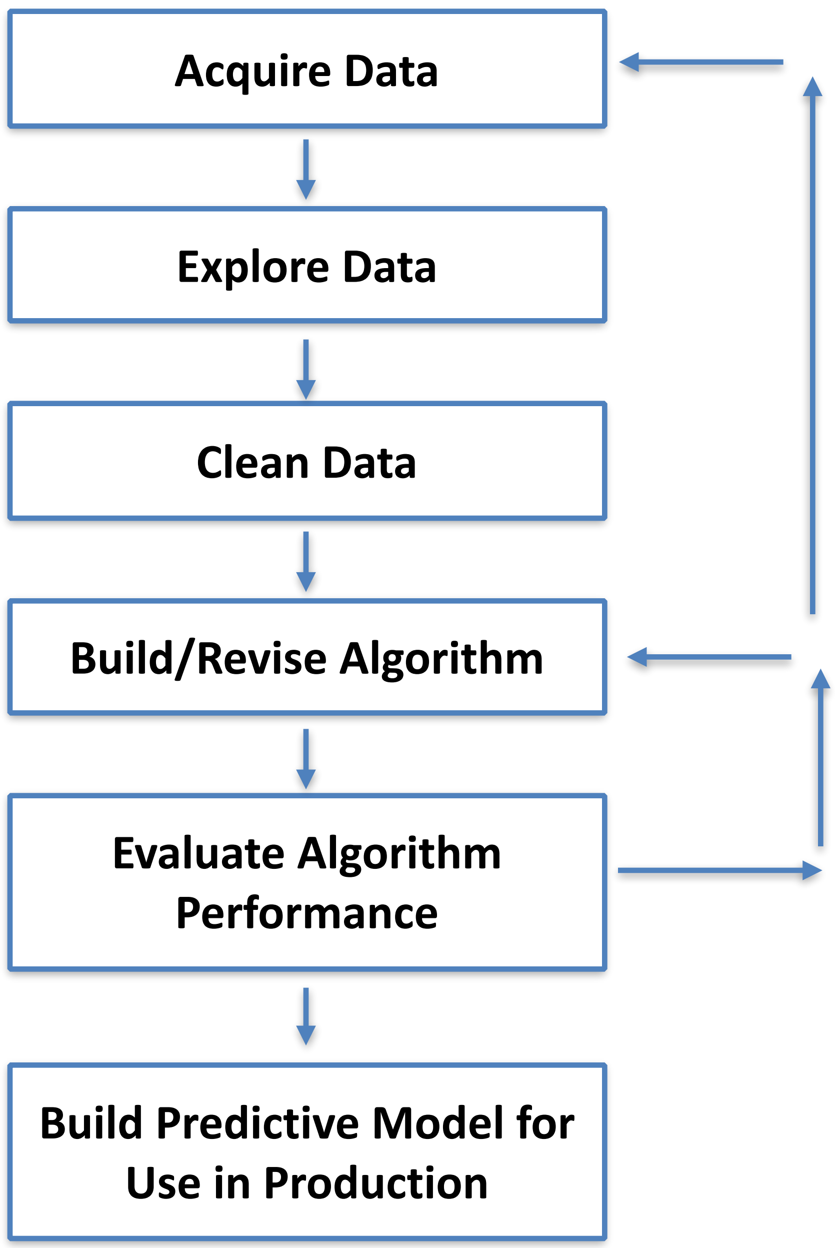


<u>Output</u>

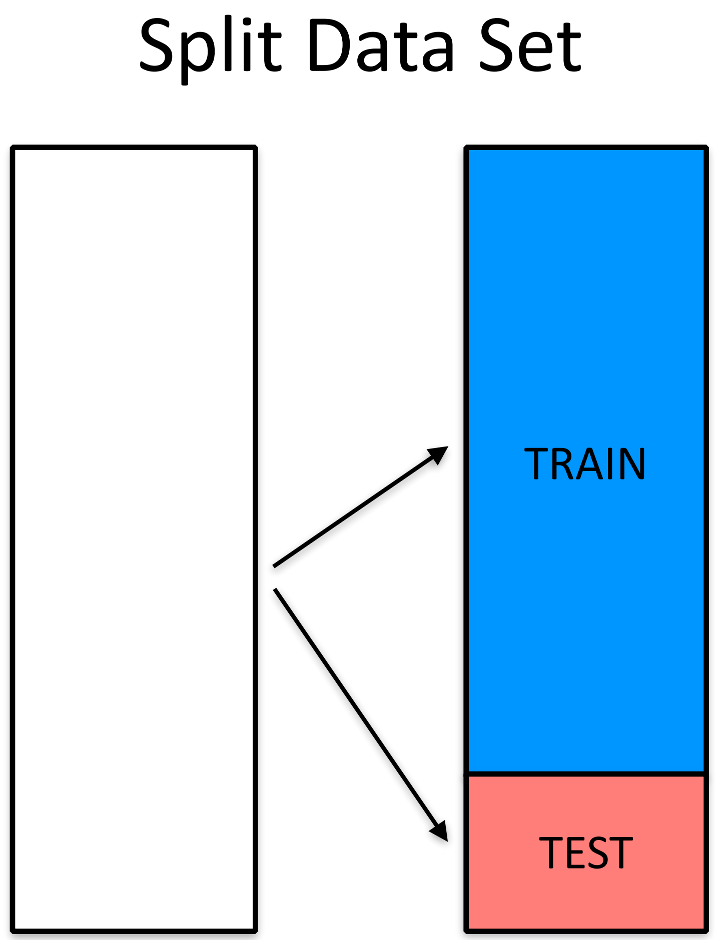
Class Label

- **Survived**
- **Deceased**

Data Science Process


Acquire Data
Explore Data
Clean Data
Build/Revise Algorithm
Evaluate Algorithm Performance
Build Predictive Model for Use in Production

