List of ML APIs from sklearn and other modules.

Module	**API**
sklearn.feature_extraction	- DictVectorizer … - FeatureHasher …
sklearn.feature_extraction.text	- TfidfVectorizer … - CountVectorizer …
sklearn.feature_selection	- VarianceThreshold … - SelectKBest … - SelectPercentile … - GenericUnivariateSelect … - mutual_info_regression … - mutual_info_classif … - RFE … - RFECV … - SelectFromModel … - SequentialFeatureSelector …
sklearn.impute	- SimpleImputer - KNNImputer - MissingIndicator
sklearn.preprocessing	- StandardScaler - MinMaxScaler - MaxAbsScaler - FunctionTransformer - PolynomialFeatures - KBinsDiscretizer - OneHotEncoder - LabelEncoder - OrdinalEncoder - LabelBinarizer - MultiLabelBinarizer - add_dummy_feature
sklearn.compose	- ColumnTransformer - TransformedTargetRegressor
sklearn.linear_model	- LinearRegression – SGDRegressor - Lasso, LassoCV - Ridge, RidgeCV - LogisticRegression, LogisticRegressionCV - SGDClassifier

Module	API
	- RidgeClassifier, RidgeClassifierCV - Perceptron
sklearn.svm	- SVC
sklearn.naive_bayes	- MultinomialNB - GaussianNB - BernoulliNB
sklearn.ensemble	- RandomForestRegressor
sklearn.dummy	- DummyRegressor - DummyClassifier
sklearn.datasets	- load_iris, load_wine, load_diabetes - fetch_california_housing, fetch_20_newsgroups - fetch_openml - make_regression - make_blobs - make_classification - make_multilabel_classification
keras.datasets	mnist
sklearn.model_selection	- train_test_split - cross_validate - cross_val_score - cross_val_predict - learning_curve - validation_curve - ShuffleSplit - StratifiedShuffleSplit - permutation_test_score - GridSearchCV - RandomizedSearchCV
sklearn.metrics	- mean_squared_error - mean_absolute_error - mean_absolute_percentage_error - log_loss - hinge_loss - confusion_matrix - ConfusionMatrixDisplay - precision_score - recall_score - make_scorer - classification_report

Module	**API**
	- precision_recall_curve - roc_curve - plot_roc_curve - roc_auc_curve
sklearn.decomposition	- PCA
sklearn.pipeline	- make_pipeline - Pipeline - FeatureUnion
sklearn.set_config	- set_config
sklearn.utils	- all_estimators
sklearn.utils.multiclass	- type_of_target
scipy.stats	- uniform - loguniform
imblearn.under_sampling	- RandomUnderSampler
imblearn.over_sampling	- RandomOverSampler - SMOTE
warnings	- filterwarnings
numpy	- array - arange - linspace, logspace - to_numpy - unique - zeros, ones - where - argmax, argmin - argsort - random.seed - random.permutation - count_nonzero - var, std - row_stack (vstack), column_stack (hstack)
pandas	- DataFrame - read_csv - concat

Module	**API**
	- get_dummies - var, std
pandas.plotting	- scatter_matrix
sns	- histplot - scatterplot - heatmap - pairplot - set_style

List of object based APIs

Object	**API/property/attribute**
DataFrame	- head, tail - columns - loc, iloc - info, describe - insert, pop - copy - plot, boxplot - corr - drop, dropna - isna, isnull, notna, isna - reset_index - replace - transpose, T - hist - sum, mean, median, var, std
Numpy array	- shape - reshape - transpose, T - mean, median, var, std - ravel
Series	- value_counts - unique - count - replace - mean, median, var, std - hist

Object	**API/property/attribute**
Bunch	- frame - data - target - target_names - feature_names
DictVectorizer, VarianceThreshold	- fit, transform, fit_transform
SelectKBest SelectPercentile GenericUnivariateSelect RFE/RFECV SelectFromModel SequentialFeatureSelector PolynomialFeatures	- fit, transform, fit_transform - get_feature_names_out
SimpleImputer	- fit, transform, fit_transform - statistics_ - median
PCA	- fit, transform, fit_transform - explained_variance
RandomUnderSampler RandomOverSampler SMOTE	- fit_resample
KNNImputer MissingIndicator	- fit, transform, fit_transform
StandardScaler, MinMaxScaler, MaxAbsScaler	- fit, partial_fit, transform, fit_transform
csr_matrix	- toarray, todense
DictVectorizer	- fit, transform, fit_transform
OrdinalEncoder, OneHotEncoder, LabelEncoder,	- fit, transform, fit_transform - categories_
LabelBinarizer, MultilabelBinarizer	- fit, transform, fit_transform - classes_

