کتابخانه Scikit-Learn

معرفی کتابخانه Scikit-Learn

«سایکیت‌لِرن» (Scikit-learn)، یک کتابخانه «متن‌باز» (Open Source) برای زبان پایتون است که طیفی از الگوریتم‌های «یادگیری ماشین» (Machine Learning)، «پیش‌پردازش داده‌ها» (Data Pre-Processing)، «اعتبارسنجی متقابل» (Cross Validation) و «بصری‌سازی» (Visualization) را با استفاده از یک رابط یکپارچه پیاده‌سازی می‌کند.

یک مثال پایه‌ای

```
>>> 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)
```

بارگذاری داده‌ها (Loading The Data)

داده‌ها باید به صورت عددی و آرایه‌های NumPy یا ماتریس خلوت SciPy ذخیره شده باشند. دیگر انواعی که قابل تبدیل به آرایه‌های عددی هستند، مانند دیتافریم Pandas نیز قابل پذیرش هستند.

```
>>> import numpy as np
>>> X = np.random.random((10,5))
>>> y = np.array(['M','M','F','F','M','F','M','M','F','F','F'])
>>> X[X < 0.7] = 0
```

داده‌های آموزش و آزمون (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)
```

پیش‌پردازش داده‌ها (Preprocessing The Data)

استانداردسازی (Standardization)

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

رمزنگاری ویژگی‌های طبقه‌ای (Encoding Categorical Features)

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

نرمال‌سازی (Normalization)

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

جایگذاری مقادیر ناموجود (Missing Values)

```
>>> from sklearn.preprocessing import Imputer
>>> imp = Imputer(missing_values=0, strategy='mean', axis=0)
>>> imp.fit_transform(X_train)
```

دودویی‌سازی (Binarization)

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

ساخت ویژگی‌های چند جمله‌ای (Polynomial Features)

```
>>>from sklearn.preprocessing import PolynomialFeatures
>>> poly = PolynomialFeatures(5)
>>> poly.fit_transform(X)
```

ساخت مدل

برآوردگرهای یادگیری نظارت شده (Supervised Learning Estimators)

رگرسیون خطی (Linear Regression)

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

ماشین بردار پشتیبان (Support Vector Machines | SVM)

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

نایو بیز (Naive Bayes)

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

K نزدیک‌ترین همسایگی (K-Nearest Neighbors | KNN)

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

برآوردگرهای یادگیری نظارت نشده (Unsupervised Learning Estimators)

تحلیل مولفه اساسی (Principal Component Analysis)

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

K میانگین (K Means)

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

برازش مدل (Model Fitting)

یادگیری نظارت شده (Supervised Learning)	
>>> lr.fit(X, y)	برازش مدل برای داده‌ها
>>> knn.fit(X_train, y_train)	
یادگیری نظارت نشده (Unsupervised Learning)	
>>> k_means.fit(X_train)	برازش مدل برای داده‌ها
>>> pca_model = pca.fit_transform(X_train)	برازش برای داده‌ها و سپس تبدیل آن

پیش‌بینی (Prediction)

برآوردگرهای نظارت شده	
>>> y_pred = svc.predict(np.random.random((2,5)))	پیش‌بینی برچسب‌ها
>>> y_pred = lr.predict(X_test)	پیش‌بینی برچسب‌ها
>>> y_pred = knn.predict_proba(X_test)	تخمین احتمال یک برچسب
برآوردگرهای نظارت نشده	
>>> y_pred = k_means.predict(X_test)	پیش‌بینی برچسب‌ها در الگوریتم‌های خوشه‌بندی

ارزیابی کارایی مدل (Model Performance Evaluation)

سنجه‌های دسته‌بندی (Classification Metrics)

امتیاز صحت (Accuracy Score)	
>>> knn.score(X_test, y_test)	روش امتیاز برآوردگر
>>> from sklearn.metrics import accuracy_score	توابع امتیازدهی متریک
>>> accuracy_score(y_test, y_pred)	
گزارش دسته‌بندی (Classification Report)	
>>> from sklearn.metrics import classification_report	«صحت» (Precision)، «دقت» (Recall)، f_1-Score و «پشتیبان» (Support)
>>> print(classification_report(y_test, y_pred))	
ماتریس درهم‌ریختگی (Confusion Matrix)	
>>> from sklearn.metrics import confusion_matrix	
>>> print(confusion_matrix(y_test, y_pred))	

سنجه‌های رگرسیون (Regression Metrics)

امتیاز R^2 (R^2 Score)

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

میانگین قدر مطلق خطا (Mean Absolute Error)

```
>>> from sklearn.metrics import mean_absolute_error
>>> y_true = [3, -0.5, 2]
>>> mean_absolute_error(y_true, y_pred)
```

میانگین خطای مربعات (Mean Squared Error)

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

سنجه‌های خوشه‌بندی (Clustering Metrics)

اندیس تصادفی شده Rand (Adjusted Rand Index)

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

همگن بودن (Homogeneity)

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

اندازه V (V-measure)

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

اعتبارسنجی متقابل (Cross-Validation)

```
>>> from sklearn.cross_validation import cross_val_score
>>> print(cross_val_score(knn, X_train, y_train, cv=4))
>>> print(cross_val_score(lr, X, y, cv=2))
```

تنظیم مدل (Model Tuning)

جست‌وجوی شبکه‌ای (Grid Search)

```
>>> from sklearn.grid_search import GridSearchCV
>>> params = {"n_neighbors": np.arange(1,3),
"metric": ["euclidean", "cityblock"]}
>>> grid = GridSearchCV(estimator=knn,
param_grid=params)
>>> grid.fit(X_train, y_train)
>>> print(grid.best_score_)
>>> print(grid.best_estimator_.n_neighbors)
```

بهینه‌سازی پارامتر تصادفی شده (Randomized Parameter Optimization)

```
>>> from sklearn.grid_search import RandomizedSearchCV
>>> params = {"n_neighbors": range(1,5),
"weights": ["uniform", "distance"]}
>>> rsearch = RandomizedSearchCV(estimator=knn,
param_distributions=params,
cv=4,
n_iter=8,
random_state=5)
>>> rsearch.fit(X_train, y_train)
>>> print(rsearch.best_score_)
```

مجموعه آموزش‌های داده‌کاوی فرادرس (+کلیک کنید)

برای مشاهده دیگر «تقلب‌نامه‌های» مجله فرادرس، به این لینک مراجعه فرمایید.

جهت آگاهی از آخرین تقلب‌نامه‌های منتشر شده، در کانال تلگرام مجله فرادرس عضو شوید.

تهیه و تنظیم: مجله فرادرس

منبع

برای مشاهده دیگر «تقلب‌نامه‌های» مجله فرادرس، به این لینک مراجعه فرمایید.