Projet Science des données

Gilles Vanwormhoudt, Christelle Garnier

6 juin 2023

1 Descriptif du projet

L'objectif est de vous placer dans le contexte d'un projet de science des données de type **classification supervisée**. Il s'agit de vous faire pratiquer les principales étapes de ce type de projet en s'appuyant sur des données réelles comme celles que l'on peut trouver dans différents entrepôts en accès libre.

Vous allez travailler sur un jeu de données disponible sur MyLearningSpace, un fichier présentant les différentes variables est également disponible. Pour des raisons didactiques, nous avons légèrement modifié ce jeu de données mais la plupart des valeurs sont d'origine.

Il est important de préciser que le travail que vous allez accomplir au cours de ce projet n'est pas spécifiquement lié au jeu de données sélectionné. On retrouve systématiquement la plupart des étapes de ce travail dans un projet de classification de données. Il est donc conseillé de bien comprendre l'objectif de chaque étape et la façon d'atteindre cet objectif. Cela vous servira certainement dans le cadre d'autres projets de même nature.

2 Travail à réaliser

Le travail à réaliser se décompose en plusieurs étapes qui sont caractéristiques d'un projet de science des données (comme indiqué dans le cours). Il y a généralement des étapes consacrées à la compréhension des données, à la préparation des données, à l'entraînement et la comparaison de différents modèles puis à la sauvegarde du meilleur modèle obtenu pour l'inclure dans le système en production.

Les étapes sont décrites dans les sections suivantes. Il est conseillé de suivre l'ordre indiqué. Pour chaque étape, des explications et des indications de fonctions à utiliser dans différentes bibliothèques sont données. Votre travail consiste à comprendre les explications, à explorer la documentation des fonctions mentionnées et à produire le code pour réaliser l'objectif de l'étape. A chaque étape, pensez à vérifier que votre code produit bien le résultat attendu. Pour écrire ce code, il est parfois nécessaire de s'appuyer sur le code écrit lors d'étapes précédentes. C'est à vous d'évaluer le code qui doit être réutilisé.

En fin de projet, il faudra fournir vos codes commentés ainsi qu'un rapport clair et concis rédigé à la manière d'un compte-rendu d'étude. En particulier, vous vous attacherez à analyser les résultats obtenus et à répondre aux différentes questions posées à chaque étape.

3 Importation des données

Le jeu de données est disponible sur le site du cours sous la forme d'un fichier CSV (Comma separated values). En utilisant la bibliothèque Pandas et sa fonction `read_csv()`, vous devez importer le fichier en mémoire pour permettre la manipulation des données.

Afin de réaliser cet import, prenez le temps d'analyser le fichier de façon à identifier les paramètres à fournir à la fonction `read_csv()`. Si tout s'est bien passé lors de l'importation, la fonction `read_csv()` renvoie un objet de la classe DataFrame.

Assurez-vous que l'importation est correcte en utilisant les fonctionnalités offertes pour les objets `DataFrame` (voir le cours à cet effet), notamment l'affichage des informations sur les variables et leur type, ainsi que l'affichage des premières lignes.

Il est important d'identifier les données d'entrée et la variable de sortie correspondant à l'étiquette de classes. Après identification, précisez clairement l'objectif du problème de classification à traiter.

4 Examen des données

Après avoir importé les données, il est important de comprendre leur structure. Plus votre compréhension des données sera fine, plus vous serez en mesure de les transformer pour obtenir des modèles précis. Les éléments à connaître sur vos données sont généralement :

- la taille du jeu de données
- le type des données (numérique : int, float ou qualitatif/catégoriel : object)
- la qualité des données (est-ce qu'il y a des données manquantes ou aberrantes ?)
- la distribution des données
- ...

En utilisant les méthodes de la classe `DataFrame`, procédez à l'examen des données et notez les informations qui vous paraissent pertinentes. En particulier, il est important d'identifier les données qualitatives, les données manquantes (représentées par le symbole 'NA' : Not Available) et les données aberrantes qui devront toutes être pré-traitées avant la classification (pré-traitements à réaliser dans la section 5).

