Machine learning I
TP 3 - Classification supervisé

Issam Falih

Instructions : Préparez un rapport incluant le code source et vos résultats, et déposez-le sur Moodle. Il est recommandé d'utliliser streamlit pour afficher vos résultats. Le TP est à faire d'une manière individuelle.

Part I

Dans cette partie du TP, nous allons analyser un extrait de données réelles : le registre des passagers du Titanic. Les données sont divisées en deux ensembles contenant des informations sur les passagers (nom, sexe, âge, classe, port d'embarquement, etc.). Notre objectif sera d'étudier et de construire un modèle de classification prédictif en utilisant le premier ensemble de données, puis de tester et valider le résultat sur le second ensemble afin de prédire qui survivra et qui mourra.

Analyse préliminaire

1. Chargez le fichier "titanic_train.csv" dans un DataFrame pandas, les données sont à télécharger depuis boostcamp. Décrivez les différents attributs :
   - Utilisez la méthode **.info()** pour afficher les types des différents attributs.
   - Vous pouvez utiliser des méthodes comme **.describe()** pour un résumé statistique, et des fonctions des bibliothèques **matplotlib** ou **seaborn** (par exemple **hist()** ou **countplot()**) pour visualiser la distribution des attributs.
   - Commentez la proportion de données manquantes (la méthode **.isnull().sum()** peut être utile ici).
2. Combien de personnes ont survécu et combien sont mortes dans cet ensemble de données ?
   - Vous pouvez utiliser la méthode **.value_counts()** sur la colonne appropriée pour afficher le nombre total de morts et de survivants.
   - En ajoutant l'argument **normalize=True** à la méthode précédente, vous afficherez les pourcentages.

Les femmes et les enfants d'abord !

1. Comparez les taux de survie entre les hommes et les femmes. Commentez.
2. Nous voulons maintenant évaluer les taux de survie en fonction de l'âge des passagers.
   - Ajoutez une colonne "Child" dans votre DataFrame, dont la valeur sera "child" si le passager a moins de 18 ans et "adult" sinon.
   - Évaluez le taux de survie en fonction des valeurs de votre nouvelle colonne (pensez à utiliser `.groupby()`). Commentez.
3. Quelle hypothèse potentiellement biaisée a été considérée dans la question précédente ? Proposez une solution pour corriger ce problème et réévaluez le taux de survie en conséquence. Les résultats sont-ils très différents ?

4. Revenez à la variable "Child" de la question B.2. Utilisez la méthode `.groupby()` pour évaluer les taux de survie des passagers en fonction de leur âge ('Child') et de leur sexe. Que pouvez-vous conclure sur l'efficacité de la politique "Les femmes et les enfants d'abord" lors de l'évacuation du Titanic ?

Nous allons maintenant appliquer la classification naïve bayésienne pour prédire les chances de survie des passagers. L'ensemble de données "titanic_test.csv" sera utilisé comme ensemble de validation.

5. Chargez les données du fichier "titanic_test.csv" et ajoutez la colonne "Child" comme dans la question B.2. Convertissez cette variable ainsi que la variable "Survived" en type `'category'` en utilisant la méthode `.astype('category')`. Cette conversion est une bonne pratique et doit être appliquée `aux deux DataFrames`.

6. Importez la classe de classifieur bayésien de votre choix depuis `scikit-learn` (par exemple, 'CategoricalNB' depuis 'sklearn.naive_bayes'). Entraînez votre classifieur sur les données d'entraînement :

   `'classifier.fit(X_train, y_train)'`, où 'X_train' contient les colonnes 'Child' et 'Sex' et 'y_train' la colonne 'Survived'.

   Note : Scikit-learn nécessite des données numériques. Vous devrez probablement transformer vos variables catégorielles en utilisant 'pd.get_dummies()' ou un 'LabelEncoder' avant l'entraînement.

   Décrivez le contenu de l'objet "classifier". Que représentent les probabilités affichées par ses attributs (ex: '.class_prior_') et comment se comparent-elles aux probabilités calculées dans les questions précédentes ? Commentez.

7. Appliquez votre modèle de classification à l'ensemble de validation en utilisant la méthode :

   `'pred = classifier.predict(X_test)'`

   Comparez le vecteur de prédictions avec la vraie solution et calculez la précision (accuracy) de votre classifieur. Vous pouvez utiliser les fonctions du module 'sklearn.metrics'.

Taux de survie et classe sociale

1. Affichez les taux de survie pour les 3 classes de passagers.

2. Le classifieur Naïf Bayésien fonctionne mieux avec des données catégorielles

   (a) Transformez la variable "Pclass" en utilisant la méthode `.astype('category')`.

   (b) Créez un nouvel attribut "Fare2" dont la valeur peut être "¡10", "10-20", "20-30", "30+" ou "NA" en fonction du prix payé par le passager (colonne "Fare"). Utilisez la fonction `pd.cut()` pour cela. N'oubliez pas de convertir également ce nouvel attribut en type 'category'.

3. Les nouveaux attributs "Pclass" et "Fare2" sont-ils indépendants ? Justifiez (vous pouvez utiliser un test du chi-carré ou une table de contingence avec `pd.crosstab()`).

4. Affichez les taux de survie en fonction de la valeur de "Fare2". Commentez.

5. En vous inspirant des questions B.6 et B.7, entraînez un classifieur Naïf Bayésien avec ces deux attributs ('Pclass' et 'Fare2') et évaluez sa précision sur l'ensemble de validation "titanic_test.csv".

   (a) N'oubliez pas d'ajouter et de convertir les variables "Pclass" et "Fare2" dans l'ensemble de validation !

   (b) Comparez les résultats et les performances de ce modèle avec celui de la section B. Commentez.

