Guideline for Risk Prediction Expert System

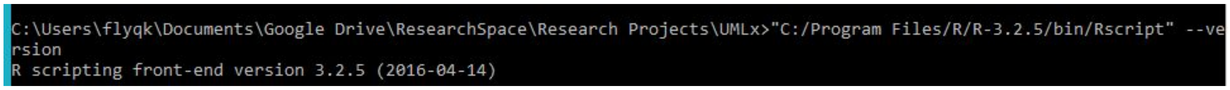
Overview

This document aims at providing a tutorial and examples for the reader to access the operational details of risk prediction model.
Section I - VI give the tutorial and example based on the given data in the attached package.
Section VII provides instruction about where and how to collect data from your own data sources.

I Environment Setup

Step 1: Install R (3.2.5)

1. Current experiment is based on R-3.2.5, which is available at:
   https://cran.r-project.org/bin/windows/base/old/3.2.5/
2. Open Command Prompt of Windows OS, and type the following command to see if R is correctly installed.
   "C:/Program Files/R/R-3.2.5/bin/Rscript" --version



```
C:\Users\flyqk\Documents\Google Drive\ResearchSpace\Research Projects\UMLx>"C:/Program Files/R/R-3.2.5/bin/Rscript" --ve
rsion
R scripting front-end version 3.2.5 (2016-04-14)
```

Step 2: Install R packages

1. Login R environment.
2. Type command:
   install.packages("neuralnet")
3. Type command:
   install.packages("ggplot2")

II Data Preparation

In the attached package, the documents used as the inputs of the model training and testing processes are as follows:

1. The training dataset for the USC-CSSE risk prediction model:
   "*Data/training_data_set_usc_model.csv*"
2. The training dataset for the Open Source risk prediction model:
   "*Data/training_data_set_open_source_model.cs*v"
3. The example input for testing the risk prediction model:
   "*Data/Input_Data_Example_5_10.csv*"

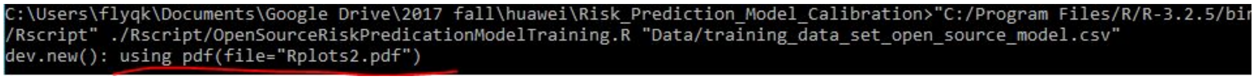
III Train risk prediction models

Train Risk Prediction Model Based on Open Source Data

Step 1: Run the open-source risk prediction model training script.

1. Open Command Prompt of Windows OS, and type:
   "C:/Program Files/R/R-3.2.5/bin/Rscript" ./Rscript/OpenSourceRiskPredicationModelTraining.R "Data/training_data_set_open_source_model.csv"

   **Please follow the steps introduced in slides 33 - 40 of "Advanced Tollgate for 05-03-2018.pptx" or Section VII to create your own dataset.*



```
C:\Users\flyqk\Documents\Google Drive\2017 fall\huawei\Risk_Prediction_Model_Calibration>"C:/Program Files/R/R-3.2.5/bir
/Rscript" ./Rscript/OpenSourceRiskPredicationModelTraining.R "Data/training_data_set_open_source_model.csv"
dev.new(): using pdf(file="Rplots2.pdf")
```

Step 2: Check the output files.

2. Graphic representation file for the trained model:
   "./Rplots2.pdf".
3. The model training report:
   "./Temp/open-source-risk-prediction-model-training-report.txt"
4. The trained model:
   "./Model/riskPredictionModel_open_source.rds"

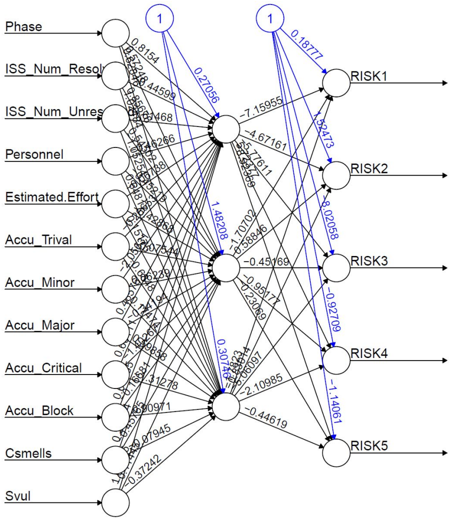


Figure 1. An example of the trained open source risk prediction model

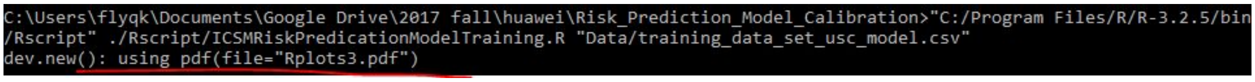
**If you run your data, the results will be produced in the same files but with different results.*

Train Risk Prediction Model Based On USC-CSSE projects

Step 1: Run the USC-CSSE risk prediction model training script.

1. Open Command Prompt of Windows, and type:

   "C:/Program Files/R/R-3.2.5/bin/Rscript" ./Rscript/ICSMRiskPredicationModelTraining.R "Data/training_data_set_usc_model.csv"



```
C:\Users\flyqk\Documents\Google Drive\2017 fall\huawei\Risk_Prediction_Model_Calibration>"C:/Program Files/R/R-3.2.5/bin
/Rscript" ./Rscript/ICSMRiskPredicationModelTraining.R "Data/training_data_set_usc_model.csv"
dev.new(): using pdf(file="Rplots3.pdf")
```

   *Please follow the instructions in Section VII to collect the empirical data to train your own model.

Step 2: Check the output files.

1. Graphic representation file for the trained model:
   “./Rplots3.pdf”
2. The model training report:
   “./Temp/icsm-risk-prediction-model-training-report.txt”
3. The trained model:
   “./Model/riskPredictionMode_icsm_projects.rds”

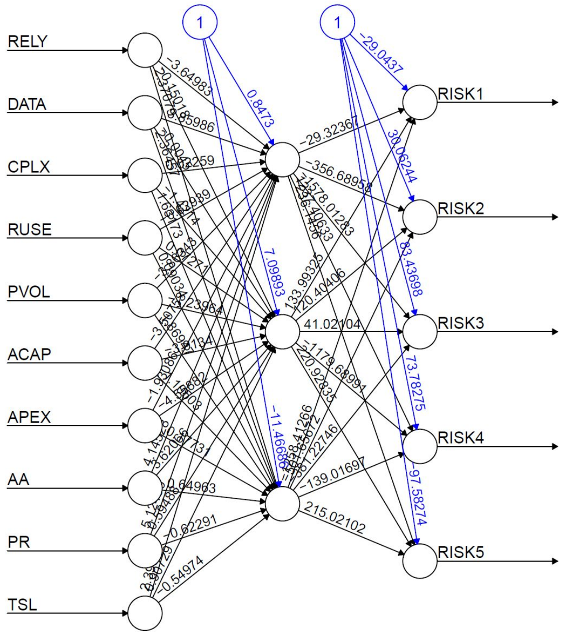


Figure 2. an example of trained USC-CSSE risk prediction model

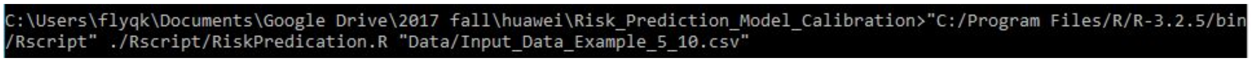
IV Run the trained model for risk prediction

Step 1: Run the risk prediction model test script.

