Curriculum détaillé du cours

Introduction à l'apprentissage automatique et à l'intelligence artificielle

Contenu du document

Ce document cherche à définir plus précisément les différentes étapes du contenu théorique et pratique appris dans le cours d'introduction à l'apprentissage automatique et à l'intelligence artificielle. Pour chacune des leçons, les objectifs pratiques, les notions apprises et les critères de réussite seront énumérés.

Leçons

1. L'écosystème Python pour l'apprentissage automatique

Objectifs pratiques

- Installer l'environnement de travail avec Google Colaboratory.
- Utiliser un Notebook Python.

Notions

- Introduire Google Colaboratory.
- Survoler les fonctionnalités d'un Notebook Python.
- Apprendre comment ajouter du texte et des descriptions dans un Notebook avec le langage Markdown.

Critères de réussite

- Pouvoir ouvrir un Notebook Python avec Google Colaboratory.
- Montrer une maîtrise des fonctionnalités d'un Notebook.
- Maîtriser et démontrer une utilisation du langage Markdown dans les Notebooks.

2. Python et Scipy 101

Objectifs pratiques

- Utiliser Python dans un environnement de développement pour l'apprentissage automatique.
- Programmer avec le langage informatique Python.
- Se familiariser avec les librairies importantes (numpy, pandas et matplotlib).

Notions

- Python
    - Type de données (attribution simple, attribution multiple, valeurs nulles).
    - Contrôles de conditions (if-else, boucles pour, boucles quand).
    - Structure de données (listes, dictionnaires).
    - Fonctions
- Numpy
    - Définir un array.
    - Accéder aux valeurs dans un array.
    - Effectuer des calculs arithmétiques avec des arrays.
- Matplotlib
    - Créer des graphiques de base.
- Pandas
    - Comprendre les séries avec Pandas.
    - Introduction aux tables de données dans Pandas.
    - Accéder aux données d'une table de données Pandas (loc, iloc).

Critères de réussite

- Être en mesure d'écrire du code avec le langage de programmation Python.
- Acquérir la capacité à lire et interpréter le code Python.
- Pouvoir déboguer soi-même les erreurs obtenues en développant du code Python.
- Savoir quand utiliser les librairies principales utilisées dans la manipulation des données avec Python (Numpy, Pandas, Matplotlib).
- Démontrer la maîtrise des librairies de base pour l'analyse de données.

3. Charger les données

Objectifs pratiques

- Télécharger un jeu de données dans un Notebook à partir d'un fichier .CSV.
- Expérimenter le téléchargement de données avec Pandas.
- Explorer le jeu de données importé.
- Effectuer des modifications simples sur le jeu de données de base.

Notions

- Vérifier l'intégrité du jeu de données (*shape*).
- Obtenir un aperçu du jeu de données (*head, tail*).
- Manipuler et renommer les colonnes (*columns*).

Critères de réussite

- Être capable de télécharger un jeu de données avec succès.

- Explorer sommairement la structure du jeu de données pour connaître son intégrité.
- Pouvoir confirmer que le jeu de données entier a été téléchargé en le comparant avec le fichier .CSV initial ou avec l'information disponible à son sujet au préalable.
- Pouvoir changer les noms des colonnes d'un jeu de données.
- Maîtriser les critères de base quant au format idéal de noms de colonnes.

4. Statistiques descriptives

Objectifs pratiques

- Obtenir un aperçu des données.
- Réviser les dimensions du jeu de données.
- Identifier les types de variables dans le jeu de données.
- Faire un sommaire des classes de la variable à prédire.
- Effectuer des statistiques descriptives sur les variables du jeu de données.
- Observer la distribution des variables du jeu de données.

Notions

- Utilisation des fonctions Python pour obtenir un aperçu initial des données (*head, tail, dtypes, info, shape, index, columns*).
- Obtenir les statistiques descriptives sommaires (*describe*).
- Faire un sommaire des classes à prédire (*groupby, size*).
- Établir des corrélation entre les variables continues du jeu de données (*corr*).
- Créer des graphiques de base de distribution.
- Obtenir les statistiques numériques de l'asymétrie des données.

Critères de réussite

- Maîtriser les étapes préliminaires d'exploration des données.
- Être capable d'interpréter concrètement les statistiques de base obtenues.
- Interpréter la distribution des classes ou valeurs à prédire.
- Pouvoir programmer l'ensemble des statistiques descriptives pour un jeu de données.
- Identifier clairement comment les corrélations entre les variables pourraient affecter les résultats prédictifs éventuellement.
- Décrire le rôle des analyses de distribution et comment elles affectent les modèles.

5. Visualisation des données

Objectifs pratiques

- Créer des graphiques pour visualiser les données.
- Utiliser les librairies Python pour la visualisation de données.
- Personnaliser les graphiques selon les données.

