Software defect prediction based on counterfactual explanations

Wei Zheng
Nanchang Hangkong University

Tengfei Chen (✉ 2656090080@qq.com)
Nanchang Hangkong University

Meiting Hu
Nanchang Hangkong University

Fengyu Yang
Nanchang Hangkong University

Xin Fan
Nanchang Hangkong University

Peng Xiao
Nanchang Hangkong University

Research Article

Keywords: Software defect prediction, Hybrid defect prediction model, Counterfactual explanations, Network representation technique

Posted Date: July 27th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1870038/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License
Software defect prediction based on counterfactual explanations

Wei Zheng\textsuperscript{a,b}, Tengfei Chen\textsuperscript{a,b,*}, Meiting Hu\textsuperscript{a,b}, Fengyu Yang\textsuperscript{a,b}, Xin Fan\textsuperscript{a,b}, Peng Xiao\textsuperscript{a,b}
\textsuperscript{a}School of software, Nanchang Hangkong University, Nanchang 330063, P.R.China
\textsuperscript{b}Software Testing and Evaluation Center, Nanchang Hangkong University, Nanchang 330063, P.R.China
\textsuperscript{*}Corresponding author. Tengfei Chen. Email: 2656090080@qq.com

ABSTRACT

Software defect prediction is critical to ensure the software quality. Researchers have worked on building various defect prediction models to improve the performance of defect prediction. The existing defect prediction models are mainly divided into two categories: one is the models constructed based on the artificial statistical features and the other is the models constructed based on the semantic features. DP-CNN\textsuperscript{[29]} is one of the best defect prediction models, because it combines both the artificial statistical features and semantic features, so its performance is greatly improved compared with traditional defect prediction models. In this paper, we are inspired by the DP-CNN model and make the following two improvements: firstly, we use a new network representation technique Struc2vec to mine the information existing between software modules, which specializes in learning node representations from structural identity and can further extract the structural features associated with defects. We let the DP-CNN model again incorporate the newly mined structural features. Then, we propose a feature selection method based on counterfactual explanations which preserves reasonable causal relationships between features and thus produces a better subset of features compared to traditional feature selection methods. We used it to optimize the artificial statistical features inside the DP-CNN model. Based on the above methods, we propose a new hybrid defect prediction model DPS-CNN-STR. We evaluate our model on six open-source projects in terms of F1-score in defect prediction. The experimental results show that DPS-CNN-STR improves the state-of-the-art method by 3.6% on average.

Keywords: Software defect prediction; Hybrid defect prediction model; Counterfactual explanations; Network representation technique.
1 Introduction

With the expansion of the software system scale, there are more and more defects in the software. The presence of defects can cause serious economic losses and even threaten people's lives [1-3]. It has been found that more than 80% of the cost in the whole life cycle of software is spent on defect fixing [4-7], and if these defects are detected and changed at the early stage of the software development, the repair costs will be significantly reduced. Therefore, related researchers have devoted themselves to building the various defect prediction models [8-11], to help developers identify potential defects in the software as early as possible in order to reduce the losses caused by defects.

To solve the above problems [12], researchers have used methods such as machine learning [13-16] to construct classifiers for defect prediction, such as logistic regression (LR) [17], support vector machines (SVM) [18], random forests (RF) [19], and neural networks (NN) [20]. However, due to the limitations of these models, the performance of defect prediction is far from reaching the expected result [21-22].

With the advent of the era of big data [23-24], deep learning [25-27] has been gradually applied to the field of the software defect prediction, and good results have been achieved. Wang et al. [28] proposed a software defect prediction model based on Deep Belief Network (DBN), which can automatically learn the fixed syntax and rich semantic features of programs, and then use these features to build the software defect prediction model. Li et al. [29] proposed a DP-CNN model: using CNN to automatically extract local semantic features and combine them with the artificial statistical features for software defect prediction, and the results showed that CNN can capture local semantic features more effectively than DBN.

However, most of the above defect prediction models are constructed based on the internal features of the software modules, and the single source of features leads to the unsatisfactory defect prediction results.

To solve this problem, network representation learning [30-31] is formally applied in the field of the software defect prediction. The software network graph is first constructed based on the dependencies existing between the software modules, and through representation learning, the graph information is effectively characterized and more important structural features are extracted. Typical methods are DeepWalk [32], LINE [33], Node2Vec [34], and SDNE [35], which are all based on the assumption of the similarity of nearest neighbors. In fact, in some scenarios, two vertices that are not nearest neighbors may also have high similarity, and for this type of the similarity, the above methods are unable to capture it. Therefore, we use a new network representation learning technique, Struc2vec [36], which specifically constructs node sequences from another perspective and focuses more on the structural information of nodes, overcoming the limitations of the traditional network representation learning methods.

Software defect prediction models are mainly constructed by features, so the selection of features will directly affect the defect prediction results. The feature selection [37-40] affects the performance of defect prediction by influencing the accuracy
and generalization of the machine learning models. When the dataset has only a small or redundant number of features, the machine model is unable to learn the general pattern, a situation that can lead to insufficient or overfitting. However, the traditional feature selection methods (e.g., wrapper method) cause little contribution to the improvement of defect prediction performance due to their own limitations.

