1. Deployment Requirements

1.1 Hardware Requirements

- Intel Core i5 CPU 1.70GHz
- RAM: 4GB is recommended
- Any Video card

1.2 Software Requirements

- OS Required: Windows 7 or later
- Python 2.7.X (Limited to this version)
- MS Excel (optional- good to have)

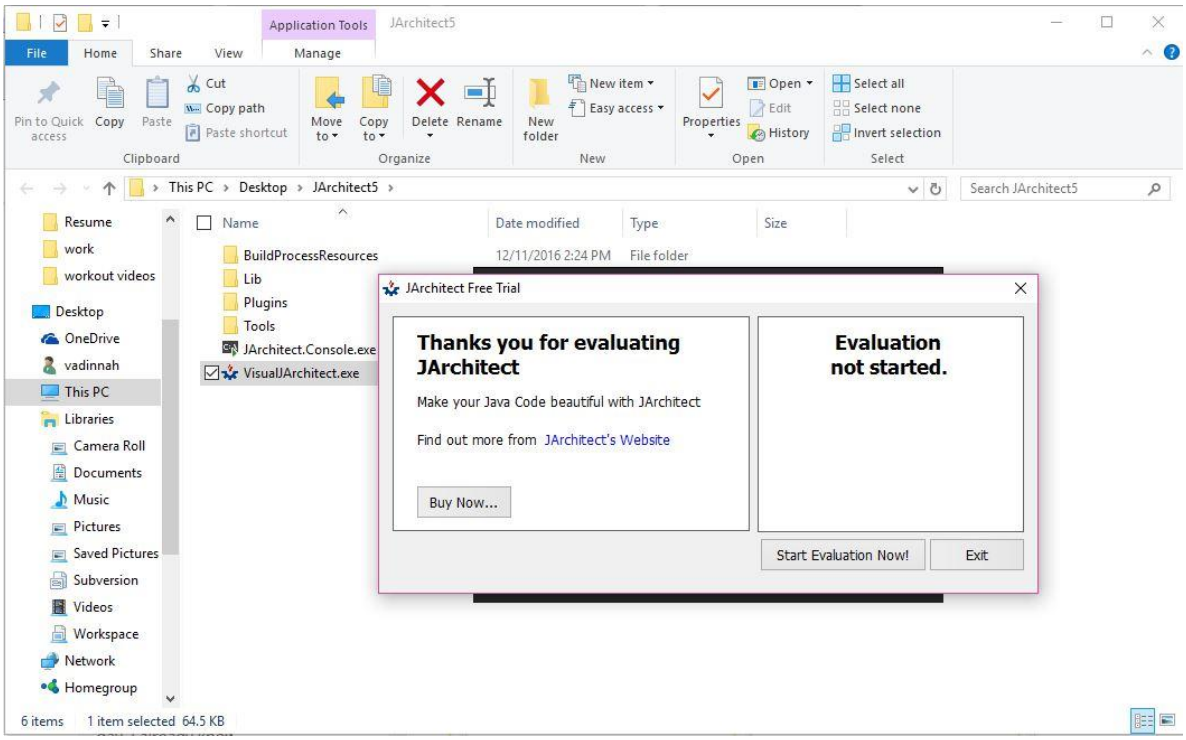
1.3 Setup and Deployment Instructions

1.3.0 Initial download :

1. Download "CloneDetective-master" named zipped file on the windows system, unzip and place it anywhere you would like to.(**placing it in C: drive is <u>strongly</u> recommended).**

1.3.1 JArchitect instructions: Has to be done when using for first time

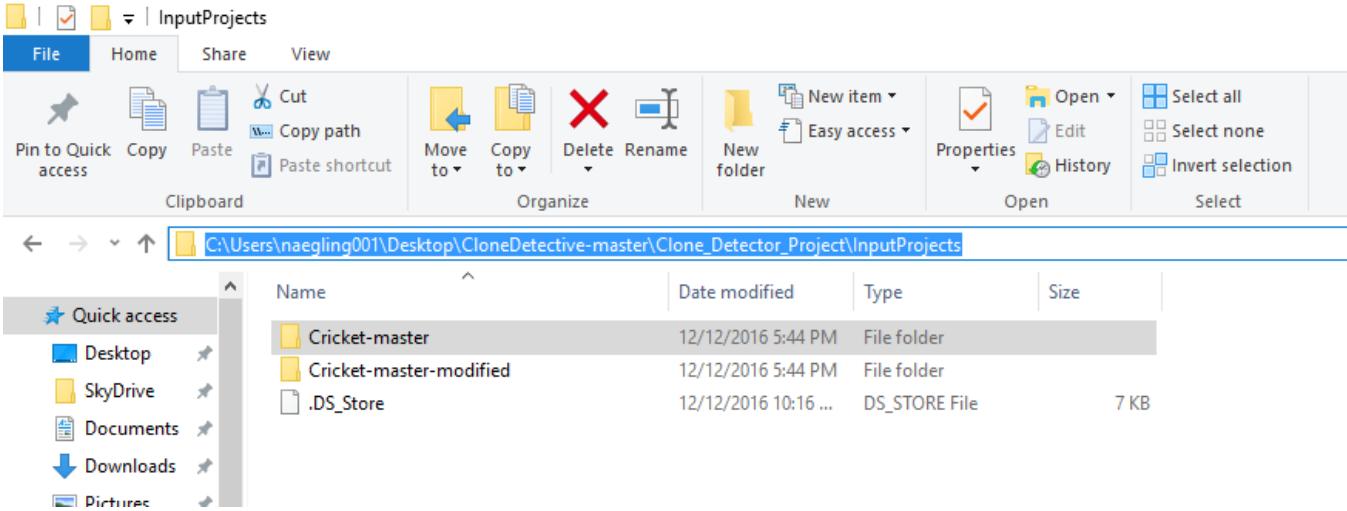
1. Make sure you have Internet Connection.
2. Go to JArchitect5 folder which is located at **...\CloneDetective-master\Clone_Detector_Project\JArchitect5.**
    a. Launch the **VisualJArchitect.exe**
3. You will get a prompt to start a trial (Example shown in picture below).
    a. Click Start Evaluation Now