Notions

- Créer un histogramme de distribution (*displot*).
- Créer un graphique de densité (*displot*).
- Créer un diagramme boîte à moustache (*boxplot*).
- Créer un diagramme à barre (*barplot*).
- Produire une matrice de corrélation pour les variables continues d'un jeu de données (*corr, heatmap*).
- Produire une matrice de nuages de points (*pairplot*).
- Créer un graphique nuage de points (*scatterplot, lineplot*).

Critères de réussite

- Établir des liens clairs et précis entre les visualisations et les statistiques descriptives.
- Interpréter la distribution des variables d'un jeu de données à l'aide des graphiques de distribution et de densité.
- Maîtriser les librairies de visualisation avec Python (*matplotlib, seaborn*).
- Démontrer l'habileté à produire chaque type de graphiques énumérés dans les notions.
- Être en mesure d'interpréter les graphiques et d'en tirer des conclusions.
- Pouvoir personnaliser les graphiques en fonction du contexte et de la problématique.

6. Préparation des données

Objectifs pratiques

- Explorer les types de données dans un contexte de programmation.
- Se familiariser avec la préparation des données.
- Maîtriser les différentes méthodes de transformation des données.
- Comprendre l'importance de préparer les données.

Notions

- Imprimer les types des variables du jeu de données.
- Identifier les différents types de variables comprises dans le jeu de données.
- Préparer et encoder les variables binaires ou catégoriques textuelles.
    - Encoder les variables binaires manuellement (*Pandas - replace*).
    - Encoder les variables catégoriques ordinales automatiquement (*Scikit-learn - LabelEncoder*) ou manuellement (*Pandas - replace*).
    - Encoder les variables catégoriques avec un encodeur "One-hot" (*Pandas - get_dummies*).
- Transformer les variables du jeu de données.
    - Redimensionner les données (*Scikit-learn - rescale*)
    - Standardiser les données (*Scikit-learn - standardize*)
    - Normaliser les données (*Scikit-learn - normalize*)
    - Binariser les données (*Scikit-learn - binarize*)

- Séparer les variables d'entrée et de sortie manuellement avec l'indexation de listes.

Critères de réussite

- Pouvoir identifier les bénéfices liés à l'étape de préparation des données dans le contexte précis de la problématique.
- Discerner et identifier les différents types de variables dans un jeu de données.
- Encoder toutes les variables binaires ou catégoriques de sorte qu'il n'y ait plus aucune valeur textuelle dans le jeu de données.
- Identifier les méthodes de transformation adéquates pour chacune des variables du jeu de données.
- Pouvoir personnaliser les paramètres des méthodes de transformation.
- Comprendre le fonctionnement de la ou des méthodes de transformation sélectionnées.
- Différencier les variables d'entrée de la variable de sortie dans le jeu de données.

7. Sélection des variables

Objectifs pratiques

- Apprivoiser les méthodes automatiques de sélection de variables.
- Comparer différentes méthodes de sélection.
- Présenter des résultats en tableau Pandas.

Notions

- Sélection univariée (*SelectKBest* et arguments de score)
- Séparer les données en sous-ensemble X et y.
- Élimination récursive des variables (*RFE*).
- Utiliser un algorithme (arbre ou ensemble) pour extraire l'importance des variables.
- Résumer des résultats sous forme de tableau personnalisé.

Critères de réussite

- Pouvoir utiliser des méthodes automatiques pour sélectionner adéquatement les variables les plus importantes dans un jeu de données.
- Pouvoir expliquer la différence entre chaque méthode de sélection de variables.
- Être en mesure de produire des tableaux de résultats personnalisés avec Pandas.

8. Méthodes d'échantillonnage

Objectifs pratiques

- Séparer le jeu de données en ensemble d'entraînement et de test.

- Apprendre à utiliser la validation croisée pour tester la performance d'un modèle et la précision des évaluations.

Notions

- Séparer le jeu de données en ensemble d'entraînement et de test (*train_test_split*).
    a. Définir les arguments de la fonction (*train_test_split*).
    b. Appliquer la séparation du jeu de données sur un algorithme.
    c. Observer la variance potentielle de la précision des prédictions avec (*train_test_split*).
- Utiliser la validation croisée (*cross_val_score*).
    a. Définir les arguments de la fonction (*cross_val_score*).
    b. Obtenir les résultats de la validation croisée (moyenne et écart-type des scores).

Critères de réussite

- Identifier les différences entre séparer le jeu de données en jeux d'entraînement et de test et la validation croisée.
- Démontrer du jugement et du recul dans le choix de la méthode d'échantillonnage.
- Connaître les avantages et les inconvénients liés aux deux méthodes d'échantillonnage.
- Être en mesure d'ajuster les paramètres des méthodes d'échantillonnage.
- Être capable d'interpréter correctement les mesures de performance pour chacune des deux méthodes d'échantillonnage.

