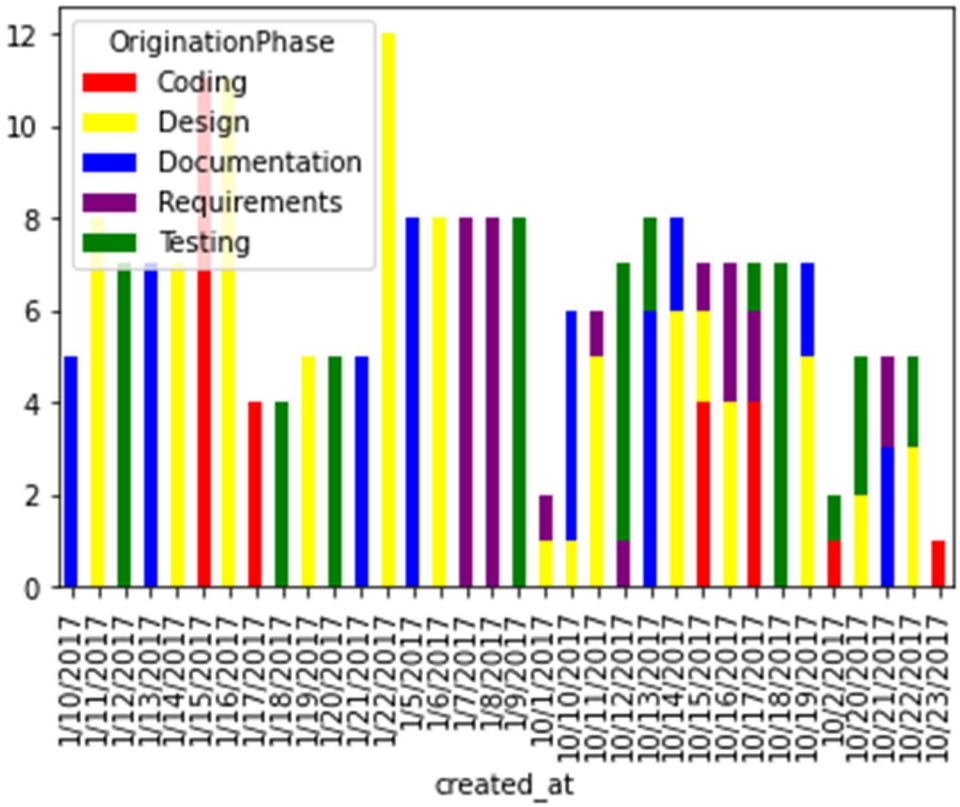
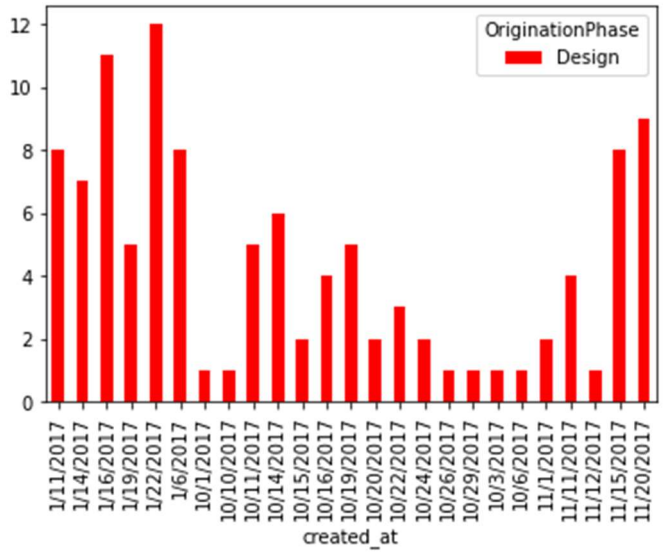
Part – 2 Data Analysis & statistical control process

Requirements: Issues Tracking and Report Design

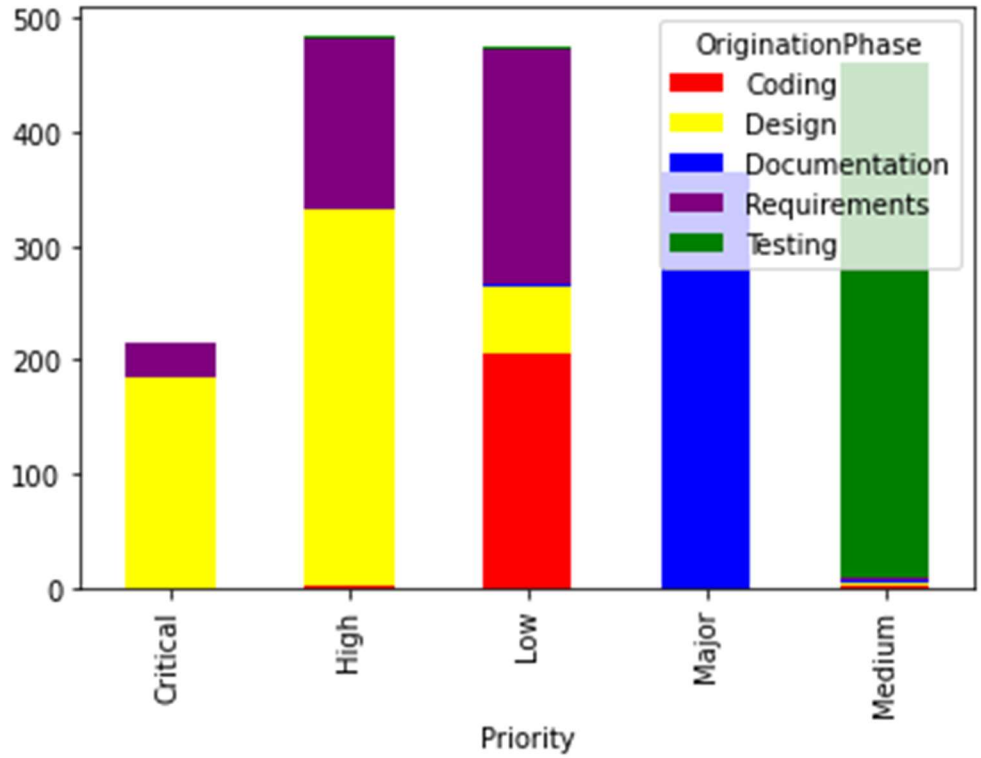
i. **Plot in Bar Chart the total number of issues created every day**