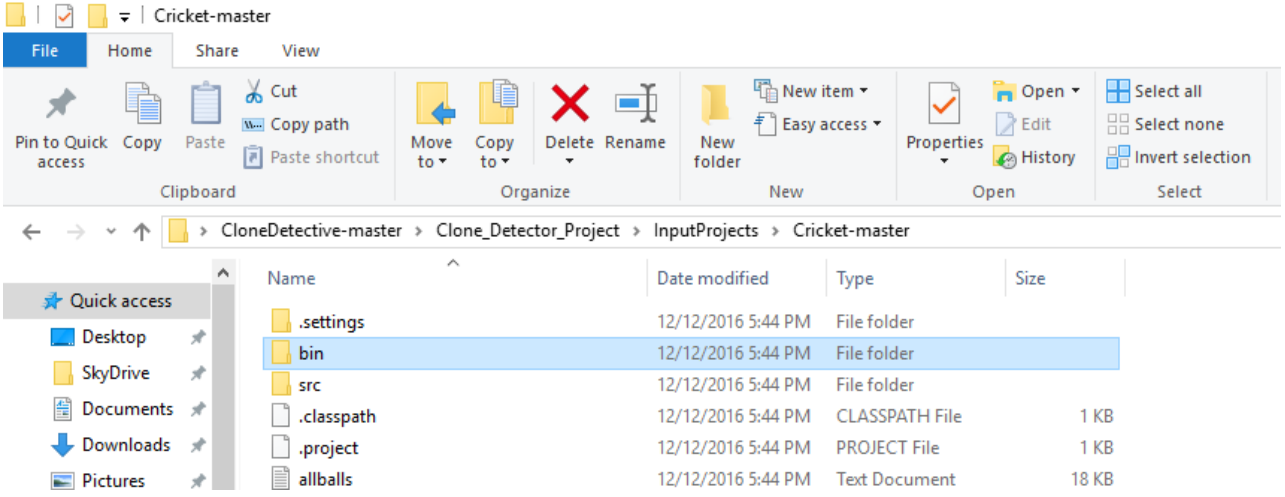


4. After the JArchitect app gets launched, close it.

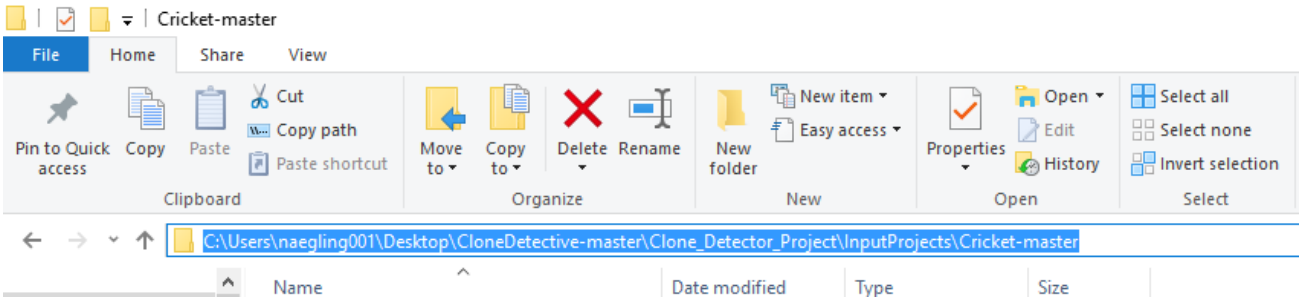
1.3.2 Main setup(Execution/Deployment) instructions:

1. Ensure that you have python 2.7 installed and specified in the system environment and path variables.
    a. You can find the download link here - https://www.python.org/downloads/
    b. You can set the path variables in this manner(please look at step 2) - http://pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-external-command/
    c. Ideally you will have python 2.7 .12 installed in C: drive like in this path **C:\Python27**
2. Remove all system environment and path variables for python 3.4.(As python 3.4 is not backward compatible it creates issues in running the tool)
    a. Go into properties→ Advanced system setting→ Environment Variables→ look for a variable names as “path”.
    b. click on “path” and then click on the edit button.
    c. look for any paths that point to C:\Python34. (any python path other than Python27) and delete them.
3. ***Automated Execution Using Batch File : (Goto Step 4 if this does not work)***
    a. Please move a built project which has a bin folder and that you wish to analyze into the ...\CloneDetective-master\Clone_Detector_Project\InputProjects folder.

