وزارة التعليم العالي والبحث العلمي
جامعة الموصل
كلية علوم الحاسوب والرياضيات
قسم البرمجيات

نهج التعلم الفوقي في التنبؤ بإعادة هيكلة البرمجيات

رسالة مقدمة
الى مجلس كلية علوم الحاسوب والرياضيات في جامعة الموصل
كجزء من متطلبات نيل شهادة ماجستير علوم في
البرمجيات

من قبل

رشا أحمد محمود عباس

بإشراف

أ. د. شهباء إبراهيم خليل حمودي

١٤٤٧هـ ٢٠٢٥ م

المستخلص

تُعدُّ عملية إعادة الهيكلة عنصراً اساسياً في عالم تطوير البرمجيات، إذ تهدف الى تحسين جودة الشفرة البرمجية دون التأثير في سلوكها الخارجي. وفي حال غياب عمليات اعادة الهيكلة تصبح الشفرة البرمجية بمرور الوقت اكثر تعقيداً وأقل تنظيماً، مما يزيد من صعوبة فهمها وصيانتها وتعديلها من قبل المطورين. لذلك، يُعدُّ اجراء اعادة الهيكلة بشكل دوري أمرًا ضروريًا للحفاظ على تنظيم الشفرة وجودتها على المدى الطويل. وفي هذا السياق، يؤدي التعلم الفوقي دوراً رئيساً في تحسين التنبؤ بإعادة الهيكلة، إذ يمكن استخدامه لتحسين جودة البرمجيات من خلال بناء نماذج دقيقة قادرة على التنبؤ باعادة الهيكلة، يؤدي ذلك إلى تعزيز الشفرة البرمجية وتقليل المشكلات والتعقيدات المحتملة في أثناء التطوير والصيانة.

تهدف هذه الرسالة الى تطوير نموذج مقترح لتحسين دقة التنبؤ بعمليات اعادة هيكلة البرمجيات على مستوى الفئة Class-Level ومستوى الدالة Method-Level باستخدام تقنيات التعلم الفوقي، تضمن النموذج المقترح مرحلتين: تهدف المرحلة الاولى الى التنبؤ بوجود الحاجة الى إعادة الهيكلة، وذلك من خلال ثلاث تقنيات من التعلم الفوقي، نموذج التكديس Stacking بالاعتماد على النماذج الأساسية Support Vector Classification (SVC)، Radial Basis Function(RBF)، Logistic Regression(LR)، و Extreme Gradient Boosting (XGBoost)، مع نموذج Logistic Regression(LR) كمُتعلم فوقي، نموذج التعزيز Boosting باستخدام خوارزميات Gradient Boosting(GB)، Categorical Boosting (CatBoost)، Light Gradient Boosting Machine (LightGBM)، Adaptive Boosting (AdaBoost). نموذج التعلم باللقطات القليلة Few-shot Learning بالاعتماد على نماذج Model-Agnostic Meta-Learning (MAML)، Reptile Model(RM)، Relation Network Model (RNM)، Prototypical Network Model(PNM) كمُتعلمين أساسيين، ونموذج LR كمُتعلم فوقي. المرحلة الثانية تستخدم لتصنيف انواع اعادة الهيكلة المتوقعة، بناءً على ما تم التنبؤ به في المرحلة الاولى، باستخدام نموذجين هما نموذج الغابة العشوائية RFM ونموذج آلة المتجه الداعم SVM. تم استخدام أربع مجموعات بيانات جُمعت مسبقًا من مشاريع مفتوحة المصدر، حيث تضمنت هذه المجموعات معلومات عن خصائص الشيفرة البرمجية وحالات إعادة الهيكلة، وقد استُخدمت في عملية تدريب النماذج واختبارها.

لتحسين أداء النموذج وزيادة دقته في التنبؤ، تم التركيز على اختيار واستخلاص الميزات الأكثر أهمية من البيانات، وذلك باستخدام أربع خوارزميات مختلفة من خوارزميات تحسين السرب، وهي: خوارزمية تحسين الأسدLOA، خوارزمية القرد العنكبوتيSMO، خوارزمية تحسين السالبSOA، وطريقة هجينة مقترحة SMOA دمجت بين خوارزمية القرد العنكبوتي وخوارزمية تحسين السالب. وقد أظهرت الطريقة الهجينة المقترحة فعالية

عالية في تعزيز دقة النموذج في التنبؤ. تم تدريب نماذج التعلم الفوقي باستخدام بيانات تدريب بنسبة 80% من مجموع البيانات، واختباره على بيانات الاختبار بنسبة 20% من البيانات. أما في المرحلة الثانية، فقد تم تدريب نماذج التصنيف بالاعتماد على البيانات الناتجة من المرحلة الأولى، والتي تمثل مخرجات أفضل نموذج تنبؤي، تم استخدام نسبة 70% من البيانات لتدريب النماذج ، واختباره بنسبة 30% من بيانات الاختبار. تم قياس اداء نماذج التعلم الفوقي باستخدام مقاييس عدة.

