Cheatography

Scikit-Learn Python Cheat Sheet

by Manasa via cheatography.com/121399/cs/22207/

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 Classi¬fic¬ation, Regres¬sion, Cluste¬ring, Dimens¬ion¬ality reduct¬ion¬,Model Selection and prepro¬cessing by supervised and unsupe¬rvised 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
>>> from sklearn.preprocessing import StandardScaler	>>> from sklearn.preprocessing import Normalizer	>>> from sklearn.preprocessing import Binarizer
>>> scaler = StandardScaler().fit(X_train)	>>> scaler = Normalizer().fit(X_train)	>>> binarizer = Binarizer(threshold=0.0).fit(X)
>>> standardized_X = scaler.transform(X_train)	>>> normalized_X = scaler.transform(X_train)	>>> binary_X = binarizer.transform(X)
>>> standardized_X_test = scaler.transform(X_test)	>>> normalized_X_test = scaler.transform(X_test)	

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
>>> from sklearn.decomposition import PCA	>>> from sklearn.cluster import KMeans
>>> pca = PCA(n_components=0.95)	>>> k_means = KMeans(n_clusters=3, random_state=0)

Model Fitting

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

Predicting output

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

Classification Metrics Model Performance

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

Clustering Metrics Model Performance

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