Cross-validation

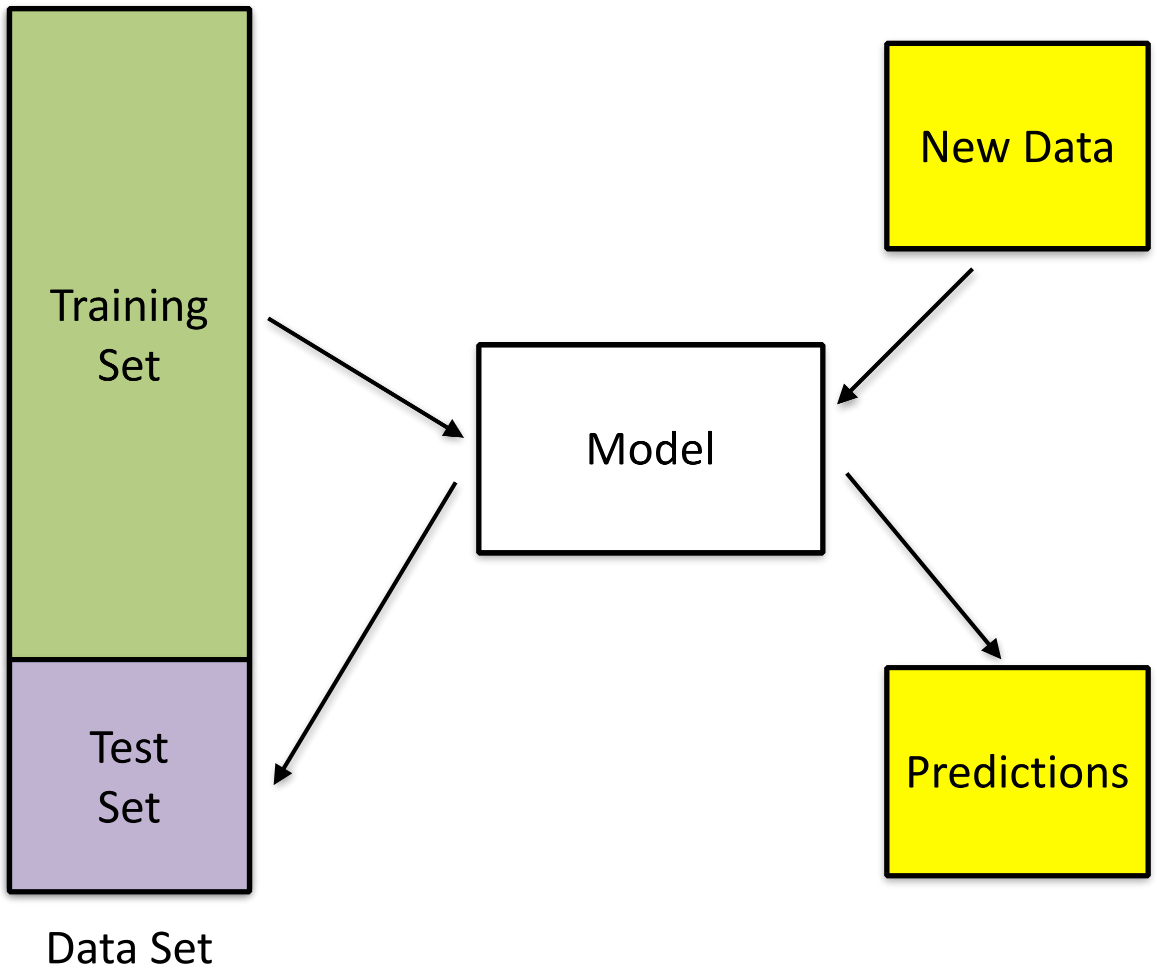


- Separate the data into two groups
- Use part of data to train the model
- Use trained model to predict out come for test data
- Compare predictions with actual results

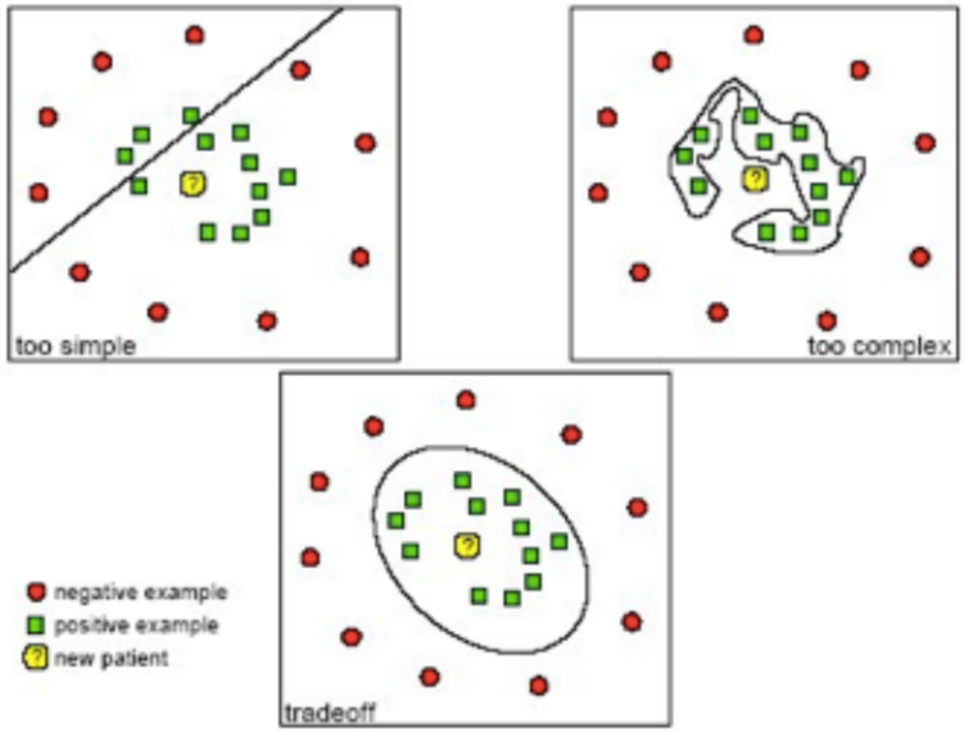
Measure model performance on accuracy of predictions

How to Build a Classification Model

1. Get Data Set
2. Split Data
3. Train Model
4. Test Model
5. Iterate Until Model is Optimized
6. Train model with all data
7. Make Predictions on New Data with optimized model