En utilisant la méthode `isna()` de la classe `DataFrame`, vous pouvez détecter s'il existe des données manquantes.

Une façon d'examiner les données à traiter est de construire une visualisation de type histogramme pour chaque variable numérique. Pour rappel, un histogramme montre le nombre d'observations qui appartiennent à chaque plage de valeurs donnée. Etudiez la méthode `hist()` de la classe `DataFrame` et utilisez-la pour obtenir les histogrammes des variables numériques.

Quelles observations pouvez-vous faire à partir de ces diagrammes ? Vous pouvez éventuellement identifier les données aberrantes, c'est-à-dire en dehors de l'échelle de valeurs prises habituellement par une variable.

5 Préparation des données

Pour faire fonctionner correctement les algorithmes de classification, il est nécessaire d'avoir des données de type numérique et de qualité, c'est-à-dire des données qui ne contiennent pas de valeurs manquantes ni de valeurs aberrantes.

Données manquantes : En analysant les données, vous avez dû noter si une variable possède des valeurs manquantes ('NA'). En utilisant la méthode `isna()` de la classe `DataFrame` et la fonction `sum()`, vous pouvez obtenir le nombre de valeurs manquantes pour chacune des variables. Pour résoudre le problème des valeurs manquantes, plusieurs solutions sont possibles :

- se débarraser des échantillons correspondants
- supprimer la variable en question
- attribuer une valeur conforme à la distribution de la variable (moyenne, médiane, valeur la plus probable...)

En général, si le nombre de valeurs manquantes est très important (plus d'un tiers des données environ), on choisit la 2ème solution. La méthode `drop()` de la classe `DataFrame` permet de supprimer une ou plusieurs colonnes facilement. Vous profiterez de cette étape pour supprimer en même temps les colonnes contenant des variables qui ne vous paraissent pas utiles pour résoudre le problème de classification.

Sinon la dernière solution est celle à privilégier. Pour les variables numériques, il suffit de calculer la médiane ou la moyenne de la variable puis de remplacer les valeurs manquantes par le résultat du calcul. Pour les variables qualitatives, de type texte le plus souvent, vous pouvez choisir de remplacer les valeurs NA par le texte le plus fréquent. Dans les deux cas, pour effectuer le remplacement, vous pouvez utiliser la méthode `fillna()` de la classe `DataFrame` (voir la documentation pour les paramètres).

Données aberrantes : Lors de la visualisation des histogrammes, vous avez dû observer si une variable prend des valeurs aberrantes. Il est nécessaire de remplacer ces valeurs aberrantes par des valeurs appropriées.

Transformation des variables qualitatives en variables numériques : Les algorithmes d'apprentissage ne traitent que des grandeurs numériques. Il faut donc transformer les variables qualitatives en variables numériques. Dans

le cas de variables booléennes, le remplacement peut se faire directement avec la méthode `replace()`. Dans les autres cas, il est plus facile d'utiliser la classe `LabelEncoder`. Etudiez cette classe et modifiez les colonnes concernées de l'objet `DataFrame`.

Normalisation des données : Un dernier point concernant la préparation des données est le recalibrage des variables. Généralement, les algorithmes d'apprentissage fonctionnent moins bien lorsque les variables numériques en entrée ont des échelles très différentes. Il est donc intéressant de mettre toutes les variables à la même échelle. Pour réaliser cette opération, Scikit-Learn offre plusieurs méthodes mais nous proposons d'utiliser la classe `StandardScaler`. Etudiez cette classe et utilisez-la pour normaliser l'ensemble des données d'entrée (qui devront d'abord être transformées en tableau Numpy).

Après avoir réalisé toutes ces transformations, il est intéressant d'examiner à nouveau toutes les variables qui seront utilisées pour la classification.