Therefore, we propose a feature selection method based on counterfactual explanations \[41\] with the following main idea: generating a different model output by minimizing changes in the input features, a set of different counterfactual explanations can be generated for one piece of data in the same algorithm. Its biggest advantage over the traditional feature selection methods is that reasonable causal relationships between features are preserved in order to ensure that the generated counterfactual samples still match the shape of the data stream of the input samples, then the importance score corresponding to each feature can be determined by global approximation of the generated counterfactual samples. Since these feature importance scores reflect how necessary a feature value is to the outcome, so we can use them to guide the generation of feature subsets with better prediction performance.

In summary, this paper proposes a new hybrid defect prediction model DPS-CNN-STR based on the artificial statistical features, semantic features and structural features. Among them, the artificial statistical features are optimized by the feature selection method based on counterfactual explanations, and the structural features are extracted by Struc2vec. our main contributions are as follows:

- To the best of our knowledge, we are one of the few people to consider the Struc2vec method for extracting the structural features and applying it to the field of the software defect prediction. Moreover, In addition, In addition, we are the first person to propose a feature selection method based on counterfactual explanations, and achieved good results, providing a new idea for the study of feature selection.

- We propose a new hybrid defect prediction model DPS-CNN-STR, whose F1 score is improved by 3.6% on average over the optimal model DP-CNN.

The rest of the paper is organized as follows: in Section 2, the related works are introduced; Section 3 focuses on the process of the hybrid defect prediction model construction; Section 4 conducts the experimental setup, and Section 5 analyzes the experimental; then some threats that may change the experimental results are briefly described in Section 6; finally, in Section 7, a summary of the work and an outlook for the future are presented.
2 Related work

In this section, we mainly introduce the following contents: the software defect prediction technology, counterfactual explanations and the relevant background of the Struc2vec.

2.1 Software defect prediction

Software defect prediction techniques are mainly used to identify the potential defects in the software in a timely manner, and help testers to perform the purposeful testing activities [42]. Fig.1 shows the common file level software defect prediction process in literature [43-45], which mainly includes the following three steps.

1. First, mark each source file in the project according to the software history warehouse. The defective is marked as buggy and the flawless is marked as clean.
2. Then, by analyzing the software source code or historical data, the features related to the software defects from source files are extracted. The most common features are code metrics (such as Halstead features [46], McCabe features [47] and CK features [48]). The obtained features and labels are trained in the various machine learning algorithms (such as LR, SVM and RF), in order to construct the classifier.
3. Finally, features are extracted from the test files in the project to be predicted. After the feature extraction of all test files is completed, the predicted results can be obtained by putting them into the classifier.

![Fig. 1. Software defect prediction process](image-url)
2.2 Counterfactual explanations

Machine learning model is a black box \cite{49}, and people cannot explain its internal working principle. Due to the unexplainability of the models, it may often lead to irreparable consequences. Therefore, it is necessary to provide explanations for machine learning models in order to reduce the potential threats brought by the models.

The most popular is the introduction of counterfactual explanations \cite{49-50}. Its core idea is to make the output of the model opposite to that before. How to realize this idea? The output of the model can be changed by modifying features, but it pursues the change of minimizing the features. For example, in the adult income \cite{51} data set, if you want to change a low-income person to a high-income person, you can increase the value of working hours. We can imagine that this value must have a critical value. When it is equal to or greater than this value, the result of the model will change, but we only choose the critical value. Or we can change the value of multiple features at the same time to change the output of the model, but counterfactual explanations pursue the minimum disturbance of features. People can explain the decision of the model through the generated counterfactual samples.

Recently, someone has improved the counterfactual explanations and provided a framework \cite{41}, which can not only generate multiple counterfactual explanations for one data, but also the generated counterfactual samples, like the original data, retain the reasonable causal relationship between features. For example, take the adult income data set. When you want to change someone's low income to high income, you can't change their gender or race. You can consider changing the weekly working hours, but its value also should conform to the range of the human normal working hours.

In addition, the generated counterfactual samples can be globally approximated to determine the importance score of each feature, and the feature importance scores reflect the necessity of the eigenvalue to the result. Inspired by the above, and in order to solve the limitations of the traditional feature selection methods, we propose a feature selection method based on counterfactual explanations, which can generate a better subset of features guided by the feature importance scores. Details will be presented in Section 3.2.

2.3 Struc2vec

The traditional network representation learning algorithms have a disadvantage. Because the sampling length of walk is limited, it is unable to effectively model long-distance nodes with structural similarity. However, the reason why the previous algorithms perform better is that the most data sets prefer the characterization of homogeneity, that is, the nodes with similar distance are also similar in the feature space, which is enough to cover the most data sets. In the process of constructing the graph, Struc2vec does not need the location information and label information of nodes, but only relies on the concept of node degree to construct the multi-layer graph. An intuitive concept: if two nodes have the same degree, then the two nodes are more similar in
structure. Further, if all the neighbor nodes of the two nodes also have the same degree, then the two nodes should be more similar in structure. In short, the nodes with neighbors with similar node sets should have similar potential representations, and Struc2vec specifically learns the node representations from the structure identification, and achieved good results.

