Projet programmation en Python

Nous vous proposons un sujet sur le traitement de données et la mise en place d'un modèle *d'apprentissage automatique* (aka. machine learning). Ce sujet ressemble à ce qui peut être proposé en entretien d'embauche pour des postes/stages *d'analyse ou fouille de données* (aka. Data Scientist).

> **Attention** : Le travail rendu devra être sous la forme d'un jupyter notebook contenant votre code et les réponses aux questions. Le travail demandé est assez libre et sur certaines parties sera différent pour chacun de vous, il nécessite donc une forte prise d'initiative avec notamment la découverte et l'application de nouveaux concepts au fur et à mesure du projet.

Introduction

Le projet que nous vous proposons concerne la musique. Il a pour but de mieux comprendre ce qui rend une musique populaire. Les données sont des données de chansons sur plus d'un siècle avec de nombreux attributs.

Le projet est séparé en trois parties. La première concerne le traitement des données. La deuxième partie concerne l'exploration des données avec pandas. La troisième partie concerne la mise en place d'un modèle d'apprentissage automatique concernant un sous-ensemble d'attributs cibles (attributs choisis au préalable).

Traitement des données [1] (5 points)

La première partie correspond au nettoyage et traitement des données à l'aide de Pandas. Les données sont disponibles sur Moodle. Elles sont inspirées du Spotify Dataset 1921-2020 de Kaggle. Vous pourrez vous y référer pour avoir plus d'informations sur les variables. Cependant, utilisez bien le *le jeu de données* (aka. dataset) de Moodle et non celui de Kaggle.

1. Présentez les données (description des différents attributs, aperçu des données, types des attributs...)
2. Nettoyez les données (gestion des types, des valeurs manquantes, doublons...) en expliquant chaque étape puis validez ce nettoyage.

Exploration des données [2] (8 points)

La deuxième partie correspond à l'exploration des données et est souvent une des parties les plus importantes lors du travail de Data Scientist. *Le but final est d'étudier la popularité d'une musique*. Ce but doit être un minimum pris en compte dans l'analyse.

1. Réaliser une exploration des données. Cette partie est très libre. Des exemples d'analyses possibles sont le tracé d'histogrammes des distributions, de boxplots de chaque attribut avec moyenne et variance pour les valeurs numériques, heatmap de la correlation entres les attributs, pairplots des attributs, analyse des variables categorielles (nombre de catégories...), analyse temporelle des données, etc. L'objectif étant d'extraire des connaissances sur les données.
   Une partie, toute aussi importante que le tracé de figures, est leurs analyse. Un exemple parmi une multitude est de remarquer le lien entre *energy* et *acousticness*.
2. Analysez votre artiste préféré. A minima, votre analyse doit contenir au minimum le nombre de chansons présentes dans le jeu de données, sa popularité au cours du temps (par année), et ses attributs moyens. Expliquez ces données avec le style de l'artiste, période d'activité, et toute autre information pertinente. Idéalement, poussez plus loin l'analyse avec, par exemple, l'évolution temporelle de certains attributs qui pourrait être liée avec une évolution stylistique de l'artiste.

1. Aka. Data Cleaning.
2. Aka. EDA, Exploratory Data Ananlysis.

Mise en place d'un modèle d'apprentissage automatique (7 points)

Cette partie consiste à mettre en place un modèle d'apprentissage automatique pour prédire la popularité d'une chanson en fonction de ses attributs. Vous devrez choisir 5 attributs pour ce modèle. Le modèle que vous implémenterez sera un modèle type *Forêt aléatoire*[3] avec la librairie `scikit-learn`.

Base de l'apprentissage

Vous trouverez, dans cette section, quelques rudiments de la théorie de l'apprentissage automatique. N'hésitez pas à vous documenter, sans vous perdre bien sûr !

Problème de régression

Le problème étudié dans ce projet est un problème de régression. Il s'agit de prédire un nombre réel, la popularité, à partir d'attributs. De manière plus précise, on veut prédire un réel y à partir d'attributs $X = (x_1, ..., x_p)$, qui peuvent être réels, entiers ou catégoriels. On prédit y grâce à la dépendance f (ou le modèle) qui décrit le lien entre la variable y et les attributs $x_1, \ldots, x_p$. Nous considérerons que des *modèles paramétriques*, c'est-à-dire, la correspondance f dépend de paramètres qui la détermine totalement comme suit :

$$y = f_\theta(x_1, ..., x_p) \tag{1}$$

où θ est le vecteur des paramètres du modèle.

A titre d'exemple, la droite de régression linéaire est un modèle totalement déterminé par deux paramètres qui sont l'ordonnée à l'origine et la pente de la droite.

Dans exemples de régression linéaire sont disponibles en scikit-learn, la documentation est disponible à l'adresse :

☞ `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html`

Entraînement d'un modèle et prédiction

En théorie de l'apprentissage paramétrique, comme dit plus haut, le modèle utilisé est complément déterminé par ses paramètres. La phase dite *d'entraînement* est la phase où l'on calcule les valeurs des paramètres. Par exemple, pour trouver les paramètres optimaux de la régression linéaire, on effectue une méthode des moindres carrées[4] et on obtient les coefficients de la pente et l'ordonnée à l'origine.

