AI into Software Testing

Pre-Interview exercises answers

Name- Kurupitage Kavindhya Dhananjanee

NIC - 996550052v

Table of Content

Table of Figures

Problem 1: Defect Prediction

The main objective of the defect prediction is a identifying potential issues or errors in the software development process. Defect prediction involves the analysis of historical data to predict which parts of the software codebase are more likely to be prone to defects or bugs. By leveraging various metrics and features such as code churn, code complexity, defect history, fix history, developer experience, developer activity, code review comments, static code analysis results, test coverage, number of test cases, number of contributors, collaboration history, time since last change, time since last defect fix, project size, project phase, number of revisions, external library usage helps software teams focus their testing and quality assurance efforts more effectively. Defect prediction is the software quality and reduce the bugs in production. Through the use of machine learning algorithms or statistical models, defect prediction contributes to a more proactive and preemptive approach to software quality management. The defect prediction can be categorized to several stages. They are,

- Describing data
- Feature engineering
- Data preprocessing
- Suitable algorithm selection
- Model training and model evaluation

1.1. Describing data

In this scenario I have used my own python generated dataset as an example. But in realistic world we can get more accurate datasets with more features. In the data understanding stage we should have a expand knowledge about the features that are included in the dataset. There are several feature in my dataset which I have generated,

1.Code churn- amount of code changes like additions, deletions, modifications.

2.Code Complexity- complex code may be more error prone.

3.Defect History- the historical count of defects

4.Fix History- historical count of how many times defects in a code

5.Developer Experience- level of expertise

6.Developer Activity- the overall engagement and contribution of a developer in the project.

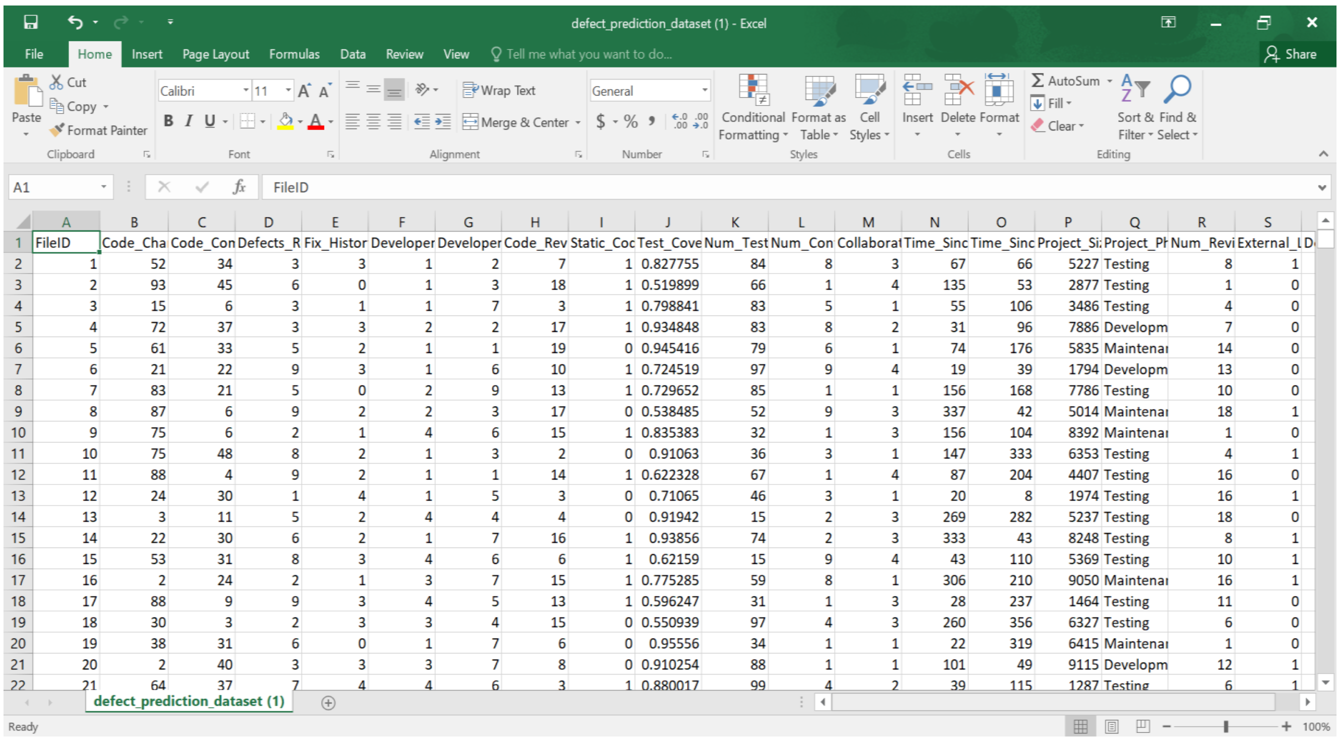
7.Code Review Comments- the number of comments made during the code review process.

8.Static Code Analysis Results- this highlight areas that may have code quality concerns and could be prone to defects.

9.Test Coverage- percentage of code exercised by automated tests.

10.Number of Test Cases- quantity of test cases

I have generated a example dataset according to this scenario . It will be a great help to explain the real plan for defects prediction. This is not a random dataset. In real world problems we can use random datasets for defects prediction. In this case I have used several important features for the dataset and I gave some conditions and generated a dataset using python programming language. I have included a screen shot of my dataset below.