أبرزت النتائج أن الطريقة الهجينة أدت الى تحقيق نتائج افضل في تحسين دقة النموذج على التنبؤ لذلك تم اعتمادها لاختيار الميزات في مجاميع البيانات، حققت نتائج تنفيذ النماذج الثلاثة دقة متباينة على مجاميع البيانات، لكن نموذج التعلم باللقطات القليلة اثبت فعاليته لكون لديه القدرة على التعلم من بيانات محدودة. إذ حققت النماذج دقة بنسبة تتراوح بين (98%-1.00%)على مستوى الفئة باستثناء نموذج LR، و حقق دقة بنسبة تتراوح بين (98%-99%) على مستوى الدالة لمجموعة البيانات DTOS1 باستثناء نموذج LR إذ كان الاقل اداءً من بين النماذج المستخدمة، لكن مجموعة البيانات DTOS4، DTOS3 كانت افضل في تحقيق نتائج مثالية، فقد تم تحقيق دقة مثالية بنسبة 1.00% على مستوى الفئة ومستوى الدالة لاغلب النماذج باستثناء نموذج LR. مجموعة البيانات DTOS2 هي الاقل نتائج فقد حققت دقة بنسبة تتراوح بين (95%-98%) على مستوى الفئة، وحققت نسبة (98%-99%) على مستوى الدالة باستثناء نموذج LR. مرحلة تصنيف انواع اعادة الهيكلة اثبت النموذجان RF,SVM كفاءتهم فقد تم تحقيق دقة مثالية لجميع البيانات على مستوى الفئة والدالة باستثناء مجموعة البيانات DTOS2 فقد حققت دقة بنسبة 87% على مستوى الدالة.

Ministry of Higher Education and
Scientific Research
University of Mosul
College of Computer Science and
Mathematics
Department of Software

A Meta-Learning Approach in Predicting Software Refactoring

**A Thesis Submitted to the Council of the College of
Computer Science and Mathematics
University of Mosul
as a Partial Fulfillment of Requirements
for the Degree of Master of Science
in
Software**

By

Rasha Ahmed Mahmoud Abbas

Supervised by

Prof. Dr. Shahbaa Ibraheem Khaleel Hamoodi

2025 A.D. **1447 A.H**

ABSTRACT

Refactoring is a fundamental component of software development, aiming to improve the quality of code without affecting its external behavior. Without refactoring, code becomes more complex and less organized over time, making it more difficult for developers to understand, maintain, and modify. Therefore, performing regular refactoring is essential to maintain code organization and quality over the long term. In this context, meta-learning plays a key role in improving refactoring prediction. It can be used to improve software quality by building accurate models capable of predicting refactorings. This strengthens code and reduces potential problems and complications during development and maintenance. This thesis aims to develop a proposed model to improve the accuracy of predicting software refactoring operations at the Class-Level and Method-Level levels using meta-learning techniques. The proposed model includes two stages: the first stage aims to predict the need for refactoring, using three meta-learning techniques: the Stacking model based on the basic models Support Vector Classification (SVC), Radial Basis Function (RBF), Logistic Regression (LR), and Extreme Gradient Boosting (XGBoost), with the Logistic Regression (LR) model as an meta-learner; the Boosting model using Gradient Boosting (GB), Categorical Boosting (CatBoost), and Light Gradient Boosting Machine (LightGBM), Adaptive Boosting (AdaBoost) algorithms; and the Few-shot Learning model based on Model-Agnostic Meta-Learning (MAML), Reptile Model (RM), Relation Network Model (RNM), and Prototypical Network. The model (PNM) serves as the base learner, and the LR model serves as the meta-learner. The second stage is used to classify the predicted refactoring types based on the predictions made in the first stage, using two models: the Random Forest Model (RFM) and the Support Vector Machine (SVM). Four datasets previously collected from open source projects, containing information on code characteristics and refactoring instances, were used in the model training and testing process.

