Multiclass Classification

Sergey Ivanov

Plan

1. Presentation on Multiclass Classification
    a. Error Rates and the Bayes Classifier
    b. Gaussian and Linear Classifiers. Linear Discriminant Analysis. Logistic Regression;
    c. Multi-class classification models and methods;
    d. Multi-class strategies: one-versus-all, one-versus-one, error-correction-codes
2. Linear Classifiers and Multi-classification Tutorial
3. In-class exercise

References

1. Multilabel Classification format
2. Classifier Comparison
3. LDA as dimensionality reduction
4. LDA vs PCA
5. Logistic Regression for 3 classes
6. Linear models
7. LDA and QDA
8. Naive Regression
9. Cross Validation in Python

Naive Bayes

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) P(x_1, \dots x_n \mid y)}{P(x_1, \dots, x_n)}$$

Naive Bayes

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) P(x_1, \dots x_n \mid y)}{P(x_1, \dots, x_n)}$$

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

Naive Bayes

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) P(x_1, \dots x_n \mid y)}{P(x_1, \dots, x_n)}$$

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

$$P(x_i | y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i | y)$$

Naive Bayes

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) P(x_1, \dots x_n \mid y)}{P(x_1, \dots, x_n)}$$

$$P(x_i | y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i | y)$$

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

Naive Bayes

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) P(x_1, \dots x_n \mid y)}{P(x_1, \dots, x_n)}$$

$$P(x_i | y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i | y)$$

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

Naive Bayes

1. Gaussian NB

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

2. Bernoulli NB

$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i)$$

Naive Bayes

Pros:

1. Fast
2. Prevent curse of dimensionality
3. Decent classifier for several tasks (e.g. text classification)
4. Inherently multiclass

Cons:

1. Bad estimator of probabilities to the class.

Linear/Quadratic Discriminant Analysis (LDA/QDA)

$$P(y=k|X) = \frac{P(X|y=k)P(y=k)}{P(X)} = \frac{P(X|y=k)P(y=k)}{\sum_l P(X|y=l) \cdot P(y=l)}$$

Linear/Quadratic Discriminant Analysis (LDA/QDA)

$$P(y=k|X) = \frac{P(X|y=k)P(y=k)}{P(X)} = \frac{P(X|y=k)P(y=k)}{\sum_l P(X|y=l) \cdot P(y=l)}$$

$$p(X|y=k) = \frac{1}{(2\pi)^n |\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu_k)^t \Sigma_k^{-1} (X-\mu_k)\right)$$

Linear/Quadratic Discriminant Analysis (LDA/QDA)

$$P(y=k|X) = \frac{P(X|y=k)P(y=k)}{P(X)} = \frac{P(X|y=k)P(y=k)}{\sum_l P(X|y=l) \cdot P(y=l)}$$

$$p(X|y=k) = \frac{1}{(2\pi)^n |\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu_k)^t \Sigma_k^{-1} (X-\mu_k)\right)$$

- LDA = each class has the same covariance equals to averaged covariance of the classes

$$\Sigma_k = \Sigma$$

- QDA = each class has its own covariance

Linear/Quadratic Discriminant Analysis (LDA/QDA)

Pros:

1. Closed-Form solution
2. Inherently Multiclass
3. No hyperparameters tuning
4. Can be used as dimensionality reduction

Cons:

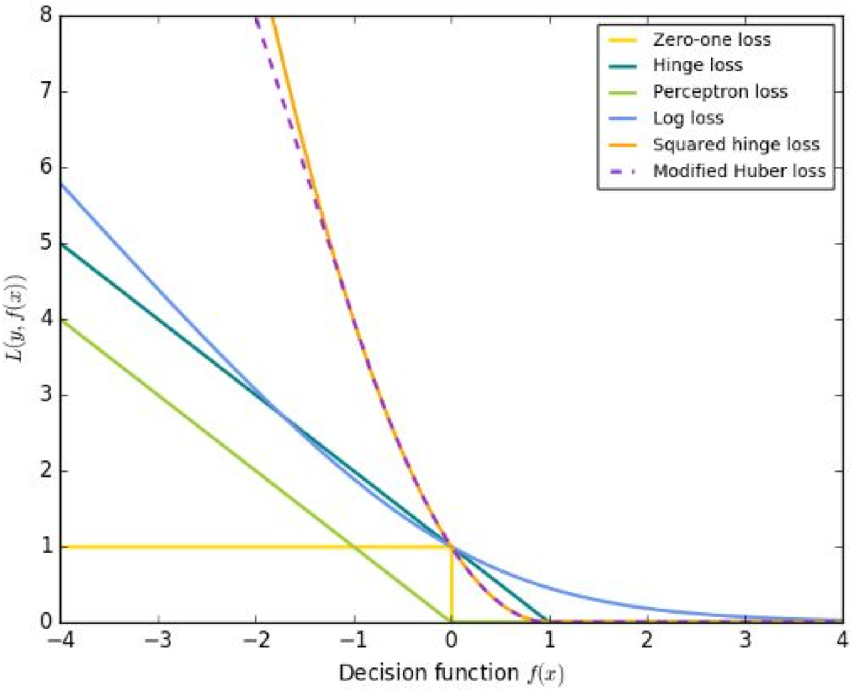
1. Assume unimodal Gaussian distribution for each class
2. Cannot reduce dimensions to more than the number of classes.
3. Not useful if “information” is in data variance instead of the mean of classes.