6 Recherche de corrélations

Pour aller plus loin dans l'analyse des données, il faut s'intéresser aux relations qui existent entre les variables. Pour cela, on peut calculer le coefficient de corrélation entre chaque couple de variables numériques en utilisant la méthode `corr()` de la classe `DataFrame`. Pour des questions de lisibilité, comme les variables sont nombreuses, il est préférable de récupérer et d'afficher la corrélation d'une variable particulière avec chacune des autres variables.

Rappelez la signification d'un coefficient de corrélation. Quelles variables d'entrée sont les plus corrélées entre elles ? les plus corrélées avec la variable de sortie représentant la classe ? A votre avis, quelles sont les variables d'entrée les plus pertinentes pour la classification ?

Une autre façon de vérifier les corrélations entre variables est d'utiliser la fonction `scatter_matrix()` de Pandas. Cette fonction croise deux à deux les variables numériques et affiche les nuages de points correspondants. Là aussi comme il y a de nombreuses variables, il faut se concentrer sur un sous-ensemble de variables, notamment celles que vous avez identifiées comme étant les plus prometteuses pour la classification. Après avoir sélectionné les variables à visualiser, utilisez la fonction `scatter_matrix()`.

Commentez la visualisation ainsi obtenue.

7 Extraction des jeux d'apprentissage et de test

Pour entraîner un modèle et l'évaluer, nous avons besoin d'un jeu d'apprentissage et d'un jeu de test. Créer ces deux jeux consiste à choisir au hasard des éléments dans le jeu de données et à les placer dans deux ensembles distincts. Vous pouvez assez facilement obtenir ces jeux en utilisant les fonctionnalités de

Pandas avec un générateur de nombres aléatoires. Cependant, comme indiqué en cours, Scikit-Learn fournit des fonctions prêtes à l'emploi. La fonction la plus simple est `train_test_split()`. Il existe plusieurs façons de paramétrer cette fonction qui prend en entrée des tableaux Numpy (pour les données d'entrée et de sortie).

Regardez sa documentation puis utilisez cette fonction pour obtenir le jeu d'apprentissage et le jeu de test sous forme de tableaux Numpy.

Quelle proportion du jeu de données initial constitue le jeu d'apprentissage ? le jeu de test ?

8 Entraînement d'un modèle

Nous sommes à présent en mesure d'utiliser le jeu d'apprentissage pour entraîner un modèle destiné à la classification binaire. Il existe de nombreux algorithmes pour ce problème dont celui du Perceptron que nous avons étudié en TP. Dans le cadre de ce projet, nous proposons d'utiliser un autre algorithme basé sur la **régression logistique** (appelé également logit).

Dans la librairie Scikit-Learn, la classe `LogisticRegression()` implante cet algorithme. Entraînez un modèle de régression logistique sur le jeu d'apprentissage en utilisant cette classe.

Il est important de comprendre comment fonctionne l'algorithme. Pour cela, vous pouvez vous aider du document qui présente la méthode de la régression logistique disponible sur MLS, que vous pouvez compléter avec d'autres documents disponibles sur le web. L'idée est de reprendre le principe de la régression linéaire et de l'appliquer à un problème de classification. L'algorithme de régression logistique est basé sur le critère du maximum de vraisemblance. Dans le cas binaire, il suppose que la sortie y peut prendre 2 valeurs : 0 et 1.

Pour vérifier votre compréhension de l'algorithme, répondez aux questions suivantes :

- Hypothèse : Quelle hypothèse fait-on sur le logarithme du rapport des vraisemblances (appelé fonction logit) : $\log \frac{p(y=1|x)}{p(y=0|x)}$?
- Minimisation de la fonction de coût : Quelle technique l'algorithme peut-il utiliser pour minimiser la fonction de coût ?
- Apprentissage : Quels paramètres sont calculés pendant la phase d'apprentissage de l'algorithme ?

9 Evaluation du modèle

Après la phase d'apprentissage, le modèle entraîné peut être utilisé pour prédire les classes sur le jeu de test.

