IA pour le génie logiciel

Extraction de métriques et utilisation d'algorithmes prédictifs pour la gestion de projet

Quentin Perez et Christelle Urtado

8 novembre 2021

Objectif de ce TP

L'objectif de ce TP est de montrer un exemple d'utilisation de l'intelligence artificielle dans le domaine du génie logiciel. Nous allons utiliser l'apprentissage supervisé afin de prédire les développeurs expérimentés dans un projet industriel disponible sur GitHub. Nous allons également créer un prédicteur simple du nombre de développeurs expérimentés par rapport à la taille du projet. Les données que nous utiliserons sont des métriques logicielles.

Environnement de travail

Dans ce TP, nous allons travailler avec un environnement Python 3 et avec des *notebooks* Jupyter. Jupyter permet de créer des *notebooks* exécutables via un navigateur web ou des IDE le supportant. L'environnement Jupyter permet l'exécution, la visualisation des résultats et leur sauvegarde. Un carnet peut ainsi être exécuté et sauvegardé avec ses résultats (graphiques ou non) sans avoir à ré-exécuter l'application pour les obtenir. Nous allons également utiliser différents *packages* Python pour :

- Le clonage de projet et la manipulation de données Git, notamment les tags à l'aide de GitPython ;
- L'ouverture de fichiers CSV et la manipulation de données avec Pandas ;
- L'apprentissage machine avec la bibliothèque Scikit-Learn ;
- La visualisation avec Matplotlib.

Mise en place de l'environnement de travail

Avant toute chose vous devez obligatoirement avoir Python 3 d'installé sur votre machine. Si ce n'est pas le cas vous pouvez vous référer à la documentation d'installation ici. L'ensemble du TP est dispo-

nible sur GitHub grâce à l'URL suivante : https://github.com/qperez/TP-Master-MTP-GL-IA4GL. Clonez le dépôt sur votre machine puis placez-vous dans le répertoire.
Afin de s'abstraire des dépendances Python déjà présentes sur votre machine, nous allons créer dans le répertoire du TP un environnement virtuel Python ou Venv. Cet environnement virtuel vous permettra d'installer les dépendances nécessaires au TP.

Création du Venv

Pour la création de l'environnement virtuel, nous allons utiliser la documentation Python disponible ici. Ouvrez un terminal ou une invite de commande et placez-vous dans le répertoire du TP.

Machine Linux / Mac

```
mkdir venv && python3 -m venv ./venv && source venv/bin/activate
```

Machine Windows

```
mkdir venv && c:\\Python35\\python -m venv .\venv && venv\\Scripts\\activate.bat
```

Installation des dépendances Python

La liste des dépendances Python nécessaires à la bonne mise en œuvre du TP se trouve dans le fichier `requirements.txt`. Pour installer les dépendances, utilisez la commande suivante à la racine du répertoire du TP : `pip install -r requirements.txt`

Environnement de développement

Pour ce TP, vous êtes libres d'utiliser l'environnement de développement que vous souhaitez. **Cependant**, nous recommandons très fortement l'utilisation de l'IDE Visual Studio Code pour la suite de ce TP. En effet, cet IDE embarque un serveur Jupyter vous permettant ainsi d'utiliser facilement les *notebooks* Jupyter et d'exécuter le code depuis l'IDE. L'installation du TP et des *plugins* nécessaires dans Visual Studio Code est montrée par la vidéo suivante : https://youtu.be/kPqv0VMiKOc

1 Récupération des métriques de BroadleafCommerce

Nous travaillerons avec les versions majeures et mineures (voir *semantic versioning*) du projet BroadleafCommerce disponibles sur GitHub : `https://github.com/BroadleafCommerce/BroadleafCommerce`. Pour cela, nous allons cloner le dépôt puis récupérer les tags des versions et les filtrer par une expression régulière. Pour chacun de ces tags, "checkouter" la version correspondante. Nous lancerons ensuite l'extraction des métriques à l'aide de l'application Java : `ck` (`https://github.com/mauricioaniche/ck`). Cette application créée par Maurício Aniche est dédiée à l'extraction de plusieurs métriques logicielles dont le nombre de lignes de code que nous utiliserons plus tard.