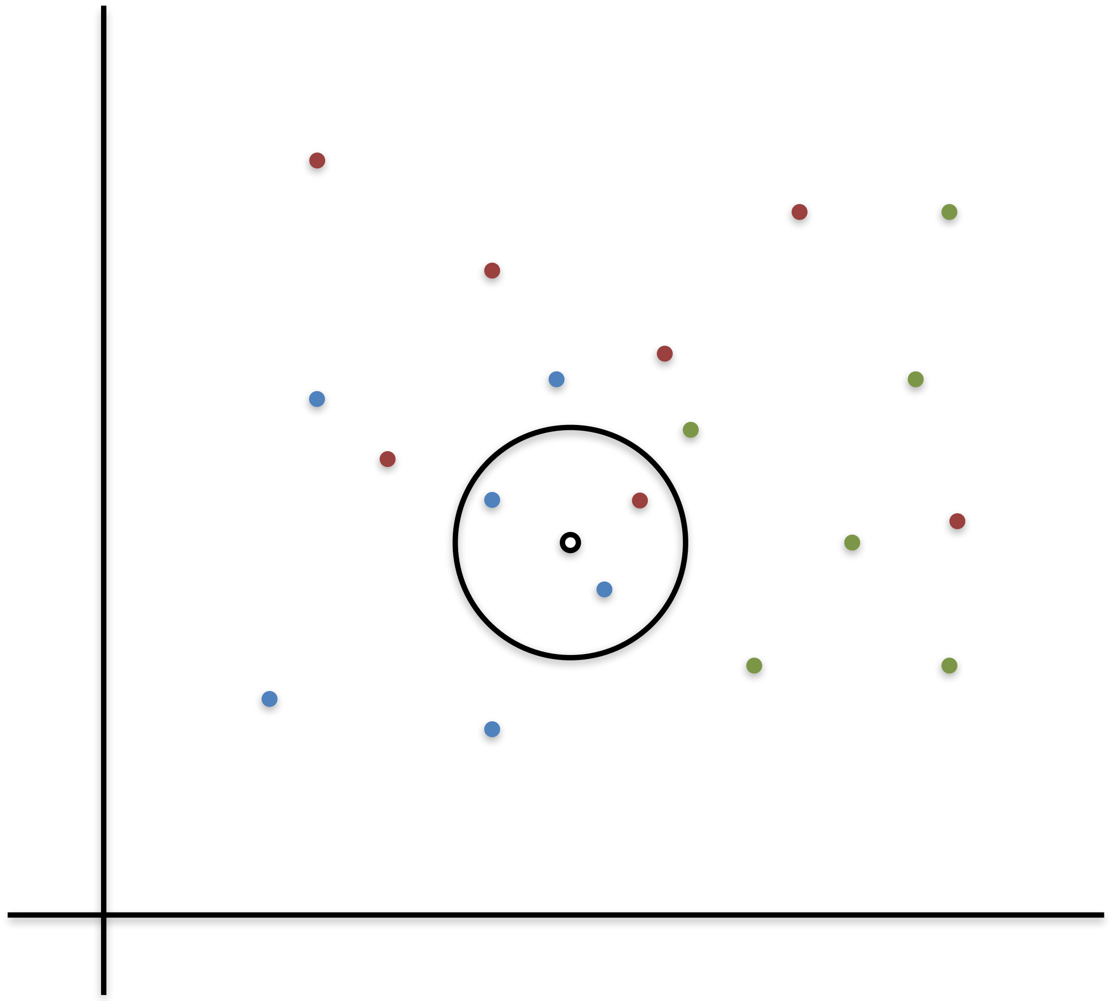
Underfitting and Overfitting



Classification Algorithms

- Logistic Regression
- Naive-Bayes
- Decision Trees
- Support Vector Machines
- Neural Networks
- K-Nearest Neighbors
- Random Forest

K Nearest Neighbors



Suppose you want to predict the white dot

1. Pick a value for k

2. Find colors of the k nearest neighbors

3. Assign the most common color

KNN Considerations

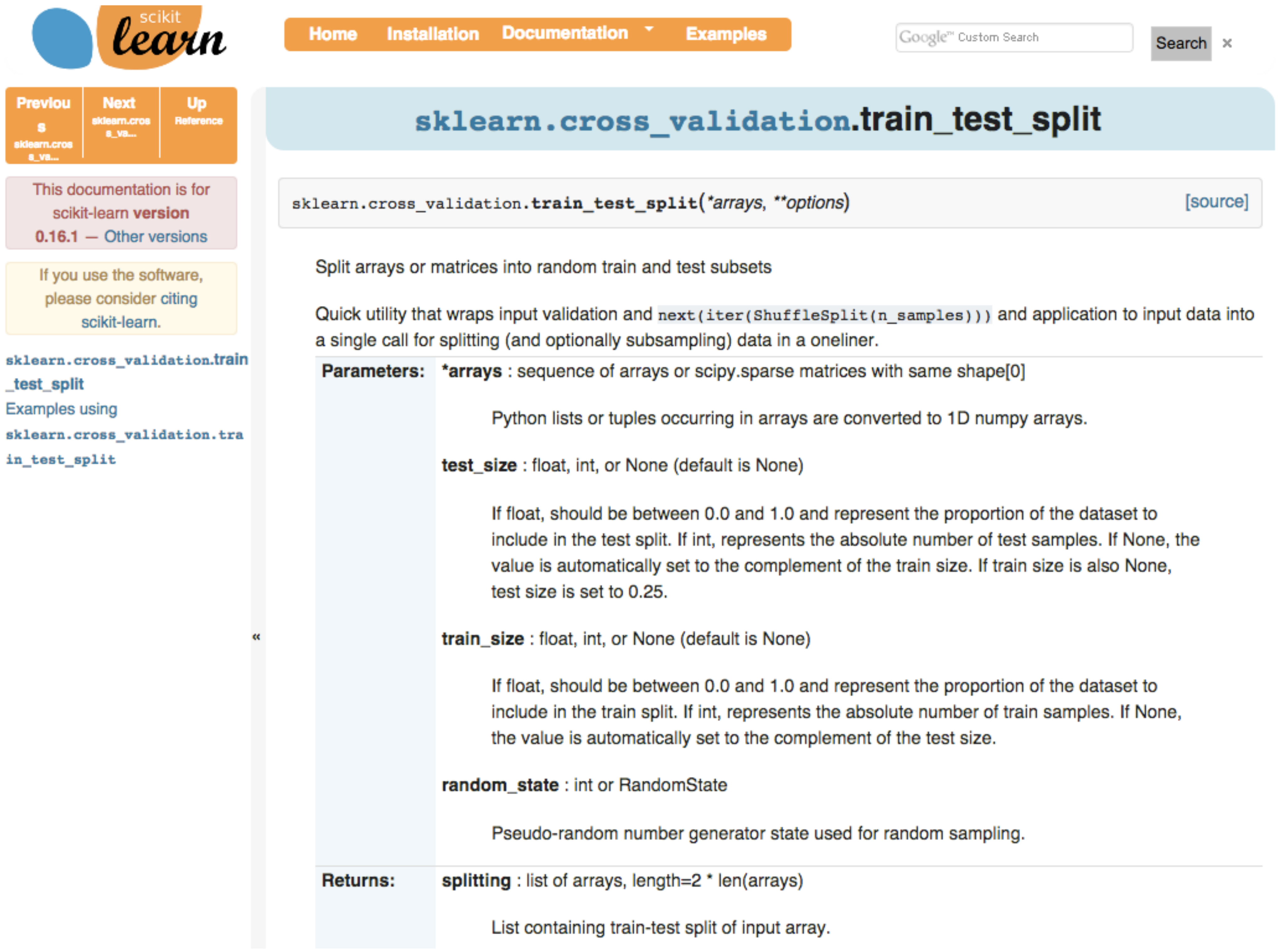
- Scaling of Data has large impact on algorithm (normalization is frequently used)
- Adjust the value of k to optimize the algorithm
- Can become computationally expensive at large scale