Stochastic Gradient Descent (SGD)

$$E(w,b) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha R(w)$$

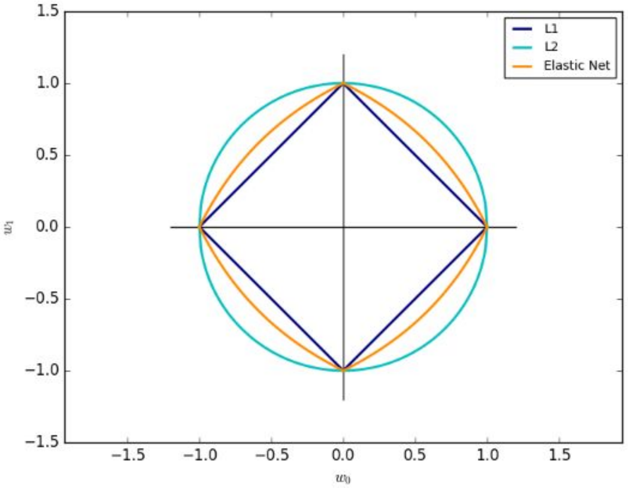
Loss functions L:

- Hinge: (soft-margin) Support Vector Machines.
- Log: Logistic Regression.
- Least-Squares: Ridge Regression.
- Epsilon-Insensitive: (soft-margin) Support Vector Regression.



Stochastic Gradient Descent (SGD)

$$E(w,b) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha R(w)$$



Regularization Term R:

- L2 norm: $R(w) := \frac{1}{2}\sum_{i=1}^{n} w_i^2$.
- L1 norm: $R(w) := \sum_{i=1}^{n} |w_i|$, which leads to sparse solutions.
- Elastic Net: $R(w) := \frac{\rho}{2}\sum_{i=1}^{n} w_i^2 + (1-\rho)\sum_{i=1}^{n} |w_i|$, a convex combination of L2 and L1, where ρ is given by `1 - l1_ratio`.

Stochastic Gradient Descent (SGD)

$$E(w,b) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha R(w)$$

$$w \leftarrow w - \eta(\alpha\frac{\partial R(w)}{\partial w} + \frac{\partial L(w^T x_i + b, y_i)}{\partial w})$$

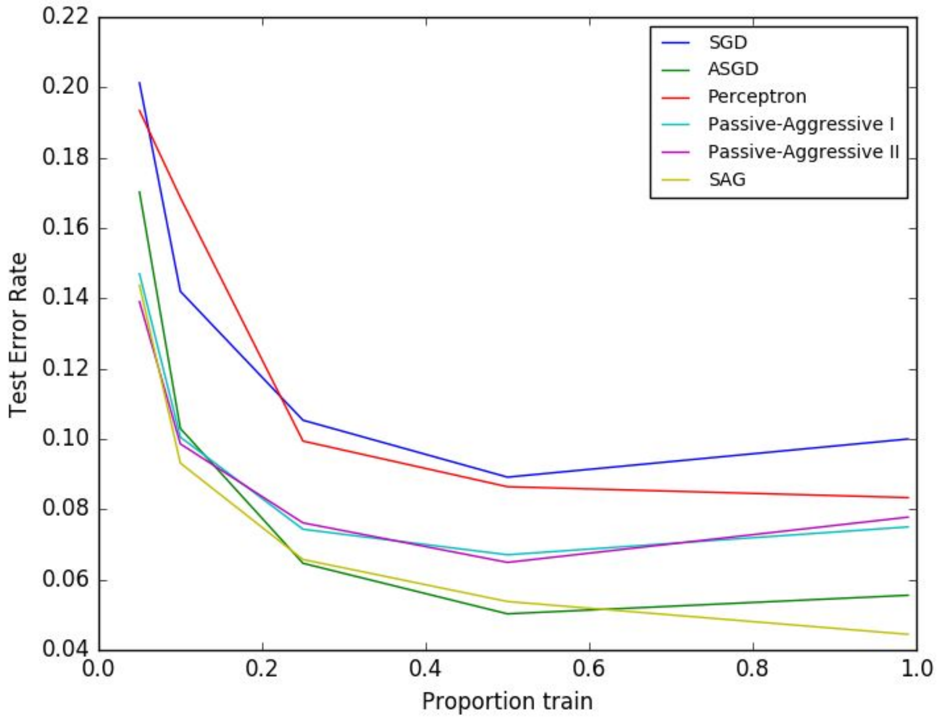
$$\eta^{(t)} = \frac{1}{\alpha(t_0 + t)}$$