FileID	Code_Cha	Code_Con	Defects_R	Fix_Histor	Developer	Developer	Code_Rev	Static_Coc	Test_Cove	Num_Test	Num_Con	Collaborat	Time_Sinc	Time_Sinc	Project_Si	Project_Ph	Num_Revi	External_l
1	52	34	3	3	1	2	7	1	0.827755	84	8	3	67	66	5227	Testing	8	1
2	93	45	6	0	1	3	18	1	0.519899	66	1	4	135	53	2877	Testing	1	0
3	15	6	3	1	1	7	3	1	0.798841	83	5	1	55	106	3486	Testing	4	0
4	72	37	3	3	2	2	17	1	0.934848	83	8	2	31	96	7886	Developm	7	0
5	61	33	5	2	1	1	19	0	0.945416	79	6	1	74	176	5835	Maintenai	14	0
6	21	22	9	3	1	6	10	1	0.724519	97	9	4	19	39	1794	Developm	13	0
7	83	21	5	0	2	9	13	1	0.729652	85	1	1	156	168	7786	Testing	10	0
8	87	6	9	2	2	3	17	0	0.538485	52	9	3	337	42	5014	Maintenai	18	1
9	75	6	2	1	4	6	15	1	0.835383	32	1	3	156	104	8392	Maintenai	1	0
10	75	48	8	2	1	3	2	0	0.91063	36	3	1	147	333	6353	Testing	4	1
11	88	4	9	2	1	1	14	1	0.622328	67	1	4	87	204	4407	Testing	16	0
12	24	30	1	4	1	5	3	0	0.71065	46	3	1	20	8	1974	Testing	16	1
13	3	11	5	2	4	4	4	0	0.91942	15	2	3	269	282	5237	Testing	18	0
14	22	30	6	2	1	7	16	1	0.93856	74	2	3	333	43	8248	Testing	8	1
15	53	31	8	3	4	6	6	1	0.62159	15	9	4	43	110	5369	Testing	10	1
16	2	24	2	1	3	7	15	1	0.775285	59	8	1	306	210	9050	Maintenai	16	1
17	88	9	9	3	4	5	13	1	0.596247	31	1	3	28	237	1464	Testing	11	0
18	30	3	2	3	3	4	15	0	0.550939	97	4	3	260	356	6327	Testing	6	0
19	38	31	6	0	1	7	6	0	0.95556	34	1	1	22	319	6415	Maintenai	1	0
20	2	40	3	3	3	7	8	0	0.910254	88	1	1	101	49	9115	Developm	12	1
21	64	37	7	4	4	6	3	1	0.880017	99	4	2	39	115	1287	Testing	6	1

Figure 1-Generated example dataset

1.2. Feature engineering

In the feature engineering we can create new features and modifying the existing features, apart from that feature engineering enhance the predictive power of the AI model.

Below I have mentioned the importance of the features in our dataset.

1.Code churn- this indicate the frequent changes in the code

2.Code Complexity- complex code may be more error-prone

3.Defect History-defect history can suggest future defects

4.Fix History- it reduce the recurring defects and indicate ongoing efforts to improve code quality.

5.Developer Experience- they have best practices, therefore they make fewer defects

6.Developer Activity- active developers have very high impact on the quality of code

7.Code Review Comments-.this reduce the defects count

8.Static Code Analysis Results- this highlight the code areas that can have defects

9.Test Coverage- large test coverage may indicate more robust testing

10.Number of Test Cases- number of test cases mainly effect to the defect prevention

1.3. Data preprocessing

In this stage , we can clean our data to make it suitable for machine learning model. Under the data preprocessing stage we can handle the missing data values, convert categorical variables into numerical format using one-hot encoding or label encoding, scaling numerical features, identify and address outliers, adjust the imbalanced data. There are few example python codes for data preprocessing.