1. Open Command Prompt of Windows, and type:

   "C:/Program Files/R/R-3.2.5/bin/Rscript" ./Rscript/RiskPredication.R "Data/Input_Data_Example_5_10.csv"

   **Please follow the steps introduced in slides 33 - 40 of “Advanced Tollgate for 05-03-2018.pptx” or Section VII to create your own dataset.*

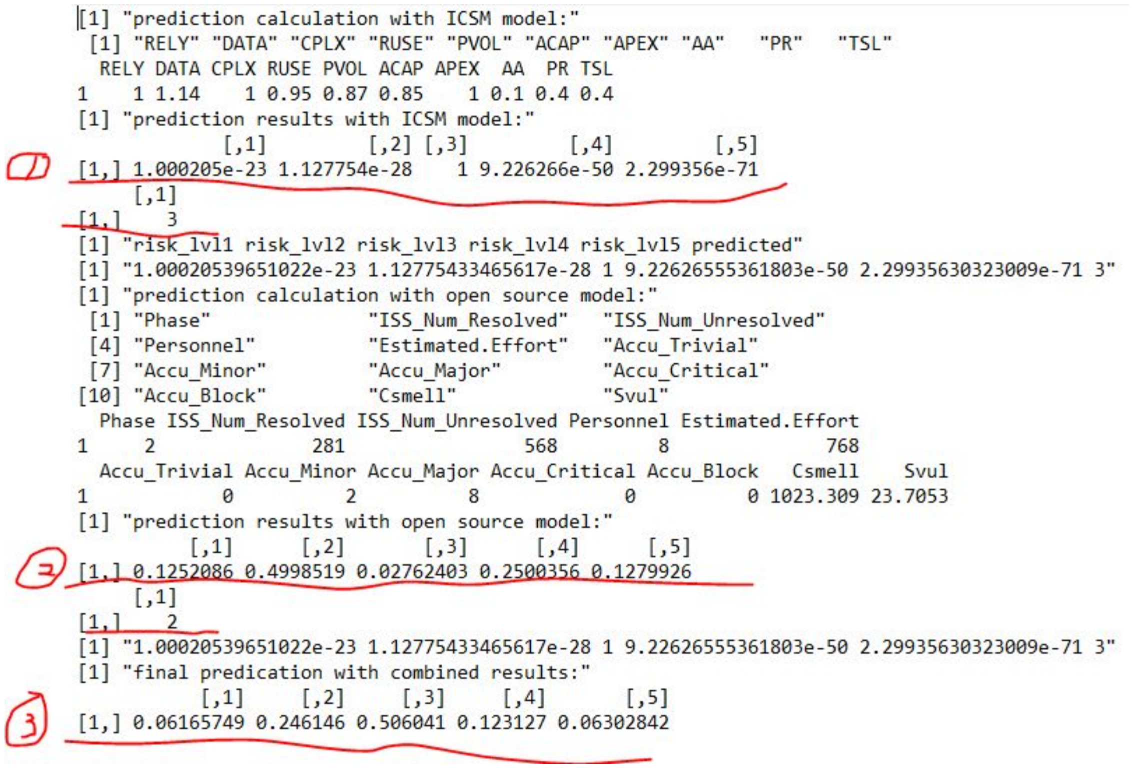


```
C:\Users\flyqk\Documents\Google Drive\2017 fall\huawei\Risk_Prediction_Model_Calibration>"C:/Program Files/R/R-3.2.5/bin
/Rscript" ./Rscript/RiskPredication.R "Data/Input_Data_Example_5_10.csv"
```

Step 2: Check the output files.

1. The risk prediction report:
   “./Data/risk-prediction-report.txt”

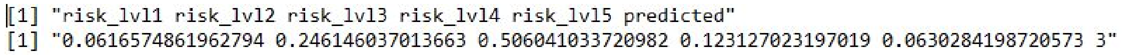
> *An example of the “risk-prediction-report.txt” is as follows:*



```
[1] "prediction calculation with ICSM model:"
 [1] "RELY" "DATA" "CPLX" "RUSE" "PVOL" "ACAP" "APEX" "AA"   "PR"   "TSL"
  RELY DATA CPLX RUSE PVOL ACAP APEX  AA  PR TSL
1    1 1.14    1 0.95 0.87 0.85    1 0.1 0.4 0.4
[1] "prediction results with ICSM model:"
             [,1]         [,2] [,3]         [,4]         [,5]
[1,] 1.000205e-23 1.127754e-28    1 9.226266e-50 2.299356e-71
     [,1]
[1,]    3
[1] "risk_lvl1 risk_lvl2 risk_lvl3 risk_lvl4 risk_lvl5 predicted"
[1] "1.00020539651022e-23 1.12775433465617e-28 1 9.22626555361803e-50 2.29935630323009e-71 3"
[1] "prediction calculation with open source model:"
 [1] "Phase"              "ISS_Num_Resolved"   "ISS_Num_Unresolved"
 [4] "Personnel"          "Estimated.Effort"   "Accu_Trivial"
 [7] "Accu_Minor"         "Accu_Major"         "Accu_Critical"
[10] "Accu_Block"         "Csmell"             "Svul"
  Phase ISS_Num_Resolved ISS_Num_Unresolved Personnel Estimated.Effort
1     2              281                568         8              768
  Accu_Trivial Accu_Minor Accu_Major Accu_Critical Accu_Block   Csmell    Svul
1            0          2          8             0          0 1023.309 23.7053
[1] "prediction results with open source model:"
          [,1]      [,2]       [,3]      [,4]      [,5]
[1,] 0.1252086 0.4998519 0.02762403 0.2500356 0.1279926
     [,1]
[1,]    2
[1] "1.00020539651022e-23 1.12775433465617e-28 1 9.22626555361803e-50 2.29935630323009e-71 3"
[1] "final predication with combined results:"
           [,1]     [,2]     [,3]     [,4]       [,5]
[1,] 0.06165749 0.246146 0.506041 0.123127 0.06302842
```

1. Is the predicted probabilities of the five levels of risk based on the ICSM risk prediction model.
2. Is the predicted probabilities of the five levels of risk based on the open source risk prediction.
3. Bayesian averaged estimates of the probabilities for the five levels of risk. We choose the most probable as the final estimate.

2. The risk prediction report:
   “./Temp/risk-prediction-results.txt”



```
[1] "risk_lvl1 risk_lvl2 risk_lvl3 risk_lvl4 risk_lvl5 predicted"
[1] "0.0616574861962794 0.246146037013663 0.506041033720982 0.123127023197019 0.0630284198720573 3"
```

The simplified output of the risk prediction results, which is used for the risk prediction api.

V Test the risk prediction models

Step 1: Run the risk prediction model test script.

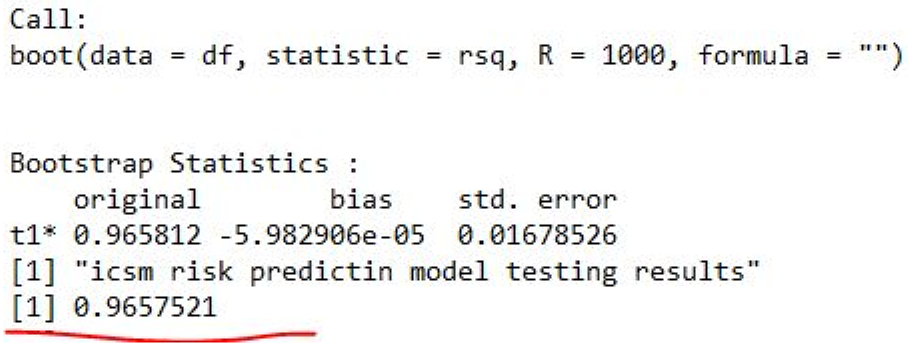
1. Open Command Prompt of Windows, and type:
   "C:/Program Files/R/R-3.2.5/bin/Rscript"
   ./Rscript/RiskPredicationModelTestingBootstrap.R
   "Data/training_data_set_usc_model.csv"
   "Data/training_data_set_open_source_model.csv"

Step 2: Check the output files.