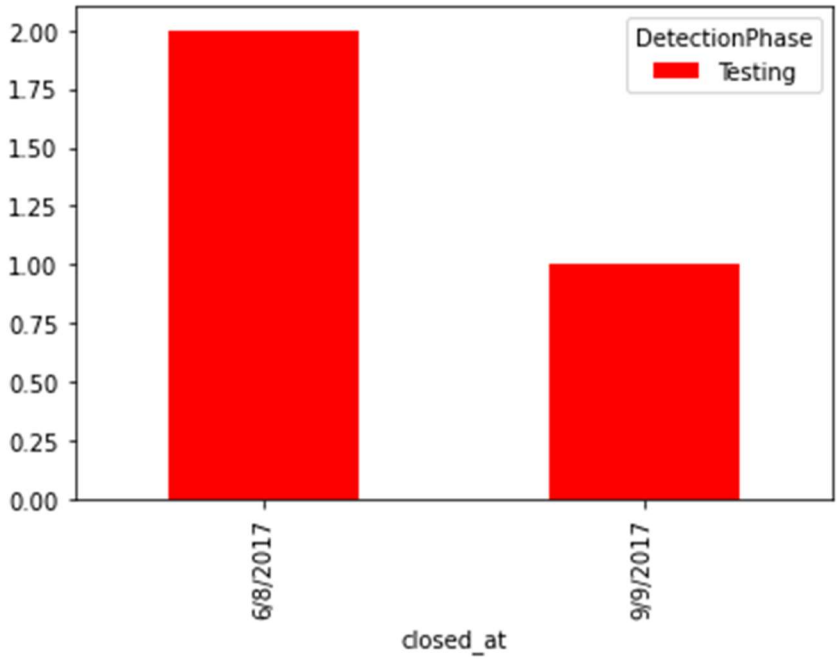
ii. **Plot in Bar Chart the total number of issues created every day and originated in Design phase**



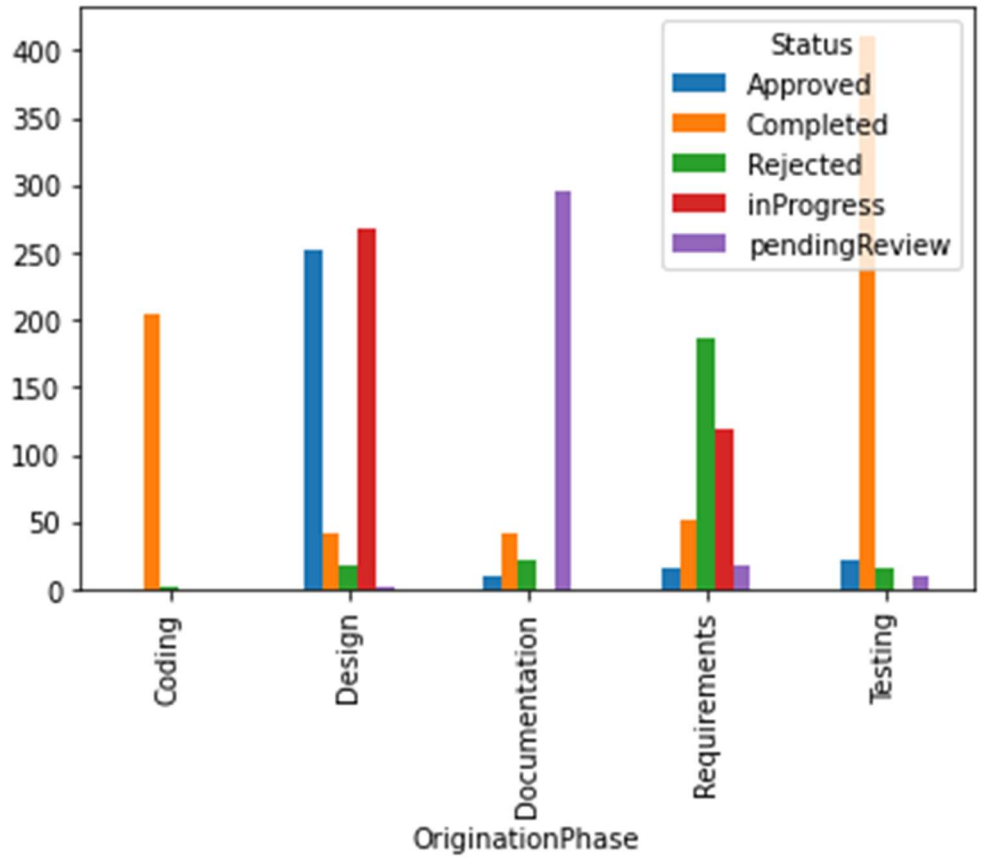
iii. **Plot in Stacked Bar Chart the total number of issues based on their priorities created for every originating phase**