![Example of two nodes (m and n) with similar structure](image)

**Fig. 2.** Example of two nodes (m and n) with similar structure

Fig. 2: Nodes m and n have similar local structures, node m has degree 4 and node n has degree 3, and nodes m and n are connected to the software network with 3 and 2 triangles respectively. It can be seen that these two nodes have high structural similarity, but because there are no common nodes in their neighborhoods, traditional network representation techniques cannot learn the potential representation of nodes with similar structures, but Struc2vec solves this problem.

In our work, in order to solve the problem of single feature source, we can use Struc2vec to extract relevant structural features, in order to improve the performance of defect prediction model.


3 Proposed method

In this section, we will introduce our proposed hybrid defect prediction model in detail. Our defect prediction model DPS-CNN-STR is based on the artificial statistical features, semantic features and structural features. It is inspired by the DP-CNN model proposed by Li et al, and has made the following improvements:

1. The artificial statistical features are optimized by the feature selection method based on counterfactual explanations.

2. Using Struc2vec to learn the structural features from the software network, a new hybrid defect prediction model is jointly constructed based on the optimized DP-CNN model, combined with the newly learned structural features.

3.1 Hybrid defect prediction model

In order to improve the performance of defect prediction models, Fig.3 shows a new hybrid defect prediction model. First, the artificial statistical features are optimized through counterfactual explanations, then the semantic features are learned from the source program using CNN, and finally the structural features are learned from the software network using Struc2vec. A hybrid defect prediction called DPS-CNN-STR is constructed.

Fig. 3. Hybrid defect prediction model

These three types of features are combined as the input of the SoftMax neural network, and then the defect prediction results are obtained through an embedding layer (i.e., word embedding), two convolutional layers, a max-pooling layer, a fully-connected hidden layer. Except the output layer uses the SigMoid activation function, all other layers use ReLu activation function. We build the neural network based on the Keras, the biggest advantage of which is its simplicity and speed, and we also keep the exact same parameter settings as in the literature [29].
3.2 Feature selection based on counterfactual explanations

Fig.4 shows a feature selection method based on counterfactual explanations, consisting of the following three main steps:

(1) In this paper, defect prediction is based on the iterations of versions within the project, so the artificial statistical table of the old version is input into the counterfactual explanations framework given in the literature [41], and then each feature is given a corresponding importance score.

(2) Features are combined one by one according to their importance scores from highest to lowest. Suppose there are n features, then there are n feature subsets correspondingly.

(3) The old version of the artificial statistical features (the subset of the features described in step 2) and their labels are used to construct the classifier, then the new version of the artificial statistical features and their labels are used as the test set, and the corresponding optimal subset of features is selected by choosing the highest F1 score.

![Feature selection based on counterfactual explanations](image)

**Fig. 4.** Feature selection based on counterfactual explanations

3.3 Structural features extraction

Using the dependency relationships existing between the software modules, we construct the software network with the software modules as the basic unit, and then uses Struc2vec to learn the potential representation of node structure, in order to extract the structural features under the unsupervised learning.

First, based on the data flow relationships that exist between the modules, the software network $G=(V, E)$ is constructed, where $G$ represents the constructed software network, $V=\{v_i \mid i=1,2,3,...,n\}$ is the set of nodes in the software network, the element $v_i$ represents each node in the software network, $n=|V|$ is the number of nodes in the
constructed software network, and $k^*$ is the diameter. $E = \{ e_{ij} \mid v_iv_j = 1, i, j \in [1, n] \}$ represents the set of edges, when the value is 1, it represents that there is a relationship between node $i$ and node $j$, when the value is 0, then there is no relationship between them, and the relationships that exist are as follows:

1. There is a dependency between node $i$ and node $j$;
2. There is a combination relationship between node $i$ and node $j$;
3. There is an inheritance relationship between node $i$ and node $j$;

Fig.5 shows the software networks built by some of the Apache open-source software projects according to the above rules.

![Software Networks](image)

**Fig. 5.** The software network of poi, lucene and synapse

The extraction of the structural features from the software network using Struc2vec is divided into four main steps as follows:

1. Measuring structural similarity. The $R_k(u)$ represents the set of nodes whose distance from node $u$ is $k$, $R_1(u)$ represents the set of directly connected nearest neighbors of $u$, and the $s(S)$ represents the ordered degree sequences of the set $S$ of nodes. The distance $f_k(u, v)$ between all nodes is calculated by introducing a hierarchical structure, and this distance can reflect the structural similarity situation between nodes, defined as follows:

$$f_k(u, v) = f_{k-1}(u, v) + g(s(R_k(u)), s(R_k(v))), k \geq 0 \text{ and } |R_k(u)|, |R_k(v)| > 0$$  \hspace{1cm} (1)

The $g(D1, D2) \geq 0$ is a function that measures the distance of the ordered degree sequences $D1, D2$. Since $s(R_k(u))$ and $s(R_k(v))$ have different lengths and may contain duplicate elements. To solve this problem, so a distance calculation formula called DTW is used, defined as follows:

$$d(a, b) = \frac{\max(a, b)}{\min(a, b)} - 1$$  \hspace{1cm} (2)
(2) Constructing the context graph. A multilayer weighted graph $M$ is constructed based on the obtained node-pair distances, which is mainly intended to encode the structural similarity between nodes. The edge weight of two nodes in a certain layer $k$ is defined as:

$$w_k(u,v) = e^{-f_k(u,v)}, k = 0, ..., k^*$$  \hspace{1cm} (3)

