### Scikit-learn

Scikit-learn is an open source Python library that implements a range of machine learning, preprocessing, cross-validation and visualization algorithms using a unified interface.

#### A Basic Example

```python
>>> from sklearn import neighbors, datasets, preprocessing
>>> from sklearn.cross_validation import train_test_split
>>> y = np.array([1, 2, 2, 3, 3, 4])
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
>>> knn.fit(X_train, y_train)
>>> y_pred = knn.predict(X_test)
```

### Preprocessing The Data

#### Encoding Categorical Features

```python
>>> from sklearn.preprocessing import LabelEncoder
>>> enc = LabelEncoder()
>>> y = enc.fit_transform(y)
```

#### Normalization

```python
>>> from sklearn.preprocessing import Normalizer
>>> scaler = Normalizer().fit(X_train)
>>> normalized_X = scaler.transform(X_train)
```

#### Binarization

```python
>>> from sklearn.preprocessing import Binarizer
>>> binarizer = Binarizer(threshold=0.0).fit(X)
>>> binary_X = binarizer.transform(X)
```

### Supervised Learning Estimators

#### Linear Regression

```python
>>> from sklearn.linear_model import LinearRegression
>>> lr = LinearRegression(normalize=True)
```

#### Support Vector Machines (SVM)

```python
>>> from sklearn.svm import SVC
>>> svc = SVC(kernel='linear')
```

#### Naive Bayes

```python
>>> from sklearn.naive_bayes import GaussianNB
>>> gnb = GaussianNB()
```

#### KNN

```python
>>> from sklearn.neighbors import KNeighborsClassifier
>>> knn = KNeighborsClassifier(n_neighbors=5)
```

#### Naive Bayes

```python
>>> from sklearn.naive_bayes import GaussianNB
>>> gnb = GaussianNB()
```

#### Standardization

```python
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().fit(X_train)
>>> standardized_X = scaler.transform(X_train)
```

#### Principal Component Analysis (PCA)

```python
>>> from sklearn.decomposition import PCA
>>> pca = PCA(n_components=0.95)
```

#### K Means

```python
>>> from sklearn.cluster import KMeans
>>> kmeans = KMeans(n_clusters=3, random_state=0)
```

### Unsupervised Learning Estimators

#### Supervised fitting

```python
>>> lr.fit(X, y)
>>> knn.fit(X_train, y_train)
```

#### Unsupervised Learning

```python
>>> from sklearn.cluster import KMeans
>>> kmeans = KMeans(n_clusters=3, random_state=0)
>>> kmeans.fit(X)
```

### Prediction

```python
>>> y_pred = knn.predict(X_test)
```

### Evaluation Your Model’s Performance

#### Accuracy Score

```python
>>> from sklearn.metrics import accuracy_score
>>> accuracy_score(y_test, y_pred)
```

#### Mean Absolute Error

```python
>>> from sklearn.metrics import mean_absolute_error
>>> mean_absolute_error(y_true, y_pred)
```

#### Mean Squared Error

```python
>>> from sklearn.metrics import mean_squared_error
>>> mean_squared_error(y_true, y_pred)
```

#### R² Score

```python
>>> from sklearn.metrics import r2_score
>>> r2_score(y_true, y_pred)
```

#### Adjusted Rand Index

```python
>>> from sklearn.metrics import adjusted_rand_score
>>> adjusted_rand_score(y_true, y_pred)
```

#### V-measure

```python
>>> from sklearn.metrics import v_measure_score
>>> v_measure_score(y_true, y_pred)
```

#### Cross-Validation

```python
>>> from sklearn.cross_validation import cross_val_score
>>> cross_val_score(knn, X_train, y_train, cv=4)
```

#### Tune Your Model

#### Grid Search

```python
>>> from sklearn.grid_search import GridSearchCV
>>> grid = GridSearchCV(knn, param_grid=params)
>>> grid.fit(X_train, y_train)
```

#### Randomized Parameter Optimization

```python
>>> from sklearn.grid_search import RandomizedSearchCV
>>> rsearch = RandomizedSearchCV(knn, params=params, n_iter=8, cv=4, n_jobs=1, random_state=5)
>>> rsearch.fit(X_train, y_train)
>>> print(rsearch.best_score_)
>>> print(rsearch.best_estimator_.n_neighbors)
```

### DataCamp