Effectuez des prédictions sur les données d'entrée du jeu de test.

On cherche maintenant à évaluer la performance du modèle en comparant les classes prédites par le modèle avec les classes réelles fournies dans le jeu

de test. Dans un premier temps, vous pouvez simplement écrire une boucle qui affiche pour chaque échantillon la classe prédite et la classe réelle.

Scikit-Learn propose plusieurs métriques pour obtenir une évaluation quantitative du modèle : `accuracy_score()`, `confusion_matrix()`, `precision_score()`, `recall_score()`, `f1_score()`.

Analysez la documentation de ces fonctions puis appliquez les sur votre jeu de test. Rappelez la signification de ces différentes métriques et analysez les résultats obtenus.

10 Amélioration de l'évaluation

Une façon d'améliorer l'évaluation du modèle est de recourir à la méthode de validation croisée. Cette méthode a été expliquée en cours. Elle consiste à découper aléatoirement le jeu d'entraînement en plusieurs sous-ensembles distincts puis à entraîner et à évaluer le modèle en passes successives. A chaque passe, un bloc est réservé pour l'évaluation et les blocs restants sont utilisés pour l'entraînement.

Scikit-Learn fournit des éléments pour appliquer la validation croisée. La classe `KFold` permet de réaliser une validation croisée en K passes et la fonction `cross_val_score()` permet d'obtenir les résultats d'une validation croisée.

Après avoir examiné la documentation de ces deux éléments, utilisez-les pour construire une validation croisée sur le jeu de données. La fonction `cross_val_score()` retourne un objet dans lequel sont stockés les résultats. Analysez ces résultats et comparez avec les résultats obtenus précédemment sans validation croisée.

11 Comparaison avec d'autres algorithmes

En général, il n'est pas immédiat de trouver l'algorithme de classification qui donnera les meilleurs résultats avec votre jeu de données. Il faut souvent essayer plusieurs algorithmes, plusieurs configurations d'hyperparamètres et comparer la précision des modèles correspondants. L'objectif de cette partie est de comparer 3 classifieurs de la libraire Scikit-Learn qui sont :

— la régression logistique étudiée précédemment
— le perceptron étudié pendant le premier TP
— les K plus proches voisins

La classification à l'aide des K plus proches voisins consiste à classer une donnée dans la classe à laquelle appartiennent ses K plus proches voisins dans l'espace des caractéristiques identifiées par apprentissage. Dans Scikit-Learn, cet algorithme est mis en oeuvre par la classe `KNeighborsClassifier`.

Un grand intérêt de la bibliothèque Scikit-Learn est de fournir une même interface pour appliquer de nombreux algorithmes d'apprentissage et pour calculer la précision des modèles obtenus (méthodes `fit()`, `predict()`, `cross_val_score()`, ...). En revanche, chaque algorithme possède un jeu d'hyperparamètres spécifiques

qu'il faudra spécifier lors de la construction de l'objet classifieur. Il est conseillé d'expérimenter plusieurs configurations pour ces hyperparamètres.

En utilisant ces interfaces, nous vous demandons de construire une portion de code qui évalue conjointement les trois algorithmes de classification sur le jeu de données en suivant les 3 étapes suivantes :

— Apprentissage du modèle sur le jeu d'entraînement
— Evaluation du modèle par validation croisée
— Affichage du score

12 Sauvegarde du modèle entraîné

Lorsque vous avez trouvé un bon modèle pour votre problème, il faut penser à sa mise en production dans une application. Cela implique de sauvegarder le modèle entraîné et de vérifier son chargement pour réaliser des prédictions avec de nouvelles données dans le système en production.

Un modèle produit avec la bibliothèque Scikit-Learn peut être sauvegardé en utilisant la bibliothèque de sérialisation d'objets qui s'appelle Pickle. Cette bibliothèque dispose des fonctions `dump()` et `load()` pour respectivement sauvegarder et charger le modèle. Examinez la documentation de ces fonctions pour connaître leurs paramètres et appliquez les.