Machine Learning - TP2

K-NN pour la classification et la régression, Construction et Évaluation de Modèles.

Avant de quitter la classe, veuillez enregistrer votre notebook sous le nom de `TP2 <Prénom> <Nom>.ipynb' et le soumettre via la plateforme.

Estimateurs dans scikit-learn

Tous les algorithmes dans scikit-learn, qu'il s'agisse de prétraitement, d'apprentissage supervisé ou non supervisé, héritent de la classe BaseEstimator. Lorsqu'ils sont instanciés, ces objets de classe sont généralement appelés des estimateurs.

Pour appliquer un estimateur, qu'il s'agisse d'un classifieur ou d'un régresseur, vous devez d'abord instancier un objet de la classe respective.

Par exemple, si vous voulez utiliser un classifieur des k-NN alors vous devez instancier la classe KNeighborsClassifier du module sklearn.neighbors comme suit :

```
from sklearn.neighbors import KNeighborsClassifier
estimator = KNeighborsClassifier(n_neighbors=3)
```

Comme le montre l'exemple, vous devez passer tous les paramètres du modèle en entrée, tels que le nombre de voisins les plus proches désirés (k), lors de la construction de l'estimateur.

Tout estimateur a certaines méthodes de la classe de base. Chaque estimateur a la méthode fit(data_entraînement, étiquettes). La méthode fit est utilisée pour construire et adapter le modèle sur les données d'entraînement. En cas d'algorithmes non supervisés, les étiquettes sont nulles. Les algorithmes d'apprentissage supervisé et les méthodes de regroupement ont également la méthode predict(nouvelles_données) pour prédire de nouvelles propriétés sur de nouvelles données invisibles. Les méthodes supervisées ont la méthode score(données_test, étiquettes_test) pour mesurer leurs performances sur des données invisibles. Les transformateurs (algorithmes comme le prétraitement, la sélection/extraction de caractéristiques, la réduction de la dimensionnalité) ont également la méthode transform(données). Le tableau suivant résume l'API de l'estimateur.

Méthode	Estimateur
estimator.fit(X_train, y_train)	Tous
estimator.predict(X_test)	Classification, Régression, Regroupement
estimator.score(X_test, y_test)	Classification, Régression

Méthode	Estimateur
estimator.transform(X)	Prétraitement, Réduction de la dimensionnalité, Sélection de caractéristiques, Extraction de caractéristiques

Pour le regroupement, les méthodes fit et predict sont combinées dans la méthode fit_predict(X, none).

Comme pour la famille d'algorithmes des k-NN, chaque famille a son propre module dans scikit-learn. Ainsi, vous pouvez vouloir rechercher les implémentations des algorithmes linéaires dans sklearn.linear_model, les machines à vecteurs de support dans sklearn.svm, les arbres de décision dans sklearn.tree, etc.

Jeux de données d'exemple dans scikit-learn

Scikit-learn fournit une série de jeux de données d'exemple pour chaque catégorie d'apprentissage spécifique. Vous pouvez trouver les jeux de données d'exemple dans le module sklearn.datasets. De plus, vous pouvez créer vos propres jeux de données en utilisant les méthodes dédiées fournies dans le même module.

Notez que dans scikit-learn, par convention, les vecteurs de caractéristiques sont désignés par X majuscule, tandis que la caractéristique cible est désignée par y minuscule. Les ensembles de données sont en outre divisés en ensemble d'entraînement désigné par X_train et y_train, et ensemble de test désigné par X_test et y_test.

Exercices

N'oubliez pas d'ajouter des commentaires et des explications si nécessaire ! Dans Jupyter, les en-têtes et les descriptions textuelles doivent être écrites dans des cellules Markdown.

Exercice 1: Application du classifieur k-NN de scikit-learn

Dans cet exercice, vous allez appliquer l'estimateur de classifieur k-NN sur un jeu de données d'exemple avec deux classes et deux caractéristiques (informatives), que vous allez créer. À cette fin, appelez la méthode sklearn.datasets.make_classification() (voir l'API pour plus de détails) et attribuez le résultat au vecteur de caractéristiques X et à la cible y. Visualisez le jeu de données en utilisant la méthode scatter du module matplotlib.pyplot. N'oubliez pas d'appeler matplotlib.pyplot.show() pour afficher le diagramme et de définir les attributs supplémentaires pour nommer les axes et le titre du diagramme.

Ce jeu de données doit être divisé en ensemble d'entraînement et ensemble de test. Pour ce faire, utilisez la méthode sklearn.model_selection.train_test_split() avec les paramètres d'entrée X, y, taille du test, état aléatoire. Cette méthode effectuera une division aléatoire des données. Pour rendre votre expérience reproductible, définissez le paramètre d'état aléatoire sur une constante, par exemple 42. De plus, définissez la taille du test de manière à conserver 70 % de vos données pour l'entraînement et le reste pour le test.

Vérifiez la taille de votre ensemble de données, le nombre de vos caractéristiques et les différentes classes dans les valeurs cibles, en appelant les méthodes appropriées et en affichant les attributs appropriés.

Entraînez un classifieur 3-NN sur votre ensemble de données d'entraînement, prédisez les étiquettes pour votre ensemble de données de test et trouvez le score du modèle sur l'ensemble de données de test. Quel type de distance est utilisé pour trouver les voisins les plus proches ? Quel type de mesure d'évaluation est utilisé dans la méthode score ?

Tracez la courbe d'apprentissage pour votre modèle, en utilisant des tailles augmentées de sous-ensembles de vos données d'entraînement (par exemple, 10 %, 20 %, etc.). Expliquez ce que vous voyez et pourquoi vous croyez que cela se produit. Remarque : Vous pouvez utiliser la méthode learning_curve du module sklearn.model_selection, ou implémenter votre propre fonction.

Exercice 2: Implémentez votre propre régresseur k-NN en Python

Dans cet exercice, vous êtes invité à construire votre propre régresseur K-NN en Python. Permettez à l'utilisateur de choisir sa distance métrique préférée (entre la distance de Manhattan et la distance euclidienne). Pour l'évaluation du modèle, implémentez l'erreur quadratique moyenne.

Testez votre modèle sur le jeu de données de test de type onde que vous créerez en appelant la méthode sklearn.datasets.make_regression(n_samples=100) et en attribuant le résultat à X et y (comme précédemment). Divisez à nouveau le jeu de données en données d'entraînement/test avec un ratio de (70/30). Utilisez 3 valeurs différentes de k et comparez les scores que vous obtenez. Pour quel paramètre obtenez-vous le meilleur modèle ? Expliquez les raisons selon vous.

Exercice 3: Validation croisée Kfold

Pour l'exercice 1, effectuez une validation croisée Kfold pour sélectionner le meilleur k pour votre modèle. Quelle est la valeur de K pour les plis de la validation croisée que vous avez sélectionnés ? Quelles sont les valeurs pour le nombre de voisins les plus

proches que vous avez sélectionnées pour vérification ? Retraînez et retestez votre modèle pour le meilleur k sélectionné.

Remarque : Pour effectuer la validation croisée KFold, utilisez l'optimiseur hyperparamétrique sklearn.model_selection.GridSearchCV.