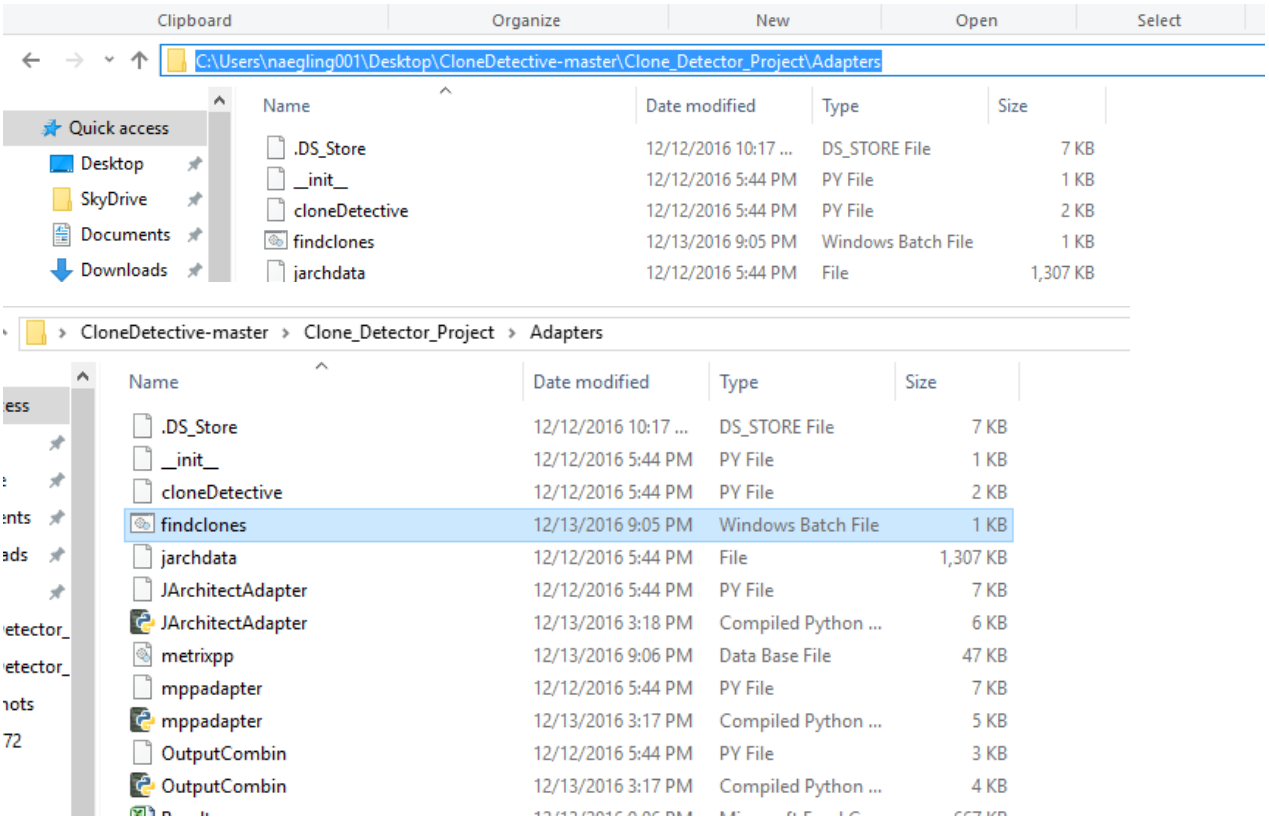




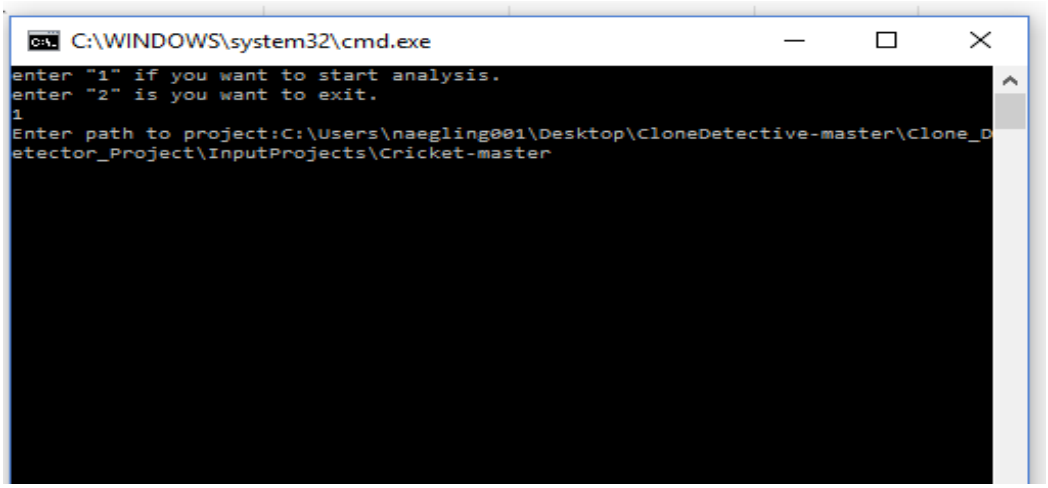
b. Now, copy the path to the project folder of the one that you moved into the ...\CloneDetective-master\Clone_Detector_Project\InputProjects folder.



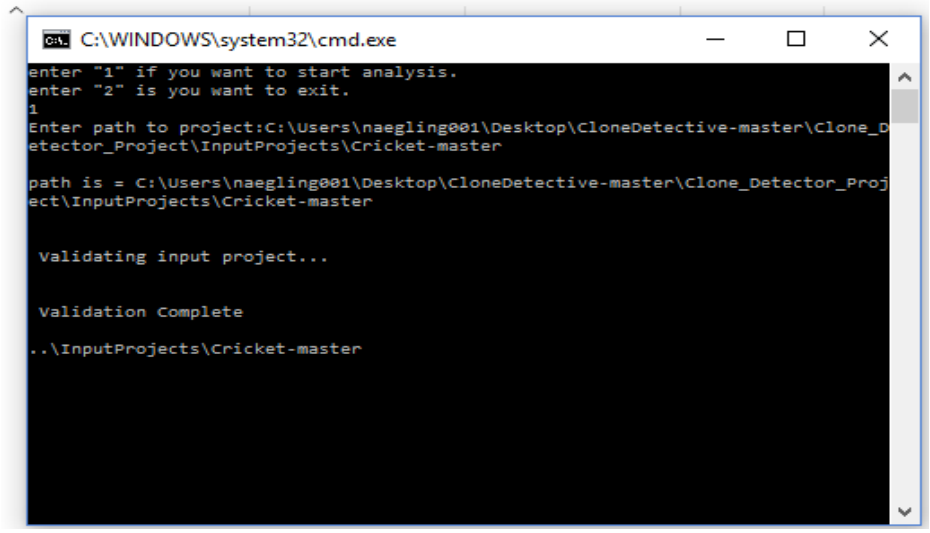
c. Please move into the ...\CloneDetective-master\Clone_Detector_Project\Adapters folder and open the findclones.bat.



