Projet final MGL869 :

Amélioration du modèle de prédiction des bogues

Automne 2024

L’objectif du projet final est d’améliorer le modèle de prédiction de bogues qui a été développé dans le laboratoire. Pour une telle amélioration, il faut ajouter des nouvelles métriques et utiliser le modèle pour un nouvel cas d'utilisation qui est la priorization des fichiers à tester avant de réaliser une nouvelle version du logiciel Hive. En résumé, ce projet est divisé en deux parties : amélioration du modèle en utilisant des métriques supplémentaires et l’utilisation du modèle pour prioriser les fichiers à tester selon la sévérité des problèmes futurs.

1. **Ajout de métriques pour améliorer le modèle :**

Inspiré de l’étude [1], considérez des métriques liées aux changements de fichiers. Les entrées que vous avez considéré dans le modèle du laboratoire sont basées sur fichier (chaque ligne du fichier de données représente un fichier) dans une version du Hive. Dans cet exercice, ajoutez des métriques liées aux changements du fichier durant la version étudiée. En particulier, collectez les métriques suivantes pour chaque fichier F et une version de Hive V (**À noter que le même fichier F va avoir différentes valeurs des métriques suivantes d’une version à une autre)** :

- Nombre de lignes ajoutées au fichier F dans la version V.
- Nombre de lignes supprimés du fichier F dans la version V.
- Nombre de commits qui ont changé le fichier F dans la version V.
- Parmi les commits précédents, quel est le nombre de commits qui ont corrigé un bug. Pour trouver ces commits, il faut définir un ensemble de mots clés à chercher dans les messages de commits tel que “fix”. Définissez et justifiez vos mots clés.
- Le nombre de commits qui ont changé le fichier F dans la version V et les versions avant V.
- Le nombre de développeurs qui ont changé le fichier F dans la version V.
- Le nombre de développeurs qui ont changé le fichier F dans la version V et toutes les versions avant V.
- Le temps moyen entre chaque deux changements au fichier F dans la version V. Si dans la version V, il y a trois commits c1, c2, c3 qui ont modifié le fichier F. Le temps moyen est la moyenne du temps entre “c1 et c2” et entre “c2 et c3”.
- Même métrique pour chaque fichier et toutes les versions avant V.
- La moyenne de l’expertise des développeurs qui ont changé le fichier F dans la version V. Pour trouver l’expertise d’un développeur, il suffit de calculer le nombre de commits que le développeur en question a réalisé avant la version V.
- Calculer aussi le minimum de l’expertise des développeurs qui ont changé le fichier F dans la version V.

Inspiré de l'étude [2] qui a trouvé un lien entre les changements aux commentaires et la qualité des bogues, collectez les métriques suivantes pour chaque fichier F et version V.

- Quel est le nombre de commits au fichier F durant la version V qui ont changé un commentaire du code.
- Quel est le nombre de commits au fichier F durant la version V qui n'ont pas changé un commentaire du code.

En utilisant ces métriques et le pipeline vu dans le cours, créer, évaluer et interpréter votre modèle. Discuter la différence entre les résultats du modèle de votre laboratoire avec le nouveau modèle.

2. **Utilisation du modèle pour trier les fichiers selon le nombre de bogues future :**

Le modèle du laboratoire et le modèle de question 1 fait une prédiction binaire indiquant si un fichier F dans une version V serait responsable d'un bogue dans les versions suivantes (après V). Modifier votre variable indépendante pour prédire la sévérité du bogue introduit par un fichier. Par exemple, est-ce que le fichier F dans la version V va introduire un bogue de priorité "Blocker", "Critical", "Major", "Minor" **et/ou** "Trivial". La priorité du bogue est dans la plateforme des issues de Hive. Pour ceci, il faut utiliser la régression nominale (Nominal regression) ou de la classification. En utilisant votre modèle, les développeurs peuvent voir quel fichier va causer des problèmes plus sévères et concentrer leurs efforts dans les tests d'un tel fichier.

Construisez votre modèle, évaluez ses performances et son interprétation. Comparez votre modèle avec le modèle de la question 1.

Références :

[1] Kamei, Yasutaka, et al. "A large-scale empirical study of just-in-time quality assurance." IEEE Transactions on Software Engineering 39.6 (2012): 757-773.
[2] Ibrahim, Walid M., et al. "On the relationship between comment update practices and software bugs." Journal of Systems and Software 85.10 (2012): 2293-2304.

3. **Remises et dates limites** :

La qualité de la présentation (10 points).
Qualité du rapport (10 points).

Date limites :

- **28 Novembre et 05 Décembre** - présentation finale.
- **19 Décembre** - remise du rapport finale.

Instructions sur le rapport finale :

Le rapport final doit contenir une section d'introduction, une section détaillant votre méthodologie, une section sur les résultats, une section sur la discussion des résultats et finalement une conclusion. **Le maximum de pages dans le rapport final est 5 pages.**

Évaluation :

Collection des nouvelles métriques	**30**
Construction des modèles	**15**
Construction du modèle de regression nominale	**15**
Analyse des résultats	**20**
Qualité de la présentation et du rapport	**20**
Total	**100%**