1. The testing report by bootstrapping:
   "./Temp/risk-prediction-model-testing-report.txt"

An example of the "risk-prediction-model-texting-report.txt" is as follows:

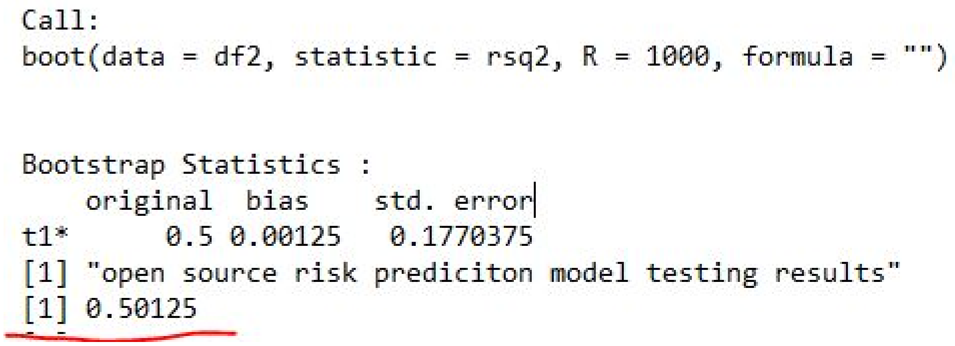
1. The risk prediction accuracy for the ICSM risk prediction model (1000 resampling for bootstrapping estimate).



```
Call:
boot(data = df, statistic = rsq, R = 1000, formula = "")


Bootstrap Statistics :
    original        bias    std. error
t1* 0.965812 -5.982906e-05  0.01678526
[1] "icsm risk predictin model testing results"
[1] 0.9657521
```

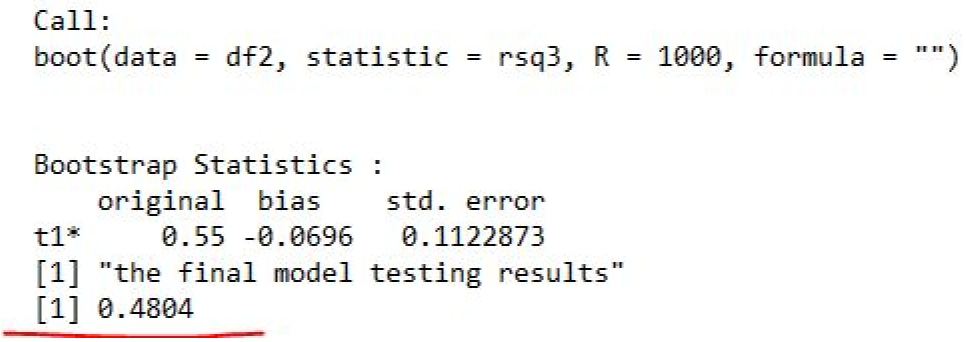
2. The risk prediction accuracy for the open source risk prediction model (1000 resampling for bootstrapping estimate).



```
Call:
boot(data = df2, statistic = rsq2, R = 1000, formula = "")


Bootstrap Statistics :
    original  bias    std. error
t1*      0.5 0.00125   0.1770375
[1] "open source risk prediciton model testing results"
[1] 0.50125
```

3. The risk prediction accuracy for the combined risk prediction model (1000 resampling for bootstrapping estimate).



```
Call:
boot(data = df2, statistic = rsq3, R = 1000, formula = "")


Bootstrap Statistics :
    original  bias    std. error
t1*     0.55 -0.0696   0.1122873
[1] "the final model testing results"
[1] 0.4804
```

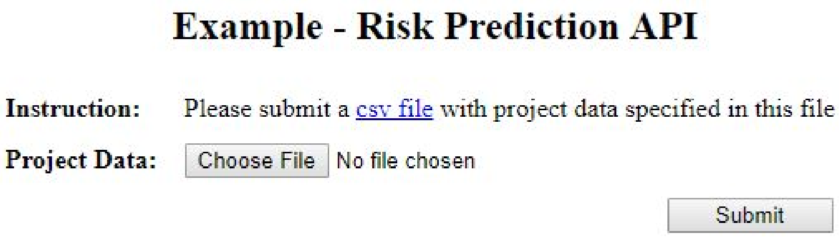
VI Risk Prediction Model API Prototype

This section gives a quick example about how to use a sample data point to request for its risk level from a Amazon EC2 Service, on which we deployed the current risk prediction model.

Step 1: Land the risk prediction API page

Go to this link or type the following address on the browser to load the Risk Prediction API Prototype Page:

http://ec2-54-67-99-52.us-west-1.compute.amazonaws.com:8081



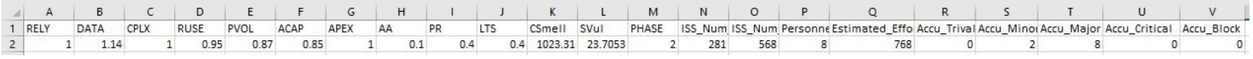
Example - Risk Prediction API

Instruction: Please submit a csv file with project data specified in this file

Project Data:

Step 2: Input sample data

As the input, upload a csv file including the defined factors (Refer to Slide#10 - 21).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	RELY	DATA	CPLX	RUSE	PVOL	ACAP	APEX	AA	PR	LTS	CSmell	SVul	PHASE	ISS_Num	ISS_Num	Personne	Estimated_Effo	Accu_Trival	Accu_Minoı	Accu_Major	Accu_Critical	Accu_Block
2	1	1.14	1	0.95	0.87	0.85	1	0.1	0.4	0.4	1023.31	23.7053	2	281	568	8	768	0	2	8	0	0

A sample input file can be downloaded from the prototype page. It includes the following factors:

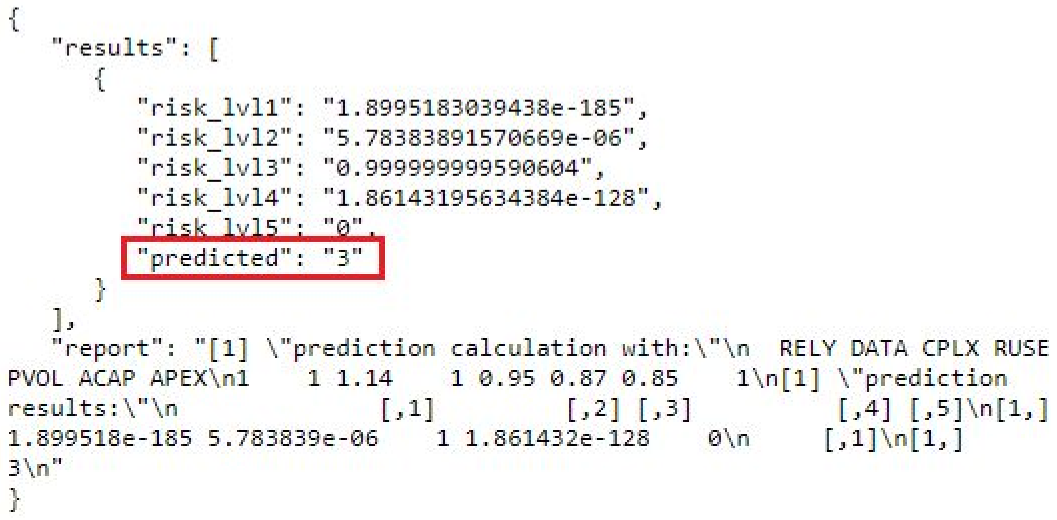
Table 1. Factors for Risk Prediction Models

USC-CSSE Risk Prediction Model			**Open Source Risk Prediction Model**	
Factor	**Description**		**Factor**	**Description**
RELY	Required Software Reliability		Phase	# of Phase
DATA	Database Size		ISS_Num_Resolved	# of resolved issues
CPLX	Product Complexity		ISS_Num_Unresolved	# of unresolved issues
RUSE	Developed for Reusability		Personnel	# of contributors
PVOL	Platform Volatility		Estimated_Effort	Total estimated effort
ACAP	Analyst Capability		Accu_Trivial	# of accumulated trivial issues
APEX	Applications Experience		Accu_Minor	# of accumulated minor issues

AA	Level of Automated Analysis		Accu_Major	# of accumulated major issues
PR	Level of Peer Review		Accu_Critical	# of accumulated critical issues
TSL	Level of Test Sophistication		Accu_Block	# of accumulated block issues
			CSmell	Code Smells
			SVul	Code Security Vulnerability