9. Évaluation d'algorithmes

- Sélectionner le ou les métriques pour évaluer la performance des modèles.
- Identifier les formules et/ou fonctions Python nécessaires pour utiliser le ou les métriques sélectionnés.

Objectifs pratiques

- Se familiariser avec les métriques d'évaluation utilisées dans les problèmes de classification et de régression.
- Apprendre à interpréter les métriques d'évaluation.
- Être en mesure de sélectionner le bon métrique d'évaluation en fonction de la problématique et de l'algorithme.

Notions

- Métriques de classification
    - Précision de classification
    - Matrice de confusion (vrai positif, vrai négatif, faux positif, faux négatif, erreur de type 1, erreur de type 2).

    - Rapport de classification (score f1, support, sensibilité / retour, spécificité, précision, exactitude).
    - Aire sous la courbe ROC
- Métriques de régression
    - Erreur moyenne absolue (MAE)
    - Erreur quadratique moyenne (MSE / RMSE)
    - R carré (R2)

Critères de réussite

- Distinguer les métriques d'évaluation de classification et de régression.
- Démontrer une démarche statistique derrière le choix d'un métrique d'évaluation.
- Sélectionner le métrique de classification ou de régression adéquat pour la problématique en question en se basant sur le domaine / sujet.
- Pouvoir appliquer les métriques d'évaluation à des algorithmes.
- Interpréter la valeur de sortie d'un métrique d'évaluation dans le contexte de la problématique en jeu.
- Être en mesure d'extraire les valeurs désirées de la matrice de confusion et du rapport de classification.

10. Algorithmes de classification

Objectifs pratiques

- Découvrir différents algorithmes de classification.
- Expérimenter les codes pour les différents algorithmes de classification.
- Identifier manuellement les algorithmes qui performent le mieux pour la problématique et les données utilisées.
- Rouler les algorithmes de façon automatique.
- Visualiser les résultats sous forme de graphiques à pattes (boxplot).

Notions

- Création des modèles de classification.
    - Régression logistique (*Logistic Regression*)
    - Analyse linéaire discriminante (*Linear Discriminant Analysis - LDA*)
    - K plus proches voisins (*k-Nearest Neighbors - KNN*)
    - Classification naïve bayesienne (*Naive Bayes Classifier*)
    - Arbres de décision (*Decision Trees*)
    - Machines à vecteurs de support (*Support Vector Machine - SVM*)
- Comparer les algorithmes d'apprentissage automatique.
- Regrouper les modèles dans un tableau.
- Rouler tous les modèles en même temps avec une boucle.
- Visualiser les performances des modèles (*matplotlib, seaborn*).

Critères de réussite

- Pouvoir rouler tous les modèles de classification énumérés dans ce chapitre.
- Faire la sélection d'un métrique de performance approprié à la problématique et à des algorithmes de classification.
- Parvenir à classer les algorithmes en ordre croissant ou décroissant de performance sur le jeu de données utilisé.
- Être en mesure de programmer une boucle pour pouvoir faire rouler plusieurs algorithmes les uns à la suite des autres automatiquement.
- Démontrer la capacité de rechercher de nouveaux modèles pour la classification dans la documentation de la librairie Scikit-Learn.
- Utiliser la documentation de la librairie Scikit-Learn.
- Différencier un jeu de données de format long ou large.
- Présenter les résultats obtenus en testant les algorithmes avec un graphique boîte de moustaches.
- Déterminer s'il y a un ou plusieurs modèles qui doivent être utilisés pour faire des prédictions et poursuivre vers les étapes suivantes.

11. Algorithmes de régression

Objectifs pratiques

- Découvrir différents algorithmes de régression.
- Expérimenter les codes pour les différents algorithmes de régression.
- Identifier manuellement les algorithmes qui performent le mieux pour la problématique et les données utilisées.
- Rouler les algorithmes de façon automatique.
- Visualiser les résultats sous forme de graphiques à pattes (boxplot).

Notions

- Création des modèles de régression.
    - Régression linéaire (*Linear Regression*)
    - Régression de Crête (*Ridge Regression*)
    - Régression Linéaire Lasso (*Lasso Linear Regression*)
    - Régression Elastic Net (*Elastic Net Regression*)
    - K plus proches voisins (*k-Nearest Neighbors - KNN*)
    - Arbres de décision (*Decision Trees*)
    - Machines à vecteurs de support (*Support Vector Machine - SVM*)
- Comparer les algorithmes d'apprentissage automatique.
- Regrouper les modèles dans un tableau.
- Rouler tous les modèles en même temps avec une boucle.
- Visualiser les performances des modèles (*matplotlib, seaborn*).