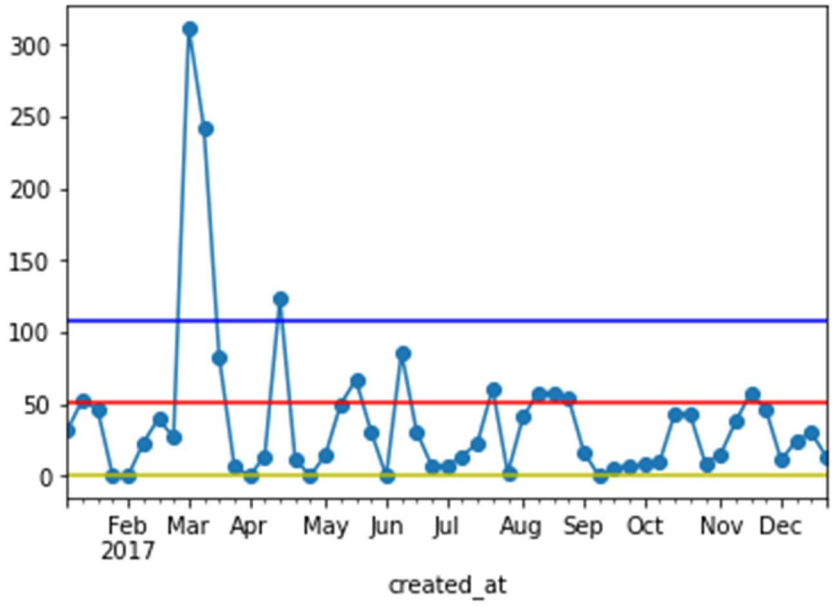
iv. **Plot in Bar Chart the total number of issues closed in every day for every DetectionPhase that have labels (Category:Bug,Priority:Critical, Status:Completed)**



v. **Plot in Pivot Chart the total number of issue status created for every OriginationPhase (group by originating phase)**

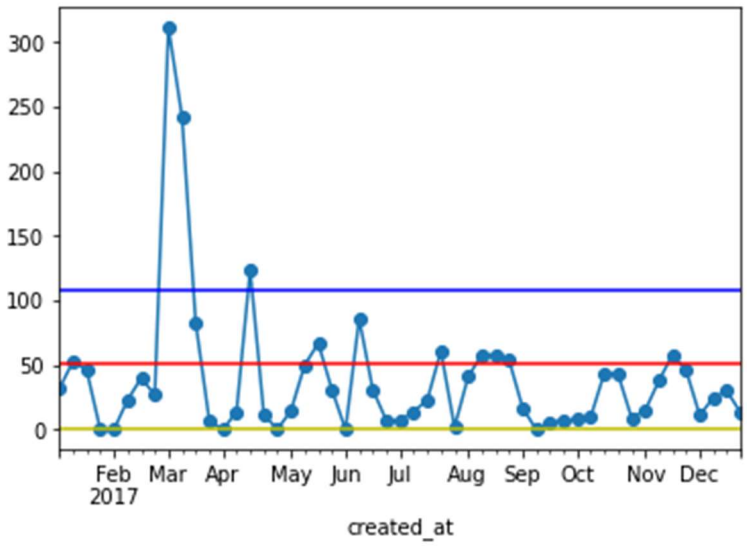


vi. **Plot in Control Chart for the total number of Critical issues created every week and originated in Design phase. Your Control chart must plot/show the UCL (Upper Control Limit) and LCL (Lower Control Limit)**

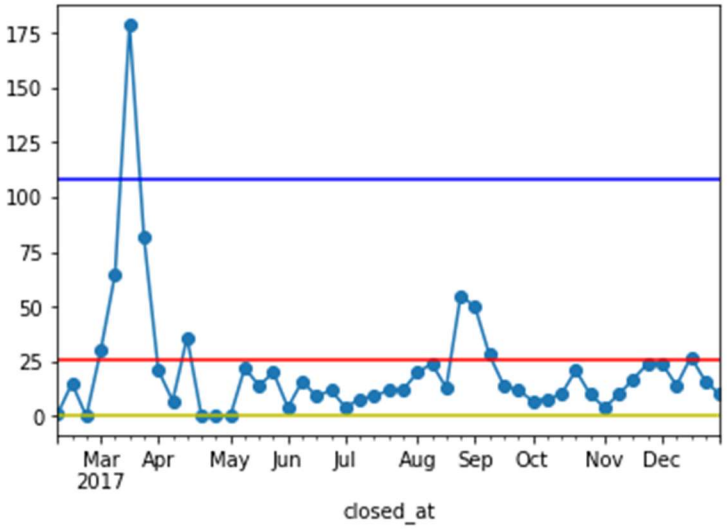


1) Create the analysis report to Support the Decision Making process, issues forecasting, and project planning.

Q1. Are there any useful patterns of outliers in the dataset?



It shows that issues were created maximum in march month and then in April month. In the later part of projects, issues were not created significantly which indicates a good overall software engineering process for this project.



It shows many issues were closed in in April which indicates the issues created in march were resolved in April in a significant number. These also suggest that many issues were raised during initial phases of SDLC which further shows the importance of Review process.

2. What are the UCL (Upper Control Limit) and LCL (Lower Control Limit) for certain issue metrics?

UCL for issues closed at is 107.65 and LCL for issues closed at is -65.87.