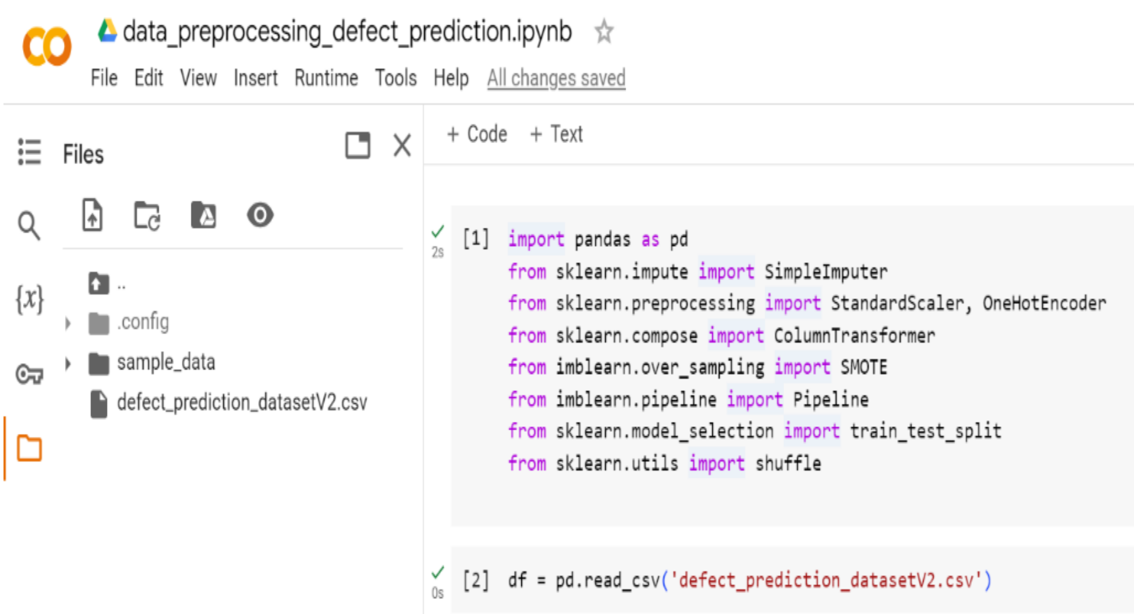


Figure 2-Import related libraries and dataset

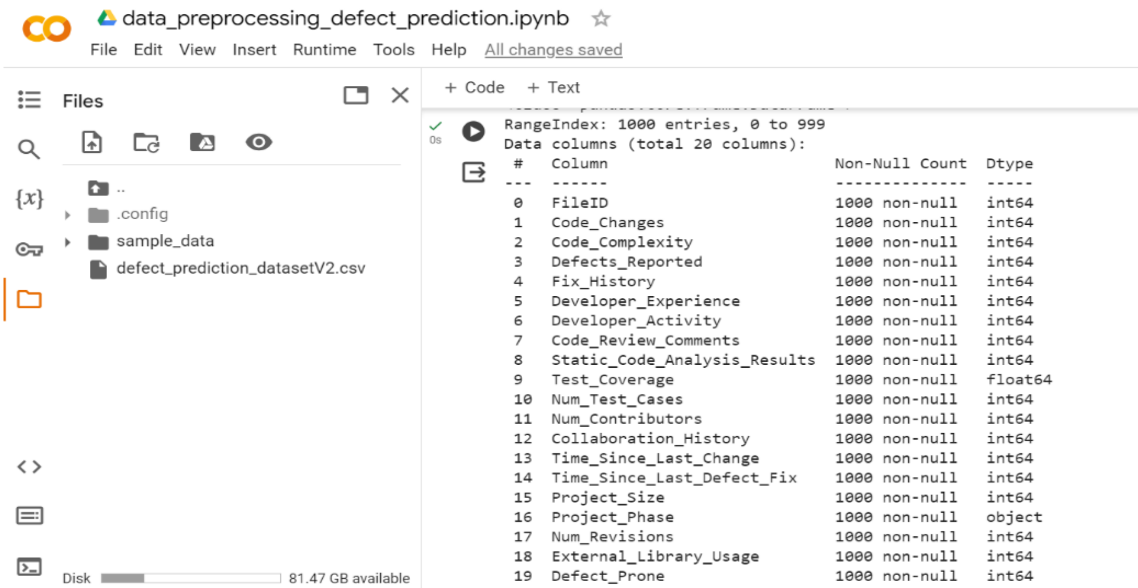


Figure 3-Initial dataset information printing (nun values and data types)