The same node belonging to different layers is connected by directed edges, and the edge weight is defined as:

$$w(u_k, u_{k+1}) = \log(\Gamma_k(u) + e), k = 0, ..., k^* - 1$$  \hspace{1cm} (4)

$$w(u_k, u_{k-1}) = 1, k = 1, ..., k^*$$  \hspace{1cm} (5)

The $\Gamma_k(u)$ is the number of edges related to node $u$, and its weight is greater than the average edge weight of the complete graph in layer $k$, and defined as:

$$\Gamma_k(u) = \sum_{v \in V} w_k(u, v)$$  \hspace{1cm} (6)

(3) Generating context for nodes. A biased random walking strategy is applied to all nodes in the graph $M$ as a way to generate the contextual representation of each node. At each sampling, if the decision is to wander at the current layer, and assuming that it is currently at layer $k$, the probability of going from node $u$ to node $v$ is:

$$P_k(u, v) = \frac{e^{-f_k(u,v)}}{Z_k(u)}$$  \hspace{1cm} (7)

The $Z_k(u)$ is the normalization factor of node $u$ in layer $k$, which is obtained by the following formula:

$$Z_k(u) = \sum_{v \in V, v \neq u} e^{-f_k(u,v)}$$  \hspace{1cm} (8)

If it is decided to switch different layers, select $k+1$ layer or $k-1$ layer with the following probability:

$$P_k(u_k, u_{k+1}) = \frac{w(u_k, u_{k+1})}{w(u_k, u_{k+1}) + w(u_k, u_{k-1})}$$  \hspace{1cm} (9)

$$P_k(u_k, u_{k-1}) = 1 - P_k(u_k, u_{k+1})$$  \hspace{1cm} (10)

(4) Learning a language model. Finally, using the Skip-Gram technique, the potential representation of each node is learned from the generated contextual representation.
3.4 Semantic features extraction

The source program is the main cause of software defects, each source file program code can be parsed into a series of word sequence representations, and the word sequence representations are converted into the semantic features by the efficient feature extraction capability of CNN, the process of which is shown in Fig.6.

Fig. 6. The semantic features extraction process

1) Traverse every source file in the software project, code in each source file is parsed into AST \textsuperscript{[52-53]} nodes by an open-source Python package named javalang. According to the optimal approach \textsuperscript{[54]}, only three main node types \textsuperscript{[55-57]} are selected as the word sequences for the software modules, one is the node type of method invocation and class instance creation, adding their specific method names and class names to the word sequences; one is the node type of declaration, such as method declaration, interface description, constructor declaration, etc., adding their values to the word sequences; and one is the node type of control flow. For example, ForStatement, IfStatement, WhileStatement, etc. are added to the word sequences, and part of the node types are shown in Table 1.

<table>
<thead>
<tr>
<th>Number</th>
<th>Node Type</th>
<th>Number</th>
<th>Node Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PackageDeclaration</td>
<td>11</td>
<td>ForStatement</td>
</tr>
<tr>
<td>2</td>
<td>ClassDeclaration</td>
<td>12</td>
<td>ContinueStatement</td>
</tr>
<tr>
<td>3</td>
<td>InterfaceDeclaration</td>
<td>13</td>
<td>ReturnStatement</td>
</tr>
<tr>
<td>4</td>
<td>MethodInvocation</td>
<td>14</td>
<td>SwitchStatement</td>
</tr>
<tr>
<td>5</td>
<td>MemberReference</td>
<td>15</td>
<td>BlockStatement</td>
</tr>
<tr>
<td>6</td>
<td>ReferenceType</td>
<td>16</td>
<td>TryResource</td>
</tr>
<tr>
<td>7</td>
<td>MethodDeclaration</td>
<td>17</td>
<td>CatchClause</td>
</tr>
<tr>
<td>8</td>
<td>IfStatement</td>
<td>18</td>
<td>ForControl</td>
</tr>
<tr>
<td>9</td>
<td>WhileStatement</td>
<td>19</td>
<td>BasicType</td>
</tr>
<tr>
<td>10</td>
<td>DoStatement</td>
<td>20</td>
<td>FormalParameter</td>
</tr>
</tbody>
</table>

Table 1 Part of the node types
(2) Since the CNN model only receives the numeric input, all words in the word sequences need to be converted to the numeric values. The solution is to encode the words in a non-repeating way starting with a value of 1 and increasing, so as to ensure a one-to-one correspondence between words and encoded values. In order to distinguish words coding and labels coding, so we decided to encode the labels with One-Hot encoding \[^{[58-59]}\]. Results in the above steps, the source files are parsed into a series of numeric vectors.

(3) In addition, CNN requires the input vectors to maintain a consistent length. However, because the code of each source file is different, it is impossible to keep the same length. In order to solve this problem, we first set the fixed length of the input vectors, and then if the length of the input vectors is greater than the set value, the redundant part will be discarded; Otherwise, it is supplemented with 0. Finally, the vectors are used as the input of CNN model, and the semantic features are obtained through an embedding layer, a convolutional layer, an activation layer, a pooling layer and a fully connected layer, respectively.
4 Experimental setup

In this section, a series of experiments are designed to evaluate the effectiveness of our proposed hybrid defect prediction model.