UCL for issues created at is 209.41 and LCL for issues created at is -124.31

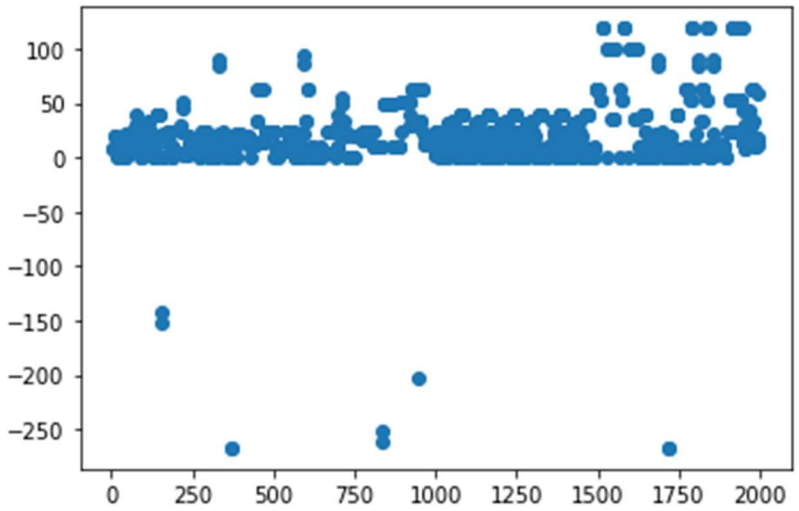
(Calculation reference from code file analysis.ipynb)

3. Can the outliers detect hidden problems in the given dataset?

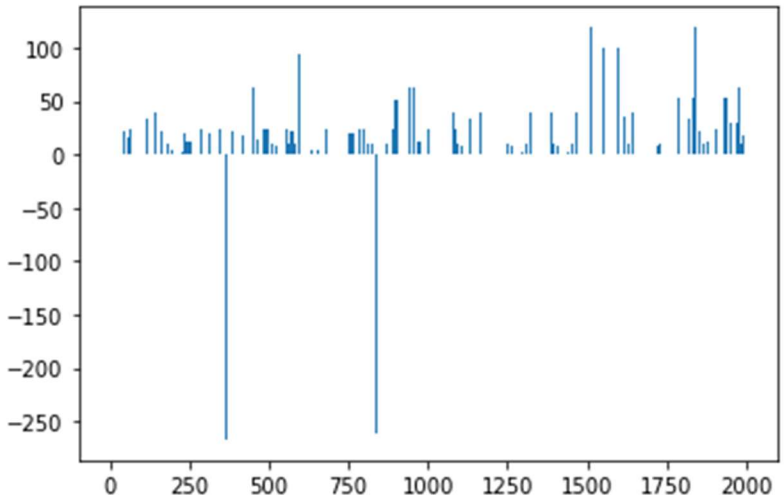
Yes, outliers can definitely detect hidden problems in the given dataset. For example, usually 10 number of days are taken to solve issues on average. Outliers for this analysis will indicate if any issue has taken significantly more or less amount of time.

For example, average days to close issues are 19 days. Outliers shall help to find certain issues which have taken considerably longer duration than 19 days.

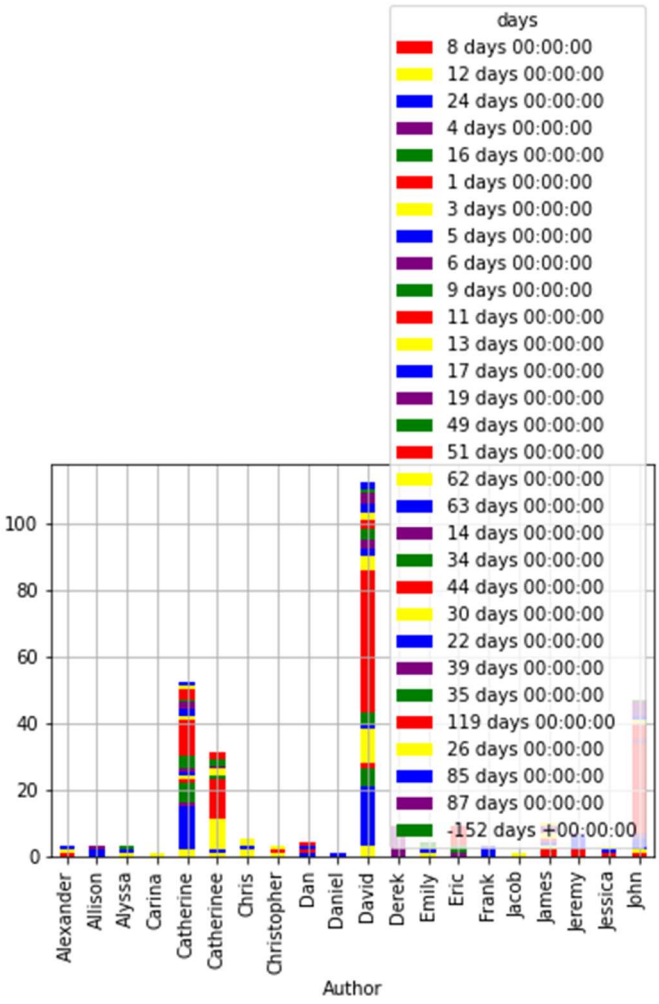
As below plot suggests that there were few issues which took extra time to be resolved and closed.



Below, bar chart shows the same thing but it also shows that which issues are still open.



4. What is the correlation between outliers and hidden problems in the given dataset?

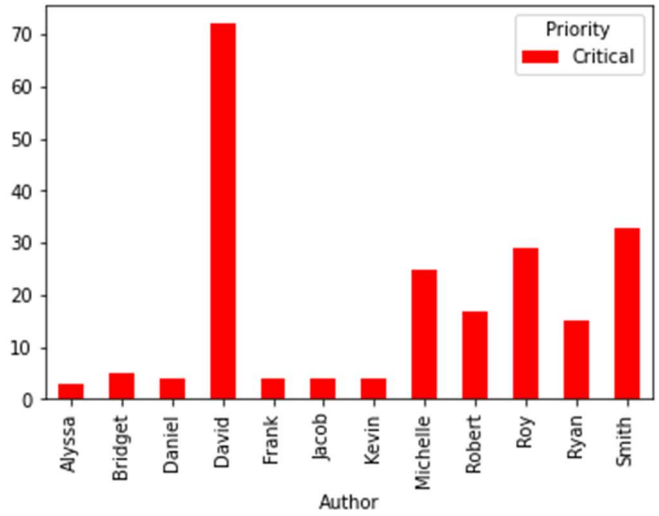


This bar chart helps to understand that how many total days are taken by which author to close the issues. It shows David has to work most to close the issues.

