

Machine Learning

Supervised Learning	Unsupervised learning
The model maps input to an output based on the previous input-output pairs	No training is given to the model and it has to discover the features of input by self training mechanism.

Scikit learn can be used in Classification, Regression, Clustering, Dimensionality reduction, Model Selection and preprocessing by supervised and unsupervised training models.

Basic Commands

```
>>> from sklearn import neighbors, datasets, preprocessing
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.metrics import accuracy_score
>>> iris = datasets.load_iris()
>>> X, y = iris.data[:, :2], iris.target
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=33)
>>> scaler = preprocessing.StandardScaler().fit(X_train)
>>> X_train = scaler.transform(X_train)
>>> X_test = scaler.transform(X_test)
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
>>> knn.fit(X_train, y_train)
>>> y_pred = knn.predict(X_test)
>>> accuracy_score(y_test, y_pred)
```

Loading Data example

```
>>> import numpy as np
>>> X = np.random.random((20,2))
>>> y = np.array(['A','B','C','D','E','F','G','-A','C','A','B'])
>>> X[X < 0.7] = 0
```

The data being loaded should be numeric and has to be stored as NumPy arrays or SciPy sparse matrices.

Processing Loaded Data

Standardization	Normalization	Binarization
<pre>>>> from sklearn.preprocessing import StandardScaler >>> scaler = StandardScaler().fit(X_train) >>> standardized_X = scaler.transform(X_train) >>> standardized_X_test = scaler.transform(X_test)</pre>	<pre>>>> from sklearn.preprocessing import Normalizer >>> scaler = Normalizer().fit(X_train) >>> normalized_X = scaler.transform(X_train) >>> normalized_X_test = scaler.transform(X_test)</pre>	<pre>>>> from sklearn.preprocessing import Binarizer >>> binarizer = Binarizer(threshold=0.0).fit(X) >>> binary_X = binarizer.transform(X) >>> binary_X_test = binarizer.transform(X_test)</pre>

Training And Test Data

```
>>> from sklearn.model_selection import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=0)
```

Creating Model

Supervised Learning Estimators

Linear Regression	Support Vector Machines (SVM)	Naive Bayes
-------------------	-------------------------------	-------------

```
>>> from sklearn.linear_model import LinearRegression
>>> from sklearn.svm import SVC
>>> from sklearn.naive_bayes import GaussianNB
>>> lr = LinearRegression(normalize=True)
>>> svc = SVC(kernel='linear')
>>> gnb = GaussianNB()
```

Creating Model

Unsupervised Learning Estimators

Principal Component Analysis (PCA)	K Means
<pre>>>> from sklearn.decomposition import PCA >>> pca = PCA(n_components=0.95)</pre>	<pre>>>> from sklearn.cluster import KMeans >>> k_means = KMeans(n_clusters=3, random_state=0)</pre>

Model Fitting

Supervised Learning	Unsupervised learning
<pre>>>> lr.fit(X, y) >>> knn.fit(X_train, y_train) >>> svc.fit(X_train, y_train)</pre>	<pre>>>> k_means.fit(X_train) >>> pca_model = pca.fit_transform(X_train)</pre>



By **Manasa**
cheatography.com/manasa/

Not published yet.
Last updated 30th March, 2020.
Page 1 of 2.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>

Predicting output

Supervised Estimators	Unsupervised Estimators
<pre>>>> y_pred = svc.predict(np.random.random((-2,5)))</pre>	<pre>>>> y_pred = k_means.predict(X_test)</pre>
<pre>>>> y_pred = lr.predict(X_test)</pre>	
<pre>>>> y_pred = knn.predict_proba(X_test)</pre>	

Classification Metrics Model Performance

Accuracy Score	Classification Report	Confusion Matrix
<pre>>>> knn.score(X_test, y_test)</pre>	<pre>>>> from sklearn.metrics import classification_report</pre>	<pre>>>> from sklearn.metrics import confusion_matrix</pre>
<pre>>>> from sklearn.metrics import accuracy_score</pre>	<pre>>>> print(classification_report(y_test, y_pred))</pre>	<pre>>>> print(confusion_matrix(y_test, y_pred))</pre>
<pre>>>> accuracy_score(y_test, y_pred)</pre>		

Clustering Metrics Model Performance

Adjusted Rand Index	Homogeneity	Cross-Validation
<pre>>>> from sklearn.metrics import adjusted_rand_score</pre>	<pre>>>> from sklearn.metrics import homogeneity_score</pre>	<pre>>>> print(cross_val_score(knn, X_train, y_train, cv=4))</pre>
<pre>>>> adjusted_rand_score(y_true, y_pred)</pre>	<pre>>>> homogeneity_score(y_true, y_pred)</pre>	<pre>>>> print(cross_val_score(lr, X, y, cv=2))</pre>