4.1 Datasets

The experimental data of this paper comes from the open-source software projects under the Apache foundation. Considering the various factors such as the stability of the data set, we finally selected six open-source software projects. Because this paper is based on the software defect prediction within the version iteration, each project has two consecutive versions, in which the old version is used as the training set and the new version is used as the test set to evaluate the model performance. Table 2 shows in detail the descriptions, versions, average number of files and buggy rates of the six Apache open source software projects.

<table>
<thead>
<tr>
<th>Project</th>
<th>Description</th>
<th>Versions</th>
<th>Avg Files</th>
<th>Buggy Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>poi</td>
<td>Java library to access Microsoft format files</td>
<td>2.5, 3.0</td>
<td>409</td>
<td>64.7</td>
</tr>
<tr>
<td>lucene</td>
<td>Text search engine library</td>
<td>2.0, 2.2</td>
<td>210</td>
<td>55.7</td>
</tr>
<tr>
<td>synapse</td>
<td>Data transport adapters</td>
<td>1.1, 1.2</td>
<td>239</td>
<td>30.5</td>
</tr>
<tr>
<td>camel</td>
<td>Enterprise integration framework</td>
<td>1.4, 1.6</td>
<td>892</td>
<td>18.6</td>
</tr>
<tr>
<td>jedit</td>
<td>Text editor designed for programmers</td>
<td>4.0, 4.1</td>
<td>284</td>
<td>23.8</td>
</tr>
<tr>
<td>xerces</td>
<td>XML parser</td>
<td>1.2, 1.3</td>
<td>441</td>
<td>15.5</td>
</tr>
</tbody>
</table>

First of all, the number of files in the project we selected varies from 210 to 892, and its purpose is to ensure the diversity of data. Then, we also selected projects with different defect rates to test the performance of our model, with a minimum of 15.5% and a maximum of 64.7%. The first column in Table 2 shows the six projects in the dataset, the second column gives a brief description of the projects, the third and fourth columns respectively describe the version of the projects and the average number of files, and the last column describes the percentage of defect instances.

In addition, this paper collects a dataset of 20 artificial statistical features and defect statistics for these six Apache open-source software projects, with the statistical feature data coming from the tera-PROMISE project. The dataset contains metric metrics based on the software size and software complexity, and part of these statistical metrics are shown in Table 3.
<table>
<thead>
<tr>
<th>Feature Names</th>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weighted Methods per Class</td>
<td>WMC</td>
<td>The number of methods in the class</td>
</tr>
<tr>
<td>Depth of Inheritance Tree</td>
<td>DIT</td>
<td>Indicates the position of the class in the inheritance tree</td>
</tr>
<tr>
<td>Number of Children</td>
<td>NOC</td>
<td>The number of immediate descendants of the class</td>
</tr>
<tr>
<td>Coupling Between Object classes</td>
<td>CBO</td>
<td>The value increases when the methods of one class access services of another</td>
</tr>
<tr>
<td>Response for a Class</td>
<td>RFC</td>
<td>Number of methods invoked in response to a message to the object</td>
</tr>
<tr>
<td>Lack of Cohesion in Methods</td>
<td>LCOM</td>
<td>Number of pairs of methods that do not share a reference to an instance variable</td>
</tr>
<tr>
<td>Average Method Complexity</td>
<td>AMC</td>
<td>The number of JAVA byte codes</td>
</tr>
<tr>
<td>Afferent couplings</td>
<td>Ca</td>
<td>How many other classes use the specific class</td>
</tr>
<tr>
<td>Efferent couplings</td>
<td>Ce</td>
<td>How many other classes is used by the specific class</td>
</tr>
<tr>
<td>Lines of Code</td>
<td>LOC</td>
<td>Measures the volume of code</td>
</tr>
</tbody>
</table>

### 4.2 Evaluation measures

In order to evaluate the model performance, it is necessary to select a suitable metric to compare the performance of different models. Since the software defect prediction dataset is a typical unbalanced dataset, so we evaluate the performance of defect prediction using the F1 score mentioned in the literature [60-61], which is a statistical measure of a binary classification model that takes into account both the precision and recall of the classification model and is the reconciled average of them [62], with a maximum value of 1 and a minimum value of 0. The precision $P$ is calculated as follows:

$$P = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}$$ (11)

Where true positive indicates the number of correct predictions in the defective samples, false positive indicates the number of predicted defective samples in non-defective samples, and the recall $R$ is calculated as follows:

$$R = \frac{\text{true positive}}{\text{true positive} + \text{true negative}}$$ (12)
Where true negative indicates the number of defect free samples predicted in the defect samples. The formula for calculating $F1$ score is as follows:

$$F1 = \frac{2 \times P \times R}{P + R}$$ (13)

For datasets with balanced data distribution, precision and recall can be good measures of model performance, but for classical data imbalance datasets like software defect prediction, where the prior probability threshold for determining the category is not equal to 0.5, so it is better to use the $F1$ score.

4.3 Baselines

In order to verify the effectiveness of the feature selection method based on counterfactual explanations, we compare it with the method without feature selection and the wrapper method\cite{63}, where the feature selection is performed on the 20 artificial statistical features from the six open-source projects mentioned in Section 4.1.