Data Normalization

- Technique for adjusting the scale of data
- Calculate mean and standard deviation of all samples for each variable
- Subtract the mean and divide by the standard deviation

Iris Data Set

- Three species of iris
  - Setosa
  - Versicolour
  - Virginica
  - Species are encoded as 0,1 and 2 respectively
- Four measurements used to identify species
  - Sepal length
  - Sepal width
  - Petal length
  - Petal width



This documentation is for scikit-learn **version 0.16.1** — Other versions

If you use the software, please consider citing scikit-learn.

`sklearn.cross_validation`.train_test_split

Examples using `sklearn.cross_validation.train_test_split`

`sklearn.cross_validation`.train_test_split

`sklearn.cross_validation.`**`train_test_split`**(**arrays*, ***options*) [source]

Split arrays or matrices into random train and test subsets

Quick utility that wraps input validation and `next(iter(ShuffleSplit(n_samples)))` and application to input data into a single call for splitting (and optionally subsampling) data in a oneliner.

Parameters:

***arrays** : sequence of arrays or scipy.sparse matrices with same shape[0]

Python lists or tuples occurring in arrays are converted to 1D numpy arrays.

test_size : float, int, or None (default is None)

If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples. If None, the value is automatically set to the complement of the train size. If train size is also None, test size is set to 0.25.

train_size : float, int, or None (default is None)

If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the train split. If int, represents the absolute number of train samples. If None, the value is automatically set to the complement of the test size.

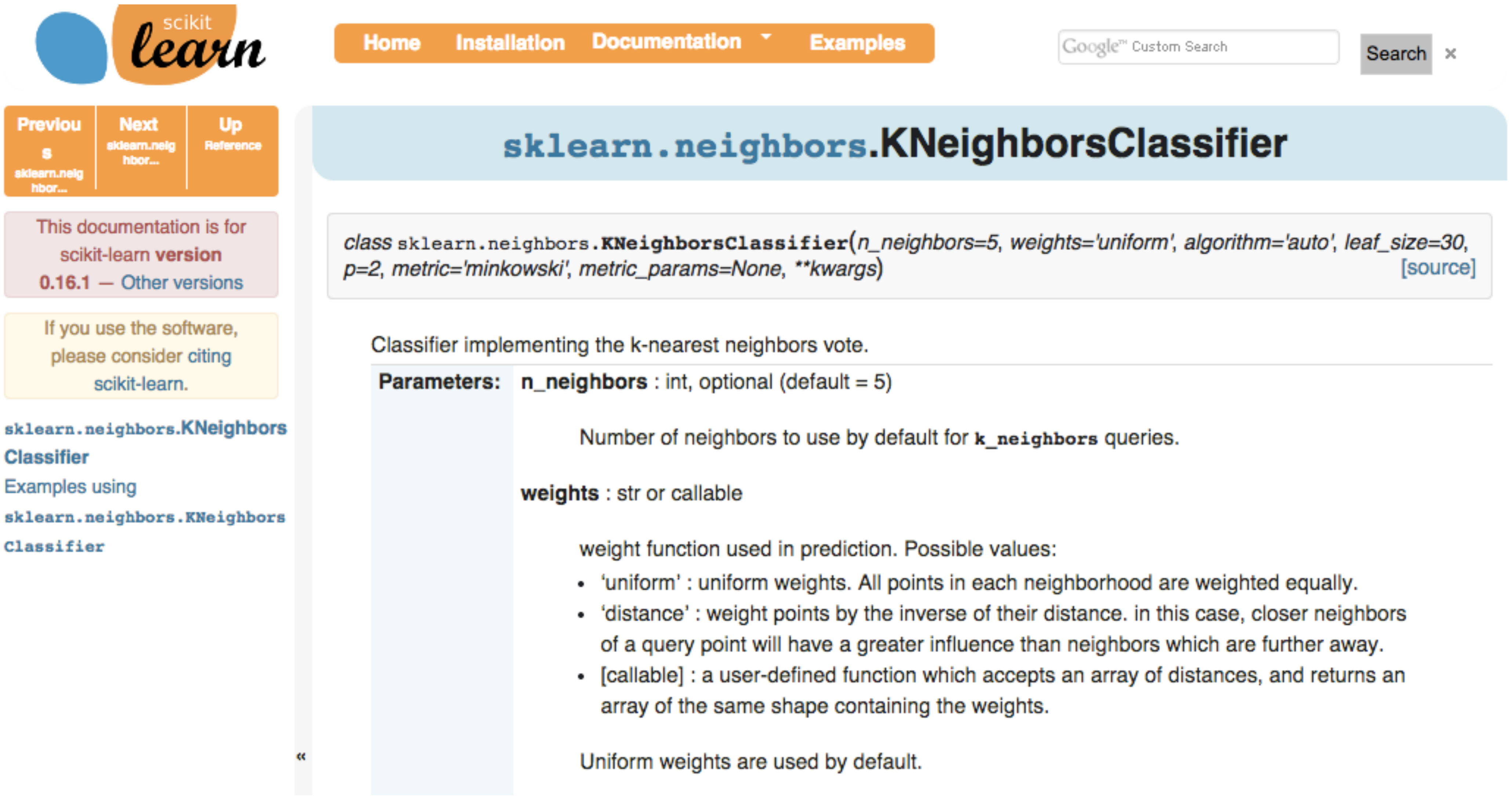
random_state : int or RandomState

Pseudo-random number generator state used for random sampling.

Returns:

splitting : list of arrays, length=2 * len(arrays)

List containing train-test split of input array.



This documentation is for scikit-learn **version 0.16.1** — Other versions

If you use the software, please consider citing scikit-learn.