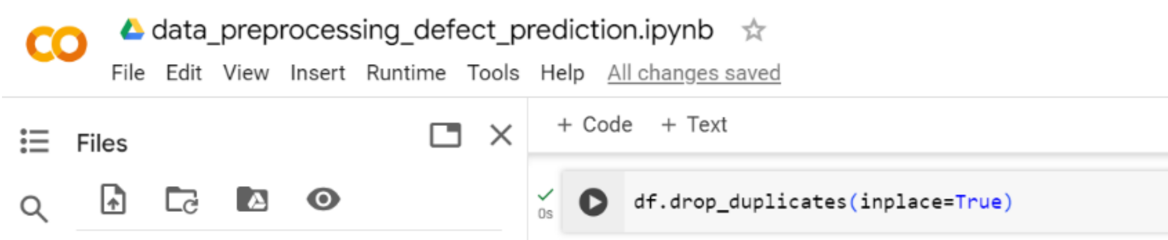


Figure 4-Duplicate values drop in the dataset

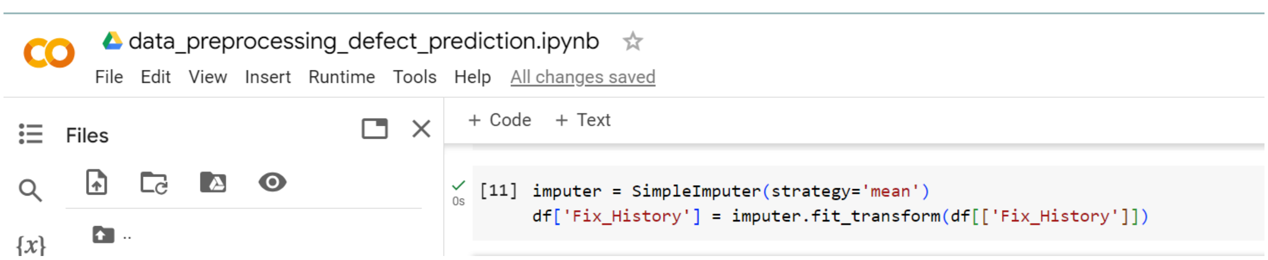


Figure 5-Missing values handle

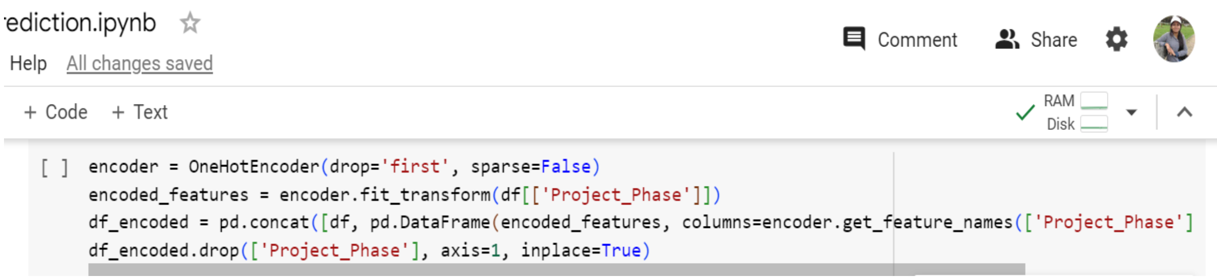


Figure 6-Encoding categorical variables using one-hot encoding

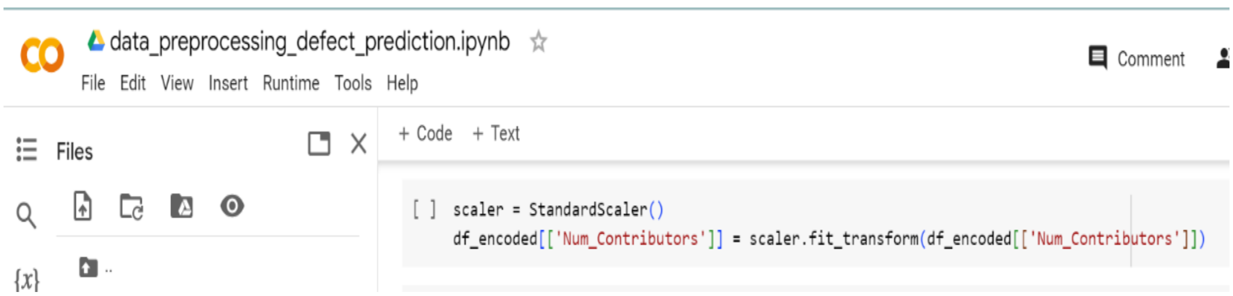


Figure 7-Scaling numerical features

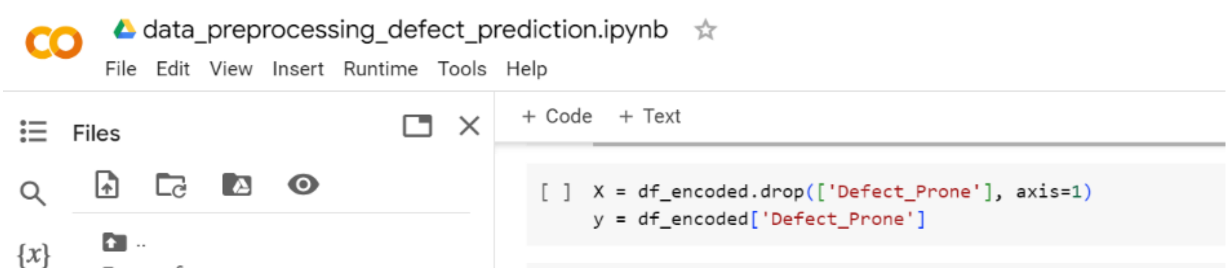


Figure 8-Dealing with imbalanced data

1.4. Suitable algorithm selection

There are lot of algorithms and methods in artificial intelligence. But when we select a algorithm or method for our problem, it should be most careful. Performance is the most obvious metric when selecting a algorithm for a task. There are several factors when implementing a algorithms. They are,

The number of data point and features

Interpretability

Data format

Training time

Prediction time

Memory requirement

Linearity of data

According to the given task, firstly we should categorized it as regression problem or whether classification problem. In the defect prediction the final prediction should be error prone or risky. That should be a binary value. Therefore the category should be the classification. Specifically, it's a binary classification problem since the outcome falls into one of two classes, error-prone or risky. In classification problems, the goal is to predict the category or class to which a new instance belongs based on input features.

According to the research papers and comparison I have selected several algorithms for this problem. I have mentioned the reasons for selecting those algorithms.

Random Forest- Work efficiently with largest data sets, robust to overfitting, familiar with non-linearity

Logistic Regression- Can handle large datasets

Naïve Bayes- This computationally very efficient , handle large datasets very well and simple algorithm

Support Vector Machine- Suitable for large data sets and effective in high dimensional spaces.

Ensemble Model- Combines multiple classifiers for improved

I have trained these four algorithms to get the best accuracy. Below I have attached the screenshots of my google colab python codes,