Stochastic Gradient Descent (SGD)

$$E(w,b) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha R(w)$$

$$w \leftarrow w - \eta(\alpha \frac{\partial R(w)}{\partial w} + \frac{\partial L(w^T x_i + b, y_i)}{\partial w})$$

$$\eta^{(t)} = \frac{1}{\alpha(t_0 + t)}$$



Stochastic Gradient Descent (SGD)

Practical Tips:

- Scale data so that each dimension has unit variance and zero mean. StandardScaler() in Python.
- Empirically, n_iter = np.ceil(10**6 / n)
- Averaged SGD works best with large number of features.
- After PCA, multiply training data by c such that L2 norm will be equals to 1.

Stochastic Gradient Descent (SGD)

Pros:

1. Fast
2. Ease of implementation
3. Sound theoretical results

Cons:

1. Hyperparameters tuning
2. Sensitive to feature scaling
3. Not multiclass

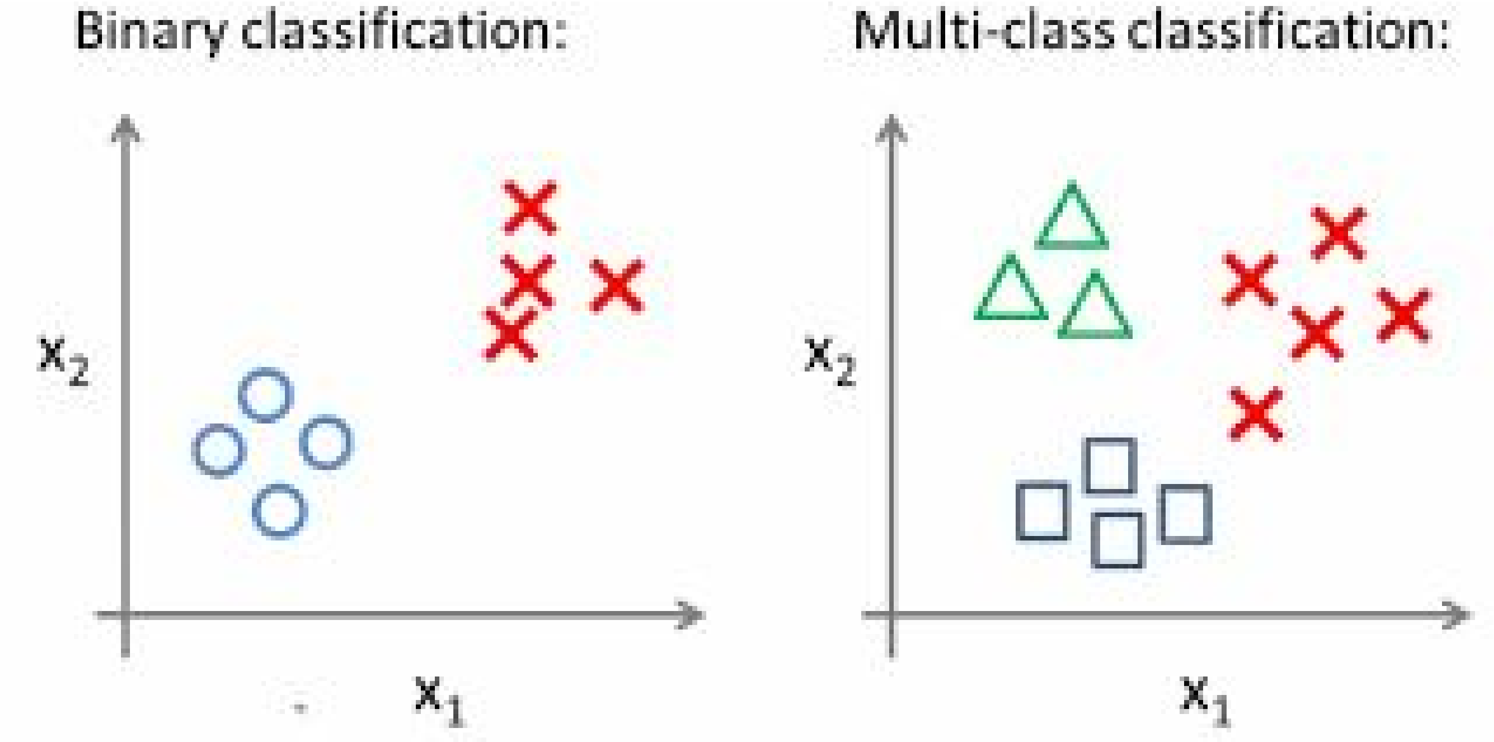
Multilabel and Multiclass classification

