Régression logistique - suite

et autres technique de validation croisé

Rafik LACHAAL

La régression Softmax ou multinomial

Le modèle de régression logistique peut être généralisé de manière à prendre en compte plusieurs classes directement sans avoir à entrainer plusieurs classificateurs binaire puis à les combiner. C'est la régression Softmax ou multinomial.

- Pour chacune des prédictions que l'on souhaite faire il y aura autant de probabilités calculées que de classes possibles.
- La matrice de confusion aura autant de lignes et de colonnes qu'il y a de classes.
- On pourra afficher avec python le taux de précision (vrais positif) avec la fonction accuracy_score().

Validation croisé

La méthode d'usage pour valider notre modèle est de partitionné mos données en trois partie :

- Apprentissage : 60%
- Validation : 20%
- Test : 20%

L'inconvénient c'est que l'on a une perte d'information l'or de notre apprentissage.

On peut aussi observé une perte de représentativité.

Autres technique de validation croisé

Autre méthode : la validation croisée

Permet d'utiliser l'intégralité du jeu de données pour l'entrainement et la validation

- « Leave-k-out crosse-validation »
- « Leave-one-out crosse-validation »
- « Stratified k-fold crosse-validation »

Autres technique de validation croisé

Leave-one-out crosse-validation :

On fait autant de groups que de points dans le jeu complet (c'est à dire k = n). Chaque group contient une seul et unique observation.

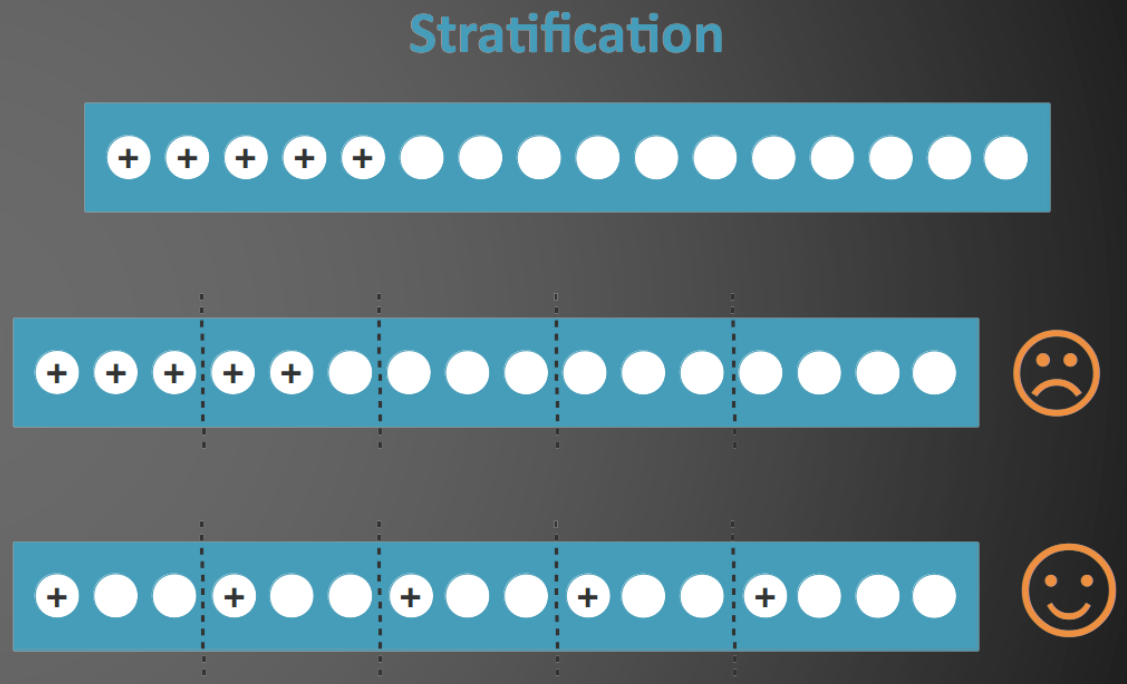
- **Avantage** : du fait qu'on ne mette qu'une seul observation de coté pour chaque apprentissage, on est sûr que l'échantillon d'apprentissage est représentatif de l'échantillon de départ.
- **Problème** :
  - On doit faire autant d'apprentissage que d'observations (100 000 obs = 100 000 modèles) -> on augmente donc le temps de calcule.
  - Des jeux d'entrainement très similaire et des jeux de test très différents. -> qualité des précisions risque de varier

Autres technique de validation croisé

Stratified k-fold crosse-validation :

Dans le cas d'un **problème de classification**, on s'efforce généralement de créer des groups qui contiennent à peu près les **mêmes proportions** d'exemples de chaque classe que le jeu de données complet.

On cherche à éviter qu'un jeu d'entraînement ne contienne que des exemples positifs et que le jeu de test correspondant ne contienne que des exemples négatifs, ce qui va affecter négativement la performance du modèle !



Techniques de ré-échantillonnage

Classification Supervisée et Asymétrie :

Au moment de l'apprentissage (et non pas de l'évaluation), on peut **compenser le déséquilibre** entre les classes dans le jeu d'entraînement en utilisant une méthode de ré-échantillonnage :

- On tire aléatoirement parmi la classe majoritaire autant d'observations que dans la classe minoritaire, ce qui crée un jeu équilibré, opération que l'on répète de nombreuses fois.
- On crée ainsi plusieurs modèles, que l'on peut ensuite combiner en moyennant leurs scores ou en choisissant l'étiquette la plus fréquemment prédite.

Techniques de ré-échantillonnage

Concrètement :

- Avec pandas :
  - Apprentissage, validation , test : df.sample(frac=0.1, replace=True)
- Avec Sickit-learn.model_selection :
  - Leave-k-out crosse-validation : KFold(n_splits=5)
  - Stratified k-fold crosse-validation : stratifiedKFold(n_splits=3)