Step 3: Check output

Once uploaded, the output (i.e. the result of risk prediction), in JSON format, given by the model, will be shown in seconds.



```
{
   "results": [
      {
         "risk_lvl1": "1.8995183039438e-185",
         "risk_lvl2": "5.78383891570669e-06",
         "risk_lvl3": "0.999999999590604",
         "risk_lvl4": "1.86143195634384e-128",
         "risk_lvl5": "0",
         "predicted": "3"
      }
   ],
   "report": "[1] \"prediction calculation with:\"\n  RELY DATA CPLX RUSE
PVOL ACAP APEX\n1    1 1.14    1 0.95 0.87 0.85    1\n[1] \"prediction
results:\"\n              [,1]         [,2] [,3]         [,4] [,5]\n[1,]
1.899518e-185 5.783839e-06    1 1.861432e-128    0\n     [,1]\n[1,]
3\n"
}
```

For example, on the above screenshot, the output lists the possibility of each Risk Level (Refer to Slide# 25 - 26) based on the input factors, and Risk Level 3 is with highest possibility. Therefore the predicted risk level is 3. The output also prints out a brief report about the input factors, as needed, for further insights.

VII Data Collection

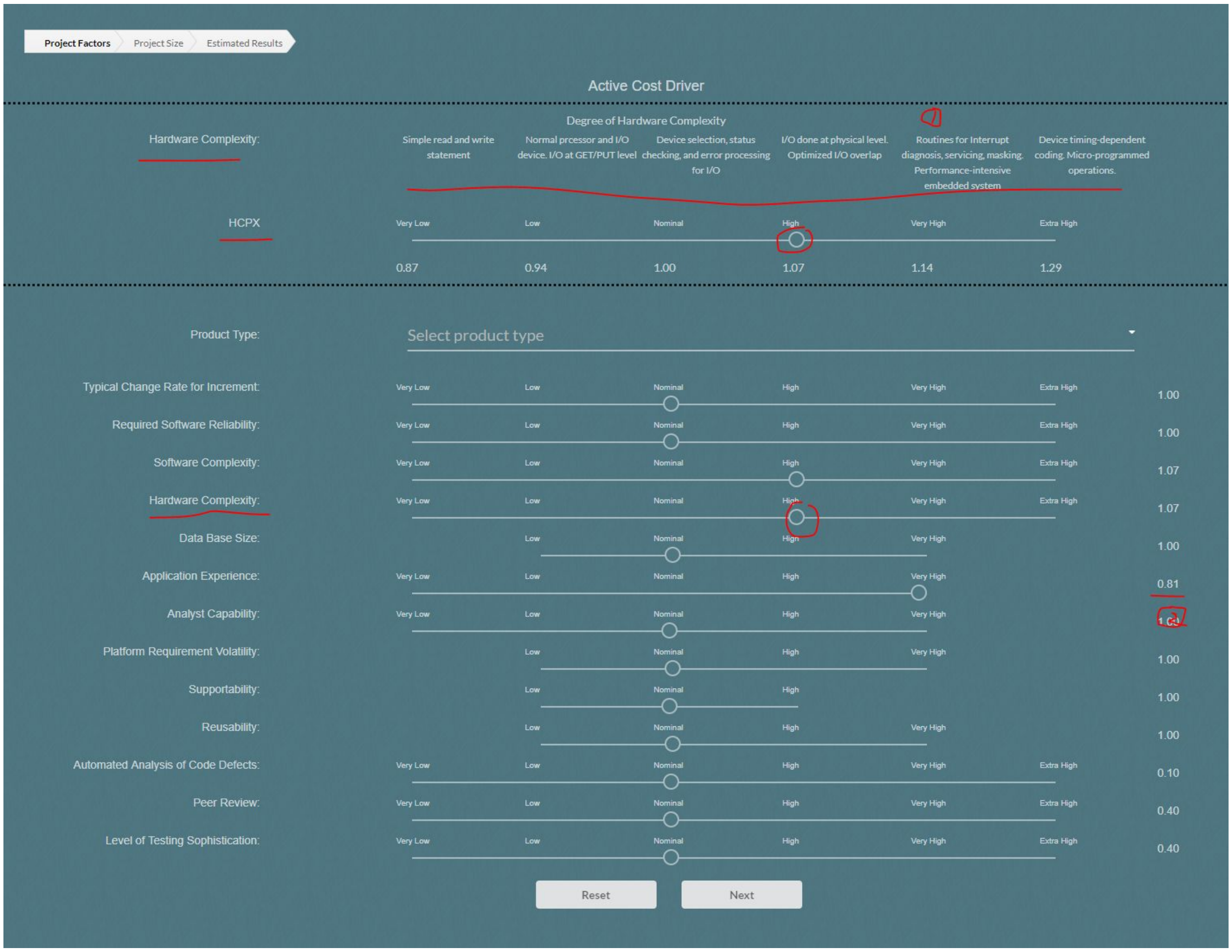
This section provides instructions about where and how to collect data from your own data sources. Specifically data corresponding to the factors listed in Table 1 need to be collected to train your own risk prediction models.

Collecting Data for USC-CSSE Risk Prediction Model.

Step1: Land on Risk Factors Setting Page using this url:

http://ec2-54-67-99-52.us-west-1.compute.amazonaws.com:8686/demo/phase5/start/factorsInputPage.html

This page provides the definitions (1) for factors. Evaluate your project from extra low - extra high for the factors and find their ratings (2).



Active Cost Driver

Hardware Complexity:

Degree of Hardware Complexity 1

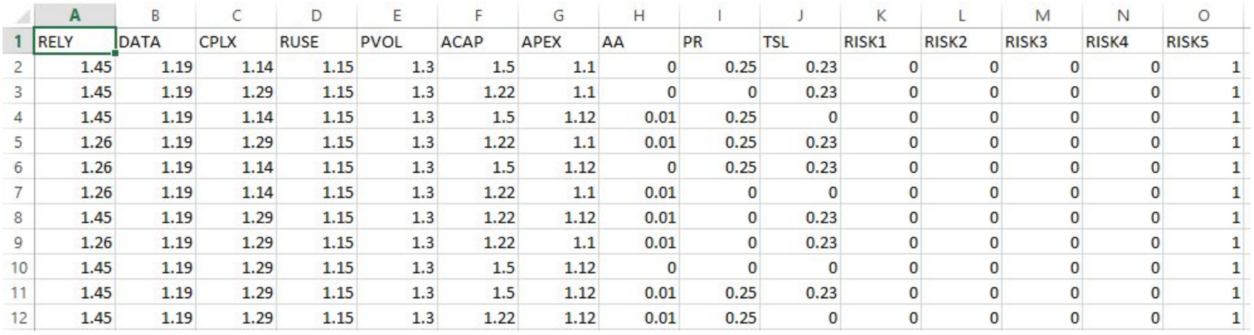
HCPX	Very Low	Low	Nominal	High	Very High	Extra High
	Simple read and write statement	Normal prcessor and I/O device. I/O at GET/PUT level	Device selection, status checking, and error processing for I/O	I/O done at physical level. Optimized I/O overlap	Routines for Interrupt diagnosis, servicing, masking. Performance-intensive embedded system	Device timing-dependent coding. Micro-programmed operations.
	0.87	0.94	1.00	1.07	1.14	1.29

Product Type: Select product type

Factor	Setting	Rating
Typical Change Rate for Increment:	Nominal	1.00
Required Software Reliability:	Nominal	1.00
Software Complexity:	High	1.07
Hardware Complexity:	High	1.07
Data Base Size:	Nominal	1.00
Application Experience:	Very High	0.81
Analyst Capability:	Nominal	1.0 2
Platform Requirement Volatility:	Nominal	1.00
Supportability:	Nominal	1.00
Reusability:	Nominal	1.00
Automated Analysis of Code Defects:	Nominal	0.10
Peer Review:	Nominal	0.40
Level of Testing Sophistication:	Nominal	0.40

Figure 3. Expert system prototype for risk evaluation.