Modèle mixte et arbres de décision

1. Entraînez un autre classifieur bayésien en utilisant toutes les variables étudiées (enfant/adulte, sexe, classe et tranche de tarif). Comparez les résultats sur l'ensemble de validation avec les deux modèles précédents. Commentez.

2. Nous voulons maintenant essayer un autre classifieur dans le but de surpasser le classifieur bayésien. Nous proposons d'utiliser un algorithme d'arbre de décision.

   (a) Importez la classe `DecisionTreeClassifier` depuis `'sklearn.tree'`.

   (b) Créez et entraînez un modèle en utilisant les variables 'Pclass', 'Fare', 'Age' et 'Sex' de l'ensemble d'entraînement. *Attention à la gestion des données manquantes dans la colonne 'Age'.*

   (c) Utilisez la fonction `plot_tree` de 'sklearn.tree' pour afficher l'arbre. Commentez sa structure.

   (d) Appliquez ce modèle sur l'ensemble de validation. Commentez ses performances.

3. Essayez d'ajouter d'autres attributs comme le port d'embarquement ('Embarked' : S=Southampton, C=Cherbourg, Q=Queenstown), le nombre de frères/sœurs/époux à bord ('SibSp'), le nombre de parents/enfants à bord ('Parch'), ou le numéro de cabine ('Cabin'), et voyez si vous pouvez obtenir un meilleur arbre. Commentez les arbres résultants et leurs performances.

   Attention, il peut y avoir des données manquantes ou des caractères vides pour certains attributs. Pensez à les gérer de manière appropriée (remplacement, suppression, etc.).

Boosting avec XGBoost

4. Importez la classe `XGBClassifier` depuis la bibliothèque `xgboost`.

5. Entraînez un modèle `XGBoost` en utilisant les mêmes variables que celles du meilleur `DecisionTreeClassifier` obtenu précédemment (incluant la gestion des données manquantes et l'encodage des variables catégorielles).

6. Appliquez le modèle sur l'ensemble de validation (`titanic_test.csv`) et calculez sa précision (`Accuracy`).

7. Affichez le graphique d'importance des variables(`plot_importance`) fourni par la librairie `xgboost`. Commentez l'importance relative des attributs.

8. Comparez la performance de l'algorithme XGBoost avec les modèles de Bayes Naïf et d'Arbre de Décision. `Quel est le modèle le plus performant pour prédire la survie des passagers du Titanic ?`

Part II : Risque Cardiaque

Nous allons maintenant appliquer la même méthodologie à un nouveau jeu de données " Heart Disease UCI ". La variable cible à prédire est 'target' (1 pour maladie cardiaque, 0 sinon).

Analyse préliminaire et facteurs de risque

1. Chargez le jeu de données "Heart Disease UCI" depuis boostcamp. Identifiez et commentez la présence de données manquantes.

2. Affichez le nombre et la proportion de patients atteints de la maladie. Commentez l'équilibre des classes.

3. Séparez le jeu de données en un ensemble d'entraînement (80 %) et un ensemble de test (20 %).

Taux de risque selon le sexe et l'âge

1. Calculez et affichez le taux de maladie cardiaque en fonction du sexe (‘sex‘). Commentez.
2. Créez une nouvelle variable catégorielle "Age_Group" en discrétisant l'âge (‘age‘) en tranches de 15 ans (ex: "Moins de 40", "40-55", "Plus de 55").
3. Évaluez le taux de maladie cardiaque en fonction de cette nouvelle variable d'âge.
4. Évaluez les taux de risque simultanément en fonction du sexe et du groupe d'âge (utilisation de ‘.groupby()‘).

Taux de risque et cholestérol

1. Affichez les taux de risque pour les différentes catégories de douleur thoracique (‘cp‘). Commentez.
2. Créez une nouvelle variable catégorielle "Cholesterol_Range" en discrétisant le taux de cholestérol (‘chol‘) en tranches pertinentes (par ex. "<200", "200-240", "240+").
3. Entraînez un classifieur Bayes Naïf en utilisant les attributs : sexe, Age_Group, cp et Cholesterol_Range (n'oubliez pas d'appliquer l'encodage One-Hot/Dummies aux variables catégorielles).
4. Évaluez la précision (Accuracy) de ce modèle sur l'ensemble de test.

Modèle mixte et arbres de décision

1. Les variables numériques comme le cholestérol (‘chol‘) et la fréquence cardiaque maximale (‘thalach‘) doivent être normalisées (ex : ‘StandardScaler‘). Appliquez cette normalisation.
2. Entraînez un classifieur ‘DecisionTreeClassifier‘ en utilisant toutes les variables pertinentes (catégorielles encodées et numériques normalisées).
3. Visualisez l'arbre de décision et identifiez la variable jugée la plus discriminante par l'algorithme.
4. Comparez la performance d'Accuracy de l'Arbre de Décision avec le modèle Bayes Naïf de la section précédente. `Quel est le modèle le plus performant pour prédire le risque cardiaque ?`

Boosting avec XGBoost pour le Risque Cardiaque

5. En vous basant sur la même préparation de données (normalisation, encodage), entraînez un modèle XGBoost (`XGBClassifier`) sur l'ensemble d'entraînement.
6. Évaluez la précision (`Accuracy`) de ce modèle sur l'ensemble de test.
7. Affichez le graphique d'importance des variables du modèle XGBoost. Les variables jugées les plus importantes par XGBoost correspondent-elles à celles de l'Arbre de Décision ? Commentez.
8. Synthétisez les performances obtenues par les trois modèles (Bayes Naïf, Arbre de Décision, XGBoost). `Quel algorithme est le plus efficace pour la classification du risque cardiaque sur ce jeu de données ?`