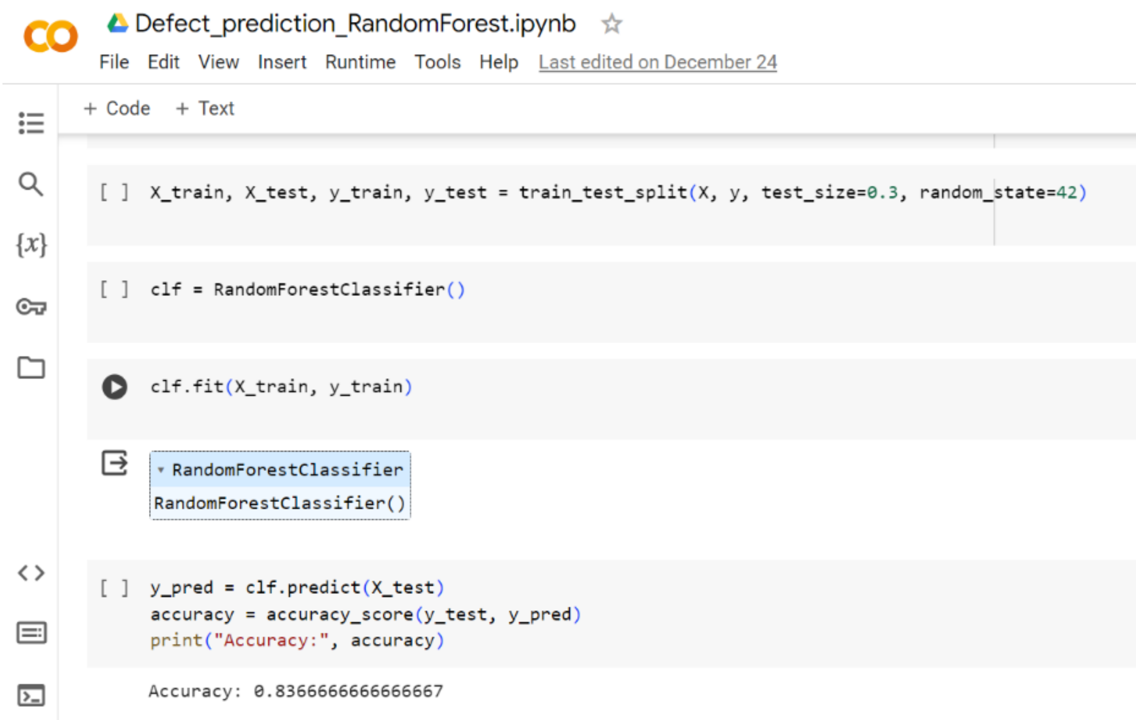


Figure 9-Dataset train using Random Forest algorithm

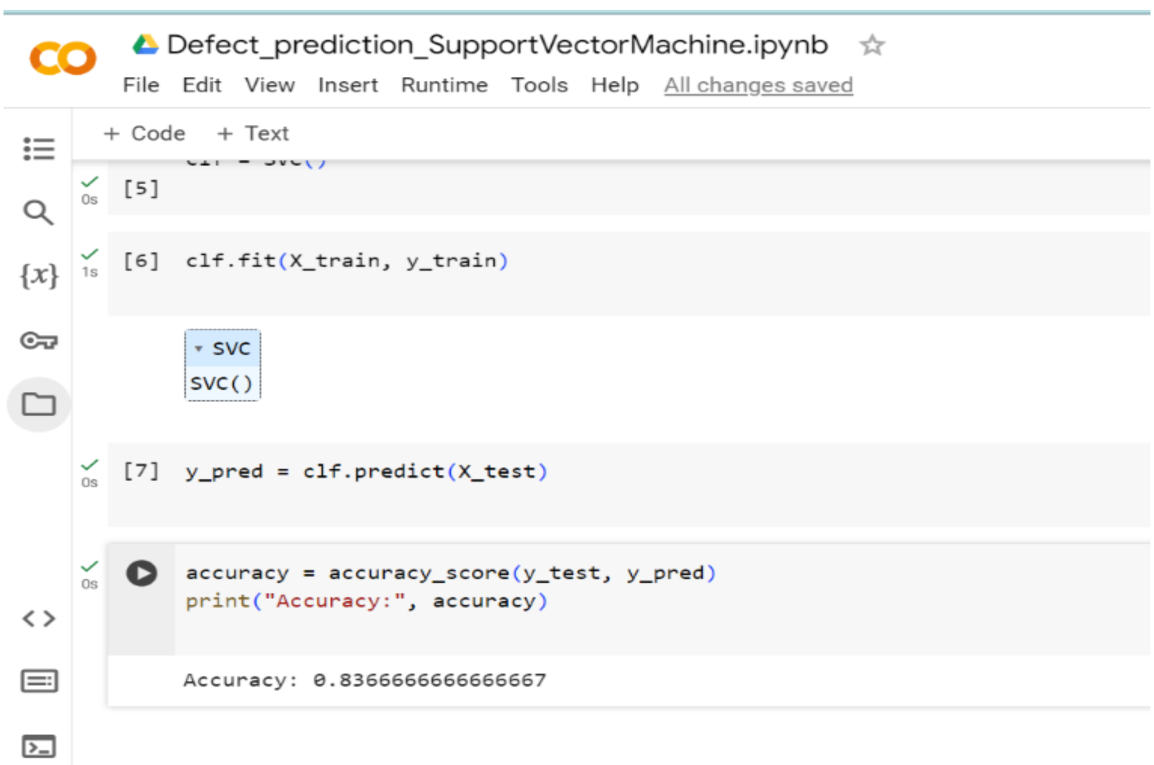


Figure 10-Dataset train using Support Vector Machine algorithm

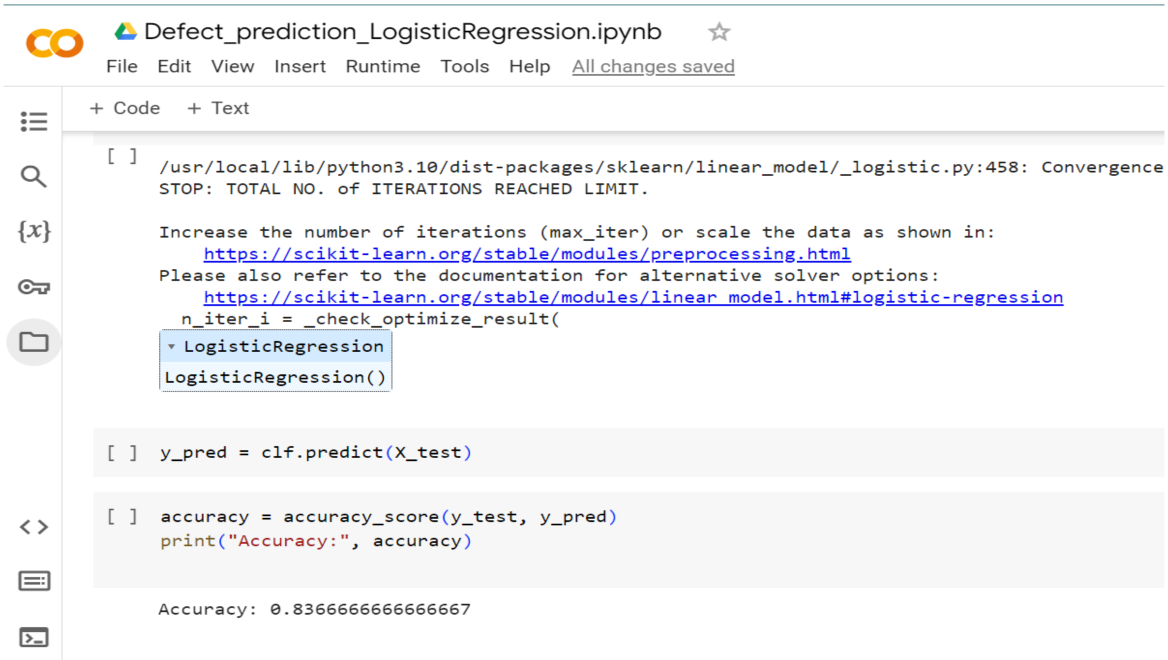


Figure 11-Dataset train using Logistic Regression algorithm

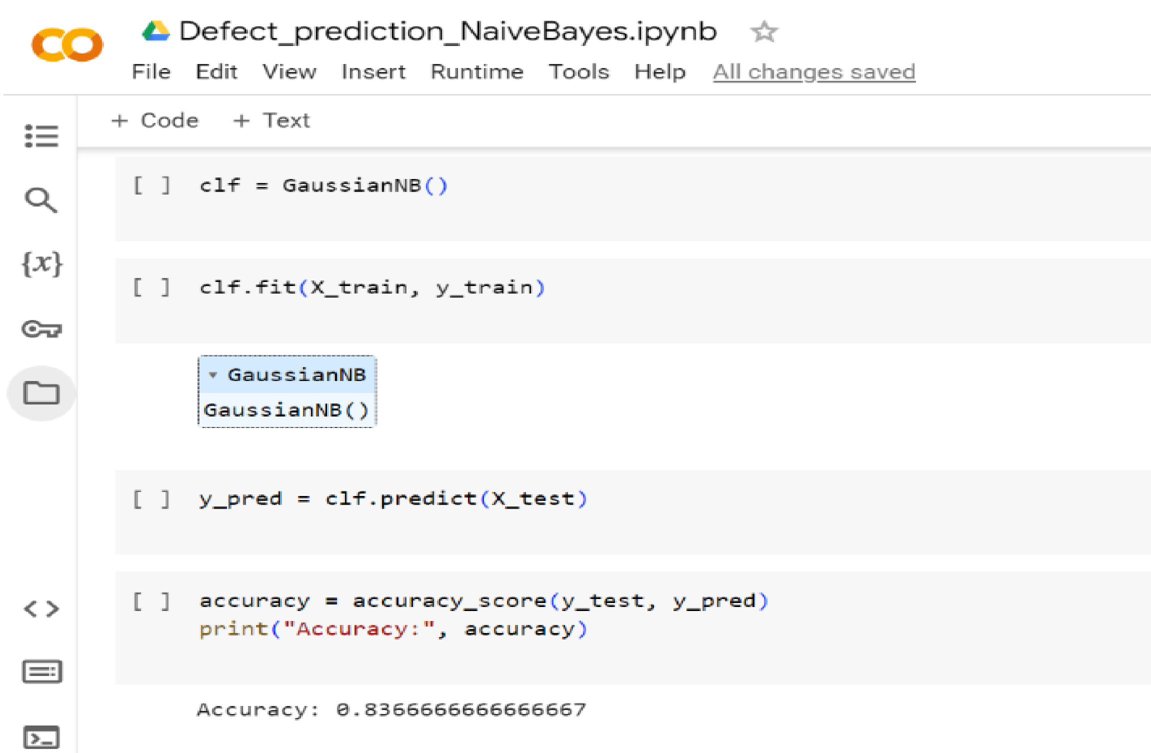


Figure 12-Dataset train using Naive Bayes algorithm

1.5. Model training and model evaluation

I have selected random forest algorithm for this scenario. Because this helps to reduce overfitting and improves performance, As well as this algorithm handle non-linearity of the dataset. It captures the complex relationships among the data points. It provides a measure of feature importance, which can be valuable for understanding which features contributes the most to the prediction.