`sklearn.neighbors`.KNeighborsClassifier

Examples using `sklearn.neighbors.KNeighborsClassifier`

`sklearn.neighbors`.KNeighborsClassifier

class `sklearn.neighbors.`**`KNeighborsClassifier`**(*n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, **kwargs*) [source]

Classifier implementing the k-nearest neighbors vote.

Parameters: **n_neighbors** : int, optional (default = 5)

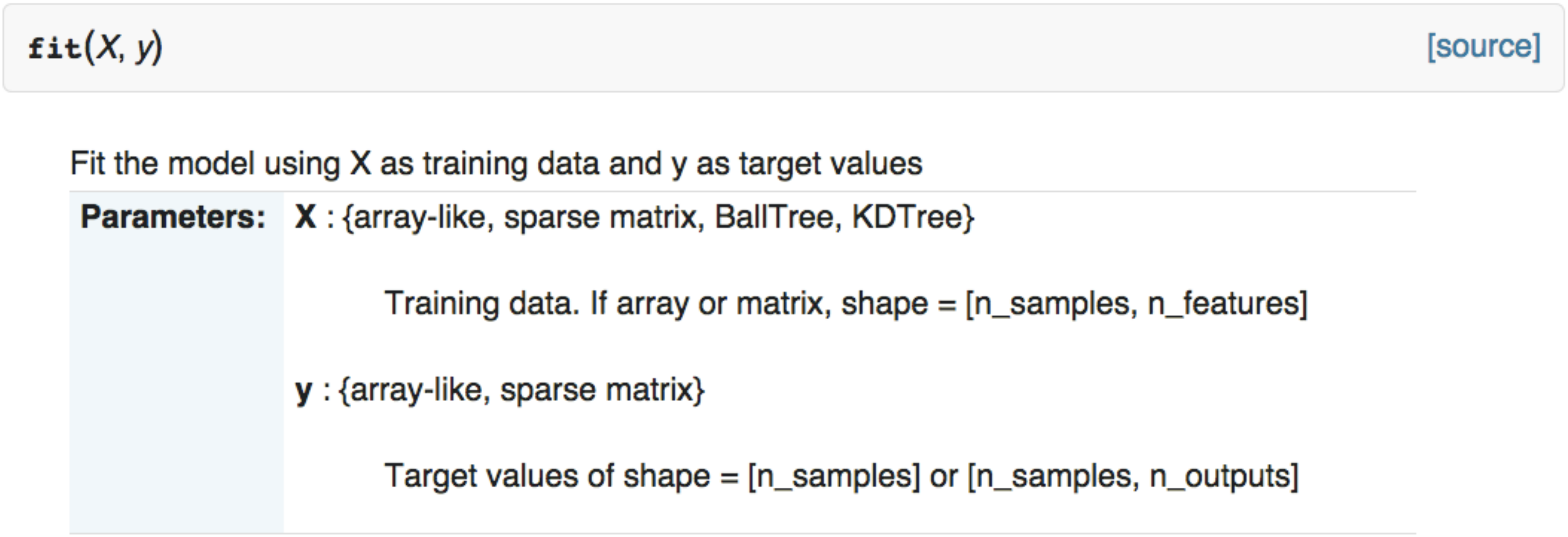
Number of neighbors to use by default for **`k_neighbors`** queries.

weights : str or callable

weight function used in prediction. Possible values:

- 'uniform' : uniform weights. All points in each neighborhood are weighted equally.
- 'distance' : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.
- [callable] : a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights.

Uniform weights are used by default.



fit(*X*, *y*) [source]

Fit the model using X as training data and y as target values

Parameters:	**X** : {array-like, sparse matrix, BallTree, KDTree}
	Training data. If array or matrix, shape = [n_samples, n_features]
	y : {array-like, sparse matrix}
	Target values of shape = [n_samples] or [n_samples, n_outputs]

predict(*X*) [source]

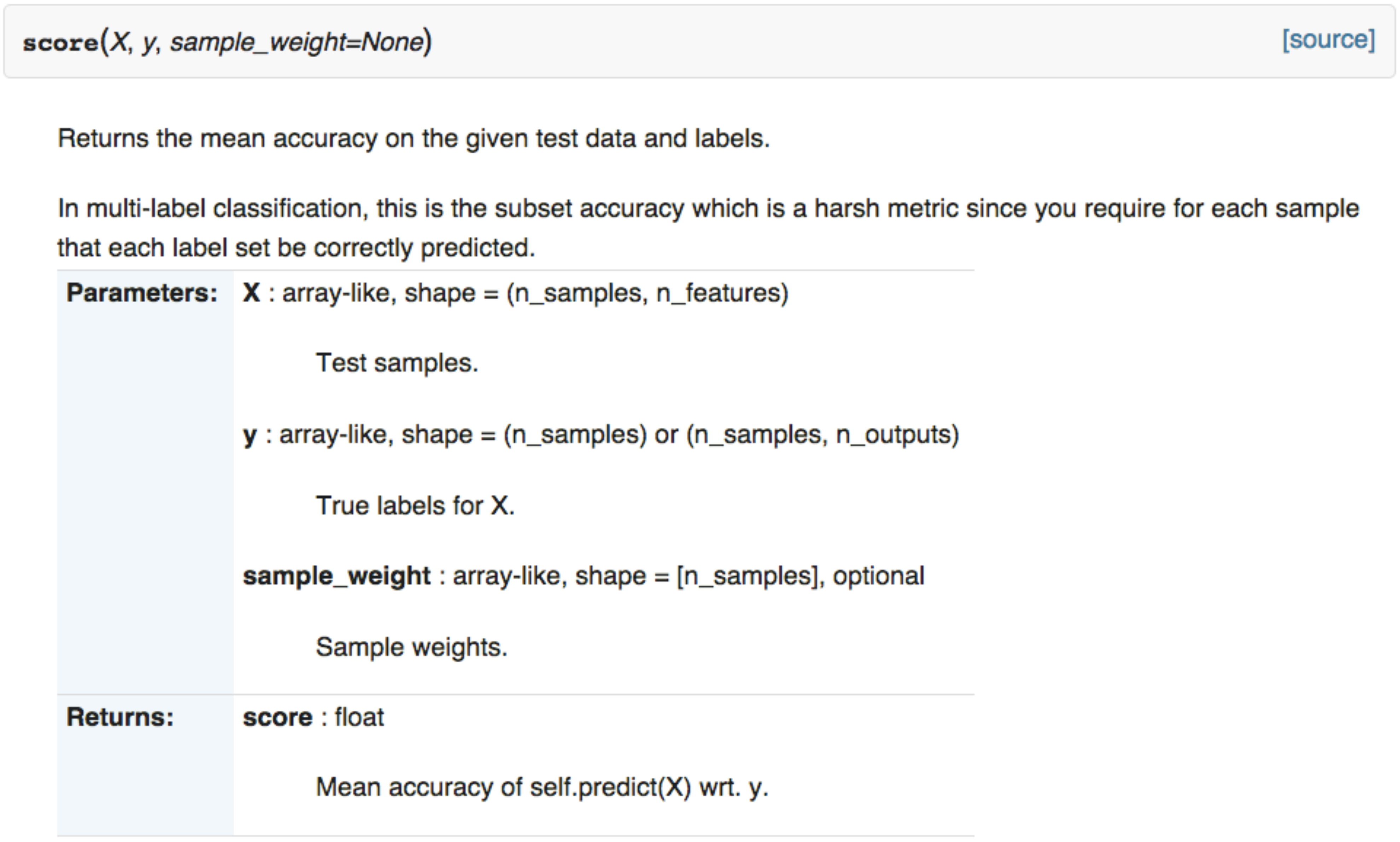
Predict the class labels for the provided data