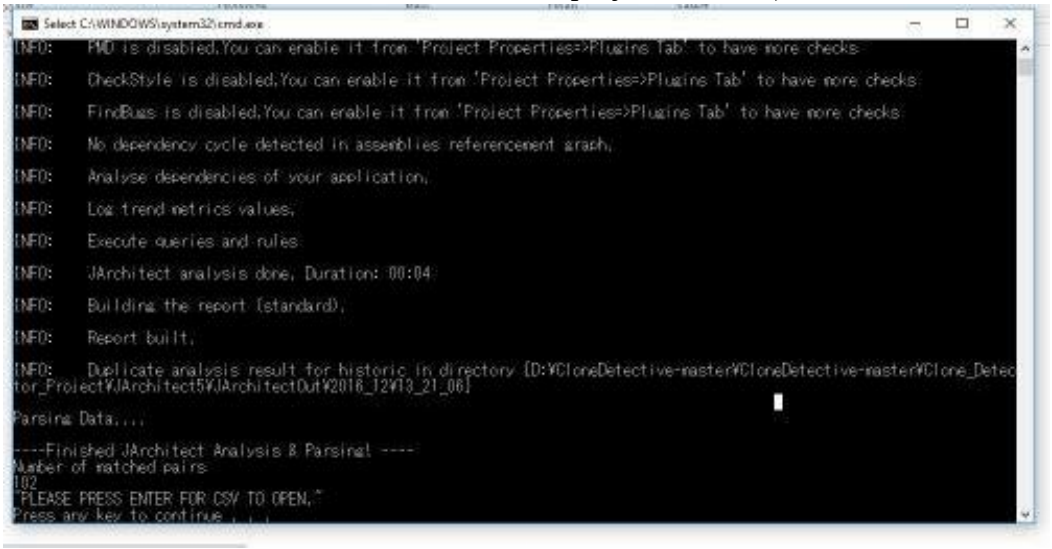
d. Press 1 when prompted for and then paste the path of the project that you have copied in previous step when you are prompted to do so.



If your input path does not have any white spaces, if you pointed it to an already built project folder which also contains the bin folder, you will see this :-



e. Sit back and wait for the execution to get completed. (this may take a few minutes based on the number of methods the project contains)

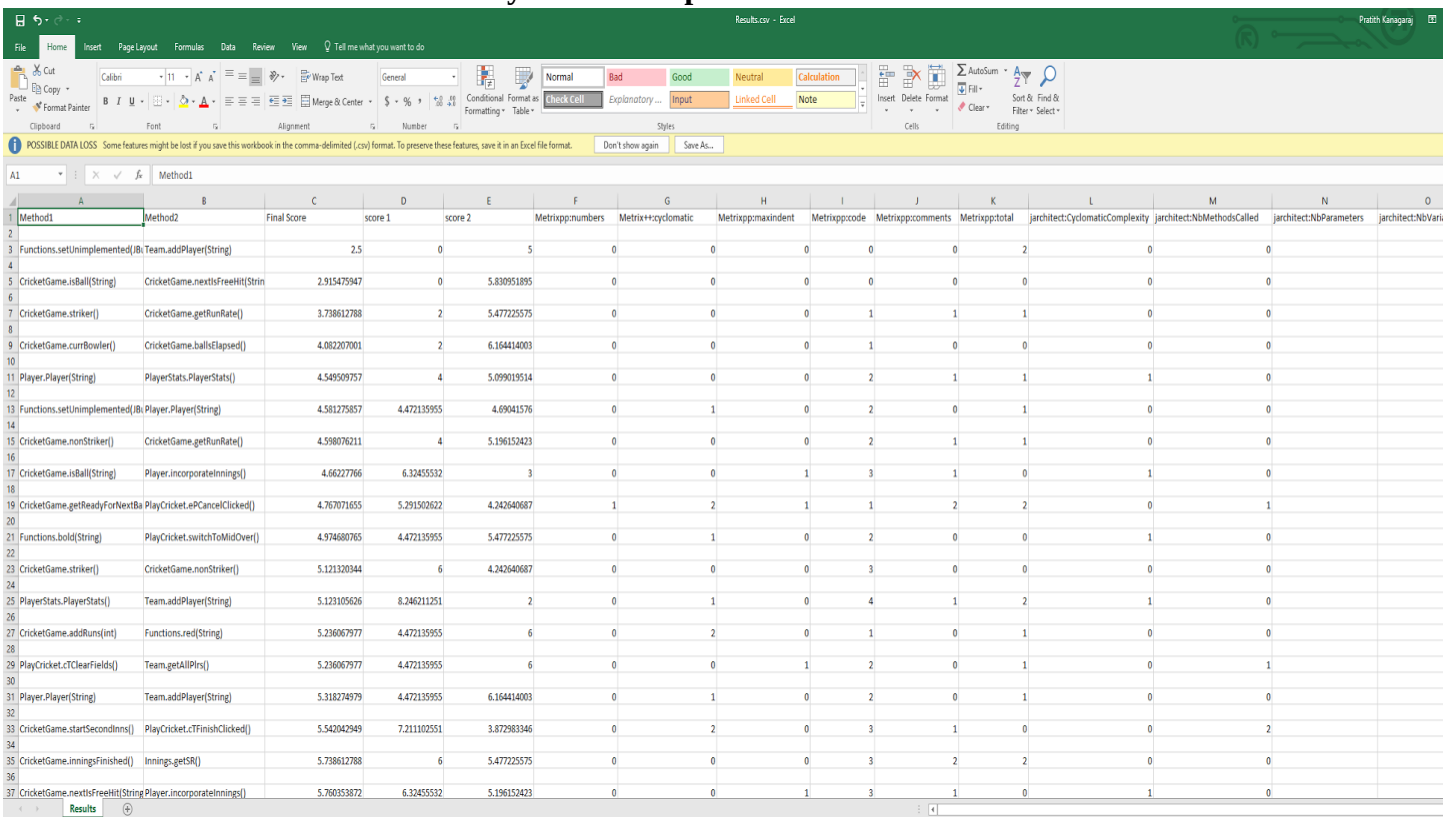


f. Press Enter to open the Results.csv for more details such as :