To improve model performance and increase its prediction accuracy, the focus was on selecting and extracting the most important features from the data. This was done using four different swarm optimization algorithms: the Lion Optimization Algorithm (LOA), the Spider Monkey Algorithm (SMO), the Negative Optimization

Algorithm (SOA), and a proposed hybrid method (SMOA), which combines the Spider Monkey Algorithm and the Negative Optimization Algorithm. The proposed hybrid method demonstrated high effectiveness in enhancing the model's prediction accuracy. The overlearning models were trained using training data representing 80% of the dataset and tested on test data representing 20% of the dataset. In the second phase, classification models were trained using the data from the first phase, which represented the output of the best predictive model. 70% of the data was used to train the models, and 30% of the test data was used to test them. The performance of the meta-learning models was measured using several metrics.

The results showed that the hybrid approach led to better results in improving the model's prediction accuracy, and therefore was adopted for feature selection in the datasets. The results of implementing the three models achieved varying accuracies on the datasets, but the few-shot learning model proved effective due to its ability to learn from limited data. The models achieved accuracies ranging from 1.00% to 98% at the class level, with the exception of the LR model. They achieved accuracies ranging from 99% to 98% at the method level for the DTOS1 dataset, with the exception of the LR model, which was the lowest-performing model. However, the DTOS4 and DTOS3 datasets performed better, achieving optimal results, with most models achieving an accuracy of 1.00% at the class and method levels, with the exception of the LR model. The DTOS2 dataset performed poorly, achieving an accuracy rate of 98%-95% at the class level and 99%-98% at the method level, with the exception of the LR model. In the Refactoring case classification phase, the RF and SVM models proved their efficiency, achieving perfect accuracy for all datasets at the class and function levels, with the exception of the DTOS2 dataset, which achieved an accuracy rate of 87% at the method level.

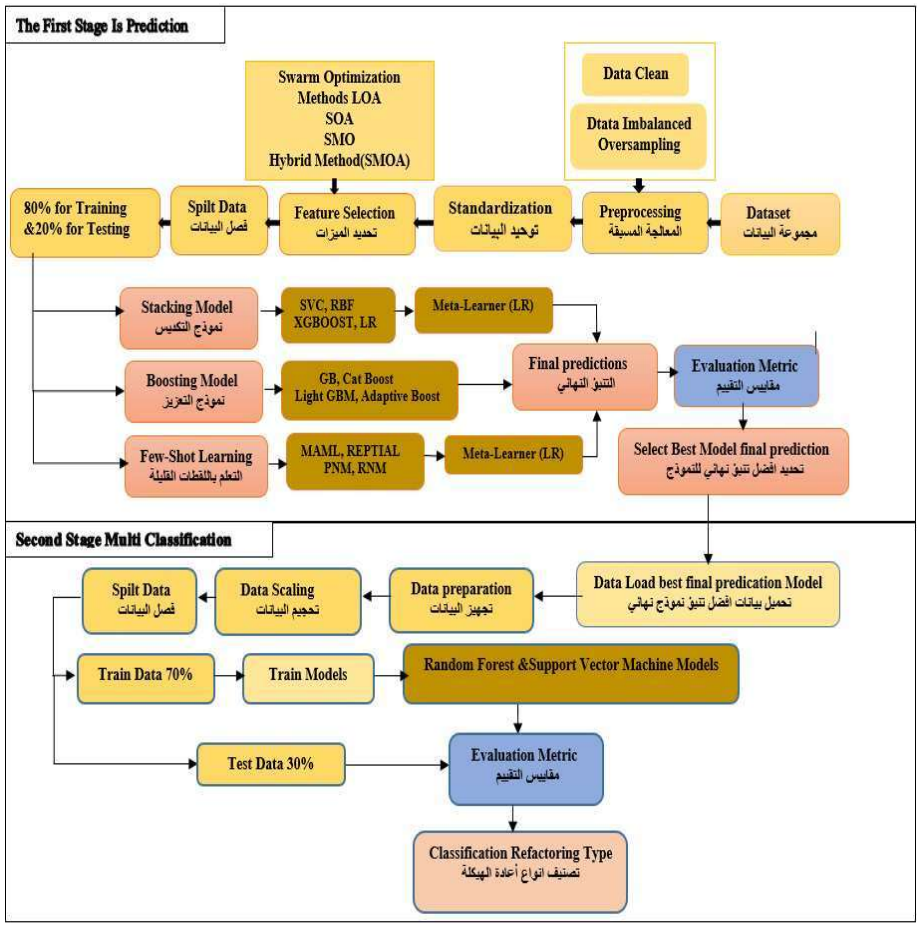
A Meta-Learning Approach in Predicting Software Refactoring