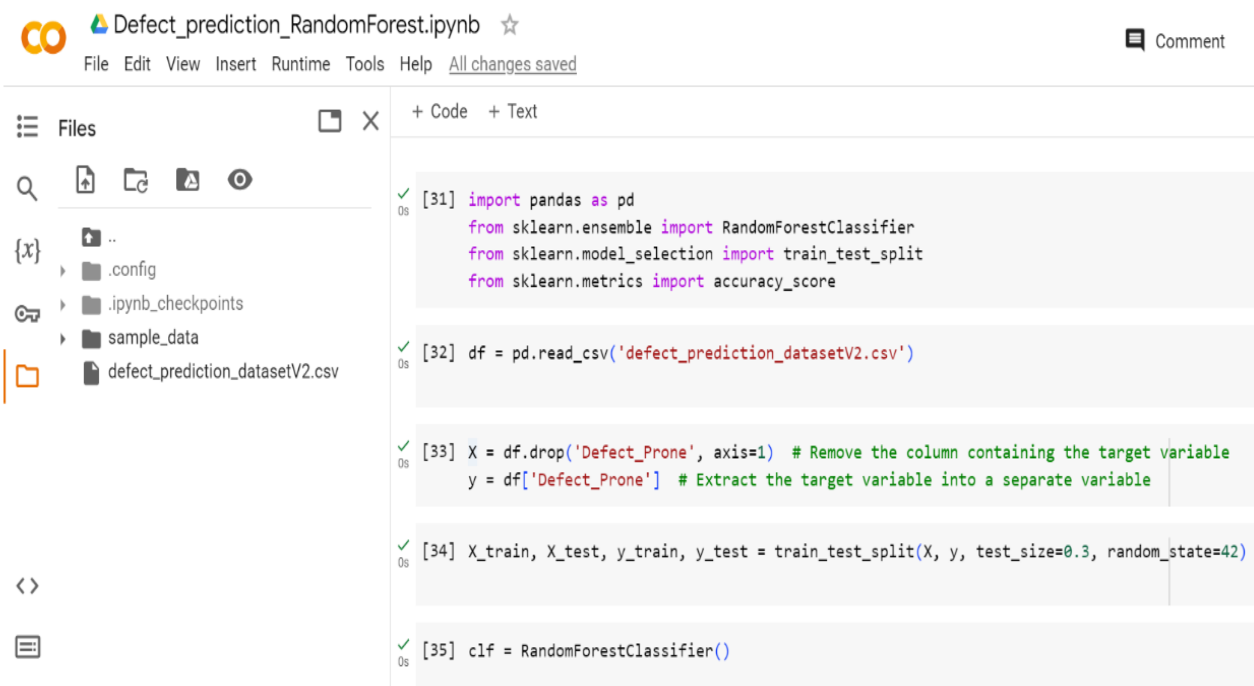


Figure 13-Random forest training 1

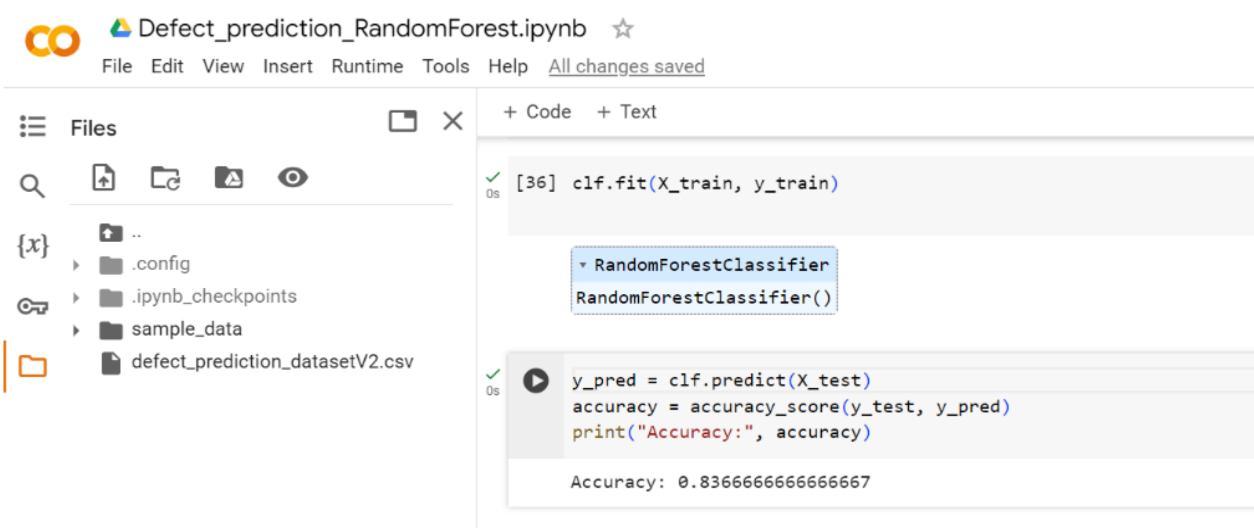


Figure 14-Random forest training 2

Apart from that we can use various hyper parameter optimization techniques like Grid search , Random search ,Bayesian Optimization, Hyperband like that. In this scenario I have used Grid search technique.

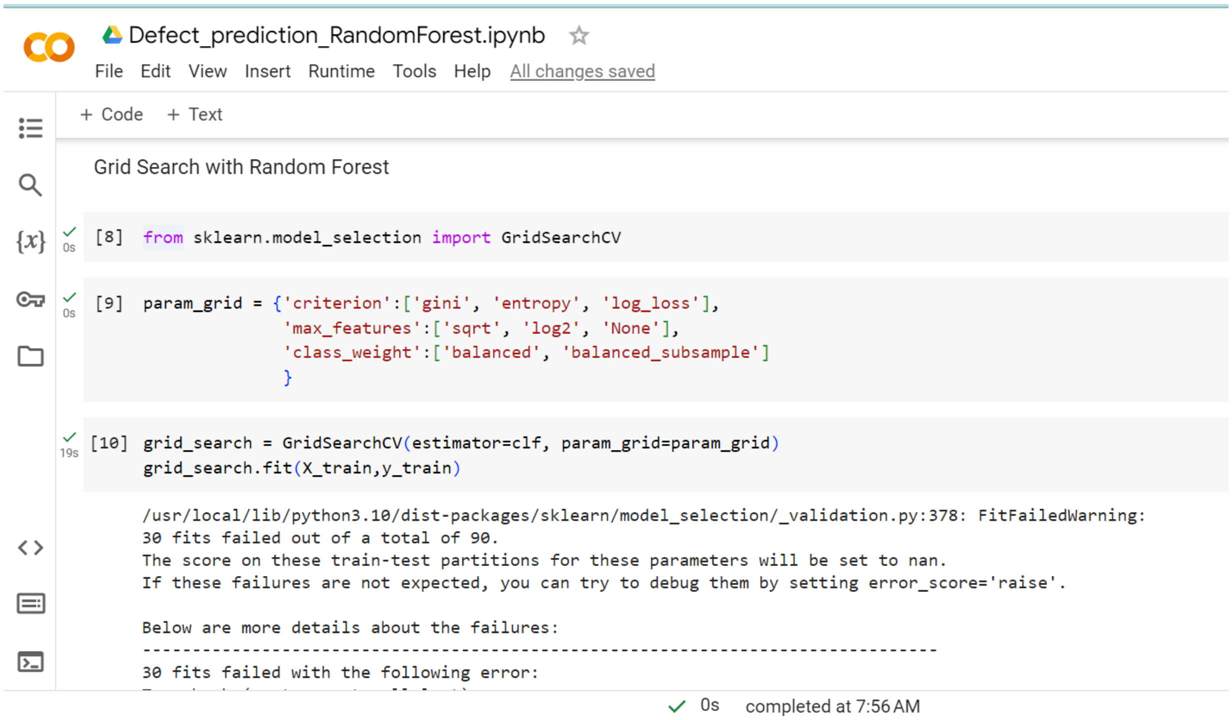


Figure 15-Random forest hyper parameter tuning using grid search technique

In the model evaluation stage, if we want we can adjust the hyper parameter and optimize the performance of the algorithm. Apart from that we can use precision, recall, F1 score and area under the ROC curve to evaluate the model's performances.

