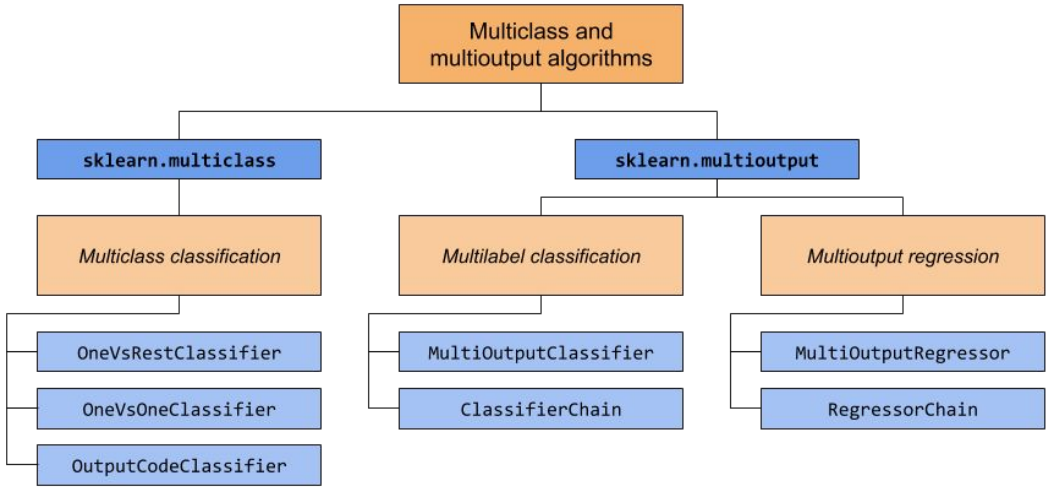
Classification

Chantri Polprasert
CPDSAI

Multinomial logistic regression

- Scikit provides the option that automatically determine if the labels are binary or multiclass
- For Logistic Regression, the training algorithm uses the one-vs-rest (OvR) scheme



https://scikit-learn.org/stable/modules/multiclass.html

Multiclass vs Multi-label classification

- Binary classification: Is this a picture of a beach? {Yes, No}
- Multi-class classification: Which class does this picture belong to? {beach, sky, foliage, mountain}
- Multi-label classification: Which labels are relevant to this picture? {beach, cloud, foliage, sand, mountain, urban}
- Multi-label classification: Which labels are relevant to this picture? {beach, cloud, foliage, sand, mountain, urban}

Each instance can have multiple labels.

Multiclass vs Multi-label classification

	Number of targets	Target cardinality	Valid `type_of_target`
Multiclass classification	1	>2	'multiclass'
Multilabel classification	>1	2 (0 or 1)	'multilabel-indicator'
Multiclass-multioutput classification	>1	>2	'multiclass-multioutput'
Multioutput regression	>1	Continuous	'continuous-multioutput'

OneVsRestClassifier

```
class sklearn.multiclass.OneVsRestClassifier(estimator, *, n_jobs=None, verbose=0)
```

- Fitting one classifier per class. For each classifier, the class is fitted against all the other classes.
- Computational efficiency (only n_classes classifiers are needed)
- Interpretability.
- Can also be used for multilabel classification
- Target format: 1d or column vector containing more than two discrete values.

```
# Initialize the KNN classifier with a specified number of neighbors
knn = KNeighborsClassifier(n_neighbors=3)

# Wrap it in OneVsRestClassifier
ovr_classifier = OneVsRestClassifier(knn)
```

```
y_pred = ovr_classifier.fit(X_train_scaled, y_train).predict(X_test_scaled)
print(y_pred)
```

```
[2 0 2 2 1 0 1 2 1 1 2 0 0 0 0 2 2 1 1 2 0 2 0 2 2 2 2 2 0 0 0 0 1 0 0 2 1
 0 0 0 2 2 1 0 0]
```

```
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
```

OneVsOneClassifier

```
class sklearn.multiclass.OneVsOneClassifier(estimator, *, n_jobs=None)
```