Step 2: Create a data sheet in .CSV format for your evaluation of the project



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	RELY	DATA	CPLX	RUSE	PVOL	ACAP	APEX	AA	PR	TSL	RISK1	RISK2	RISK3	RISK4	RISK5
2	1.45	1.19	1.14	1.15	1.3	1.5	1.1	0	0.25	0.23	0	0	0	0	1
3	1.45	1.19	1.29	1.15	1.3	1.22	1.1	0	0	0.23	0	0	0	0	1
4	1.45	1.19	1.14	1.15	1.3	1.5	1.12	0.01	0.25	0	0	0	0	0	1
5	1.26	1.19	1.29	1.15	1.3	1.22	1.1	0.01	0.25	0.23	0	0	0	0	1
6	1.26	1.19	1.14	1.15	1.3	1.5	1.12	0	0.25	0.23	0	0	0	0	1
7	1.26	1.19	1.14	1.15	1.3	1.22	1.1	0.01	0	0	0	0	0	0	1
8	1.45	1.19	1.29	1.15	1.3	1.22	1.12	0.01	0	0.23	0	0	0	0	1
9	1.26	1.19	1.29	1.15	1.3	1.22	1.1	0.01	0	0.23	0	0	0	0	1
10	1.45	1.19	1.29	1.15	1.3	1.5	1.12	0	0	0	0	0	0	0	1
11	1.45	1.19	1.29	1.15	1.3	1.5	1.12	0.01	0.25	0.23	0	0	0	0	1
12	1.45	1.19	1.29	1.15	1.3	1.22	1.12	0.01	0.25	0	0	0	0	0	1

Figure 4. Example of collected data for USC-USSE risk prediction model.

Collecting Data for Open Source Risk Prediction Model.

Step1: Find the required Jira reports and Github Commit History

1. Jira Repos

    I. Release and Milestones.

    Related factors in prediction model: ***Phase***

    II. Effort tracking report.

    Related factors in prediction model: ***Effort*** (actual effort), ***Estimated_Effort*** (estimated effort)

    III. Issue tracking reports: issue creation and resolution reports.

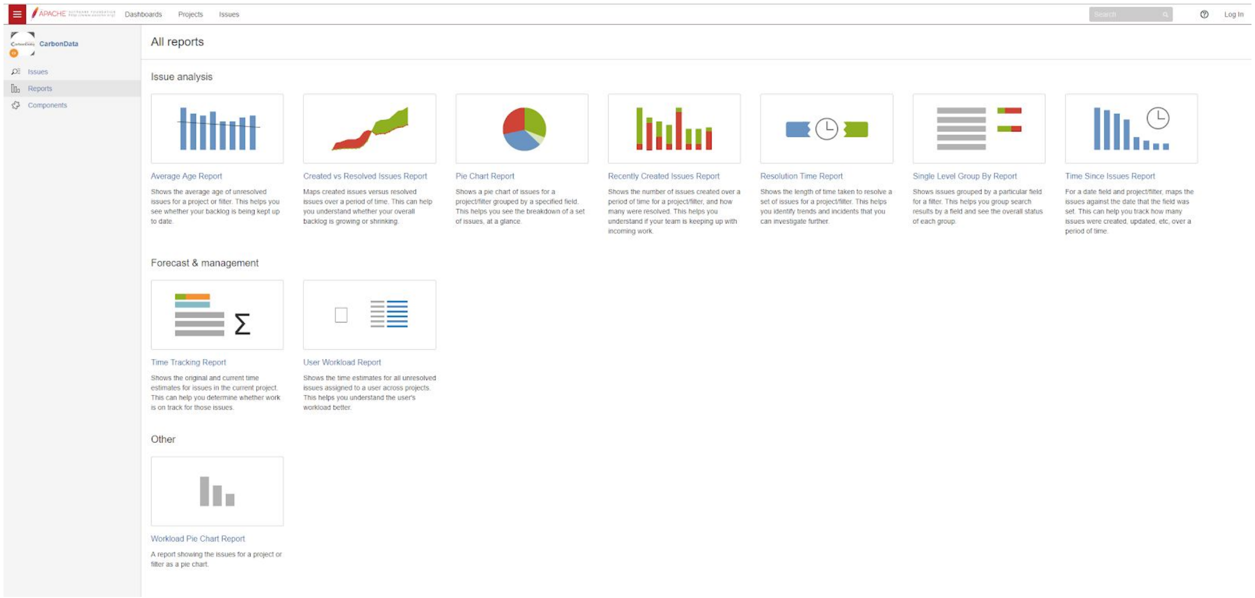
    Related factors in prediction model: ***ISS_Num_Resolved, ISS_Num_Unresolved, Accu_Trivial, Accu_Minor, Accu_Major, Accu_Critical, Accu_Block***

    IV. Personnel and Contributions.

    Related factors in prediction model: ***Personnel***

    V. System Modules.

*An example of Jira reports can be found at Url: *https://issues.apache.org/jira/projects/CARBONDATA?selectedItem=com.atlassian.jira.jira-projects-plugin:report-page*



All reports

Issue analysis

Average Age Report

Shows the average age of unresolved issues for a project or filter. This helps you see whether your backlog is being kept up to date.

Created vs Resolved Issues Report

Maps created issues versus resolved issues over a period of time. This can help you understand whether your overall backlog is growing or shrinking.

Pie Chart Report

Shows a pie chart of issues for a project/filter grouped by a specified field. This helps you see the breakdown of a set of issues, at a glance.

Recently Created Issues Report

Shows the number of issues created over a period of time for a project/filter, and how many were resolved. This helps you understand if your team is keeping up with incoming work.

Resolution Time Report

Shows the length of time taken to resolve a set of issues for a project/filter. This helps you identify trends and incidents that you can investigate further.

Single Level Group By Report

Shows issues grouped by a particular field for a filter. This helps you group search results by a field and see the overall status of each group.

Time Since Issues Report

For a date field and project/filter, maps the issues against the date that the field was set. This can help you track how many issues were created, updated, etc, over a period of time.

Forecast & management

Time Tracking Report

Shows the original and current time estimates for issues in the current project. This can help you determine whether work is on track for those issues.

User Workload Report

Shows the time estimates for all unresolved issues assigned to a user across projects. This helps you understand the user's workload better.

Other

Workload Pie Chart Report

A report showing the issues for a project or filter as a pie chart.

Figure 5. Jira reports panel to download related datasheets.

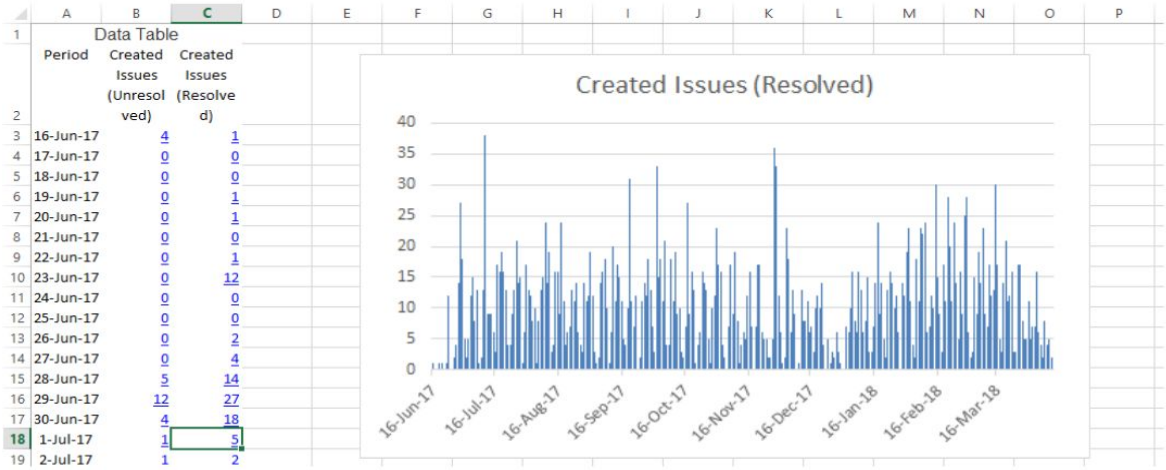
2.Github

I.Commits.