1.6. Assumptions

- In this problem I have used a generated dataset using a python code and under my own conditions. It is used only for an example to understand my plan easily.
- As well as I have assumed the features I have selected are relevant and sufficient to capture the characteristics of defect prone code.
- I have assumed the historical data on reported defects is a good indicator of future defect-prone areas.

Problem 2: Smart Test Selector

Software components can have some defects. Developers can not fix those all of errors during the development process. Therefore it should be use software testing part before the delivery the software. It can be done using manual methods and smart methods also. With the growth in emerging technologies, manual software testing process is a challenge to do test planning, test case creation, execution and reporting. Therefore we can use smart software testers with all of good attributes which have a good smart tester. There are several main components of a test strategy are,

- The scope of testing
- Test objectives
- Budget limitation
- Communication and status reporting
- Industry standards
- Testing measurement and metrics
- Defect reporting and tracking
- Configuration management
- Deadlines
- Test execution schedule
- Risk identification

Examples for smart software testers;

Ktalon Studio

JMeter

Lmbda Test

BrowserStack

Testsigma

Jira

Saoce Labs

Selenium

Appium

Robot Framework

Product	Katalon	Selenium	appium	TestComplete	cypress
Application Under Test	Web/API/ Mobile/Desktop	Web	Mobile (Android/iOS)	Web/Mobile/ Desktop	Web
Supported platform(s)	Windows/ macOS/ Linux	Windows/ macOS/ Linux/Solaris	Windows/ macOS	Windows	Windows/ macOS/ Linux
Setup & configuration	Easy	Coding Required	Coding Required	Easy	Coding Required
Low-code & Scripting mode	Both	Scripting Only	Scripting Only	Both	Scripting Only
Supported language(s)	Java & Groovy	Java, C#, Python, JavaScript, Ruby, PHP, Perl	Java, C#, Python, JavaScript, Ruby, PHP, Perl	JavaScript, Python, VBScript, JScript, Delphi, C++, C#	JavaScript
Advanced test reporting	✓	✗	✗	✗	✓
Pricing	Free and Paid	Free	Free	Paid	Free and Paid
Ratings & Reviews (Gartner)	**4.4/5 740 reviews**	**4.5/5 443 reviews**	4.4/5 90 reviews	4.4/5 45 reviews	4.6/5 27 reviews

Figure 16- Most famous test frameworks comparison

Selecting an appropriate set of tests to run efficiently is crucial for effective testing process. Firstly we should understand the code changes. We should consider about the impact of code changers to the full system and functions. And next should define the test objectives clearly. We should give priority to critical test cases. As well as This should be a risk based testing. We can focus testing efforts on areas with higher risks. Apart from that we should consider about the code coverage analysis. The special one is test automation optimization. We can use parallel execution methods also.

As well as there are several criteria for test selection. Code changes give higher priority to tests covering area directly. Historical test results also effect mainly for test selection. Other one is dependency analysis. Tests that cover modules with dependencies on the changed code are selected.

We can use various tools and techniques for smart testing;

- **CI/CD platform**
  Jenkins, GitLab Cl, Travis Cl
- **Version control system**
  Git
- **Static code analysis**
  SonarQube
- **Test automation frameworks**
  Selenium, Junit, TestNG
- **Test case management**
  TestRail, qTest

There are can be several challenges when selecting a suitable smart tester. Adapting to dynamic code change is a main challenge. So we can implement real-time monitoring to minimize this challenge. Other main challenge is maintaining test suitability. Integration with CI/CD tools is another challenge, but we can collaborate with CI/CD platform experts. Minimizing false positive in test selection is also a challenge.

However implementing a smart test selection mechanism is an iterative process that requires collaboration, monitoring and continuous improvement.

REFERENCE

[1] *Ieee.org*. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9293650. [Accessed: 27-Dec-2023].

[2] *"The Smart Tester," BrainKart. [Online]. Available: https://www.brainkart.com/article/The-Smart-Tester_9146/. [Accessed: 27-Dec-2023].*

[3] *"What are the Qualities of a Good Software Tester," QC More, 11-May-2016. [Online]. Available: https://www.qcmore.com/blog/httpwww-qcmore-comblogtraits-smart-tester/. [Accessed: 27-Dec-2023].*

[4] *"11 ways to improve software testing through planning, work environment, automated testing, and reporting," AltexSoft, 12-Feb-2021. .*

[5] *"Top 15 automation testing tools," katalon.com. .*

[6] *"QA Manager Goal Setting Template," Clickup.com. [Online]. Available: https://clickup.com/templates/goal-setting/qa-manager. [Accessed: 27-Dec-2023].*

[7] *"12 different types of testing tools (recommended by tester's)," Testsigma Blog, 06-Oct-2020. [Online]. Available: https://testsigma.com/blog/different-types-of-tools-to-help-testers-day-to-day-life-easier/. [Accessed: 27-Dec-2023].*

[8] *S. J. Bigelow, "CI/CD pipelines explained: Everything you need to know," Software Quality, 13-May-2021. [Online]. Available: https://www.techtarget.com/searchsoftwarequality/CI-CD-pipelines-explained-Everything-you-need-to-know. [Accessed: 27-Dec-2023].*