Scikit-Learn Cheat Sheet

Scikit-learn is a free software machine learning library for the Python programming language.

Data Preparation

Methods for data preprocessing

```
#Standardization
pp = preprocessing. StandardScaler()

#Normalization
pp = preprocessing.Normalizer()

# Binarization
pp = preprocessing. Binarizer(threshold=0.0)

# Encoding Categorical Features
pp = preprocessing. LabelEncoder()

# Imputing Missing Values
pp = preprocessing. Imputer(missing_values=0, strategy='mean', axis=0)

#Generating Polynomial Features
pp = PolynomialFeatures(5)
```

Data Preprocessing

```
import sklearn.preprocessing as preprocessing
pp = preprocessing.MinMaxScaler()
pp.fit(X)
X_preprocessed = pp.transform(X)
```

Pipeline

```
steps = [("feaunion", fus),
        ("another_pca", PCA(n_components=15)),
        ("lr",LinearRegression())]
pipeline = Pipeline(steps)
pipeline.fit(X, y)
```

Train-Test data

```
import numpy as np
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y)
```

Model Performance Evaluation

Supervised Metrics

```
from sklearn.metrics import metrics

# Classification Metrics
# Accuracy Score
score = knn.score(X_test, y_test)
cm = metrics.confusion_matrix(test_y, pred_y)
metrics.plot_confusion_matrix(lr, test_x, test_y)

# Classification Report
print(metrics.classification_report(y_test, y_pred))
# Confusion Matrix
print(metrics.confusion_matrix(y_test, y_pred))
# F1 score
f1 = metrics.f1_score(test_y, pred_y)

# Regression Metrics
# Mean Absolute Error
y_true = [3, -0.5, 2]
mae = metrics. mean_absolute_error(y_true, y_pred)

# Mean Squared Error
mse = metrics.mean_squared_error(y_test, y_pred)
```

# R^2 Score

```
r2 = metrics.r2_score(y_true, y_pred)
```

Unsupervised Metrics

```
from sklearn.metrics import metrics

# Clustering Metrics
# Adjusted Rand Index
metrics.adjusted_rand_score(y_true, y_pred)

#Homogeneity
metrics.homogeneity_score(y_true, y_pred)

#V-measure
metrics.v_measure_score(y_true, y_pred)
```

```
#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))
```

Model Tuning

Grid Search and Cross Validation

```
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)
```

Randomized Search and Cross Validation

```
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_)
```

Supervised Models

```
# Linear Regression
from sklearn.linear_model import LinearRegression
mod = LinearRegression(normalize=True)

# Support Vector Machines (SVM)
from sklearn.svm import SVC
mod = SVC(kernel='linear')

# Naive Bayes
from sklearn.naive_bayes import GaussianNB
mod = GaussianNB()

# KNN - Classifier
from sklearn import neighbors
mod = neighbors.KNeighborsClassifier(n_neighbors=6)

# KNN – Regressor
from sklearn.neighbors import KNeighborsRegressor
mod = KNeighborsRegressor()

# Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
mod = DecisionTreeClassifier()

# Random forest
from sklearn.ensemble import RandomForestClassifier
mod = RandomForestClassifier()

# Logistic regression
from sklearn.linear_model import LogisticRegression
mod = LogisticRegression()

# Neural networks
from sklearn.neural_network import MLPClassifier
mod = MLPClassifier(hidden_layer_sizes=(10, 10, 10), max_iter=1000)
```

Model Fitting

```
from sklearn.linear_model import LinearRegression
mod = LinearRegression()
mod.fit(X, y)
pred_y = mod.predict(X_test)
```

Unsupervised Models

```
# Principal Component Analysis (PCA)
from sklearn.decomposition import PCA
mod = PCA(n_components=0.95)

# K Means
from sklearn.cluster import KMeans
mod = KMeans(n_clusters=3, random_state=0)
```

scikit learn

Created by hectorv.com