Dans ce TP, nous allons extraire "manuellement" les métriques. Cependant, les plateformes d'intégration continue comme Jenkins ou les outils d'analyse statique comme SonarQube permettent de calculer des métriques de manière automatique à chaque version releasée ou commitée sur le système de gestion de versions.

Pour notre cas d'étude, nous allons :

1. Utiliser le `package` GitPython et sa documentation pour :
   - Cloner le dépôt Github de BroadleafCommerce à l'endroit indiqué par la variable `PATH_TO_REPO` avec l'aide de la méthode `Repo.clone_from` ;
   - Créer un objet `Repo` (documentation) qui vous permettra de récupérer les tags des versions ;
   - Créer un objet `Git` (documentation) qui vous permettra de "checkouter" la version désirée.
2. Récupérer la liste des tags du dépôt à l'aide de l'objet `Repo`.
3. Itérer sur la liste des tags, où pour chaque tag vous allez :
   - Vérifier par une expression régulière que l'on se situe sur des tags ayant la forme `broadleaf-X-Y-0-GA` où `X` et `Y` peuvent varier entre 0 et 9 et où le tag a une taille fixe (utilisation de `^` et `$` pour matérialiser le début et la fin de la chaîne de caractères) ;
   - Vérifier que l'on ne va pas extraire les métriques d'une version déjà présente dans le dossier `ck_metrics` ;
   - **Une fois ces vérifications effectuées**, "checkouter" la version désirée à l'aide du tag ;
   - Lancer l'extraction des métriques à l'aide de l'instruction python suivante : `subprocess.run(["java", "-jar", "ck.jar", ".." + os.sep + "BroadleafCommerce"])` ;
   - Renommer le fichier `class.csv` généré en `class_[tag].csv` et supprimer les fichiers `variable.csv`, `methode.csv`, `field.csv` créés par `ck`.

2 Utilisation du classifieur de développeurs (Random Forest)

Nous allons maintenant utiliser le classifieur (Random Forest) déjà entraîné et sauvegardé sous le nom `classifier_rf.pkl`. Ce classifieur permet la classification des développeurs en deux catégories :

— les développeurs expérimentés, étiquetés "SSE" pour *Senior Software Engineer* ;

— les développeurs non-expérimentés, étiquetés "NSSE" pour *Non-Senior Software Engineer*.

Pour charger le classifieur, vous utiliserez la fonction `joblib.load` (documentation).

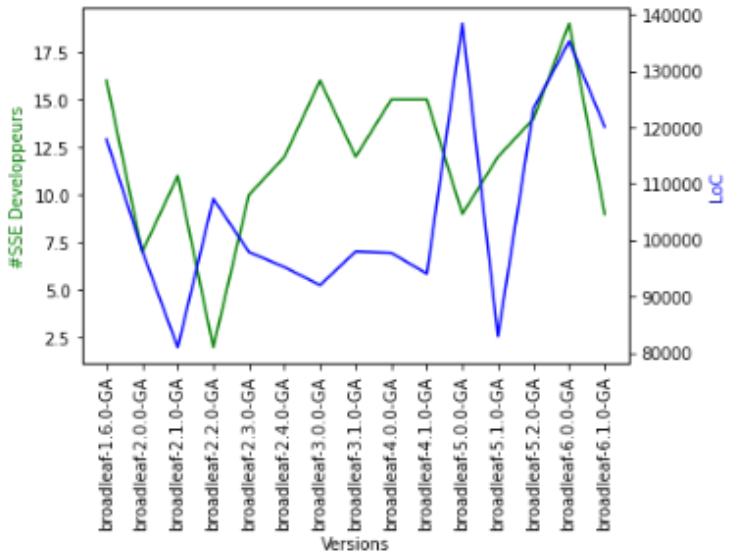
1. Nous allons utiliser les métriques (23) associées à des développeurs, celles-ci sont stockées dans le dossier `metrics_by_dev` . Chaque fichier de ce dossier est nommé en fonction de la version sur laquelle les métriques ont été extraites. Nous allons donc itérer sur la liste de ces fichiers ordonnée par ordre alphanumérique.
2. Pour chaque fichier CSV, l'ouvrir avec Pandas en tant que Dataframe via la fonction : `pd.read_csv` (documentation).
3. Ces métriques sont à l'état "brut" dans les fichiers, c'est-à-dire qu'elles ont des échelles et des unités différentes. Le classifieur Random Forest que nous avons entraîné, lui, ne travaille qu'avec des variables comprises dans $[-1, 1]$. Ici, il va donc falloir faire une mise à l'échelle des variables à l'aide d'un *scaler* de Scikit-Learn : `MinMaxScaler` (documentation). De plus, afin de réduire les écarts de valeurs sur certaines variables, nous allons appliquer un logarithme sur 11 de ces dernières à l'aide de la fonction `log_dataframe` de ce *notebook*.
4. Prédire ensuite la catégorie des développeurs ("SSE" ou "NSSE") puis la stocker dans le dictionnaire `dict_classified_dev`. Faire un affichage à l'aide de `print` pour visualiser l'évolution du nombre de développeurs dans chaque catégorie au fur et à mesure des versions.