5. How to detect if there is problem hidden in the given dataset?

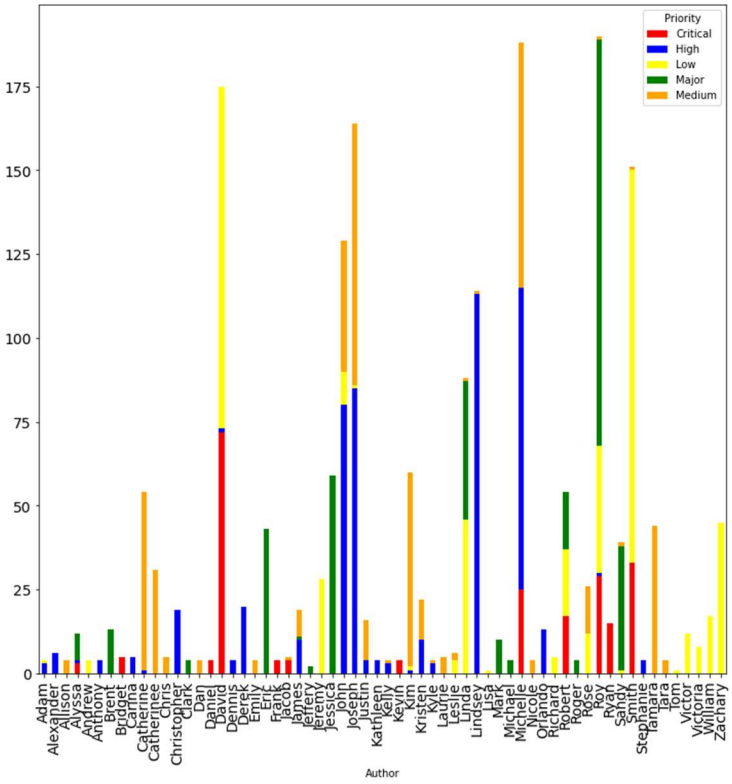
All the relative variables of the dataset shall be compared and analyzed with help of plot. If any plot shows some irregular pattern or outliers then those are the indication of hidden problems. Such hidden problems can be detected by analyzing data in depth, find correlation between variables etc.

6. How to detect if certain engineer deliberately creates issues with Priority Critical?

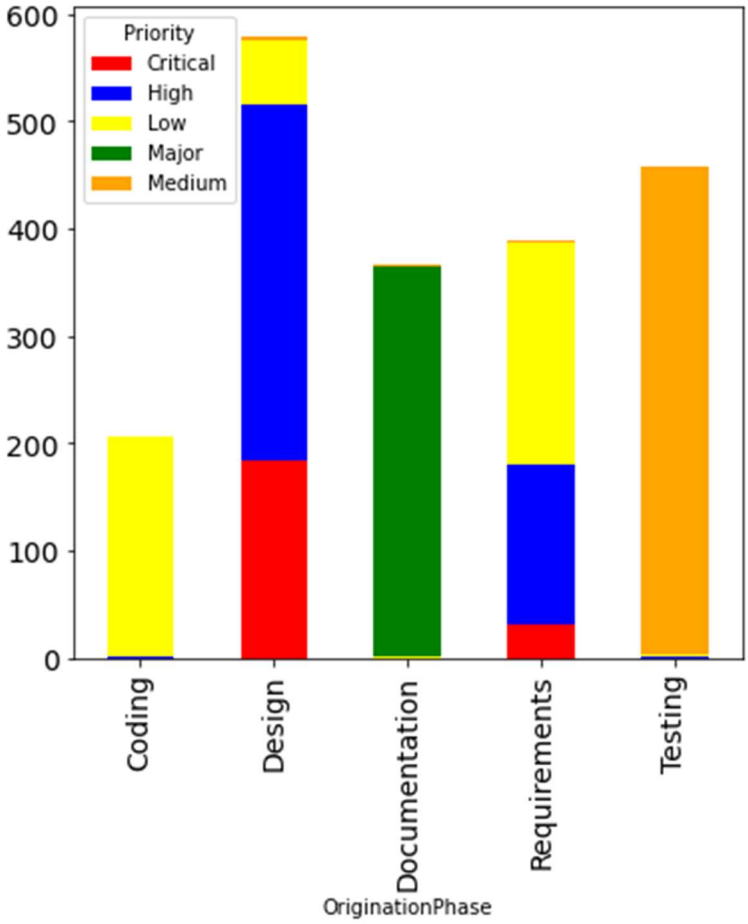


As this plot suggests, David has created maximum critical issues. But it does not show that if David had deliberately created issues with a critical priority. To know that, we shall compare all the issues' priority created by all authors. Below plot helps to understand us what priority issues are created by which author.

As per below plot, it is seen that David has also created similar number of issues with Low priority. Hence, David is not creating critical issues deliberately. Same information can be derived for any author based on below plot.