- **Multiclass**: classifying more than 2 classes. For example, classifying digits.
- **Multilabel**: assigning a set of topics to each sample. For example, assignment of topics to an article.
- **Multioutput-multiclass**: fixed number of output variables, each of which can take on arbitrary number of values. For example, predicting a fruit and its color, where each fruit can take on arbitrary set of values from {‘blue’, ‘orange’, ‘green’, ‘white’,...}.

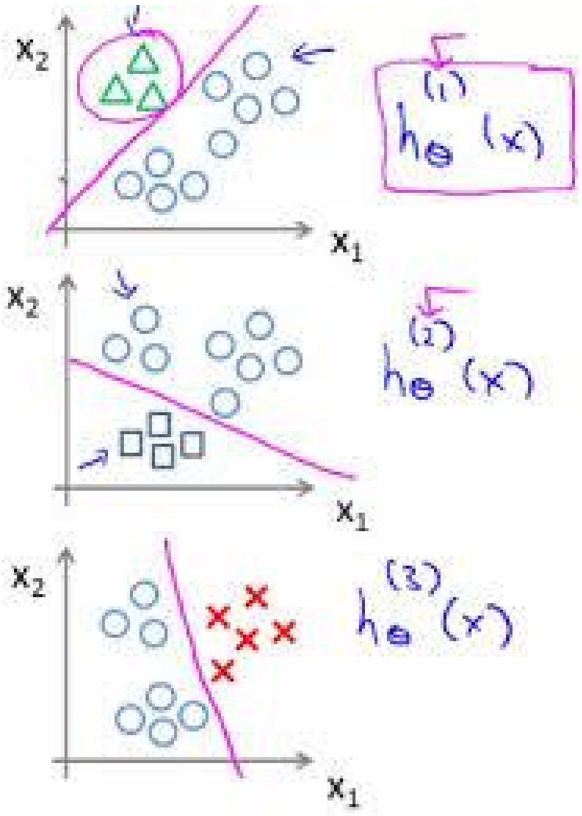
Multilabel and Multiclass classification

- Inherent Multiclass: Naive Bayes, LDA/QDA, DT, Random Forest, kNN
- One-vs-Rest
- One-vs-One
- Error-Correcting Output Codes

One-vs-Rest (OVR)



One-vs-Rest (OVR)



One-vs-Rest (OVR)