Parameters:	**X** : array of shape [n_samples, n_features] A 2-D array representing the test points.
Returns:	**y** : array of shape [n_samples] or [n_samples, n_outputs] Class labels for each data sample.

predict_proba(*X*) [source]

Return probability estimates for the test data X.

Parameters:	**X** : array, shape = (n_samples, n_features) A 2-D array representing the test points.
Returns:	**p** : array of shape = [n_samples, n_classes], or a list of n_outputs of such arrays if n_outputs > 1. The class probabilities of the input samples. Classes are ordered by lexicographic order.



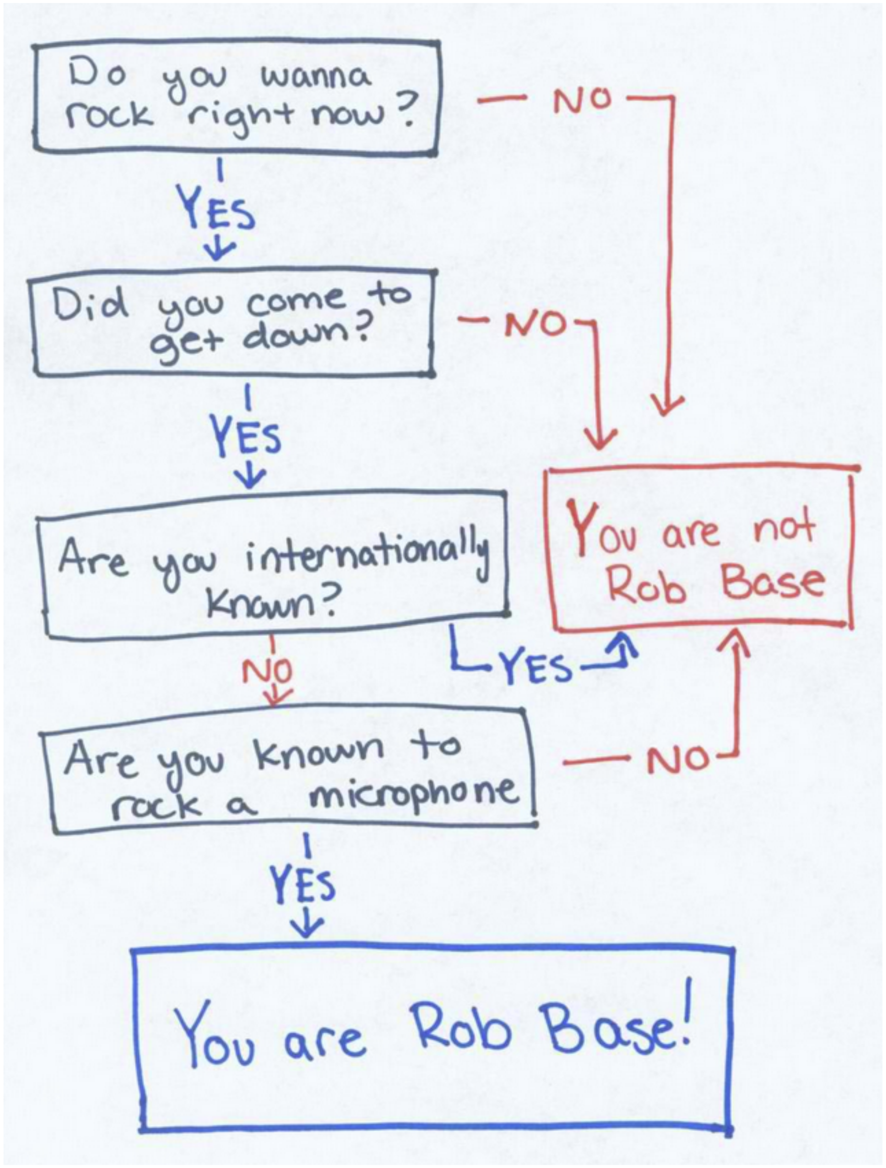
score(*X, y, sample_weight=None*) [source]

Returns the mean accuracy on the given test data and labels.

In multi-label classification, this is the subset accuracy which is a harsh metric since you require for each sample that each label set be correctly predicted.

