ChatGPT

Scikit-Learn (sklearn) Cheat Sheet

Data Preprocessing

- **StandardScaler** – Standardize features by removing the mean and scaling to unit variance [1].
- **MinMaxScaler** – Scale features to a given range (default 0 to 1) [2].
- **RobustScaler** – Remove the median and scale data according to the interquartile range (IQR), making scaling robust to outliers [3].
- **Normalizer** – Scale input vectors (samples) to unit norm (each sample's length is 1) [4].
- **Binarizer** – Binarize data by thresholding: values above threshold become 1, others 0 [5].
- **LabelEncoder** – Encode target labels with values 0 to n_classes-1 (for converting labels y to numeric form) [6].
- **OneHotEncoder** – Encode categorical features as a one-hot numeric array (adds a binary column for each category) [7].
- **OrdinalEncoder** – Encode categorical features as ordinal integers (0 to n_categories-1 for each feature) [8].
- **SimpleImputer** – Impute (fill) missing values using a specified strategy (mean, median, most_frequent, or constant) [9].
- **KNNImputer** – Impute missing values using k-Nearest Neighbors (uses the mean of the nearest neighbors) [10].
- **PolynomialFeatures** – Generate new features consisting of all polynomial combinations of the original features up to a given degree (for feature engineering) [11].

Feature Selection

- **SelectKBest** – Select features according to the k highest scores from a scoring function (e.g. chi^2 for classification, f_regression for regression) [12].
- **RFE (Recursive Feature Elimination)** – Select features by recursively removing the least important features using an estimator's feature importances or coefficients, until the desired number of features is reached [13]. (Use `RFECV` for automatic selection with cross-validation.)
- **SelectFromModel** – Select features based on an estimator's importance weights (e.g. drop features below a certain importance threshold).

Model Selection (Splitting & Cross-Validation)

- **train_test_split** – Split arrays or matrices into random train and test subsets (commonly used to create training and hold-out test sets) [14].
- **KFold** – K-Folds cross-validator: split dataset into *k* consecutive folds (for cross-validation, each fold is used as a test set once) [15].
- **StratifiedKFold** – Stratified K-Folds cross-validator: like KFold but preserves class label proportions in each fold (for classification) [16].
- **cross_val_score** – Evaluate a score (e.g. accuracy) by cross-validation (trains and tests on multiple folds, returns the score for each fold) [17].
- **cross_val_predict** – Generate cross-validated estimates for each input data point (returns predicted values for each sample, as if each were in the test fold) [18].
- **cross_validate** – Evaluate one or multiple metrics by cross-validation and also record fit/score times (returns a dictionary with scores and timing) [19].

Classification Models (Estimators)

- **LogisticRegression** – Linear model for binary or multi-class classification (uses logistic function; supports L1/L2 regularization) [20].
- **SVC (Support Vector Classifier)** – Support Vector Machine classifier for linear or non-linear classification (can use kernels like linear, RBF) [21].
- **KNeighborsClassifier** – k-Nearest Neighbors classifier; labels a sample based on the majority class among its nearest neighbors [22].
- **DecisionTreeClassifier** – Tree-based classifier that splits features to create decision rules leading to class predictions [23].
- **RandomForestClassifier** – Ensemble of many decision trees (bagging); outputs the majority vote of trees for classification (reduces overfitting) [24].
- **GradientBoostingClassifier** – Ensemble of decision trees built sequentially, where each new tree corrects errors of the previous (gradient boosting) [25].
- **GaussianNB** – Gaussian Naïve Bayes classifier; assumes features follow a Gaussian distribution, used for fast probabilistic classification [26].
- **MLPClassifier** – Multi-Layer Perceptron neural network classifier; optimizes log-loss via stochastic gradient descent or LBFGS (for non-linear classification) [27].

Regression Models

- **LinearRegression** – Ordinary least squares linear regression model (fits a line/plane to minimize squared error) [28].
- **Ridge** – Linear regression with L2 regularization (penalizes large coefficients to prevent overfitting) [29].
- **Lasso** – Linear regression with L1 regularization (encourages sparsity, can set some coefficients to zero) [30].
- **ElasticNet** – Linear regression with combined L1 and L2 regularization (balance of Ridge and Lasso penalties) [31].
- **DecisionTreeRegressor** – Decision tree for regression; splits data into regions and predicts the mean value in each region [32].
- **RandomForestRegressor** – Ensemble of decision tree regressors (bagging); outputs the average prediction of many trees [33].
- **SVR (Support Vector Regressor)** – Support Vector Machine for regression; finds a function within a tolerance tube (epsilon) to fit data with maximum margin [34].
- **MLPRegressor** – Multi-Layer Perceptron neural network for regression; optimizes squared error via gradient-based methods (can capture complex nonlinear relationships) [35].