Then, to verify the effectiveness of the structural features mined by Struc2vec, a new software defect prediction model DP-CNN-STR was proposed based on the DP-CNN model and combined with the mined structural features. The performance of the two models is compared.

Finally, in order to verify that our proposed feature selection method and the structural features both are helpful for defect prediction, we compare the DPS-CNN-STR model with the DP-CNN model.

4.4 Research questions

In studying the construction of software defect prediction models, our main concern is the performance of the model. In this paper, we propose a new software defect prediction model DPS-CNN-STR. Because this model is constructed based on the DP-CNN model and additionally undergoes feature selection based on counterfactual explanations and combines the structural features mined by Struc2vec. Therefore, we have the following three research questions:

• RQ1: Is the feature selection method based on counterfactual explanations effective?
• RQ2: Can the structural features mined by Struc2vec effectively improve the performance of defect prediction models?
• RQ3: Is the performance of hybrid defect prediction model DPS-CNN-STR better than that of model DP-CNN?
4.5 Setup

In RQ1, we compare our proposed feature selection method with the wrapper method and the method without feature selection. The method without feature selection means that only logistic regression is used as the classifier. The feature selection method based on counterfactual explanations and the wrapper method need a specific classifier as the carrier and also in order to ensure the comparability of the experiment, both feature selection methods also use the logistic regression as a specific classifier. The specific ideas are as follows:

• The feature selection method based on counterfactual explanations (LRBOC): first, the old version of the artificial statistical feature table is input into the counterfactual explanations framework given in the literature\cite{41}, and then each feature is given a corresponding importance score, which is returned in the form of a dictionary from highest to lowest, e.g. `{'feature1': 0.542, 'feature2': 0.384, ...... , 'feature20': 0.112}`. Then features are combined one by one according to the importance scores from highest to lowest. Suppose there are 20 features, then there are 20 feature subsets corresponding to the following form: [feature1], [feature1, feature2], ...... [feature1, feature2, ...... , feature20]. Then, the old version of the artificial statistical features (the feature subset described above) and their labels are used to construct the classifier, and finally the new version of the artificial statistical features and their labels are used as the test set, and the corresponding optimal feature subset is selected by choosing the highest F1 score.

• The wrapper method (WRAP): There is a parameter named ‘n_features_to_select’ in the packing method, which represents the number of features in the optimal feature subset. First, the number of features in the optimal feature subset of the old version is determined by the learning curve, then the specific optimal feature subset is determined by the feature score matrix. Suppose the number of features in the optimal feature subset is 8 and the form of the feature score matrix is [1,2,2,3,1,2,3,1,2,1,3,2,1,2,3,1,3,2,1], we can select the 8 features corresponding to the value 1 as the optimal feature subset. Then, the optimal feature subset and labels of the old version are used to build the classifier, while the optimal feature subset and labels of the new version are used as the test set, and its F1 score is recorded.

• The method without feature selection (LR): without using any feature selection method, all features and labels of the old version are used to build the classifier, while all features and labels of the new version are used as the test set and its F1 score is recorded.

The F1 score values obtained from the above three feature selection methods are compared, and the above comparison results are used to verify the effectiveness of the feature selection method based on counterfactual explanations.

In RQ2,3, we mainly set up two sets of comparison tests. One group is the comparison of F1 scores of the DP-CNN model and the DP-CNN-STR model; the other group is the comparison of F1 scores of the DP-CNN model and the DPS-CNN-STR model. The design of the above experiments is realized under the control variable method.
5 Experimental results

In this section, we give the experimental results to answer the three questions.

5.1 RQ1: Validity of feature selection

5.1.1 Comparative

As far as we know, we are the first person to propose the feature selection using counterfactual explanations. As shown in Table 4, the performance of our proposed feature selection method LRBOC is much better than the traditional feature selection method WRAP on six datasets.

<table>
<thead>
<tr>
<th>Project</th>
<th>Version</th>
<th>LR</th>
<th>WRAP</th>
<th>LRBOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>poi</td>
<td>2.5, 3.0</td>
<td>0.745</td>
<td>0.742</td>
<td>0.788</td>
</tr>
<tr>
<td>lucene</td>
<td>2.0, 2.2</td>
<td>0.603</td>
<td>0.605</td>
<td>0.609</td>
</tr>
<tr>
<td>synapse</td>
<td>1.1, 1.2</td>
<td>0.562</td>
<td>0.554</td>
<td>0.639</td>
</tr>
<tr>
<td>camel</td>
<td>1.4, 1.6</td>
<td>0.350</td>
<td>0.354</td>
<td>0.360</td>
</tr>
<tr>
<td>jedit</td>
<td>4.0, 4.1</td>
<td>0.598</td>
<td>0.610</td>
<td>0.626</td>
</tr>
<tr>
<td>xerces</td>
<td>1.2, 1.3</td>
<td>0.243</td>
<td>0.261</td>
<td>0.307</td>
</tr>
<tr>
<td>Avg</td>
<td></td>
<td>0.517</td>
<td>0.521</td>
<td>0.555</td>
</tr>
</tbody>
</table>

In order to see the results more directly, we draw them into a chart, as shown in Fig.7:

![Fig. 7. F1 scores of three feature selection methods](image)

It is clear from Fig.7 that the F1 score of the LRBOC is on average 3.4% higher than that of the WRAP, and the F1 score of the LRBOC is on average 3.8% higher than that of the LR. However, the WRAP is only 0.4% higher on average than the LR. All
the above data illustrate the effectiveness of the feature selection method based on counterfactual explanations.