- Final Score = similarity score of a pair of methods. This is calculated as the average of score1 and score 2.
- Score1 = similarity score of a method pair calculated from metrics obtained from Metrics++ by using the euclidean distance formula.
- Score2 = similarity score of a method pair calculated from metrics obtained from JArchitect by using euclidean distance formula.
- Method1 and Method2 are obviously the methods paired for comparision.
- Remaining fields are the metrics we obtain from each tool i.e metrics++ and JArchitect to calculate the similarity. Values shown for each metrics are the differences between those collected for both methods
- **Note : The method pairs are ordered in an ascending order based on their final score, the minimum score is 0 and max score is 100. The**

lower the score the higher the probability that the methods are clones, to better phrase this we can say "lower the score, higher the similarity between methods of a pair".

- **In our output, pairs that were calculated using metrics from both adapters will be listed first. After that metrics collected by only a single adapter will be listing again in descending order. These are listed last as they will be less precise.**

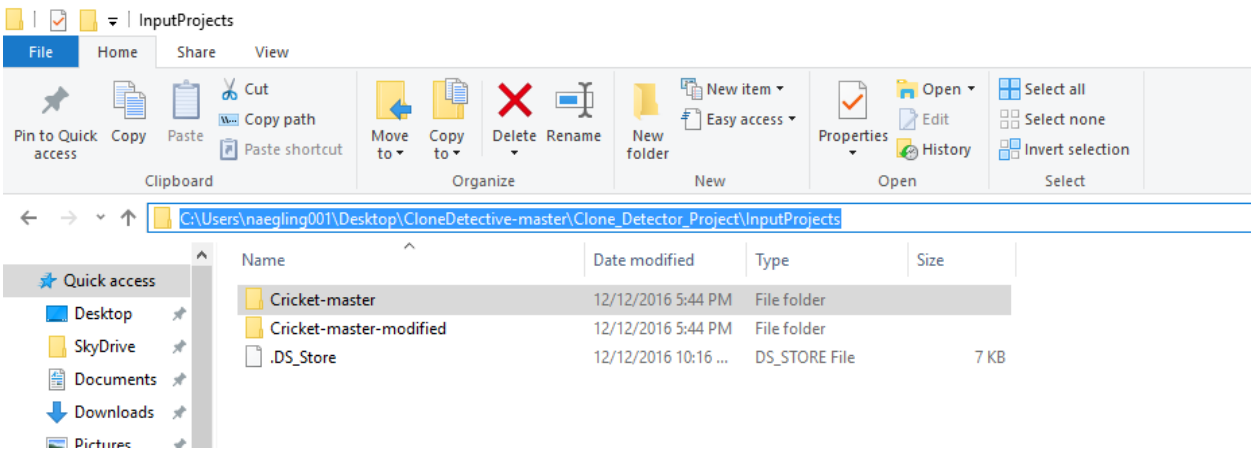


Method1	Method2	Final Score	score 1	score 2	Metrixpp:numbers	Metrix++:cyclomatic	Metrixpp:maxindent	Metrixpp:code	Metrixpp:comments	Metrixpp:total	jarchitect:CyclomaticComplexity	jarchitect:NbMethodsCalled	jarchitect:NbParameters	jarchitect:NbVari
Functions.setUnimplemented(JBu	Team.addPlayer(String)	2.5	0	5	0	0	0	0	0	2	0	0		
CricketGame.isBall(String)	CricketGame.nextIsFreeHit(Strin	2.915475947	0	5.830951895	0	0	0	0	0	0	0	0		
CricketGame.striker()	CricketGame.getRunRate()	3.738612788	2	5.477225575	0	0	0	1	1	1	0	0		
CricketGame.currBowler()	CricketGame.ballsElapsed()	4.082207001	2	6.164414003	0	0	0	1	0	0	0	0		
Player.Player(String)	PlayerStats.PlayerStats()	4.549509757	4	5.099019514	0	0	0	2	1	1	1	0		
Functions.setUnimplemented(JBu	Player.Player(String)	4.581275857	4.472135955	4.69041576	0	1	0	2	0	1	0	0		
CricketGame.nonStriker()	CricketGame.getRunRate()	4.598076211	4	5.196152423	0	0	0	2	1	1	0	0		
CricketGame.isBall(String)	Player.incorporateInnings()	4.66227766	6.32455532	3	0	0	1	3	1	0	1	0		
CricketGame.getReadyForNextBa	PlayCricket.ePCancelClicked()	4.767071655	5.291502622	4.242640687	1	2	1	1	2	2	0	1		
Functions.bold(String)	PlayCricket.switchToMidOver()	4.974680765	4.472135955	5.477225575	0	1	0	2	0	0	1	0		
CricketGame.striker()	CricketGame.nonStriker()	5.121320344	6	4.242640687	0	0	0	3	0	0	0	0		
PlayerStats.PlayerStats()	Team.addPlayer(String)	5.123105626	8.246211251	2	0	1	0	4	1	2	1	0		
CricketGame.addRuns(int)	Functions.red(String)	5.236067977	4.472135955	6	0	2	0	1	0	1	0	0		
PlayCricket.cTClearFields()	Team.getAllPlrs()	5.236067977	4.472135955	6	0	0	1	2	0	1	0	1		
Player.Player(String)	Team.addPlayer(String)	5.318274979	4.472135955	6.164414003	0	1	0	2	0	1	0	0		
CricketGame.startSecondInns()	PlayCricket.cTFinishClicked()	5.542042949	7.211102551	3.872983346	0	2	0	3	1	0	0	2		
CricketGame.inningsFinished()	Innings.getSR()	5.738612788	6	5.477225575	0	0	0	3	2	2	0	0		
CricketGame.nextIsFreeHit(String	Player.incorporateInnings()	5.760353872	6.32455532	5.196152423	0	0	1	3	1	0	1	0		