7. How to detect if certain origination phase causes majority of the in progress Critical-Bug issues?



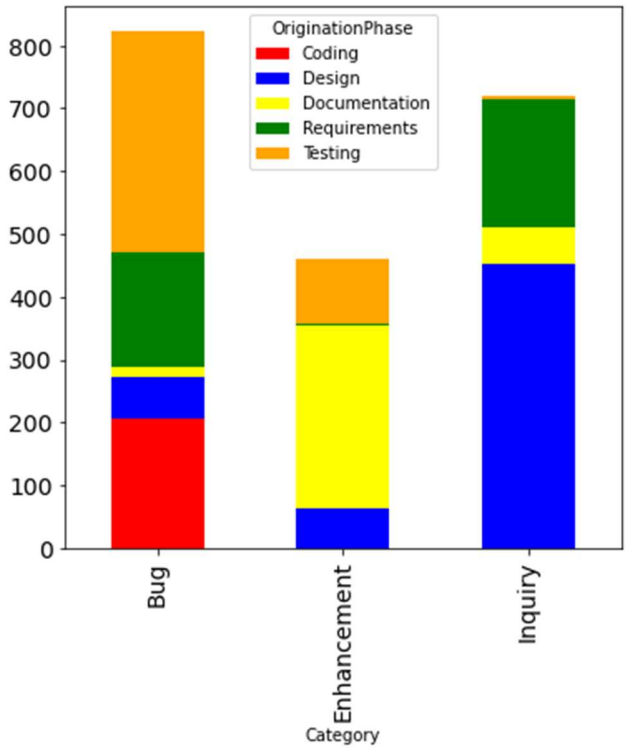
In the above plot, red bar shows critical bugs in progress. It is only seen in Design and Reqirements phase. Based on this we can conclude that certain origination phase design and requirements are causing majority of the in progress critical bug issues.

8. Can you chart the patterns of outliers in the dataset?

Outliers can be identified for particular variables in the dataset. Outliers should not have any pattern. Hence, plotting chart of outliers does not make much sense as it does while identifying hidden problems.

9. Can you create the right pivot stacked bar chart?

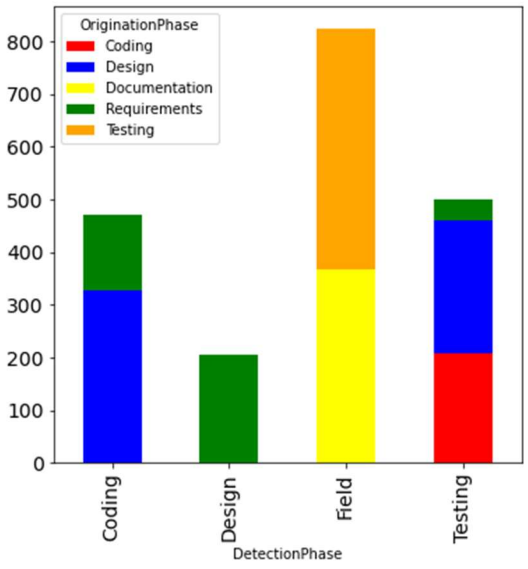
10. How to group multi-levels? Group by Origination phase or Category for example.



11. How many Levels of indexing? Should the Priority be displayed in a pivotchart of DetectionPhase for example

There are two levels of indexing. Primary and secondary. The primary indexing are divided in dense and sparse indexing. Priority of detectionphase should be displayed to understand important bugs which needs to be fixed first.

12. What is the avg number of issues opened per DetectionPhase?



Number of issues opened for each detection phase

Coding : ~500

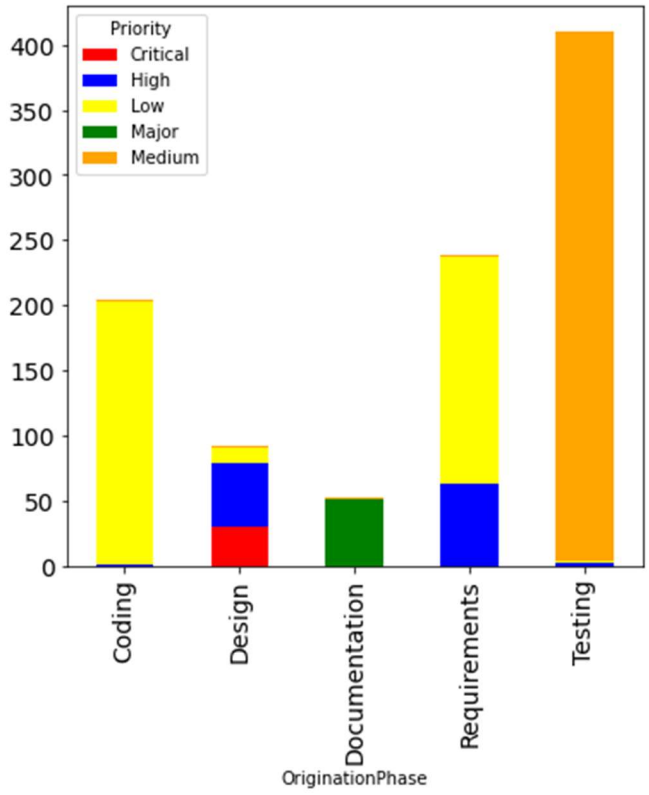
Desing : ~200

Field : ~ 800

Testing : ~500

Total – 2000

13. What is the avg turn around time per issue (time from the day the issue created till it got closed)?



Avg turn-around time per issue as per origination phase (code in the file)