5.1.2 Extension

As described in Section 3.2, the feature selection method based on counterfactual explanations mainly consists of three steps. Here, we show the process of selecting the best feature subset for different projects in the form of graphs, as shown in Fig.8.

Fig. 8. The process of selecting the best feature subset

(a) poi
(b) lucene
(c) synapse
(d) camel
(e) jedit
(f) xerces
Through the Fig.8, we can clearly see that when there are 20 features, it is equivalent to no feature selection. In the poi project, when the number of the artificial statistical features is 7, the F1 score is the biggest, these seven features are the top seven with the largest feature importance scores, and they are the optimal feature subset of the artificial statistical features for this project.

### 5.2 RQ2: Validity of structural features

In order to answer the RQ2 and RQ3, we designed two sets of comparison experiments, and the results are shown in Table 5.

<table>
<thead>
<tr>
<th>Project</th>
<th>Version</th>
<th>DP-CNN</th>
<th>DP-CNN-STR</th>
<th>DPS-CNN-STR</th>
</tr>
</thead>
<tbody>
<tr>
<td>poi</td>
<td>2.5, 3.0</td>
<td>0.784</td>
<td>0.799</td>
<td>0.806</td>
</tr>
<tr>
<td>lucene</td>
<td>2.0, 2.2</td>
<td>0.761</td>
<td>0.768</td>
<td>0.765</td>
</tr>
<tr>
<td>synapse</td>
<td>1.1, 1.2</td>
<td>0.556</td>
<td>0.570</td>
<td>0.582</td>
</tr>
<tr>
<td>camel</td>
<td>1.4, 1.6</td>
<td>0.508</td>
<td>0.548</td>
<td>0.560</td>
</tr>
<tr>
<td>jedit</td>
<td>4.0, 4.1</td>
<td>0.580</td>
<td>0.627</td>
<td>0.620</td>
</tr>
<tr>
<td>xerces</td>
<td>1.2, 1.3</td>
<td>0.374</td>
<td>0.426</td>
<td>0.448</td>
</tr>
<tr>
<td>Avg</td>
<td></td>
<td>0.594</td>
<td>0.623</td>
<td>0.630</td>
</tr>
</tbody>
</table>

In order to see the results more directly, we draw them into a chart, as shown in Fig.9:

![Fig. 9. F1 scores of three models](image)

The F1 score of DP-CNN-STR model is 2.9% higher than that of DP-CNN on average, it proves that the software structure features extracted by Using Struc2Vec are helpful for the construction of software defect prediction models.

Finally, in order to answer RQ3, DPS-CNN-STR has an average of 3.6% higher F1 score than DP-CNN, proving that our feature selection method and the structural features both can facilitate the construction of the models.
6 Threats to validity

The main internal threats come from the construction of the experimental environment and the setting of parameters. Our experiments are based on the python language environment, in order to reduce the uncontrollable factors in the implementation process, we have adopted a sufficiently mature third-party libraries, such as calling various Python packages, to achieve the required requirements. In addition, we refer to the default values of the documentation as the parameters for defect prediction, and the parameter setting often directly affects the prediction performance.

For example, in the feature selection experiment based on counterfactual explanations, there is a parameter called penalty term. In the process of machine learning, because we provide a large amount of data for training, many dimensions will be generated during the training, some are decisive, others are irrelevant. In other words, we hope to get more accurate results through a large number of data training, but at the same time, the more judgment dimensions, the worse the generalization ability of our model, this requires us to control a balance, so we introduced the penalty term. In the experiment in Section 5.1.1, we choose its default value of 1. However, when the values of penalty term are different, the results are often different, as shown in Table 6.

<table>
<thead>
<tr>
<th>Project</th>
<th>Version</th>
<th>C=0.01</th>
<th>C=0.1</th>
<th>C=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>poi</td>
<td>2.5, 3.0</td>
<td>0.770</td>
<td>0.758</td>
<td>0.788</td>
</tr>
<tr>
<td>lucene</td>
<td>2.0, 2.2</td>
<td>0.608</td>
<td>0.607</td>
<td>0.609</td>
</tr>
<tr>
<td>synapse</td>
<td>1.1, 1.2</td>
<td>0.601</td>
<td>0.635</td>
<td>0.639</td>
</tr>
<tr>
<td>camel</td>
<td>1.4, 1.6</td>
<td>0.363</td>
<td>0.360</td>
<td>0.360</td>
</tr>
<tr>
<td>jedit</td>
<td>4.0, 4.1</td>
<td>0.619</td>
<td>0.621</td>
<td>0.626</td>
</tr>
<tr>
<td>xerces</td>
<td>1.2, 1.3</td>
<td>0.339</td>
<td>0.309</td>
<td>0.307</td>
</tr>
<tr>
<td>Avg</td>
<td></td>
<td>0.550</td>
<td>0.548</td>
<td>0.555</td>
</tr>
</tbody>
</table>

In order to see the results more directly, we draw them into a chart, as shown in Fig.10:

![Fig. 10. LRBOC under different parameter settings](image_url)
In addition, when building the DPS-CNN-STR model, different internal parameter settings will also lead to the different prediction performance of the model, we will investigate how to set the values of these parameters in the most optimal way in order to produce better results in the future work.