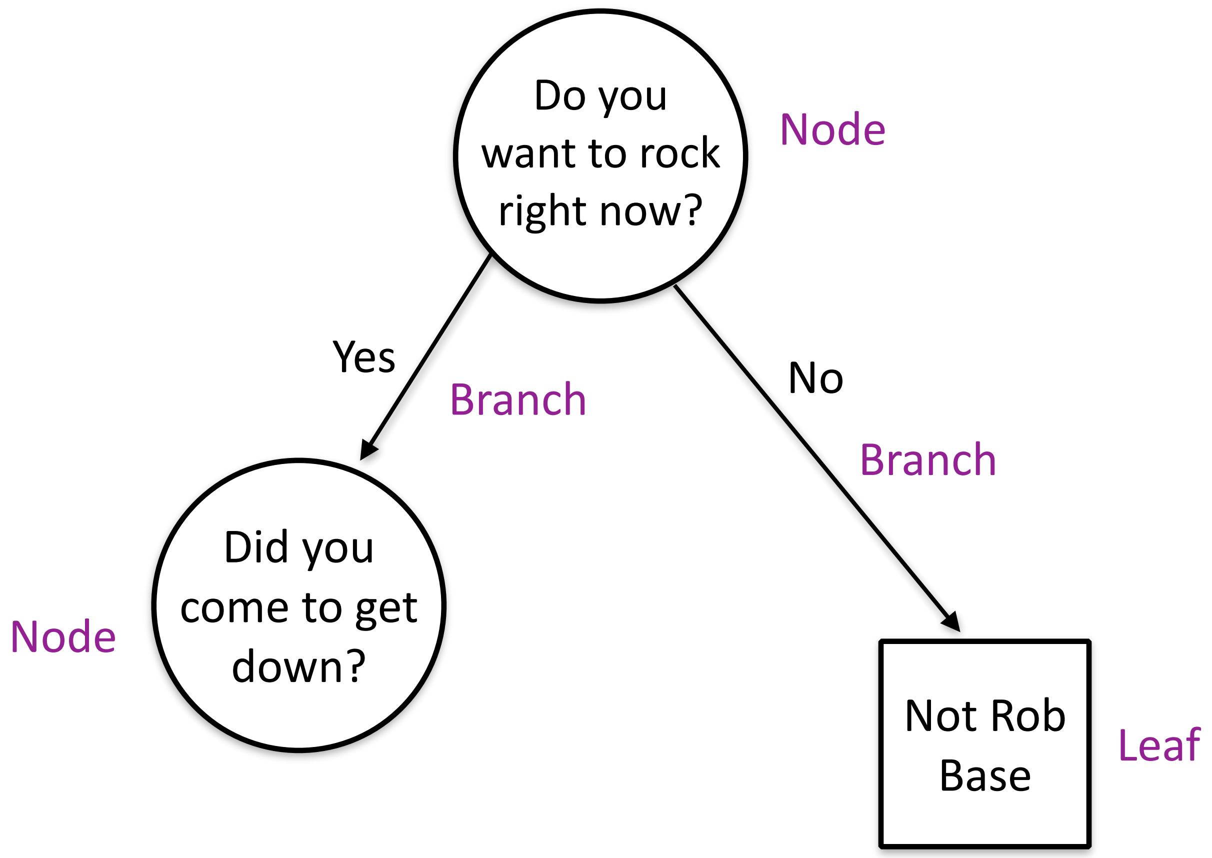
Parameters:	**X** : array-like, shape = (n_samples, n_features) Test samples. **y** : array-like, shape = (n_samples) or (n_samples, n_outputs) True labels for X. **sample_weight** : array-like, shape = [n_samples], optional Sample weights.
Returns:	**score** : float Mean accuracy of self.predict(X) wrt. y.

Is This a Decision Tree?


Do you wanna rock right now?
NO
YES
Did you come to get down?
NO
YES
Are you internationally known?
YES
NO
You are not Rob Base
Are you known to rock a microphone
NO
YES
You are Rob Base!

Decision Trees

- Trees consist of:
  - Nodes (questions)
  - Branches (answers)
  - Leaves (end points)
- Acyclic - flows in one direction
- No split is necessary when all records are from same class



Decision Trees

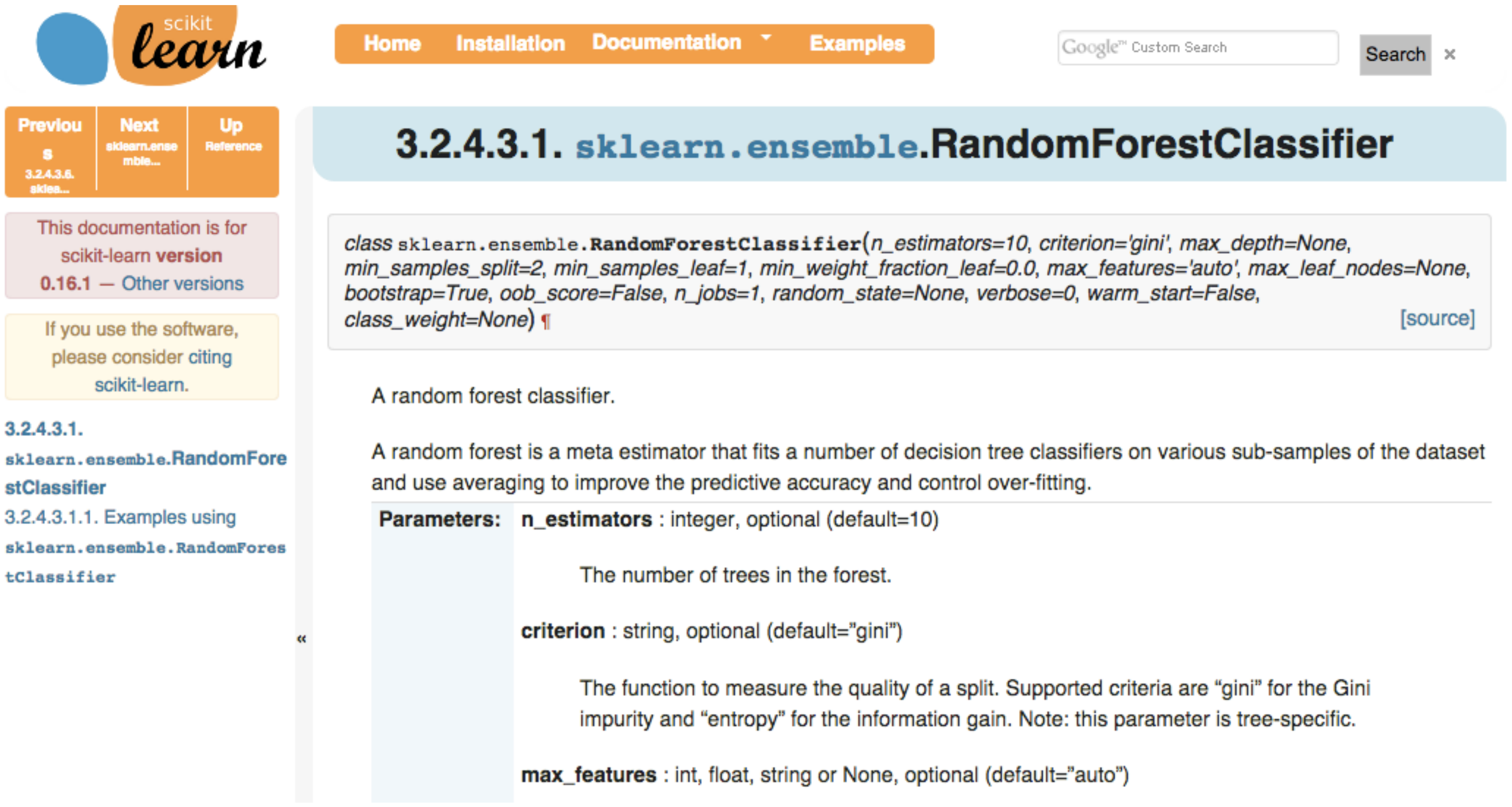
- Algorithm selects optimal node that creates the largest increase in purity
- Decision trees are susceptible to overfitting
- Techniques for preventing overfitting
    - Minimum number of records for leaf
    - Maximum depth for branches
- One technique is to purposely overfit, but manually prune branches

Random Forest

- Ensemble algorithm (mix of many models)
- Collection of decision trees
- Evaluates predictions from many models and selects the most common classification
- Features are randomly selected for each decision tree
- Bagging (Bootstrap Aggregating) is also applied to each tree
    - Sample of the data set is used for training
    - Samples are drawn with replacement

Why Random Forest Is Popular?