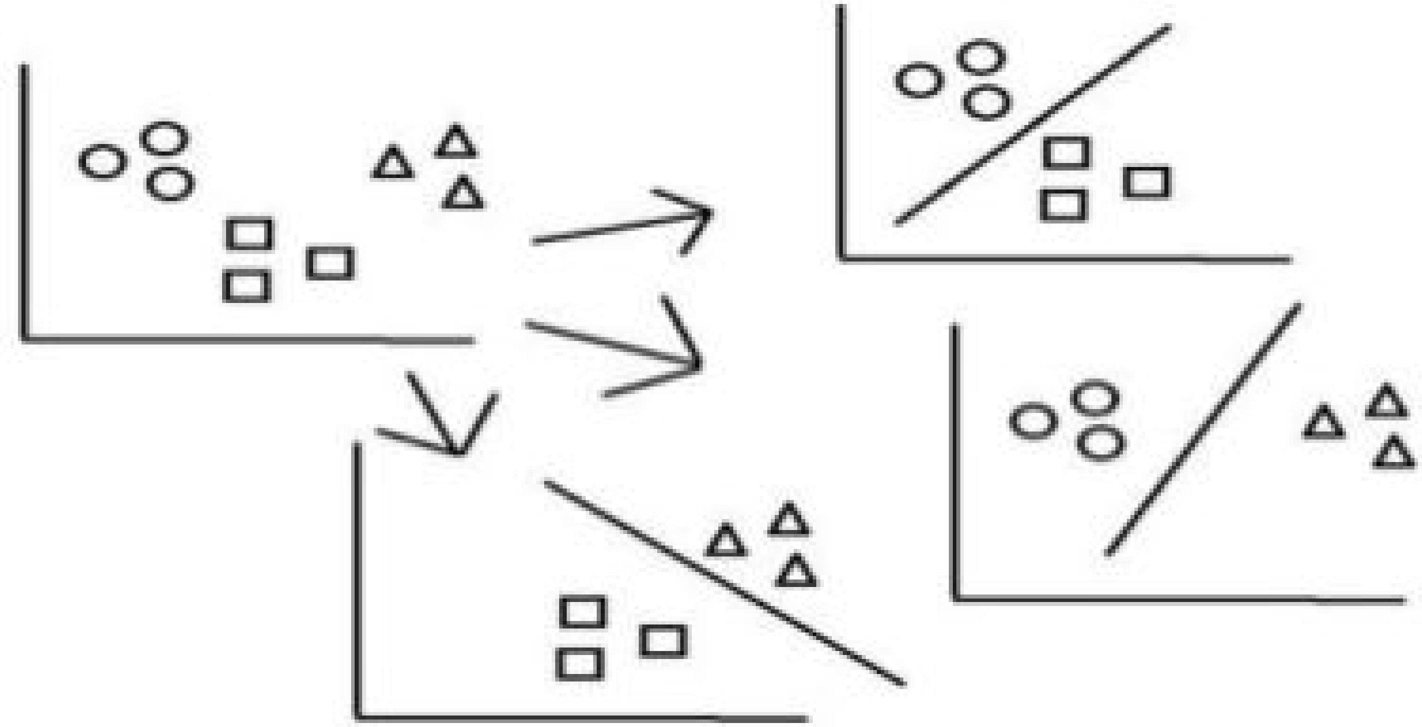
<u>Training:</u> Fits one classifier per class against all other data as a negative class. In total K classifiers.

<u>Prediction:</u> applies K classifiers to a new data point. Selects the one that got a positive class. In case of ties, selects the class with highest confidence.

Pros:

- Efficient
- Interpretable

One-vs-One (OVO)



One-vs-One (OVO)

Training: Fits (K-1) classifier per class against each other class. In total K*(K-1)/2 classifiers.

Prediction: applies K*(K-1)/2 classifiers to a new data point. Selects the class that got the majority of votes (“+1”). In case of ties, selects the class with highest confidence.

Pros:

- Used for Kernel algorithms (e.g. “SVM”).

Cons:

- Not as fast as OVR

Error-Correcting Output Codes (ECOC)

<u>Training:</u> 1) Obtain a binary codeword for each class of length *c*. 2) Learn a separate binary classifier for each position of a codeword. In total, *c* classifiers.

<u>Prediction:</u> Apply *c* classifiers to a new data point and select the class closest to a datapoint by Hamming distance.

Class	Code Word					
	vl	hl	dl	cc	ol	or
0	0	0	0	1	0	0
1	1	0	0	0	0	0
2	0	1	1	0	1	0
3	0	0	0	0	1	0
4	1	1	0	0	0	0
5	1	1	0	0	1	0
6	0	0	1	1	0	1
7	0	0	1	0	0	0
8	0	0	0	1	0	0
9	0	0	1	1	0	0

Column position	Abbreviation	Meaning
1	vl	contains vertical line
2	hl	contains horizontal line
3	dl	contains diagonal line
4	cc	contains closed curve
5	ol	contains curve open to left
6	or	contains curve open to right

Error-Correcting Output Codes (ECOC)

How to obtain codewords?

1) Row separation
2) Column separation

Class	Code Word														
	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
0	1	1	0	0	0	0	1	0	1	0	0	1	1	0	1
1	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0
2	1	0	0	1	0	0	0	1	1	1	1	0	1	0	1
3	0	0	1	1	0	1	1	1	0	0	0	0	1	0	1
4	1	1	1	0	1	0	1	1	0	0	1	0	0	0	1
5	0	1	0	0	1	1	0	1	1	1	0	0	0	0	1
6	1	0	1	1	1	0	0	0	0	1	0	1	0	0	1
7	0	0	0	1	1	1	1	0	1	0	1	1	0	0	1
8	1	1	0	1	0	1	1	0	0	1	0	0	0	1	1
9	0	1	1	1	0	0	0	0	1	0	1	0	0	1	1

Pros:

- Can be more correct than OVR

Multilabel and Multiclass classification

- Inherent Multiclass: Naive Bayes, LDA/QDA, DT, Random Forest, kNN
- One-vs-Rest
- One-vs-One
- Error-Correcting Output Codes