- Constructs one classifier per pair of classes.
- At prediction time, the class which received the most votes is selected.
- In the event of a tie (among two classes with an equal number of votes), choose the class with the highest aggregate classification confidence by summing over the pairwise classification confidence levels computed by the underlying binary classifiers.
- O(n_classes^2) complexity

```
# Initialize the KNN classifier with a specified number of neighbors
knn = KNeighborsClassifier(n_neighbors=3)

# Wrap it in OneVsOneClassifier
ovo_classifier = OneVsOneClassifier(knn)
```

```
y_pred_ovo = ovo_classifier.fit(X_train_scaled, y_train).predict(X_test_scaled)
print(y_pred_ovo)
```

```
[2 0 2 2 1 0 1 2 1 1 2 0 0 0 0 2 2 1 1 2 0 2 0 2 2 2 2 2 0 0 0 0 1 0 0 2 1
 0 0 0 2 2 1 0 0]
```

```
accuracy = accuracy_score(y_test, y_pred_ovo)
print(f'Accuracy: {accuracy:.2f}')
```

MultiOutputClassifier

```
class sklearn.multioutput.MultiOutputClassifier(estimator, *, n_jobs=None) [source]
```

Multi target classification.

- Target format: A valid representation of multilabel y is an either dense or sparse binary matrix of shape (n_samples, n_classes).

```
>>> y = np.array([[1, 0, 0, 1], [0, 0, 1, 1], [0, 0, 0, 0]])
>>> print(y)
[[1 0 0 1]
 [0 0 1 1]
 [0 0 0 0]]
```

```
>>> import numpy as np
>>> from sklearn.datasets import make_multilabel_classification
>>> from sklearn.multioutput import MultiOutputClassifier
>>> from sklearn.linear_model import LogisticRegression
>>> X, y = make_multilabel_classification(n_classes=3, random_state=0)
>>> clf = MultiOutputClassifier(LogisticRegression()).fit(X, y)
>>> clf.predict(X[-2:])
array([[1, 1, 1],
       [1, 0, 1]])
```

https://jmread.github.io/talks/Talk-Helsinki.pdf

Single-label vs Multi-label

Table : Single-label $Y \in \{0, 1\}$

X_1	X_2	X_3	X_4	X_5	beach Y
1	0.1	3	1	0	0
0	0.9	1	0	1	1
0	0.0	1	1	0	0
1	0.8	2	0	1	1
1	0.0	2	0	1	0
0	0.0	3	1	1	?

Build classifier h, such that $\hat{y} = h(\tilde{\mathbf{x}})$.

Table : Multi-label $Y_1, \ldots, Y_L \in 2^L$

X_1	X_2	X_3	X_4	X_5	beach Y_1	sunset Y_2	foliage Y_3	mountain Y_4	urban Y_5	field Y_6
1	0.1	3	1	0	0	1	1	0	1	0
0	0.9	1	0	1	1	0	0	0	0	0
0	0.0	1	1	0	0	1	0	0	0	0
1	0.8	2	0	1	1	0	0	1	0	1
1	0.0	2	0	1	0	0	0	1	0	1
0	0.0	3	1	1	?	?	?	?	?	?

Build classifier(s) h or $\mathbf{h}$, such that $\hat{\mathbf{y}} = [y_1, \ldots, y_L] = h(\tilde{\mathbf{x}})$.

MultiOutputRegressor

```
class sklearn.multioutput.MultiOutputRegressor(estimator, *, n_jobs=None)
```

Multi target regression.

Target format: A valid representation of multi-output y is a dense matrix of shape (n_samples, n_output) of floats. A column wise concatenation of continuous variables.

```
>>> y = np.array([[31.4, 94], [40.5, 109], [25.0, 30]])
>>> print(y)
[[ 31.4  94. ]
 [ 40.5 109. ]
 [ 25.   30. ]]
```

```
>>> from sklearn.datasets import load_linnerud
>>> from sklearn.multioutput import MultiOutputRegressor
>>> from sklearn.linear_model import Ridge
>>> X, y = load_linnerud(return_X_y=True)
>>> regr = MultiOutputRegressor(Ridge(random_state=123)).fit(X, y)
>>> regr.predict(X[[0]])
array([[176..., 35..., 57...]])
```

References

- https://scikit-learn.org/1.5/modules/multiclass.html
- https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LogisticRegression.html#
- https://jmread.github.io/