Coding – 1.01 days

Design – 6.29 days

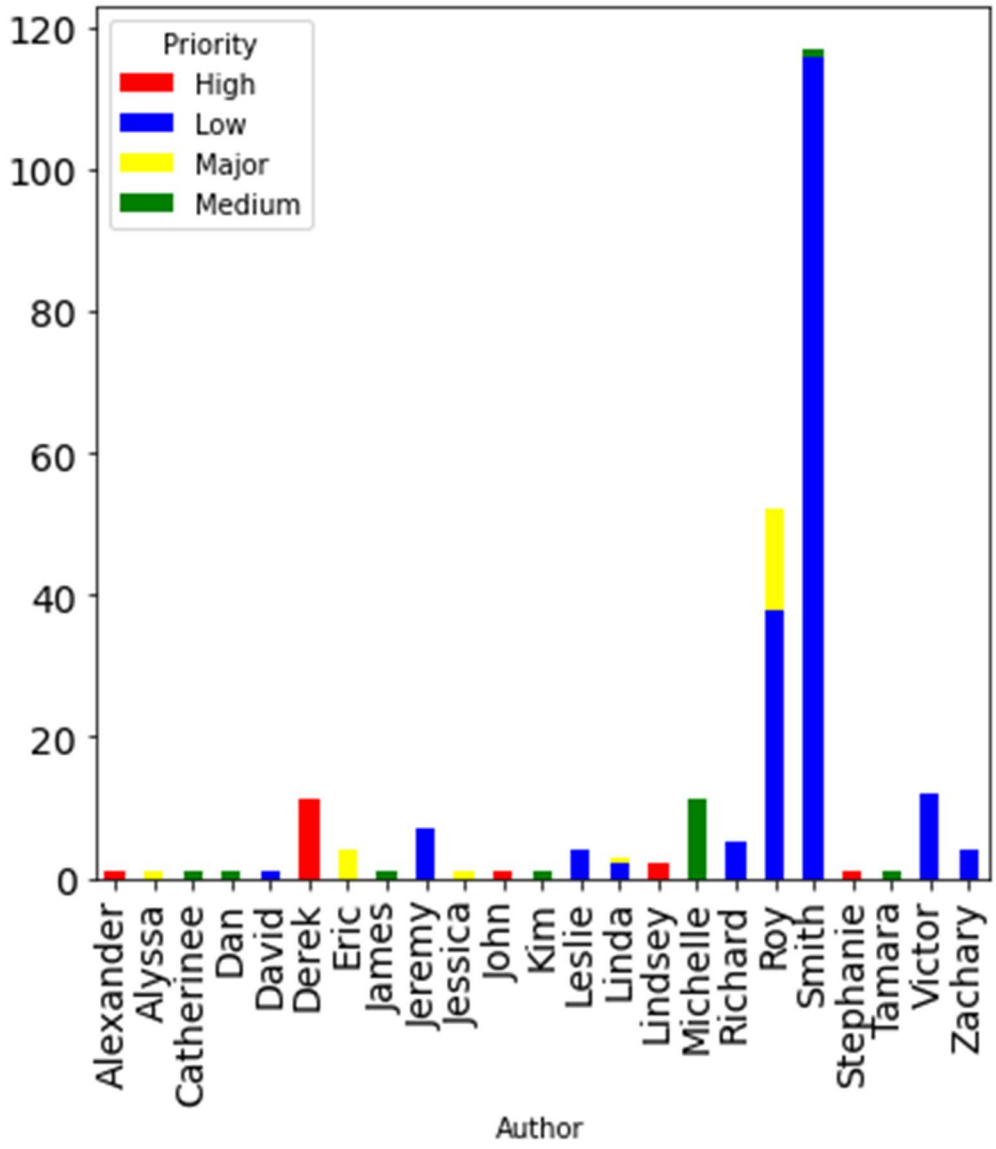
Documentation – 7.05 days

Requirements – 1.63 days

Testing – 1.11 days

14. What is the avg number of rejected issues opened per engineer?

Average number of rejected issues opened per engineer : 3.91 ~4



Smith has highest numbers of rejected issues.

15. What is the avg number of critical issues opened per engineer?

Avg. number of critical issues opened per engineer: 3.46~4 (code in the python file)

16. What is the avg number of rejected issues per Origination Phase?

Total number of rejected issues: 243

Rejected issues in Requirements: 186

Rejected issues in Design: 18

Rejected issues in Coding: 2

Rejected issues in Documentation: 21

Rejected issues in Testing: 16

Average number of rejected issues per origination phase – 243 / 5 = 48.6 ~49

17. What is the avg number of critical issues per Origination Phase?

Total number of critical issues: 215

Critical issues in Requirements: 31

Critical issues in Design: 184

Critical issues in Coding: 0

Critical issues in Documentation: 0

Critical issues in Testing: 0

Average number of Critical issues per origination phase – 215 / 5 = 43

18. What is the avg number of created issues per Origination Phase?

Average number of created issues per origination phase – 2000 / 5 = 400

19. What is the avg number of rejected critical issues per Origination Phase?

None of the critical issues has been rejected.

20. What is the ratio of total number of critical to medium issues per Origination Phase?

Design phase – total number of critical to medium issues ratio – 184 : 3

Requirement phase - total number of critical to medium issues ratio – 31 : 2

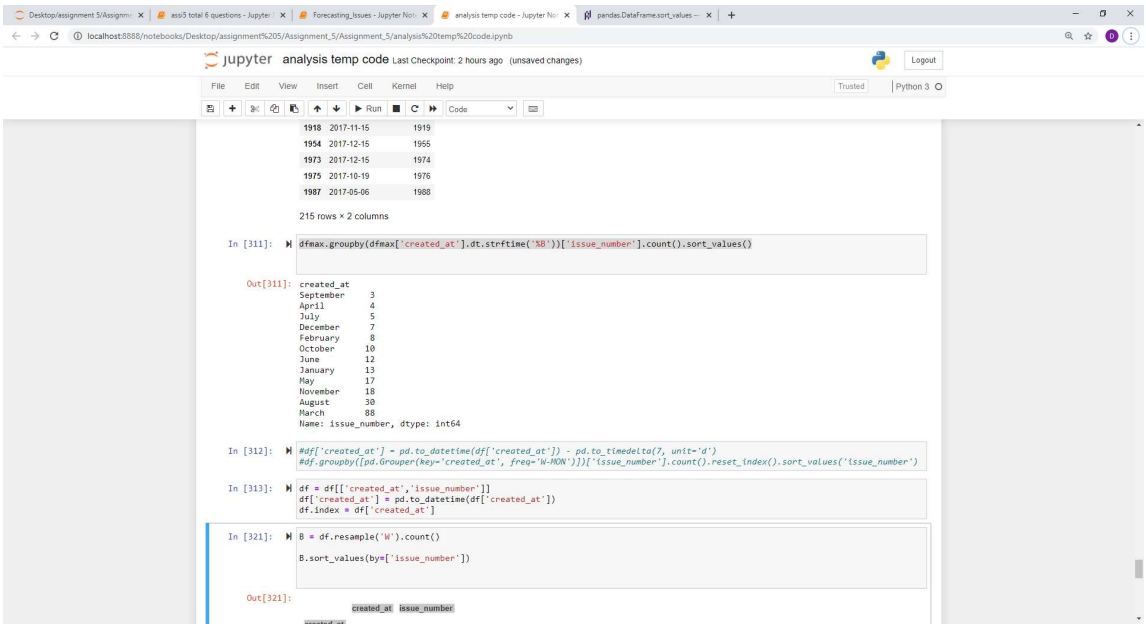
Coding phase - total number of critical to medium issues ratio – 0 : 2