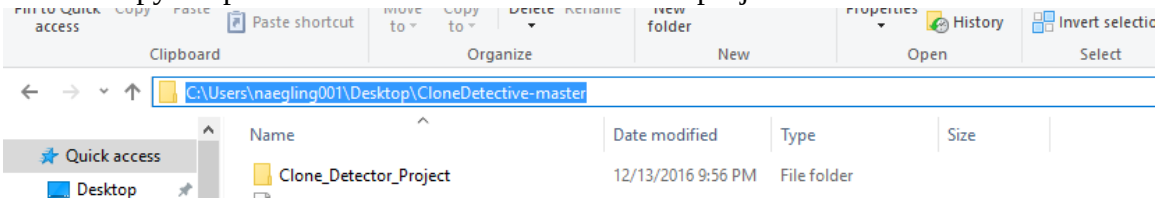
g. Use Save-as option if you want to keep your results as this csv will be overwritten next time you run the tool.

4. ***Command line execution :***

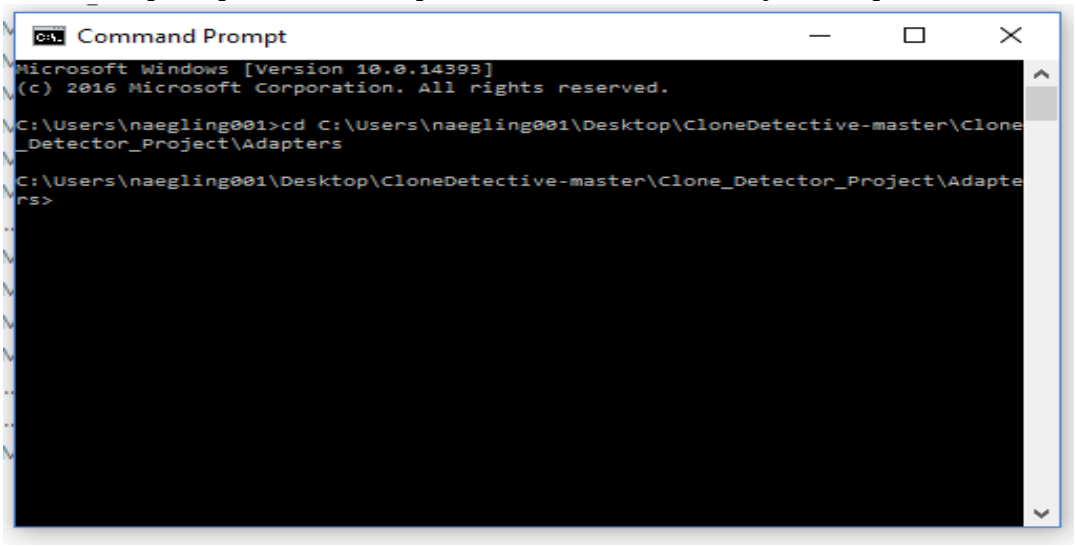
a. Now, please move a built project which has a bin folder and that you wish to analyze into the ...\CloneDetective-master\Clone_Detector_Project\InputProjects folder.



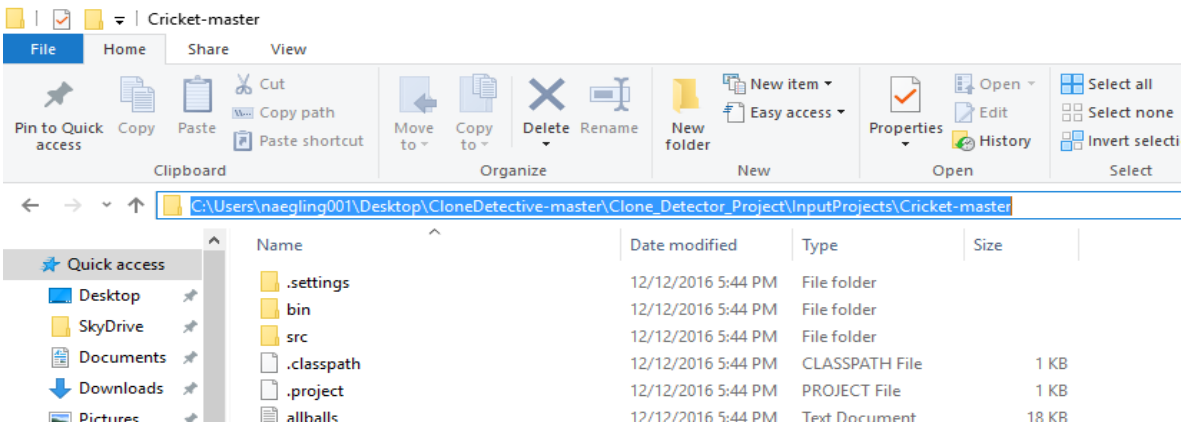
b. Copy the path of the "CloneDetective-master" project folder.



c. Open command prompt and change directory to <copied_path_without_space>\Clone_Detector_Project\Adapters.

