[Scikit-learn](http://scikit-learn.org/stable/) is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib!

As you build robust Machine Learning programs, it's helpful to have all the sklearncommands all in one place in case you forget.

### **[Linear Regression](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html)**

**Import and create the model:**

from sklearn.linear\_model import LinearRegression  
  
your\_model = LinearRegression()

**Fit:**

your\_model.fit(x\_training\_data, y\_training\_data)

* .coef\_: contains the coefficients
* .intercept\_: contains the intercept

**Predict:**

predictions = your\_model.predict(your\_x\_data)

* .score(): returns the coefficient of determination R²

### **[Naive Bayes](http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html#sklearn.naive_bayes.MultinomialNB))**

**Import and create the model:**

from sklearn.naive\_bayes import MultinomialNB  
  
your\_model = MultinomialNB()

**Fit:**

your\_model.fit(x\_training\_data, y\_training\_data)

**Predict:**

# Returns a list of predicted classes - one prediction for every data point  
predictions = your\_model.predict(your\_x\_data)  
  
# For every data point, returns a list of probabilities of each class  
probabilities = your\_model.predict\_proba(your\_x\_data)

### **[K-Nearest Neighbors](http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier)**

**Import and create the model:**

from sklearn.neigbors import KNeighborsClassifier  
  
your\_model = KNeighborsClassifier()

**Fit:**

your\_model.fit(x\_training\_data, y\_training\_data)

**Predict:**

# Returns a list of predicted classes - one prediction for every data point  
predictions = your\_model.predict(your\_x\_data)  
  
# For every data point, returns a list of probabilities of each class  
probabilities = your\_model.predict\_proba(your\_x\_data)

### **[K-Means](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html)**

**Import and create the model:**

from sklearn.cluster import KMeans  
  
your\_model = KMeans(n\_clusters=4, init='random')

* n\_clusters: number of clusters to form and number of centroids to generate
* init: method for initialization
  + k-means++: K-Means++ [default]
  + random: K-Means
* random\_state: the seed used by the random number generator [optional]

**Fit:**

your\_model.fit(x\_training\_data)

**Predict:**

predictions = your\_model.predict(your\_x\_data)

### **[Validating the Model](http://scikit-learn.org/stable/modules/classes.html#sklearn-metrics-metrics)**

**Import and print accuracy, recall, precision, and F1 score:**

from sklearn.metrics import accuracy\_score, recall\_score, precision\_score, f1\_score  
  
print(accuracy\_score(true\_labels, guesses))  
print(recall\_score(true\_labels, guesses))  
print(precision\_score(true\_labels, guesses))  
print(f1\_score(true\_labels, guesses))

**Import and print the confusion matrix:**

from sklearn.metrics import confusion\_matrix  
  
print(confusion\_matrix(true\_labels, guesses))

### **[Training Sets and Test Sets](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)**

from sklearn.model\_selection import train\_test\_split  
  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, train\_size=0.8, test\_size=0.2)

* train\_size: the proportion of the dataset to include in the train split
* test\_size: the proportion of the dataset to include in the test split
* random\_state: the seed used by the random number generator [optional]