3 Extraction du nombre total de lignes de code depuis les fichiers CSV

Nous avons classé les développeurs par catégorie pour chaque version de BroadleafCommerce. L'étape suivante est d'extraire le nombre total de lignes de code pour chaque version de BroadleafCommerce. Pour cela, chacun des fichiers **triés par ordre alphanumérique croissant** doit être ouvert avec Pandas. Vous devez ensuite faire la somme de la colonne "LoC" pour chaque version et l'ajouter à la liste `loc_by_versions`.

4 Nombre de développeurs SSE et nombre de lignes de code par version de BroadleafCommerce

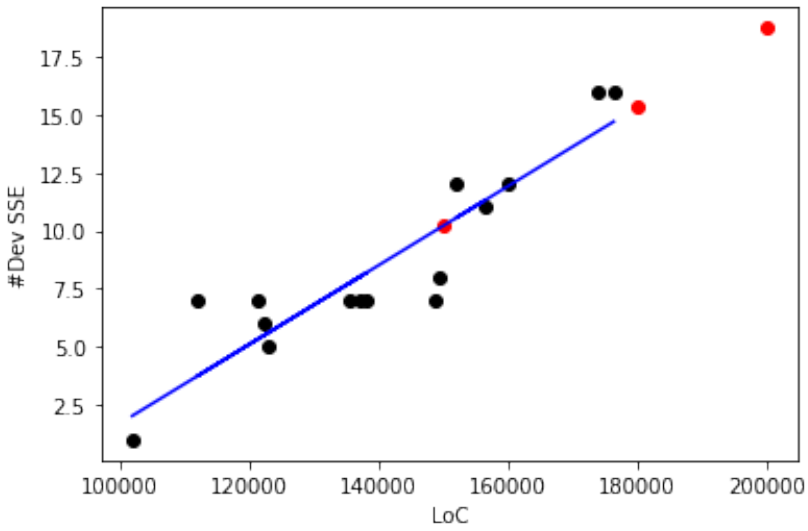
L'objectif ici est de tracer à l'aide du *package* Matplotlib (documentation) un graphique à 3 axes comme montré dans la figure d'exemple ci-dessous :



5 Création d'un prédicteur du nombre de développeurs expérimentés en fonction de la taille du projet

Nous avons maintenant l'ensemble des données permettant de créer un prédicteur du nombre de développeurs expérimentés en fonction du nombre de lignes de code du projet. Pour ce faire, nous allons mettre en œuvre un prédicteur basé sur une régression linéaire : `LinearRegression` (documentation). Ce prédicteur utilise la méthode des moindres carrés afin d'ajuster une droite d'équation $ax + b + \epsilon$ où a est le coefficient directeur, b l'ordonnée à l'origine et ϵ l'erreur liée aux moindres carrés. Pour cela, nous allons :

1. Créer un objet `LinearRegression` et l'ajuster sur `X` et `y` à l'aide de la méthode `fit`.
2. Afficher le coefficient de régression sur `X` et `y`.
3. Déterminer le coefficient de détermination linéaire avec la fonction `r2_score` (documentation) qui mesure l'ajustement entre les prédictions du classifieur sur les données `X` par rapport aux sorties `y`. Plus il est proche de 1, meilleures sont les prédictions.
4. Prédire le nombre de développeurs expérimentés (SSE) pour 150000, 180000 et 200000 lignes de code.
5. Tracer un graphique semblable à la figure d'exemple ci-dessous :


17.5
15.0
12.5
10.0
7.5
5.0
2.5
#Dev SSE
100000
120000
140000
160000
180000
200000
LoC