Clustering Models (Unsupervised)

- **KMeans** – Partitions data into *k* clusters by trying to minimize intra-cluster variance (iteratively assigns points to the nearest cluster centroid) [36].
- **DBSCAN** – Density-Based Spatial Clustering: finds clusters of high density and labels low-density points as noise (can find arbitrarily shaped clusters) [37].
- **AgglomerativeClustering** – Hierarchical clustering that recursively merges pairs of clusters; can use various linkage criteria (e.g. Ward, complete) [38].
- **MeanShift** – Clustering that iteratively shifts points towards the mode (peak) of their local density; automatically discovers the number of clusters.

Dimensionality Reduction

- **PCA (Principal Component Analysis)** – Unsupervised technique to reduce feature dimensionality by projecting data onto principal components that maximize variance [39].
- **TruncatedSVD** – Dimensionality reduction for sparse data (like text); similar to PCA but does not center data, useful for sparse matrices (e.g. TF-IDF matrices) [40].
- **t-SNE** – T-distributed Stochastic Neighbor Embedding for visualization; embeds high-dimensional data into 2D/3D while preserving local structure (good for plotting clusters) [41].
- **FeatureAgglomeration** – Reduces dimensionality by clustering features instead of samples, then aggregating them (treats features similarly to observations in clustering) [42].

Pipelines and Workflows

- **Pipeline** – Chain a sequence of transformations and an estimator into a single object. The pipeline ensures that preprocessing (e.g., scaling, encoding) on training data is **also applied to test data** in the same way. You can fit and predict with a Pipeline as one unit [43].
- **ColumnTransformer** – Apply different transformers to different columns of an array/DataFrame (e.g., numeric vs categorical columns) and combine the results back into one feature set [44]. This is useful for preprocessing heterogeneous data.
- **FeatureUnion** – Apply multiple transformer pipelines in parallel and concatenate their outputs (e.g., extract different sets of features and combine) [45].
- **make_pipeline** – Convenience function to create a Pipeline without naming the steps (automatically names steps after their estimator types).
- **make_column_selector** – Utility to select columns by dtype or column name pattern, often used with ColumnTransformer to pick features.

Example: Using a pipeline to scale features and train a classifier:

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('clf', LogisticRegression())
])
pipe.fit(X_train, y_train)
y_pred = pipe.predict(X_test)
```

Model Training & Prediction (Common Methods)

- **fit(X, y)** – Train the model/estimator using the training data X and targets y [46]. (For unsupervised transformers, `fit(X)` learns the transformation.)
- **predict(X)** – Predict labels or values for new data X using a trained model [47]. (Available for supervised estimators and some clustering models.)
- **transform(X)** – Transform data X using a fitted transformer (e.g., scale or encode features). (`fit_transform(X)` can be used to do both in one step for training data.)
- **predict_proba(X)** – For classifiers, predict class probabilities for X (each sample gets a probability distribution over classes) [48].

- **score(X, y)** – Return a default evaluation score for the model on data X with true labels y (by default, accuracy for classifiers, R^2 for regressors) [49] .

Model Evaluation Metrics

Classification Metrics:
- **accuracy_score** – Proportion of correct predictions (overall accuracy) [50] .
- **precision_score** – Precision of positive class: fraction of positive predictions that were actually positive [51] .
- **recall_score** – Recall (sensitivity) of positive class: fraction of actual positives that were correctly predicted [52] .
- **f1_score** – F1 score, the harmonic mean of precision and recall (useful for imbalanced classes) [53] .
- **confusion_matrix** – Compute the confusion matrix to summarize prediction results (counts of TP, FP, FN, TN) [50] .
- **classification_report** – Text summary of precision, recall, F1-score for each class (and overall metrics) [54] .
- **roc_auc_score** – Area Under the ROC Curve (AUC) for binary classifiers (higher = better separability of classes) [55] .
- **log_loss** – Logarithmic loss (cross-entropy) for probabilistic classifiers (smaller is better).

Regression Metrics:
- **mean_squared_error (MSE)** – Mean of squared differences between predictions and true values (penalizes large errors) [56] .
- **mean_absolute_error (MAE)** – Mean of absolute differences between predictions and true values (more interpretable, linear penalty) [57] .
- **r2_score** – R^2 (coefficient of determination): fraction of variance in target explained by the model (1.0 is perfect, 0 means no better than mean) [58] .

(For clustering, common metrics include silhouette_score for evaluating cluster cohesion/separation, etc.)

Hyperparameter Tuning

- **GridSearchCV** – Exhaustive grid search over specified hyperparameter values, with cross-validation. Trains models for **all combinations** of parameters in a grid to find the best-performing combination [59] . Access `best_params_` and `best_estimator_` after fitting to get the optimal configuration.
- **RandomizedSearchCV** – Randomized search over hyperparameters. Samples a fixed number of random combinations from the parameter distributions for cross-validation, which can be much faster than grid search for large search spaces [60] . Specify `n_iter` for number of parameter settings to try.
- **HalvingGridSearchCV / HalvingRandomSearchCV** – **Successive halving** strategies that begin with many candidates trained on small portions of data, then progressively focus resources on the most promising candidates (efficient for large datasets) [61] .
- You can supply a **scoring** parameter to these search CV tools to optimize for metrics other than default, and use `cv` to specify the cross-validation strategy.

Utility Functions & Datasets

- **Built-in Toy Datasets** (`sklearn.datasets`): Use functions to load small standard datasets for practice and experimentation. For example: `load_iris()` (iris flower classification),

`load_wine()` (wine classification), `load_breast_cancer()` (cancer diagnosis), `load_digits()` (handwritten digits) [62], `load_diabetes()` (diabetes regression), `fetch_california_housing()` (California housing regression). These return a *Bunch* object (dictionary-like) with `.data` (features) and `.target` (labels).

- **Dataset Fetchers**: For larger datasets, functions like `fetch_20newsgroups()` (text classification data) and `fetch_openml()` can download data from online repositories.
- **Synthetic Data Generators** (`sklearn.datasets.make_*`): Quickly generate artificial datasets for testing algorithms. E.g. `make_classification()` – create a random n-class classification problem with specified features/informative features [63]; `make_regression()` – create a random regression problem; `make_blobs()` – generate isotropic Gaussian blobs for clustering.
- **shuffle / resample** (`sklearn.utils`): Randomly shuffle datasets or perform bootstrap resampling. Useful for randomizing data order or creating sample splits.
- **Estimator Utilities**: `get_params()` and `set_params()` allow getting or setting hyperparameters of an estimator (useful for introspection or cloning models). `clone()` creates an unfitted copy of an estimator.

---

1 StandardScaler — scikit-learn 1.7.2 documentation
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

2 MinMaxScaler — scikit-learn 1.7.2 documentation
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

3 RobustScaler — scikit-learn 1.7.2 documentation
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html

4 normalize — scikit-learn 1.7.2 documentation
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.normalize.html

5 Binarizer — scikit-learn 1.7.2 documentation
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Binarizer.html

6 LabelEncoder — scikit-learn 1.7.2 documentation
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

7 sklearn.preprocessing.OneHotEncoder — scikit-learn 0.21.3 documentation
https://scikit-learn.org/0.21/modules/generated/sklearn.preprocessing.OneHotEncoder.html

8 sklearn.preprocessing.OrdinalEncoder — scikit-learn 0.21.3 documentation
https://scikit-learn.org/0.21/modules/generated/sklearn.preprocessing.OrdinalEncoder.html

9 10 11 20 21 22 23 24 25 26 28 29 30 32 33 34 36 37 38 39 40 41 42 46 47 48 49 50 51 52 53 54 55 56 58 59 60 62 Scikit-learn Cheatsheet [2025 Updated] - Download pdf - GeeksforGeeks
https://www.geeksforgeeks.org/blogs/scikit-learn-cheatsheet/

12 How I used SelectKBest feature selection on Kaggle's August 2021 ...
https://medium.com/mlearning-ai/how-i-used-selectkbest-feature-selection-on-kaggles-august-2021-tabular-competition-1b8a35ad429a

13 RFE — scikit-learn 1.7.2 documentation
https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html

14 sklearn.model_selection.train_test_split — scikit-learn 0.18.2 ...
https://scikit-learn.org/0.18/modules/generated/sklearn.model_selection.train_test_split.html

15 Tutorial: K Fold Cross Validation - Kaggle
https://www.kaggle.com/code/satishgunjal/tutorial-k-fold-cross-validation

16 sklearn.model_selection.StratifiedKFold — scikit-learn 0.21.3 ...
https://scikit-learn.org/0.21/modules/generated/sklearn.model_selection.StratifiedKFold.html

17 cross_val_score — scikit-learn 1.7.2 documentation
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html

18 cross_val_predict — scikit-learn 1.7.2 documentation
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_predict.html

19 cross_validate — scikit-learn 1.7.2 documentation
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_validate.html

27 MLPClassifier — scikit-learn 1.7.2 documentation
https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

31 Implementation of Elastic Net Regression From Scratch
https://www.geeksforgeeks.org/machine-learning/implementation-of-elastic-net-regression-from-scratch/

35 MLPRegressor — scikit-learn 1.7.2 documentation
https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

43 sklearn.pipeline.Pipeline — scikit-learn 0.19.2 documentation
https://scikit-learn.org/0.19/modules/generated/sklearn.pipeline.Pipeline.html

44 ColumnTransformer — scikit-learn 1.7.2 documentation
https://scikit-learn.org/stable/modules/generated/sklearn.compose.ColumnTransformer.html

45 FeatureUnion — scikit-learn 1.7.2 documentation
https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.FeatureUnion.html

57 mean_absolute_error — scikit-learn 1.7.2 documentation
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error.html

61 HalvingGridSearchCV — scikit-learn 1.7.2 documentation
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.HalvingGridSearchCV.html

63 make_classification — scikit-learn 1.7.2 documentation
https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html