SCIKIT-LEARN (sklearn) CHEAT SHEET — Regression & Classification (Stage 6)

Core imports

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score, classification_report, confusion_matrix
from sklearn.model_selection import GridSearchCV
```

Typical ML workflow
----
1) Load & inspect data (df.head(), df.info()).
2) Split into features (X) and target (y).
3) Train/test split:
   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
4) (Optional) Scale numeric features:
   scaler = StandardScaler()
5) Fit a model → predict → evaluate.

Linear Regression (predict numbers)

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2  = r2_score(y_test, y_pred)
print("coef_:", model.coef_, "intercept_:", model.intercept_, "MSE:", mse, "R2:", r2)
```

Polynomial Regression (curved trends)

```
pipe = Pipeline([
    ("poly", PolynomialFeatures(degree=2, include_bias=False)),
    ("lin", LinearRegression())
])
pipe.fit(X_train, y_train)
print("R2 test:", pipe.score(X_test, y_test))
```

Logistic Regression (binary classes)

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(max_iter=1000)
clf.fit(X_train, y_train)
pred = clf.predict(X_test)
proba = clf.predict_proba(X_test)  # probabilities
print("Accuracy:", accuracy_score(y_test, pred))
print(classification_report(y_test, pred))
```

K-Nearest Neighbours (KNN)

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
pred = knn.predict(X_test)
print("Accuracy:", accuracy_score(y_test, pred))
```

Scaling with Pipelines (recommended for KNN/logistic)

```
clf = Pipeline([
    ("scale", StandardScaler()),
    ("logit", LogisticRegression(max_iter=1000))
])
clf.fit(X_train, y_train)
```

Model selection with GridSearchCV

```
param_grid = {"n_neighbors": [3,5,7,9]}
grid = GridSearchCV(KNeighborsClassifier(), param_grid=param_grid, cv=5)
grid.fit(X_train, y_train)
print("Best params:", grid.best_params_, "CV score:", grid.best_score_)
```

Train/Test Split & Feature/Target

```
X = df[["feature1", "feature2"]]  # DataFrame → 2D
y = df["target"]                   # Series → 1D
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Common gotchas

• X must be 2D (shape: [n_samples, n_features]).
• y is 1D for regression/classification.
• Always keep a test set separate for honest evaluation.
• Use StandardScaler for distance-based models (KNN, SVM) and logistic regression.
• Set random_state for reproducibility.

Reading & saving models

```
import joblib
joblib.dump(model, "model.pkl")
loaded = joblib.load("model.pkl")
```