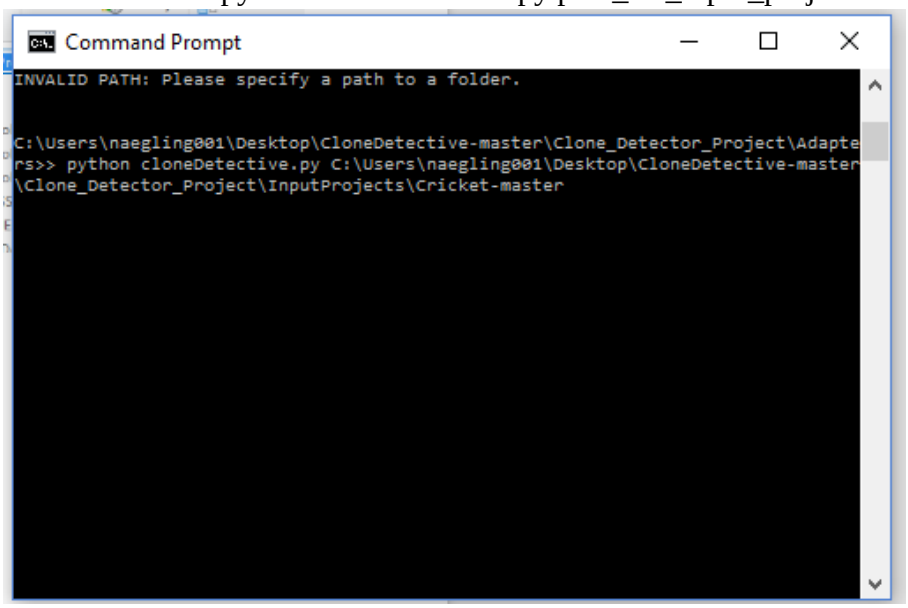


d. Now, copy the path to the project folder of the one that you moved into the ...\CloneDetective-master\Clone_Detector_Project\InputProjects folder.

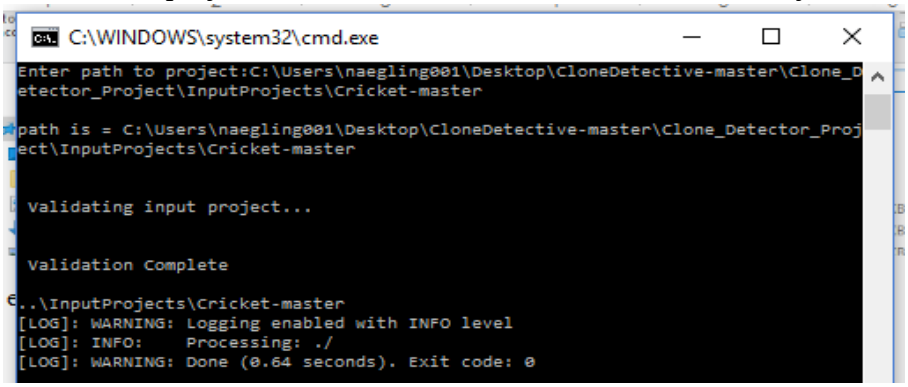


e. In Command Prompt type,

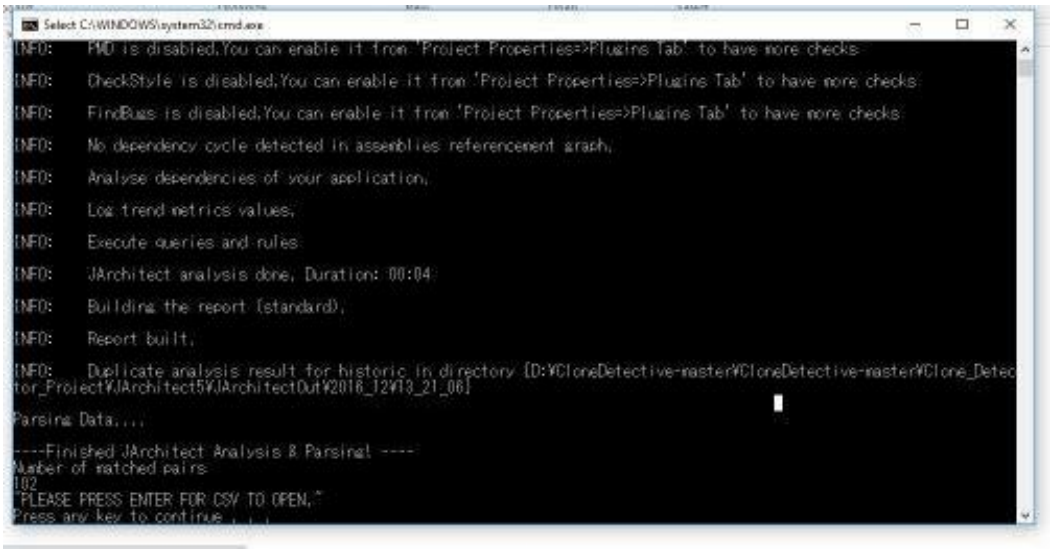
> python cloneDetective.py path_for_input_project