Author: Rasha Ahmed Mahmoud Advisor: Shahbaa Ibraheem Khaleel Publisher: University of Mosul

HIGHLITS

- The models achieved accuracies ranging from 1.00% to 98% at the class level, with the exception of the LR model.
- They achieved accuracies ranging from 99% to 98% at the method level for the DTOS1 dataset, with the exception of the LR model, which was the lowest-performing model. However, the DTOS4 and DTOS3 datasets performed better, achieving optimal results, with most models achieving an accuracy of 1.00% at the class and method levels, with the exception of the LR model.
- The DTOS2 dataset performed poorly, achieving an accuracy rate of 98%-95% at the class level and 99%-98% at the method level.

GRAPHICAL ABSTRACT



Keywords:

Meta-Learning

Refactoring

Predication

Swarm Optimization Method

Machine Learning

ABSTRACT

Refactoring is a fundamental component of software development, aiming to improve the quality of code without affecting its external behavior. Without refactoring, code becomes more complex and less organized over time, making it more difficult for developers to understand, maintain, and modify. Therefore, performing regular refactoring is essential to maintain code organization and quality over the long term. In this context, meta-learning plays a key role in improving refactoring prediction. It can be used to improve software quality by building accurate models capable of predicting refactorings. This strengthens code and reduces potential problems and complications during development and maintenance. This thesis aims to develop a proposed model to improve the accuracy of predicting software refactoring operations at the Class-Level and Method-Level levels using meta-learning techniques. The proposed model includes two stages: the first stage aims to predict the need for refactoring, using three meta-learning techniques: the Stacking model based on the basic models Support Vector Classification (SVC), Radial Basis Function (RBF), Logistic Regression (LR), and Extreme Gradient Boosting (XGBoost), with the Logistic Regression (LR) model as an meta-learner; the Boosting model using Gradient Boosting (GB), Categorical Boosting (CatBoost), and Light Gradient Boosting Machine (LightGBM), Adaptive Boosting (AdaBoost) algorithms; and the Few-shot Learning model based on Model-Agnostic Meta-Learning (MAML), Reptile Model (RM), Relation Network Model (RNM), and Prototypical Network. The model (PNM) serves as the base

	learner, and the LR model serves as the meta-learner. The second stage is used to classify the predicted refactoring types based on the predictions made in the first stage, using two models: the Random Forest Model (RFM) and the Support Vector Machine (SVM). Four datasets previously collected from open source projects, containing information on code characteristics and refactoring instances, were used in the model training and testing process. To improve model performance and increase its prediction accuracy, the focus was on selecting and extracting the most important features from the data. This was done using four different swarm optimization algorithms: the Lion Optimization Algorithm (LOA), the Spider Monkey Algorithm (SMO), the Negative Optimization Algorithm (SOA), and a proposed hybrid method (SMOA), which combines the Spider Monkey Algorithm and the Negative Optimization Algorithm. The proposed hybrid method demonstrated high effectiveness in enhancing the model's prediction accuracy. The overlearning models were trained using training data representing 80% of the dataset and tested on test data representing 20% of the dataset. In the second phase, classification models were trained using the data from the first phase, which represented the output of the best predictive model. 70% of the data was used to train the models, and 30% of the test data was used to test them. The performance of the meta-learning models was measured using several metrics. The results showed that the hybrid approach led to better results in improving the model's prediction accuracy, and therefore was adopted for feature selection in the datasets. The results of implementing the three models achieved varying accuracies on the datasets, but the few-shot learning model proved effective due to its ability to learn from limited data. The models achieved accuracies ranging from 1.00% to 98% at the class level, with the exception of the LR model. They achieved accuracies ranging from 99% to 98% at the method level for the DTOS1 dataset, with the exception of the LR model, which was the lowest-performing model. However, the DTOS4 and DTOS3 datasets performed better, achieving optimal results, with most models achieving an accuracy of 1.00% at the class and method levels, with the exception of the LR model. The DTOS2 dataset performed poorly, achieving an accuracy rate of 98%-95% at the class level and 99%-98% at the method level, with the exception of the LR model. In the Refactoring case classification phase, the RF and SVM models proved their efficiency, achieving perfect accuracy for all datasets at the class and function levels, with the exception of the DTOS2 dataset, which achieved an accuracy rate of 87% at the method level. **2025 M.Sc. Thesis @Univ. of Mosul, College of Computer Science and Mathematics., Dept. Software** **(rasha.23csp16@student.uomosul.edu.iq)**