Related factors in prediction model: ***CSmell, SVul*** *(CSmell and SVul can be drived using USC-CSSE's SQUAAD tool or SonarQube)***.**

Step 2: Collect Phase and Issues Data

1. The distribution of creation time indicates the phases - ***Phase***. (If actual definition of phases is available, it would be better)
2. Issue records provides creation times and resolution times of the issues. ***ISS_Num_Resolved, ISS_Num_Unresolved*** are calculated by ***Phase*** using the issue report

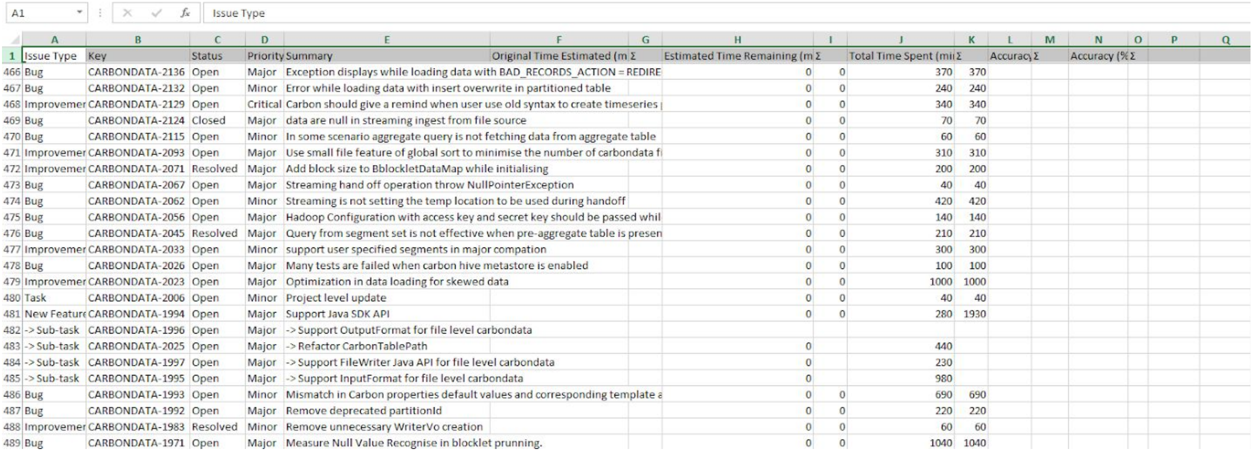


	A	B	C
1		Data Table	
2	Period	Created Issues (Unresolved)	Created Issues (Resolved)
3	16-Jun-17	4	1
4	17-Jun-17	0	0
5	18-Jun-17	0	0
6	19-Jun-17	0	1
7	20-Jun-17	0	1
8	21-Jun-17	0	0
9	22-Jun-17	0	1
10	23-Jun-17	0	12
11	24-Jun-17	0	0
12	25-Jun-17	0	0
13	26-Jun-17	0	2
14	27-Jun-17	0	4
15	28-Jun-17	5	14
16	29-Jun-17	12	27
17	30-Jun-17	4	18
18	1-Jul-17	1	5
19	2-Jul-17	1	2

Figure 6. Example of issue related datasheet.

Step 3: Collect time tracking data

Time tracking records provide estimated times and actual times spent on certain tasks.***Effort*** (actual effort), ***Estimated_Effort*** (estimated effort) for each ***Phase*** is calculated by the effort spent on the issues that belong to the ***Phase***.



A1 | Issue Type

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Issue Type	Key	Status	Priority	Summary	Original Time Estimated (m	Σ	Estimated Time Remaining (m	Σ	Total Time Spent (mi	Σ	Accuracy	Σ	Accuracy (%	Σ		
466	Bug	CARBONDATA-2136	Open	Major	Exception displays while loading data with BAD_RECORDS_ACTION = REDIRE			0	0	370	370						
467	Bug	CARBONDATA-2132	Open	Minor	Error while loading data with insert overwrite in partitioned table			0	0	240	240						
468	Improvemer	CARBONDATA-2129	Open	Critical	Carbon should give a remind when user use old syntax to create timeseries			0	0	340	340						
469	Bug	CARBONDATA-2124	Closed	Major	data are null in streaming ingest from file source			0	0	70	70						
470	Bug	CARBONDATA-2115	Open	Minor	In some scenario aggregate query is not fetching data from aggregate table			0	0	60	60						
471	Improvemer	CARBONDATA-2093	Open	Major	Use small file feature of global sort to minimise the number of carbondata fi			0	0	310	310						
472	Improvemer	CARBONDATA-2071	Resolved	Major	Add block size to BblockletDataMap while initialising			0	0	200	200						
473	Bug	CARBONDATA-2067	Open	Major	Streaming hand off operation throw NullPointerException			0	0	40	40						
474	Bug	CARBONDATA-2062	Open	Minor	Streaming is not setting the temp location to be used during handoff			0	0	420	420						
475	Bug	CARBONDATA-2056	Open	Major	Hadoop Configuration with access key and secret key should be passed whil			0	0	140	140						
476	Bug	CARBONDATA-2045	Resolved	Major	Query from segment set is not effective when pre-aggregate table is presen			0	0	210	210						
477	Improvemer	CARBONDATA-2033	Open	Minor	support user specified segments in major compation			0	0	300	300						
478	Bug	CARBONDATA-2026	Open	Major	Many tests are failed when carbon hive metastore is enabled			0	0	100	100						
479	Improvemer	CARBONDATA-2023	Open	Major	Optimization in data loading for skewed data			0	0	1000	1000						
480	Task	CARBONDATA-2006	Open	Minor	Project level update			0	0	40	40						
481	New Feature	CARBONDATA-1994	Open	Major	Support Java SDK API			0	0	280	1930						
482	-> Sub-task	CARBONDATA-1996	Open	Major	-> Support OutputFormat for file level carbondata												
483	-> Sub-task	CARBONDATA-2025	Open	Major	-> Refactor CarbonTablePath			0		440							
484	-> Sub-task	CARBONDATA-1997	Open	Major	-> Support FileWriter Java API for file level carbondata			0		230							
485	-> Sub-task	CARBONDATA-1995	Open	Major	-> Support InputFormat for file level carbondata			0		980							
486	Bug	CARBONDATA-1993	Open	Minor	Mismatch in Carbon properties default values and corresponding template a			0	0	690	690						
487	Bug	CARBONDATA-1992	Open	Major	Remove deprecated partitionId			0	0	220	220						
488	Improvemer	CARBONDATA-1983	Resolved	Minor	Remove unnecessary WriterVo creation			0	0	60	60						
489	Bug	CARBONDATA-1971	Open	Major	Measure Null Value Recognise in blocklet prunning.			0	0	1040	1040						

