Scikit-learn

Scikit-learn is an open-source, free Python library. It facilitates activities such as classifying data, clustering similar data, forecasting values, and simplifying data for tasks like dimensionality reduction. Additionally, it gives you the skills to prepare data, select the optimal model, and assess performance. Scikit-learn, which is built on top of existing Python libraries like NumPy and SciPy, is easy to use, popular, and perfect for both novices and machine learning specialists.

Installing Scikit-Learn

```
pip install scikit-learn
```

Data Preprocessing

Function	Description
StandardScaler	Standardize features by removing the mean and scaling to unit variance.
MinMaxScaler	Scale features to a specific range (e.g., 0 to 1).
Binarizer	Transform features into binary values (thresholding).
LabelEncoder	Encode target labels with values between 0 and n_classes-1.
OneHotEncoder	Perform one-hot encoding of categorical features.
PolynomialFeatures	Generate polynomial and interaction features.
SimpleImputer	Impute missing values using a strategy (mean, median, most frequent).
KNNImputer	Impute missing values using k-nearest neighbors.

Clustering Models

Function	Description
KMeans	A popular clustering algorithm that partitions data into k distinct clusters based on similarity.
DBSCAN	A density-based clustering algorithm that groups data points based on density, allowing for irregular shapes.
AgglomerativeClustering	A hierarchical clustering method that builds clusters iteratively by merging or splitting clusters.

Model Selection and Evaluation

Function	Description
train_test_split	Split data into training and testing sets
cross_val_score	Perform cross-validation on the model.
cross_val_predict	Cross-validation generator for predictions.
accuracy_score	Evaluate classification accuracy.
confusion_matrix	Generate confusion matrix for classification.
classification_report	Detailed classification report (precision, recall, F1-score).
mean_squared_error	Evaluate regression performance with mean squared error.
r2_score	Evaluate regression performance with R^2 score.
roc_auc_score	Compute area under the ROC curve for binary classification.
f1_score	Compute the F1 score for classification models.
precision_score	Compute precision score for classification models.
recall_score	Compute recall score for classification models.

Model Selection and Evaluation

Function	Description
LogisticRegression	A linear model used for binary or multi-class classification.
SVC	Support Vector Classifier, used for both linear and non-linear classification.
RandomForestClassifier	An ensemble method that builds multiple decision trees for robust classification.
GradientBoostingClassifier	An ensemble method that builds trees sequentially to correct errors of previous trees.
GaussianNB	Naive Bayes classifier based on Gaussian distribution of data.
KNeighborsClassifier	Classifier that assigns labels based on nearest neighbors' majority class.
DecisionTreeClassifier	A tree-based classifier that splits data into branches to make decisions.

Regression Models

Function	Description
LinearRegression	A linear model used to predict continuous numerical values.
Ridge	A linear regression model with L2 regularization to prevent overfitting.
Lasso	A linear regression model with L1 regularization to enhance sparsity.
DecisionTreeRegressor	A tree-based model that predicts continuous values by learning splits on the data.
RandomForestRegressor	An ensemble method that averages the predictions of multiple decision trees for better accuracy.
SVR	Support Vector Regressor, used for predicting continuous values with support vector machines.

Dimentionality Reduction

Function	Description
PCA	Principal Component Analysis (PCA) reduces the number of features by finding new dimensions that maximize variance.
TruncatedSVD	A dimensionality reduction method suited for sparse matrices, especially in text mining.
t-SNE	A technique for visualizing high-dimensional data by mapping it to a lower-dimensional space.
FeatureAgglomeration	A method for feature reduction that merges features based on their similarity.

Model Training and Prediction

Function	Description
fit()	Trains the model using the provided data (X_train, y_train).
predict()	Makes predictions based on the trained model for unseen data (X_test).
fit_predict()	Combines training and prediction into a single method, commonly used in clustering.
predict_proba()	Returns probability estimates for classification models, indicating class likelihoods.
score()	Evaluates the model's performance using a scoring metric, typically accuracy for classification or R^2 for regression.

Optimizing Models

Exhaustive Search with GridSearchCV

```
from sklearn.model_selection import GridSearchCV
param_grid = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']}
grid = GridSearchCV(SVC(), param_grid, cv=5)
grid.fit(X_train, y_train)
print("Best Parameters:", grid.best_params_)
```

Randomized Search for Hyperparameters

```
from sklearn.model_selection import RandomizedSearchCV
random = RandomizedSearchCV(SVC(), param_distributions={
        'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']}, n_iter=10)
random.fit(X_train, y_train)
```

Data Transformation Techniques

Standardization

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(X_train)
X_train_standardized = scaler.transform(X_train)
X_test_standardized = scaler.transform(X_test)
```

Normaliziation

```
from sklearn.preprocessing import Normalizer
scaler = Normalizer().fit(X_train)
X_train_normalized = scaler.transform(X_train)
X_test_normalized = scaler.transform(X_test)
```

Binarization

```
from sklearn.preprocessing import Binarizer
binarizer = Binarizer(threshold=1.0).fit(X)
X_binarized = binarizer.transform(X)
```

Encoding Non-Numerical Data

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
y_encoded = encoder.fit_transform(y)
```

Handling Missing Values:

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(X)
```

Creating Polynomial Features

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)
```

Importing and Preparing Data

Loading Built-in Datasets

```
from sklearn.datasets import load_iris
data = load_iris()
X, y = data.data, data.target
```

Splitting Data into Training and Testing

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
```

Metrics for Classification Models

Accuracy Score

```
from sklearn.metrics import accuracy_score
print("Accuracy:", accuracy_score(y_test, y_pred))
```

Classification Report (Precision, Recall, F1)

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

Confusion Matrix Insights

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_pred))
```

Supervised Learning Algorithms

Linear Regression

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Support Vector Machines (SVM)

```
from sklearn.svm import SVC
model = SVC(kernel='linear')
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

K-Nearest Neighbors

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Unsupervised Learning Algorithms

PCA

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_train)
```

K-Means

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X_train)
y_pred = kmeans.predict(X_test)
```

Metrics for Regression Models

Mean Absolute Error (MAE)

```
from sklearn.metrics import mean_absolute_error
print("MAE:", mean_absolute_error(y_test, y_pred))
```

Mean Squared Error (MSE)

```
from sklearn.metrics import mean_squared_error
print("MSE:", mean_squared_error(y_test, y_pred))
```

R^2 Score

```
from sklearn.metrics import r2_score
print("R² Score:", r2_score(y_test, y_pred))
```

Metrics for Clustering

Adjusted Rand Index

```
from sklearn.metrics import adjusted_rand_score
print("ARI:", adjusted_rand_score(y_test, y_pred))
```

Homogeneity

```
from sklearn.metrics import homogeneity_score
print(homogeneity_score(y_test, y_pred))
```

V-Measure

```
from sklearn.metrics import v_measure_score
print(v_measure_score(y_test, y_pred))
```

GeeksforGeeks