Scikit-learn 핵심 치트시트

Scikit-learn 표준 분석 워크플로우

1. **데이터 준비**: 입력 특성(X)과 예측 대상인 타겟(y)을 분리합니다.
2. **모델 선택**: 해결하려는 문제에 적합한 알고리즘 클래스를 임포트합니다.
3. **모델 인스턴스화**: 하이퍼파라미터를 설정하여 모델 객체를 생성합니다.
4. **학습**: `model.fit(X, y)`를 통해 모델을 학습시킵니다.
5. **예측**: `model.predict(X_new)`로 새로운 데이터에 대한 예측값을 생성합니다.
6. **평가**: 다양한 지표를 활용해 모델의 성능을 정밀하게 측정합니다.

데이터 전처리 (Preprocessing)

데이터의 품질을 높이기 위해 `sklearn.preprocessing` 모듈을 활용합니다.

- **스케일링 (Scaling)**:
  - `StandardScaler`: 데이터를 평균 0, 분산 1이 되도록 표준화합니다.
  - `MinMaxScaler`: 데이터를 0과 1 사이의 범위로 압축합니다.
  - `RobustScaler`: 중앙값과 사분위수를 사용하여 이상치(Outlier)의 영향을 최소화합니다.
- **인코딩 (Encoding)**:
  - `LabelEncoder`: 범주형 텍스트 라벨을 정수형 숫자로 변환합니다.
  - `OneHotEncoder`: 범주형 변수를 희소 행렬 형태의 원-핫 벡터로 변환합니다.
- **결측치 처리**:
  - `SimpleImputer`: 평균, 중앙값, 최빈값 등으로 누락된 데이터를 대체합니다.

```
from sklearn.preprocessing import
StandardScaler
scaler = StandardScaler()
# 학습과 변환을 동시에 수행
X_scaled = scaler.fit_transform(X)
```

주요 모델 선택 가이드

지도 학습 (Supervised Learning)

- **회귀 (Regression)**:
  - `LinearRegression`: 가장 기본적인 선형 회귀 모델입니다.
  - `Ridge`, `Lasso`: 과적합 방지를 위해 규제(Regularization)가 추가된 선형 모델입니다.
  - `SVR`: 서포트 벡터 머신 기반의 회귀 모델입니다.
  - `RandomForestRegressor`: 결정 트리 앙상블 기반의 강력한 회귀 모델입니다.
- **분류 (Classification)**:
  - `LogisticRegression`: 이진 분류 및 다중 분류의 기준이 되는 모델입니다.
  - `SVC`: 클래스 간 경계를 최대화하는 서포트 벡터 분류기입니다.
  - `KNeighborsClassifier`: 이웃 데이터와의 거리를 기반으로 분류합니다.
  - `RandomForestClassifier`: 여러 개의 결정 트리를 조합하여 성능을 높인 분류기입니다.

비지도 학습 (Unsupervised Learning)

- **군집화 (Clustering)**:
  - `KMeans`: 데이터를 K개의 군집으로 묶습니다.
  - `DBSCAN`: 밀도 기반으로 공간상의 군집을 찾아냅니다.
- **차원 축소 (Dimensionality Reduction)**:
  - `PCA`: 데이터의 분산을 최대한 보존하면서 차원을 축소합니다.
  - `TSNE`: 고차원 데이터를 시각화하기 위해 저차원으로 투영합니다.

데이터 분할 및 교차 검증

`sklearn.model_selection` 모듈을 통해 모델의 일반화 성능을 높입니다.

- **학습/테스트 데이터 분할**:

```
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
)
```

- **교차 검증 (Cross-Validation)**:

```
from sklearn.model_selection import
cross_val_score
# 5-겹 교차 검증으로 모델의 안정성을 평가합니다.
scores = cross_val_score(model, X, y,
cv=5)
```

모델 평가 지표 (Metrics)

`sklearn.metrics` 모듈에서 제공하는 주요 성능 측정 지표입니다.

회귀 평가 지표

- `mean_absolute_error` (MAE): 실제값과 예측값 차이의 절대량 평균
- `mean_squared_error` (MSE): 차이의 제곱 평균 (큰 오차에 민감)
- `r2_score`: 모델이 데이터의 변동성을 얼마나 설명하는지 나타내는 결정계수

분류 평가 지표

- `accuracy_score`: 전체 데이터 중 정답을 맞힌 비율 (정확도)
- `precision_score`: Positive로 예측한 것 중 실제 정답의 비율 (정밀도)
- `recall_score`: 실제 Positive인 것 중 모델이 찾아낸 비율 (재현율)
- `f1_score`: 정밀도와 재현율의 균형을 나타내는 조화 평균
- `roc_auc_score`: 분류 임계값 변화에 따른 모델의 식별 능력을 측정

```
from sklearn.metrics import
accuracy_score, classification_report

y_pred = model.predict(X_test)
print(f"정확도: {accuracy_score(y_test,
y_pred)}")
# 정밀도, 재현율 등 주요 지표를 한눈에 확인합니다.
print(classification_report(y_test,
y_pred))
```

하이퍼파라미터 최적화

- **그리드 서치 (Grid Search)**: 설정한 모든 조합을 전수 조사하여 최적의 파라미터를 찾습니다.

```
from sklearn.model_selection import
GridSearchCV
param_grid = {'n_estimators': [100,
200], 'max_depth': [3, 5]}
grid_search = GridSearchCV(model,
param_grid, cv=5)
grid_search.fit(X_train, y_train)
print(f"최적의 파라미터:
{grid_search.best_params_}")
```

- **랜덤 서치 (Randomized Search)**: 무작위 조합을 시도하여 효율적으로 파라미터를 탐색합니다.

파이프라인 (Pipeline)

전처리 단계와 학습 단계를 하나로 묶어 코드를 간결하게 관리하고 실수를 방지합니다.

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import
StandardScaler
from sklearn.svm import SVC

# 스케일링 후 SVC 모델 학습을 하나의 단위로 묶음
pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('svc', SVC())
])

pipe.fit(X_train, y_train)
print(f"테스트 점수: {pipe.score(X_test,
y_test)}")
```