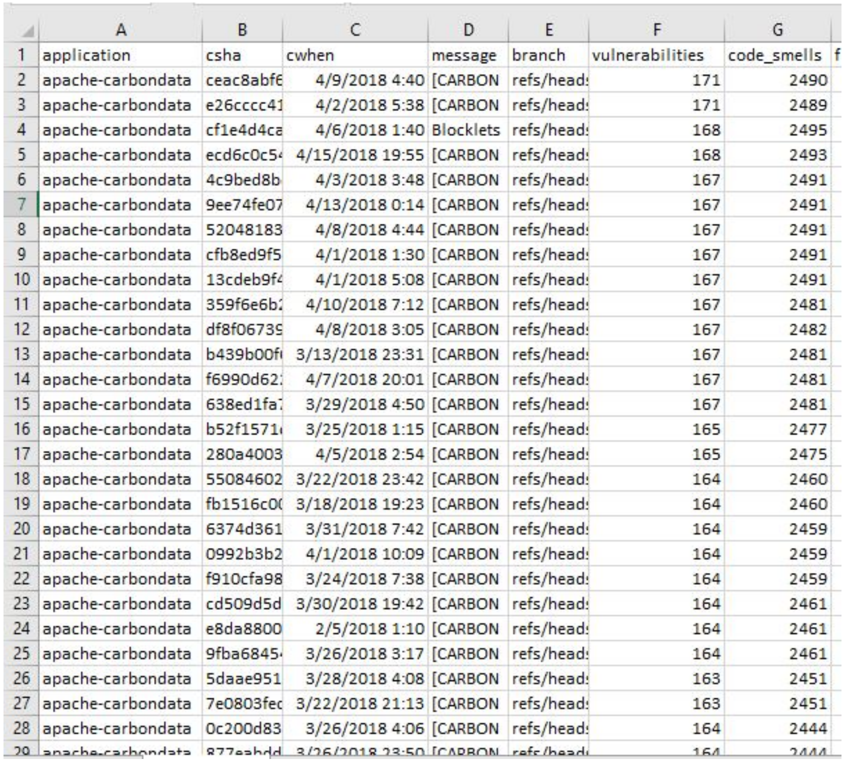
Figure 7. Example of time tracking datasheet.

Step 4: Collect code analysis data

Code metrics applied to each commit to measure code quality and technical debt.

- Number of Code smells (***CSmell***) is calculated by commits that belong to ***Phase***
- Number of Vulnerabilities (***SVul***) is calculated
- by commits that belong to ***Phase***

* This analytical data is generated by SQUAAD (or SonarQube)



	A	B	C	D	E	F	G	
1	application	csha	cwhen	message	branch	vulnerabilities	code_smells	f
2	apache-carbondata	ceac8abf6	4/9/2018 4:40	[CARBON	refs/head	171	2490	
3	apache-carbondata	e26cccc41	4/2/2018 5:38	[CARBON	refs/head	171	2489	
4	apache-carbondata	cf1e4d4ca	4/6/2018 1:40	Blocklets	refs/head	168	2495	
5	apache-carbondata	ecd6c0c54	4/15/2018 19:55	[CARBON	refs/head	168	2493	
6	apache-carbondata	4c9bed8b	4/3/2018 3:48	[CARBON	refs/head	167	2491	
7	apache-carbondata	9ee74fe07	4/13/2018 0:14	[CARBON	refs/head	167	2491	
8	apache-carbondata	52048183	4/8/2018 4:44	[CARBON	refs/head	167	2491	
9	apache-carbondata	cfb8ed9f5	4/1/2018 1:30	[CARBON	refs/head	167	2491	
10	apache-carbondata	13cdeb9f4	4/1/2018 5:08	[CARBON	refs/head	167	2491	
11	apache-carbondata	359f6e6b2	4/10/2018 7:12	[CARBON	refs/head	167	2481	
12	apache-carbondata	df8f06739	4/8/2018 3:05	[CARBON	refs/head	167	2482	
13	apache-carbondata	b439b00f	3/13/2018 23:31	[CARBON	refs/head	167	2481	
14	apache-carbondata	f6990d62	4/7/2018 20:01	[CARBON	refs/head	167	2481	
15	apache-carbondata	638ed1fa7	3/29/2018 4:50	[CARBON	refs/head	167	2481	
16	apache-carbondata	b52f1571	3/25/2018 1:15	[CARBON	refs/head	165	2477	
17	apache-carbondata	280a4003	4/5/2018 2:54	[CARBON	refs/head	165	2475	
18	apache-carbondata	55084602	3/22/2018 23:42	[CARBON	refs/head	164	2460	
19	apache-carbondata	fb1516c0	3/18/2018 19:23	[CARBON	refs/head	164	2460	
20	apache-carbondata	6374d361	3/31/2018 7:42	[CARBON	refs/head	164	2459	
21	apache-carbondata	0992b3b2	4/1/2018 10:09	[CARBON	refs/head	164	2459	
22	apache-carbondata	f910cfa98	3/24/2018 7:38	[CARBON	refs/head	164	2459	
23	apache-carbondata	cd509d5d	3/30/2018 19:42	[CARBON	refs/head	164	2461	
24	apache-carbondata	e8da8800	2/5/2018 1:10	[CARBON	refs/head	164	2461	
25	apache-carbondata	9fba6845	3/26/2018 3:17	[CARBON	refs/head	164	2461	
26	apache-carbondata	5daae951	3/28/2018 4:08	[CARBON	refs/head	163	2451	
27	apache-carbondata	7e0803fe	3/22/2018 21:13	[CARBON	refs/head	163	2451	
28	apache-carbondata	0c200d83	3/26/2018 4:06	[CARBON	refs/head	164	2444	
29	apache-carbondata	877eabdd	3/26/2018 23:50	[CARBON	refs/head	164	2444	

Figure 7. Example of commits datasheet.

Step 5: Collect accumulated bugs data

Accumulated Bugs of different severity reflect the potential technical debt. ***Accu_Trivial, Accu_Minor, Accu_Major, Accu_Critical, Accu_Block*** are determined by categorizing the bugs according to their severity.

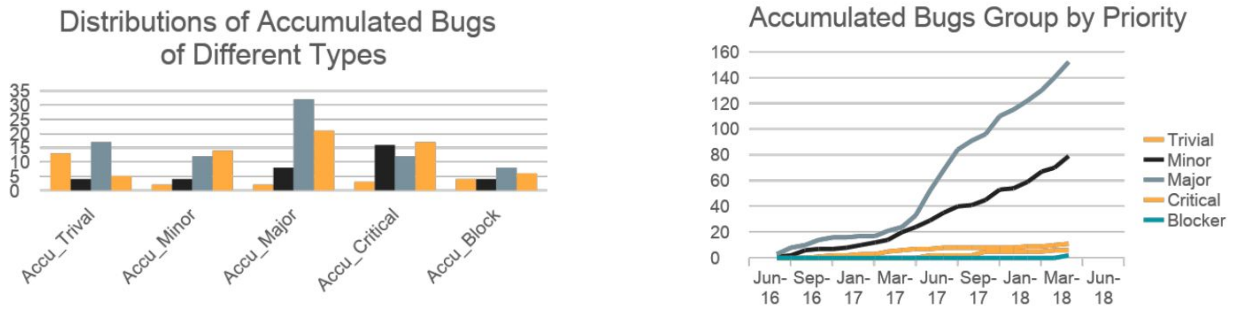
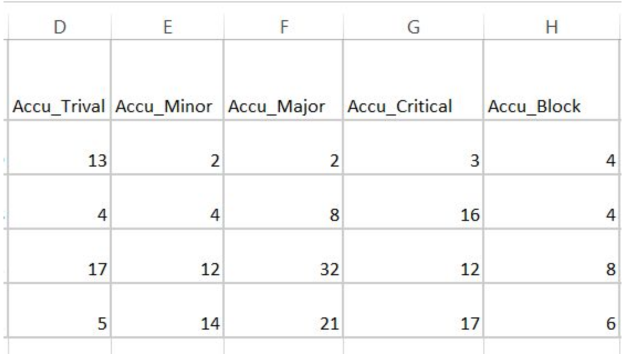


Figure 7. Example of distributions of categorized bugs.