Object	**API/property/attribute**
Pipeline	- fit, transform, fit_transform - score
FunctionTransformer, ColumnTransformer, KBinsDiscretizer	- fit, transform, fit_transform
LinearRegression Ridge Lasso	- fit - predict - score - coef_, intercept_
DummyRegressor TransformedTargetRegressor	- fit - predict - score
SGRRegressor SGDClassifier	- fit, partial_fit - predict - score - coef_, intercept_, n_iter_t_
DecisionTreeRegressor RandomForestRegressor	- fit - predict - score - feature_importances_
Perceptron	- fit, partial_fit - predict - score - classes_, coef_, intercept_
LogisticRegression	- fit - predict - score - classes_, coef_, intercept_
KNeighborsClassifier	- fit - predict - score - classes_
SVC	- fit - predict - score - classes_, coef_, intercept_ - support_, support_vectors_, n_support_

Object	**API/property/attribute**
GridSearchCV RandomizedSearchCV	- fit - predict - cv_results_ (only after fit) - best_index_ - best_estimator_ - best_params_ - best_score_
RidgeCV	- best_index_ - best_estimator_ - best_params_ - best_score_
StratifiedShuffleSplit	- split
PCA	- fit, transform, fit_transform
CountVectorizer	- fit, transform, fit_transform vocabulary_

```
>>> from sklearn.feature_selection import VarianceThreshold
>>> X = [[0, 2, 0, 3], [0, 1, 4, 3], [0, 1, 1, 3]]
>>> selector = VarianceThreshold()
>>> selector.fit_transform(X)
array([[2, 0],
       [1, 4],
       [1, 1]])
```

```
>>> from sklearn.datasets import load_digits
>>> from sklearn.feature_selection import SelectKBest, chi2
>>> X, y = load_digits(return_X_y=True)
>>> X.shape
(1797, 64)
>>> X_new = SelectKBest(chi2, k=20).fit_transform(X, y)
>>> X_new.shape
(1797, 20)
```

```
>>> from sklearn.datasets import load_digits
>>> from sklearn.feature_selection import SelectPercentile, chi2
>>> X, y = load_digits(return_X_y=True)
>>> X.shape
(1797, 64)
>>> X_new = SelectPercentile(chi2, percentile=10).fit_transform(X, y)
>>> X_new.shape
(1797, 7)
```

```
>>> from sklearn.datasets import load_breast_cancer
>>> from sklearn.feature_selection import GenericUnivariateSelect, chi2
>>> X, y = load_breast_cancer(return_X_y=True)
>>> X.shape
(569, 30)
>>> transformer = GenericUnivariateSelect(chi2, mode='k_best', param=20)
>>> X_new = transformer.fit_transform(X, y)
>>> X_new.shape
(569, 20)
```

```
>>> from sklearn.datasets import make_friedman1
>>> from sklearn.feature_selection import RFE
>>> from sklearn.svm import SVR
>>> X, y = make_friedman1(n_samples=50, n_features=10, random_state=0)
>>> estimator = SVR(kernel="linear")
>>> selector = RFE(estimator, n_features_to_select=5, step=1)
>>> selector = selector.fit(X, y)
>>> selector.support_
array([ True,  True,  True,  True,  True, False, False, False, False,
       False])
>>> selector.ranking_
array([1, 1, 1, 1, 1, 6, 4, 3, 2, 5])
```

```
>>> from sklearn.datasets import make_friedman1
>>> from sklearn.feature_selection import RFECV
>>> from sklearn.svm import SVR
>>> X, y = make_friedman1(n_samples=50, n_features=10, random_state=0)
>>> estimator = SVR(kernel="linear")
>>> selector = RFECV(estimator, step=1, cv=5)
>>> selector = selector.fit(X, y)
>>> selector.support_
array([ True,  True,  True,  True,  True, False, False, False, False,
       False])
>>> selector.ranking_
array([1, 1, 1, 1, 1, 6, 4, 3, 2, 5])
```

```
>>> from sklearn.feature_selection import SelectFromModel
>>> from sklearn.linear_model import LogisticRegression