**The main external threat** comes from the universality principle of the project. We only performed our model on 6 projects, but these projects can not summarize all types of software. In addition, we feel it necessary to explain why only 6 data sets were selected. Because the construction of our defect prediction model uses the artificial statistical features, semantic features, and structural features, which means that not only the features that need to be manually mined, but also the corresponding source code needs to be provided. However, some source code files in some projects have syntax errors and are difficult to correct, so only 6 projects are selected as our final data set after the comprehensive consideration.
7 Conclusion and Future work

Researchers have developed various models for software defect prediction to reduce the losses due to defects. In this paper, we propose a new hybrid defect prediction model DPS-CNN-STR, which is built on the basis of the DP-CNN model. The improvements are mainly reflected in the following two aspects: first, we build a software network using the data flow existing between the modules and the new network characterization technique Struc2vec is used to extract the more important structural features, combining these new mined structural features on the DP-CNN model. Then, we propose a feature selection method based on counterfactual explanations to optimize the artificial features, which achieves much better results than traditional feature selection methods because it preserves the causal relationships that exist between features. Finally, our experiments on six public datasets show that our proposed model can provide a new idea for the construction of software defect prediction models.

In the future, we will explore the potential of our DPS-CNN-STR for defect prediction on more projects. In addition, we will explore how to combine features more effectively under the premise of given feature importance scores, and how to extract the structural features of software network more effectively, in order to further improve the performance of defect prediction model.
**Declarations**

- **Competing interests:**
  1. **Funding:** This work was supported by the National Natural Science Foundation of China with the number of 6186700.
  2. **Financial interests:** It will not obtain or damage the interests of relevant companies, organizations or institutions due to the publication of this manuscript.
  3. **Employment:** The current or expected employment of any organization will not gain or lose money due to the release of the manuscript.
  4. **Non-financial interests:** I promise that my manuscript will not be treated differently due to professional interests, personal relationships or personal beliefs.
- **Data Availability:** The results/data/figures in this manuscript have not been published elsewhere, nor are they under consideration.
- **Approve:** I have read the Springer journal policies on author responsibilities and submit this manuscript in accordance with those policies.
- **Author contributions:** All authors have contributed to the study of concepts and design. The format, typesetting and grammar of the thesis are mainly solved by Fengyu Yang, Xin Fan and Peng Xiao. The first draft of the paper is mainly written by Tengfei Chen and with the assistance of Wei Zheng, the final draft of the paper is completed by Tengfei Chen. In addition, this research is supported by the National Natural Science Foundation of China, with the number of 6186700, the main contributor is Wei Zheng. There are two innovations in this paper. Among them, the idea of improving the performance of defect prediction model by combining the structural features mined by Struct2vec is provided by Tengfei Chen. In addition, the idea of feature selection using counterfactual explanations is mainly provided by Wei Zheng and Meiting Hu. Meiting Hu also found the relevant source code of counterfactual explanations. Based on the above, Tengfei Chen completed all the experiments in this paper. All the authors commented on the first draft, and all the authors carefully read the final draft and agreed.
8 References

[18] Lloyd B G R. Support Vector Machines for classification and regression[J]. Analyst, 2010,


9 Thanks

First of all, I would like to thank my graduate tutor Wei Zheng, who brings me into the path of scientific research and gives me careful guidance. With his help, the two of us finished this paper together.

Then, I want to thank Meiting Hu. We are from the same division. Her research direction is the causal inference. It is her inspiration that makes me combine the counterfactual explanations with software defect prediction, and achieved good results. Then, I would like to thank three teachers, Fengyu Yang, Xin Fan and Peng Xiao, for their guidance on formatting, writing, and other aspects of my paper.
10 Member Profile

1. Wei Zheng: Born in 1982, male, native of Pingxiang, Jiangxi, PhD, professor, master's supervisor, dean of the School of Software in Nanchang Hangkong University. He has published more than 10 SCI/EI papers, including 4 SCI Zone II papers, and has been approved 3 software copyrights.

2. Tengfei Chen: Born in 1998, male, native of Shangrao, Jiangxi, studying for his master's degree in Nanchang Hangkong University. Research direction is software defect prediction.


4. Fengyu Yang: Born in 1980, male, from Jiujiang, Jiangxi Province, master, associate professor, master's supervisor, director of the Department of data science and big data technology, School of software in Nanchang Hangkong University. He has published 8 sci/ei papers, 2 patents and 3 software copyrights.
5. Xin Fan: Born in 1981, male, from Jingzhou, Hubei Province, master, associate professor, master's supervisor. Published 8 sci/ei papers.

6. Peng Xiao: Born in 1988, male, from Ji'an, Jiangxi Province, PhD, lecturer, master's supervisor. He has published more than 5 sci/ei papers, including 2 SCI Zone I papers.