Critères de réussite

- Pouvoir rouler tous les modèles de régression énumérés dans ce chapitre.
- Faire la sélection d'un métrique de performance approprié à la problématique et à des algorithmes de régression.
- Parvenir à classer les algorithmes en ordre croissant ou décroissant de performance sur le jeu de données utilisé.
- Être en mesure de programmer une boucle pour pouvoir faire rouler plusieurs algorithmes les uns à la suite des autres automatiquement.
- Démontrer la capacité de rechercher de nouveaux modèles pour la classification dans la documentation de la librairie Scikit-Learn.
- Utiliser la documentation de la librairie Scikit-Learn.
- Différencier un jeu de données de format long ou large.
- Présenter les résultats obtenus en testant les algorithmes avec un graphique boîte de moustaches.
- Déterminer s'il y a un ou plusieurs modèles qui doivent être utilisés pour faire des prédictions et poursuivre vers les étapes suivantes.

12. Automatisation avec les pipelines

Objectifs pratiques

- Comprendre comment utiliser les pipelines.
- Tester les différentes composantes d'un pipeline.
- Modifier et complexifier les pipelines.

Notions

- Utilisation d'un pipeline pour préparer les données.
- Utilisation d'un pipeline pour extraire les variables.
- Utilisation d'un pipeline pour faire rouler un algorithme sur les données.
- Comparer les résultats avec et sans pipeline pour confirmer une utilisation adéquate.

Critères de réussite

- Pouvoir appliquer un pipeline en répliquant une partie ou tout le flux de travail effectué jusqu'à présent.
- Obtenir les mêmes résultats en utilisant un pipeline que lorsqu'on ne l'utilise pas.
- Évaluer la pertinence d'utiliser un pipeline en fonction du flux de travail effectué et les objectifs ultime du projet entrepris.

13. Méthodes de combinaison de modèles

Objectifs pratiques

- Tester différentes méthodes de combinaison de modèles / ou d'ensemble.
- Expérimenter l'utilisation de différents algorithmes dans la combinaison de modèles.

Notions

- Tester l'algorithme de bagging
    - Construire plusieurs modèles de même type à partir de différent sous-ensemble du jeu de données.
- Tester l'algorithme de boosting
    - Construire plusieurs modèles de même type qui vont apprendre à corriger les erreurs des modèles précédemment roulés.
    - Faire la comparaison entre un algorithme de boosting et les modèles de forêt aléatoire (*random forest*) et arbres supplémentaires (*extra trees*).
- Tester l'algorithme de votes
    - Construire plusieurs modèles de types différents qui vont être combinés à l'aide d'une statistique simple pour maximiser la performance.

Critères de réussite

- Évaluer la pertinence d'utiliser une méthode d'ensemble en fonction des résultats obtenus dans la sélection ponctuelle d'algorithmes dans la section précédente.
- Pouvoir déterminer le nombre adéquat d'itérations de modèles à effectuer dans les méthodes de bagging et de boosting.
- Pouvoir déterminer le nombre de modèles à utiliser dans une méthode d'ensemble par vote.
- Statuer sur l'utilité d'une méthode d'ensemble en fonction de la performance obtenue.

14. Optimisation d'algorithmes

Objectifs pratiques

- Explorer les méthodes d'optimisation d'un algorithme.
- Définir les paramètres des fonctions qui permettent d'optimiser les algorithmes.
- Tester l'optimisation avec une recherche par grille (*GridSearch*).
- Tester l'optimisation avec une recherche aléatoire (*RandomSearch*).
- Extraire les paramètres sélectionnés à partir des fonctions d'optimisation.

Notions

- Méthode d'optimisation avec une recherche par grille (*GridSearchCV*).
- Méthode d'optimisation avec une recherche aléatoire (*RandomSearchCV*).

Critères de réussite

- Pouvoir personnaliser l'optimisation d'un algorithme avec les paramètres des algorithmes.
- Montrer la capacité de rechercher la définition et les limites des paramètres qui sont utilisés dans un algorithme en utilisant la documentation de Scikit-Learn.
- Sélectionner les paramètres optimaux pour un algorithme précis en tenant compte de la problématique et des données utilisées.
- Au-delà de l'optimisation d'algorithmes, savoir reconnaître si on doit retourner aux étapes de préparation des données pour optimiser les données de base.

15. Finalisation du modèle

Objectifs pratiques

- Enregistrer un modèle sur Google Drive ou localement sur l'ordinateur.
- Télécharger un modèle dans un Notebook.
- Personnaliser le nom du modèle enregistré avec la date et l'heure.

Notions

- Enregistrer un modèle sur Google Drive avec la librairie Pickle.
- Télécharger un modèle dans un Notebook avec la librairie Pickle.

Critères de réussite

- Parvenir à enregistrer et télécharger un modèle dans un Notebook.
- Personnaliser automatiquement le nom du modèle enregistrer pour éviter les doublons et d'enregistrer par-dessus un modèle pré-existant.