>>> X = [[ 0.87, -1.34,  0.31 ],
...      [-2.79, -0.02, -0.85 ],
...      [-1.34, -0.48, -2.55 ],
...      [ 1.92,  1.48,  0.65 ]]
>>> y = [0, 1, 0, 1]
>>> selector = SelectFromModel(estimator=LogisticRegression()).fit(X, y)
>>> selector.estimator_.coef_
array([[-0.3252302 ,  0.83462377,  0.49750423]])
>>> selector.threshold_
0.55245...
>>> selector.get_support()
array([False,  True, False])
>>> selector.transform(X)
array([[-1.34],
       [-0.02],
       [-0.48],
       [ 1.48]])
```

```python
>>> from sklearn.feature_selection import SequentialFeatureSelector
>>> from sklearn.neighbors import KNeighborsClassifier
>>> from sklearn.datasets import load_iris
>>> X, y = load_iris(return_X_y=True)
>>> knn = KNeighborsClassifier(n_neighbors=3)
>>> sfs = SequentialFeatureSelector(knn, n_features_to_select=3)
>>> sfs.fit(X, y)
SequentialFeatureSelector(estimator=KNeighborsClassifier(n_neighbors=3),
                          n_features_to_select=3)
>>> sfs.get_support()
array([ True, False,  True,  True])
>>> sfs.transform(X).shape
(150, 3)
```

```
>>> from sklearn.feature_extraction import DictVectorizer
>>> from sklearn.feature_selection import SelectKBest, chi2
>>> v = DictVectorizer()
>>> D = [{'foo': 1, 'bar': 2}, {'foo': 3, 'baz': 1}]
>>> X = v.fit_transform(D)
>>> support = SelectKBest(chi2, k=2).fit(X, [0, 1])
>>> v.get_feature_names_out()
array(['bar', 'baz', 'foo'], ...)
>>> v.restrict(support.get_support())
DictVectorizer()
>>> v.get_feature_names_out()
array(['bar', 'foo'], ...)
```

```
>>> from sklearn.feature_extraction import FeatureHasher
>>> h = FeatureHasher(n_features=10)
>>> D = [{'dog': 1, 'cat':2, 'elephant':4},{'dog': 2, 'run': 5}]
>>> f = h.transform(D)
>>> f.toarray()
array([[ 0.,  0., -4., -1.,  0.,  0.,  0.,  0.,  0.,  2.],
       [ 0.,  0.,  0., -2., -5.,  0.,  0.,  0.,  0.,  0.]])
```

```
>>> from sklearn.feature_extraction.text import TfidfVectorizer
>>> corpus = [
...     'This is the first document.',
...     'This document is the second document.',
...     'And this is the third one.',
...     'Is this the first document?',
... ]
>>> vectorizer = TfidfVectorizer()
>>> X = vectorizer.fit_transform(corpus)
>>> vectorizer.get_feature_names_out()
array(['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third',
       'this'], ...)
>>> print(X.shape)
(4, 9)
```

```
>>> from sklearn.feature_extraction.text import CountVectorizer
>>> corpus = [
...     'This is the first document.',
...     'This document is the second document.',
...     'And this is the third one.',
...     'Is this the first document?',
... ]
>>> vectorizer = CountVectorizer()
>>> X = vectorizer.fit_transform(corpus)
>>> vectorizer.get_feature_names_out()
array(['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third',
       'this'], ...)
>>> print(X.toarray())
[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
>>> vectorizer2 = CountVectorizer(analyzer='word', ngram_range=(2, 2))
>>> X2 = vectorizer2.fit_transform(corpus)
>>> vectorizer2.get_feature_names_out()
array(['and this', 'document is', 'first document', 'is the', 'is this',
       'second document', 'the first', 'the second', 'the third', 'third one',
       'this document', 'this is', 'this the'], ...)
 >>> print(X2.toarray())
 [[0 0 1 1 0 0 1 0 0 0 0 1 0]
 [0 1 0 1 0 1 0 1 0 0 1 0 0]
 [1 0 0 1 0 0 0 0 1 1 0 1 0]
 [0 0 1 0 1 0 1 0 0 0 0 0 1]]
```