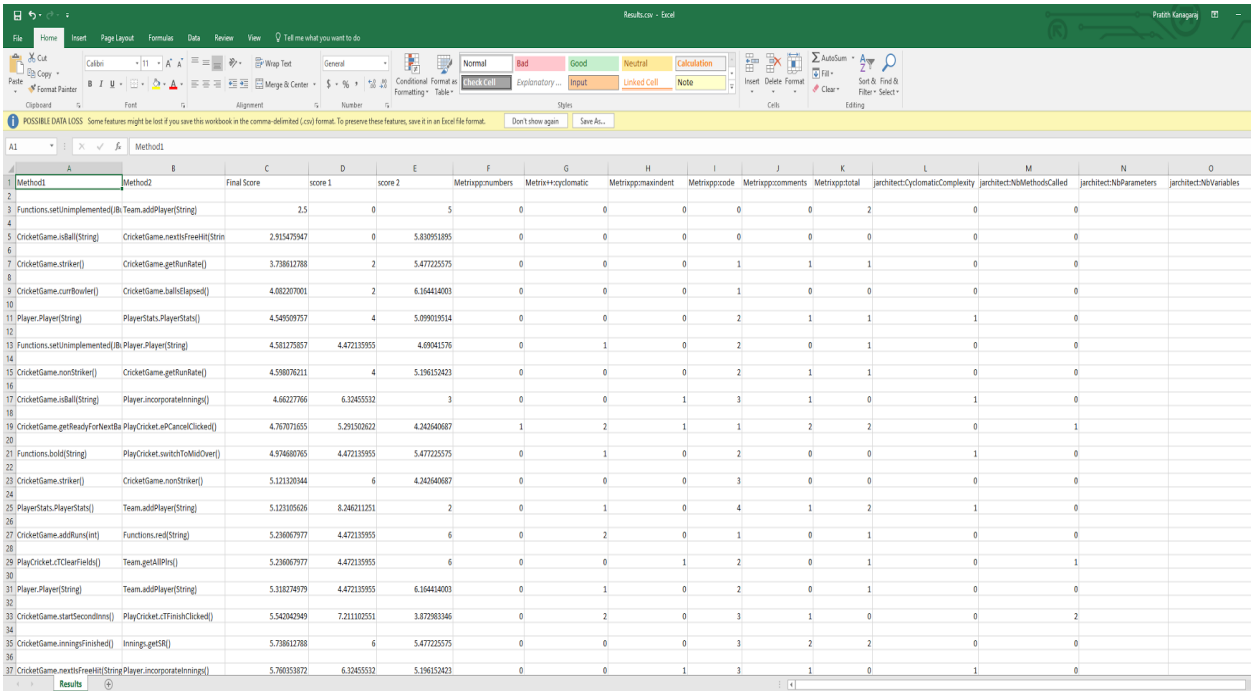
f. If your input path does not have any white spaces, if you pointed it to an already built project folder which also contains the bin folder, you will see this :



g. Sit back and wait for the execution to get completed. (this may take a few minutes based on the number of methods the project contains)



h. Look for Results.csv in …\CloneDetective-master\Clone_Detector_Projects\Adapters\ and then open the Results.csv for more details such as :
   - Final Score = similarity score of a pair of methods. This is calculated as the average of score1 and score 2.
   - Score1 = similarity score of a method pair calculated from metrics obtained from Metrics++ by using the euclidean distance formula.
   - Score2 = similarity score of a method pair calculated from metrics obtained from JArchitect by using euclidean distance formula.
   - Method1 and Method2 are obviously the methods paired for comparision.
   - Remaining fields are the metrics we obtain from each tool i.e metrics++ and JArchitect to calculate the similarity. Values shown for each metrics are the differences between those collected for both methods
   - **Note : The method pairs are ordered in an ascending order based on their final score, the minimum score is 0 and max score is 100. The lower the score the higher the probability that the methods are clones, to better phrase this we can say "lower the score, higher the similarity between methods of a pair".**
   - **In our output, pairs that were calculated using metrics from both adapters will be listed first. After that metrics collected by only a single adapter will be listing again in descending order. These are listed last as they will be less precise.**



Method1	Method2	Final Score	score 1	score 2	Metrixpp:numbers	Metrix++cyclomatic	Metrixpp:maxindent	Metrixpp:code	Metrixpp:comments	Metrixpp:total	jarchitect:CyclomaticComplexity	jarchitect:NbMethodsCalled	jarchitect:NbParameters	jarchitect:NbVariables
Functions.setUnimplemented(JB	Team.addPlayer(String)	2.5	0	5	0	0	0	0	0	2	0	0		
CricketGame.isBall(String)	CricketGame.nextIsFreeHit(Strin	2.915475947	0	5.830951895	0	0	0	0	0	0	0	0		
CricketGame.striker()	CricketGame.getRunRate()	3.738612788	2	5.477225575	0	0	0	1	1	1	0	0		
CricketGame.currBowler()	CricketGame.ballsElapsed()	4.082207001	2	6.164414003	0	0	0	1	0	0	0	0		
Player.Player(String)	PlayerStats.PlayerStats()	4.549509757	4	5.099019514	0	0	0	2	1	1	1	0		
Functions.setUnimplemented(JB	Player.Player(String)	4.581275857	4.472135955	4.69041576	0	1	0	2	0	1	0	0		
CricketGame.nonStriker()	CricketGame.getRunRate()	4.598076211	4	5.196152423	0	0	0	2	1	1	0	0		
CricketGame.isBall(String)	Player.incorporateInnings()	4.66227766	6.32455532	3	0	0	1	3	1	0	1	0		
CricketGame.getReadyForNextBa	PlayCricket.ePCancelClicked()	4.767071655	5.291502622	4.242640687	1	2	1	1	2	2	0	1		
Functions.bold(String)	PlayCricket.switchToMidOver()	4.974680765	4.472135955	5.477225575	0	1	0	2	0	0	1	0		
CricketGame.striker()	CricketGame.nonStriker()	5.121320344	6	4.242640687	0	0	0	3	0	0	0	0		
PlayerStats.PlayerStats()	Team.addPlayer(String)	5.123105626	8.246211251	2	0	1	0	4	1	2	1	0		
CricketGame.addRuns(int)	Functions.red(String)	5.236067977	4.472135955	6	0	2	0	1	0	1	0	0		
PlayCricket.cTClearFields()	Team.getAllPlrs()	5.236067977	4.472135955	6	0	0	1	2	0	1	0	1		
Player.Player(String)	Team.addPlayer(String)	5.318274979	4.472135955	6.164414003	0	1	0	2	0	1	0	0		
CricketGame.startSecondInns()	PlayCricket.cTFinishClicked()	5.542042949	7.211102551	3.872983346	0	2	0	3	1	0	0	2		
CricketGame.inningsFinished()	Innings.getSR()	5.738612788	6	5.477225575	0	0	0	3	2	2	0	0		
CricketGame.nextIsFreeHit(String	Player.incorporateInnings()	5.760353872	6.32455532	5.196152423	0	0	1	3	1	0	1	0		

i. Use Save-as option if you want to keep your results as this csv will be overwritten next time you run the tool.