Une autre approche, consiste à utiliser notre jeu de données $(y_i, X_i)_{i=1}^n$ pour calculer les valeurs des paramètres du modèle. L'objectif étant de créer un modèle qui soit à la fois performant dans la prédiction du jeu de données mais qui puisse également bien se généraliser à de nouvelles données.

En pratique, on entraîne le modèle en lui donnant les valeurs des attributs X_i et l'observation y_i associée. Puis, on détermine les paramètres en minimisant une fonction d'erreur, par exemple la moyenne des erreurs quadratiques. Une fois ce modèle entraîné, on peut prédire de nouvelles valeurs. On peut par exemple prendre des valeurs d'attributs X non vus par le modèle et obtenir une prédiction y_{pred}.

Evaluation d'un modèle

Pour distinguer un 'bon' modèle d'un 'mauvais' modèle il faut une métrique (ou un critère) ! Elles sont nombreuses. Dans le cas de la régression, une métrique classique est la moyenne des écarts quadratiques :

$$MSE(Y_{true}, Y_{pred}) = \frac{1}{n}\sum_{i=1}^{n}(Y^i_{true} - Y^i_{pred})^2.$$

Dans scikit-learn, il existe une fonction déjà implémentée ici

3. Aka. Random Forest.
4. Une méthode parmi tant d'autres !

Gestion des données

L'évaluation d'un modèle d'apprentissage doit se faire sur des données autres que celles utilisées lors de l'entraînement. En effet, l'objectif n'est pas de proposer un *modèle interpolant* les données d'entraînement mais plutôt un modèle qui prédise les bonnes valeurs sur des données autres que celles de l'entraînement. Quand un modèle retrouve de manière exacte les données sans bien se généraliser à de nouvelles données, on dit que le modèle *sur-paramétré*. Il interpole au lieu de d'apprendre la relation plus profonde entre les attributs et la prédiction.

Pour évaluer le modèle, une approche classique est donc de séparer de manière aléatoire les données en données d'entraînement et données de validation. On prend par exemple 2/3 des données originales qui serviront à entraîner le modèle et 1/3 des données originales qui serviront à évaluer le modèle. Il existe d'autres techniques plus avancées de gestion des données telles que la cross-validation.

Forêt aléatoire avec `scikit-learn`

Dans ce projet, il vous sera demandé d'implémenter un modèle de régression type *forêt aléatoire* avec la librairie scikit-learn. C'est un modèle d'ensemble fondé sur la notion d'arbres. Pour plus de détails sur la théorie, il existe plein d'explications en ligne.

De manière pratique, la documentation de ce modèle est disponible ici. Des exemples sont également fournis dans la documentation. L'idée générale à retenir, et valable pour quasiment tous les modèles de scikit-learn, est que :

1. On choisit le modèle, ici `RandomForestRegressor`.
2. Ensuite on choisit ses paramètres, vous pouvez laisser les valeurs par défaut.
3. On entraîne le modèle avec une fonction fit : `RandomForestRegressor().fit(X,y)`
4. On prédit de nouvelles valeurs avec une fonction `predict` : `RandomForestRegressor().predict(X)`

Avec un modèle type *forêt aléatoire*, il est également possible d'avoir une estimation de l'importance de chaque attribut dans la prédiction. C'est-à-dire qu'on peut obtenir un nombre qui nous dit pour chaque attribut à quel point il est important dans la prédiction. En scikit-learn, c'est l'attribut `feature_importances` du `RandomForestRegressor()` qui le donne.

N.B. : L'algorithme implémenté en `scikt-learn` ne prend pas des variables catégorielles. Il faudra donc faire attention à l'encodage de ces variables (encoding en anglais).

Implémentation

1. Choisissez 5 attributs qui feront partie de votre modèle. Il doit y avoir des attributs numériques et catégoriels. Essayez de justifier un minimum le choix de ses attributs, que ce soit une raison numérique ou totalement subjective. L'idéal étant de mélanger arguments numériques et connaissances métiers (que vous avez de manière intuitive sur la musique).
2. Implémentez et évaluez un modèle `RandomForest` en séparant vos données de manière **aléatoire** en 2/3 de données d'entraînement et 1/3 de données de validation. L'évaluation du modèle se fera avec la MSE. Commentez la valeur obtenue.
3. Comparez les résultats du modèle avec une régression linéaire multiple en `scikit-learn`. Concluez sur l'efficacité relative du modèle forêt aléatoire.
4. Étudiez l'importance des attributs pour le modèle de forêt aléatoire. Est-ce que cela paraît cohérent à votre intuition ?
5. Refaire les étapes précédentes avec seulement les chansons datant d'après 1975.
6. Utilisez une K-fold cross-validation pour entraîner et évaluer votre modèle.
7. (Bonus) Utilisez une méthode de réduction telle que PCA (principal component analysis) pour choisir les 5 attributs du modèle. Comparez et analysez la différence de résultats entre l'entraînement du modèle avec le choix fondé sur le PCA et votre choix initial.
8. (Bonus) Modifier les valeurs de certains paramètres du modèle de forêt aléatoire afin d'en améliorer les performances.
9. (Bonus) Implémentez un autre modèle d'apprentissage de votre choix et comparer les résultats avec celui de forêt aléatoire.