- Add all your data and algorithm will prioritize
- Not susceptible to overfitting
- Doesn’t require normalization
- Solid performance in wide range of applications



This documentation is for scikit-learn **version 0.16.1** — Other versions

If you use the software, please consider citing scikit-learn.

3.2.4.3.1. `sklearn.ensemble.RandomForestClassifier`

3.2.4.3.1.1. Examples using `sklearn.ensemble.RandomForestClassifier`

3.2.4.3.1. `sklearn.ensemble`.RandomForestClassifier

class `sklearn.ensemble.`**`RandomForestClassifier`**(*n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None*) ¶ [source]

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

Parameters: **n_estimators** : integer, optional (default=10)

The number of trees in the forest.

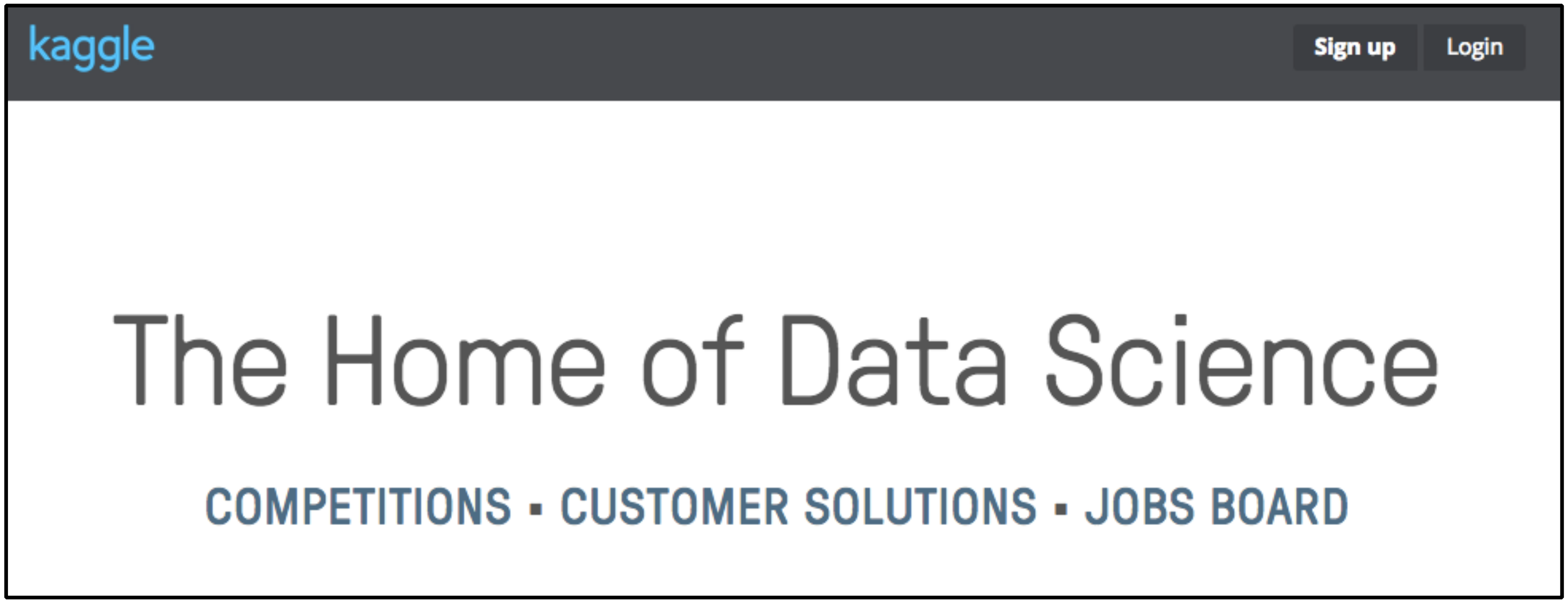
criterion : string, optional (default="gini")

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

max_features : int, float, string or None, optional (default="auto")

Kaggle

- Data prediction competitions



The Home of Data Science

COMPETITIONS ▪ CUSTOMER SOLUTIONS ▪ JOBS BOARD

Kaggle

- Create an account
- Find “Titanic: Machine Learning from Disaster”
- Submission Instructions:

  “You should submit a csv file with exactly 418 entries plus a header row. This must have exactly 2 columns: PassengerId (which can be sorted in any order), and Survived which contains your binary predictions: 1 for survived, 0 for did not.”

Make Predictions

- Read the test.csv file
- Clean test data
- Train your machine learning algorithm with training data
- Predict outcomes for test data
- Convert predictions into a DataFrame
- Save DataFrame as a cvs file (remember to exclude the index)
- Submit predictions to Kaggle site

What We've Accomplished

- Built foundation of Python skills
- Acquired and stored data in variety of formats (e.g., csv, Excel, SQL)
- Explored and summarized data
- Implemented a machine learning algorithm

Coding Best Practices

PEP-8 Style Guide

- https://www.python.org/dev/peps/pep-0008/
- maximum line length of 79 characters
- indentation
- line continuation
- commenting

Hard to Use at First But Review Once a Month and You'll Incrementally Improve

Next Steps

- Practice, Practice, Practice
  - Find a fun project
  - Lots of public data sets and web data to scrape
  - Pair programming buddy
- Join a Python Meetup
- Review Python
  - Code Academy
  - Think Python
  - Python for Data Analysis

Next Steps

- Keep Learning
  - Pycon and Pydata videos on YouTube
  - Python for Data Analysis (O’Reilly)
  - Machine Learning Classes (Coursera, Edx)
  - datasciencemasters.org
  - Kaggle competitions
- Don’t Be Afraid to Ask for Help
- Keep in Touch