SOFTWARE DEFECTS

Under Supervision:
Dr. MANJARI GUPTA
By:
JYOTIRMOI BISWAKARMA
HIMANSHU KHOLIYA

SOFTWARE DEFECTS

Software defects, also known as software bugs or software errors, are errors or flaws in software code or programs that cause the software to behave in an unintended or incorrect way. These defects can range from minor issues that don't impact functionality to critical issues that can cause the software to crash or even compromise security.

Defects can be introduced during any stage of the software development process, from requirements gathering and design to coding, testing, and maintenance. They can result from human error, such as a programmer writing incorrect code, or from environmental factors, such as hardware failures or network issues.

It is important to identify and address software defects as early as possible in the development process to minimize their impact on the final product. This is typically done through testing and debugging, where the software is systematically tested to identify and fix any defects. Software defects can also be prevented through various software development best practices, such as code reviews, automated testing, and software quality assurance processes.

TYPE OF SOFTWARE DEFECTS

- Coding defects
- Design defects
- Documentation defects
- Configuration defects
- Interface defects
- Data defects
- Performance defects
- Security defects

EXAMPLE OF SOFTWARE DEFECTS

- Crash
- Functional error
- Acknowledgement message error
- Typos
- Missing command
- Calculation error
- Hardware usage error
- Control flow error

SOME SHOCKING FACTS ABOUT SOFTWARE BUGS

- The recent iPhone bug where users could not type the letter “I”.
- Some are costly bugs and can cost a fortune to fix one such bug: the Y2K bug.
- One software bug literally led to the death of people due to the patriot missile bug; 28 people died in 1991.
- Any buggy code reflects poorly on them and their team and will eventually affect the company’s bottom line.
- Also, buggy code is inconvenient to work with and reduces productivity. The more quality code you can write, the more effective you will be. Finally, bugs are expensive.
- Various software bugs are estimated to have cost the global economy $1.7 trillion in 2017.

CAUSES OF SOFTWARE DEFECTS

- Poor software design
- Poor coding
- Lack of testing
- Unclear requirements
- Inadequate documentation
- Human errors
- Changes to existing code
- Poor maintenance

IMPACT OF SOFTWARE DEFECTS

- Software defects can have a major impact on individuals, businesses and society.
- They can lead to problems such as system crashes, data loss, security breaches, and even financial losses.
- For businesses, software defects can lead to lost revenues, customer dissatisfaction and even legal action.

HOW TO DETECT SOFTWARE DEFECTS

- Quick Attacks on Real Browsers and Devices
- Pay Attention to the Test Environment
- Do Your Own Research
- Pareto Principle
- Set Goals for Software Quality

HOW TO FIX SOFTWARE DEFECTS

1. **Identify the defect:** The first step to fixing a software defect is to identify the defect and determine the cause. This can be done by analyzing the code and understanding the program flow.

2. **Reproduce the defect:** Once the cause is identified, it is important to reproduce the defect so that further investigation can be done.

3. **Develop a fix:** Once the defect is reproduced and the cause is identified, a fix can be developed to address the issue.

4. **Test the fix:** Once the fix is developed, it is important to test the fix to ensure it works correctly and does not introduce any new defects.

5. **Deploy the fix:** Finally, once the fix has been tested and is known to work correctly, it can be deployed into production.

OBJECTIVE

Early detection and correction of software defects is essential in the software development process. In the production stage, software with defects negatively impacts on operational costs and ultimately affects customer satisfaction. Although different approaches exist to predict software defects, two essential factors are timely and accurate detection.

This presentation presents **Logistic Regression, Support Vector Machine Algorithm, Decision Tree Algorithm, Random Forest & Bagging** for enhanced prediction of software defects .

We try to enhance our prediction of software defects further by using various validation methods, resampling data.

DATASETS USED

KC1:- http://promise.site.uottawa.ca/SERepository/datasets/kc1.arff

KC2:- http://promise.site.uottawa.ca/SERepository/datasets/kc2.arff

CM1:- http://promise.site.uottawa.ca/SERepository/datasets/cm1.arff

PC1:- http://promise.site.uottawa.ca/SERepository/datasets/pc1.arff

JM1:- http://promise.site.uottawa.ca/SERepository/datasets/jm1.arff

AR1:- https://zenodo.org/record/322460

AR3:- https://zenodo.org/record/322460

AR4:- https://zenodo.org/record/322460

AR5:- https://zenodo.org/record/322460

AR6:- https://zenodo.org/record/322460

DATASET DESCRIPTION

KC1:- 2108 rows × 22 columns

KC2:- 521 rows × 22 columns

CM1:- 497 rows × 22 columns

PC1:- 1108 rows × 22 columns

JM1:- 1108 rows × 22 columns

AR1:- 120 rows × 30 columns

AR3:- 62 rows × 30 columns

AR4:- 106 rows × 30 columns

AR5:- 35 rows × 30 columns

AR6:- 100 rows × 30 columns

SOFTWARE DEFECT USING MACHINE LEARNING MODELS

We use **Logistic Regression, Support Vector Machine Algorithm, Decision Tree Algorithm, Random Forest & Bagging** for enhanced prediction of software defects .

Dissertation.

Dissertation K Cross Validation.

Dissertation Stratified.

Dissertation Stratified Train.

Dissertation Shuffle Split.

Dissertation SMOTE.

Dissertation SMOTE using K Cross Validation.

ACCURACY TABLE

	kc2	kc1	cm1	pc1	jm1	ar1	ar3	ar4	ar5	ar6
Logistic Regression	0.829	0.848	0.93	0.923	0.814	0.875	0.846	0.909	1	0.85
Support Vector Machine Algorithm	0.895	0.846	0.93	0.932	0.816	0.875	0.923	0.864	0.714	0.85
Decision Tree Algorithm	0.81	0.839	0.88	0.937	0.76	0.875	0.923	0.818	0.714	0.75
Random Forest	0.79	0.855	0.92	0.932	0.813	0.875	0.923	0.818	0.714	0.8
Bagging	0.838	0.818	0.89	0.923	0.756	0.833	0.923	0.773	0.857	0.7

ACCURACY USING STRATIFIED ON TRAIN SET

	kc2	kc1	cm1	pc1	jm1	ar1	ar3	ar4	ar5	ar6
Logistic Regression	0.829	0.848	0.93	0.923	0.814	0.875	0.846	0.909	1	0.85
Support Vector Machine Algorithm	0.895	0.846	0.93	0.932	0.816	0.875	0.923	0.864	0.714	0.85
Decision Tree Algorithm	0.81	0.839	0.88	0.937	0.76	0.875	0.923	0.818	0.714	0.75
Random Forest	0.79	0.855	0.92	0.932	0.813	0.875	0.923	0.818	0.714	0.8
Bagging	0.838	0.818	0.89	0.923	0.756	0.833	0.923	0.773	0.857	0.7

CONCLUSION

We first apply machine learning model dataset that shows good accuracy. But all the datasets have class imbalance problem. Datasets are low in instance so we cannot simply drop instance in order to class imbalance. We tried various validation and resampling(up sampling) the data