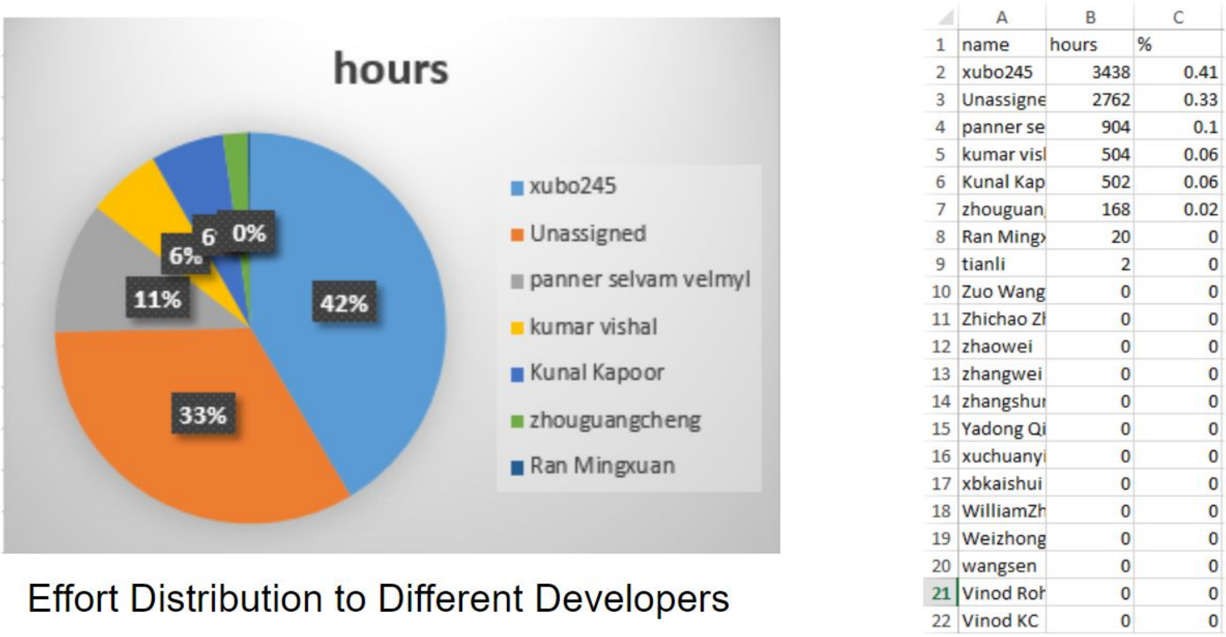


D	E	F	G	H
Accu_Trival	Accu_Minor	Accu_Major	Accu_Critical	Accu_Block
13	2	2	3	4
4	4	8	16	4
17	12	32	12	8
5	14	21	17	6

Figure 8. Example of categorized bugs based on severity.

Step 6: Collect Personnel Data

Effort distribution over team members identify the contributors, which helps measure the balance of workload. The distribution of the effort to personnel determines the core contributors, the number of which define the factor ***Personnel***. Contributors who contribute larger than 5% are determined as core contributors.



Effort Distribution to Different Developers

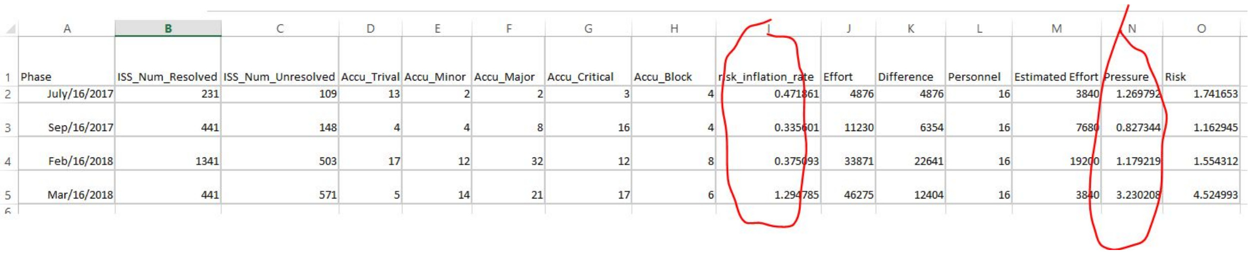
	A	B	C
1	name	hours	%
2	xubo245	3438	0.41
3	Unassigne	2762	0.33
4	panner se	904	0.1
5	kumar visl	504	0.06
6	Kunal Kap	502	0.06
7	zhouguan	168	0.02
8	Ran Mingx	20	0
9	tianli	2	0
10	Zuo Wang	0	0
11	Zhichao Zl	0	0
12	zhaowei	0	0
13	zhangwei	0	0
14	zhangshur	0	0
15	Yadong Qi	0	0
16	xuchuanyi	0	0
17	xbkaishui	0	0
18	WilliamZh	0	0
19	Weizhong	0	0
20	wangsen	0	0
21	Vinod Roh	0	0
22	Vinod KC	0	0

Figure 9. Example of personnel distribution chart and datasheet.

Step 7: Assess Level of Risk

The proposed method to measure risk.

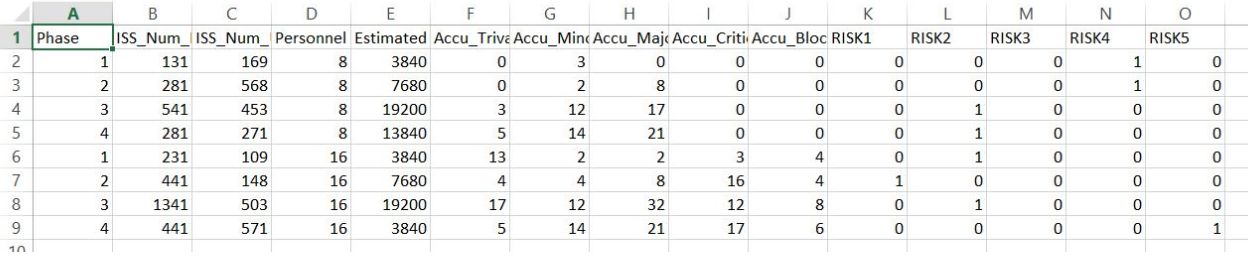
1. ***Risk_inflation_rate*** = ***ISS_Num_Unresolved*** (number of unsolved issues) / ***ISS_Num_Resolved*** (number of solved issues) (at the end of a milestone)
2. ***Pressure*** = ***Actual_Effort*** / ***Estimated_Effort*** (for each phase)
3. ***Risk*** = ***Risk_inflation_rate*** + ***Pressure***



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Phase	ISS_Num_Resolved	ISS_Num_Unresolved	Accu_Trival	Accu_Minor	Accu_Major	Accu_Critical	Accu_Block	risk_inflation_rate	Effort	Difference	Personnel	Estimated Effort	Pressure	Risk
2	July/16/2017	231	109	13	2	2	3	4	0.471861	4876	4876	16	3840	1.269792	1.741653
3	Sep/16/2017	441	148	4	4	8	16	4	0.335601	11230	6354	16	7680	0.827344	1.162945
4	Feb/16/2018	1341	503	17	12	32	12	8	0.375093	33871	22641	16	19200	1.179219	1.554312
5	Mar/16/2018	441	571	5	14	21	17	6	1.294785	46275	12404	16	3840	3.230208	4.524993

Figure 10. Example of datasheet for risk evaluation for open source risk prediction model.

Step 8: Prepare the data sheet (.csv) using the collected data



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Phase	ISS_Num_	ISS_Num_	Personnel	Estimated	Accu_Triva	Accu_Mino	Accu_Maj	Accu_Criti	Accu_Bloc	RISK1	RISK2	RISK3	RISK4	RISK5
2	1	131	169	8	3840	0	3	0	0	0	0	0	0	1	0
3	2	281	568	8	7680	0	2	8	0	0	0	0	0	1	0
4	3	541	453	8	19200	3	12	17	0	0	0	1	0	0	0
5	4	281	271	8	13840	5	14	21	0	0	0	1	0	0	0
6	1	231	109	16	3840	13	2	2	3	4	0	1	0	0	0
7	2	441	148	16	7680	4	4	8	16	4	1	0	0	0	0
8	3	1341	503	16	19200	17	12	32	12	8	0	1	0	0	0
9	4	441	571	16	3840	5	14	21	17	6	0	0	0	0	1

Figure 11. Example of input for open source risk prediction model.

*After collect the data for USC risk prediction model and Open source risk prediction model, you can use the procedures introduced in Section III to train your own risk prediction model.