Documentation phase - total number of critical to medium issues ratio – 0 : 2

Testing phase - total number of critical to medium issues ratio – 0 : 454

21. Which month got the maximum number of Critical issues created?

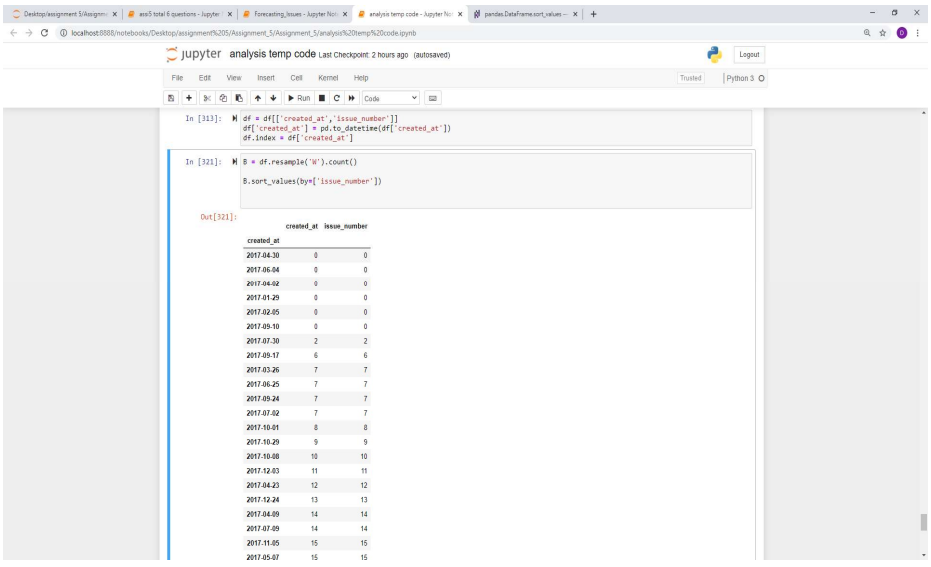
It can be concluded from below table that in the month **of March** maximum number of critical issues were created.



22. Which week got the minimum number of issues created?

From below output, it is clear that there were no issues created in total 6 weeks.

March – 4th week
June – 1st week
March – 1st week
Jan – 1st week
Feb – 1st week
Sept – 2nd week



2) Use Python and Facebook/Prophet package (https://facebook.github.io/prophet/docs/quick_start.html) to forecast the following based on the provided dataset

Note : FBProphet has considered weekdays from Monday to Friday only, whereas issues.csv file has dates which are Saturday and Sunday. Hence, answers of fbprophet and other two methods are different.

1. The day of the week maximum number of issues created

The day of the week maximum number of issues ceated : Monday

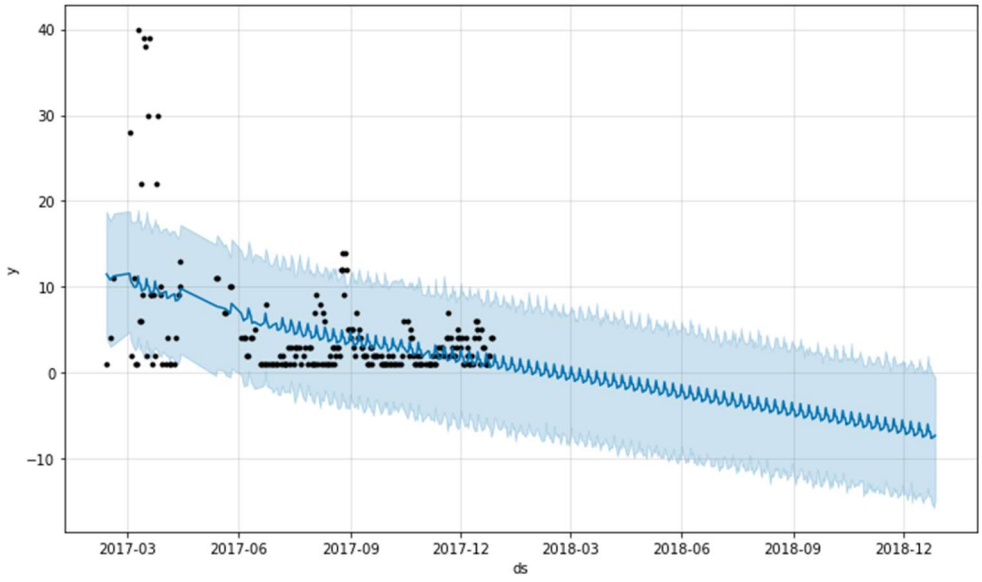
2. The day of the week maximum number of issues closed

day when maximum issues were closed is Friday

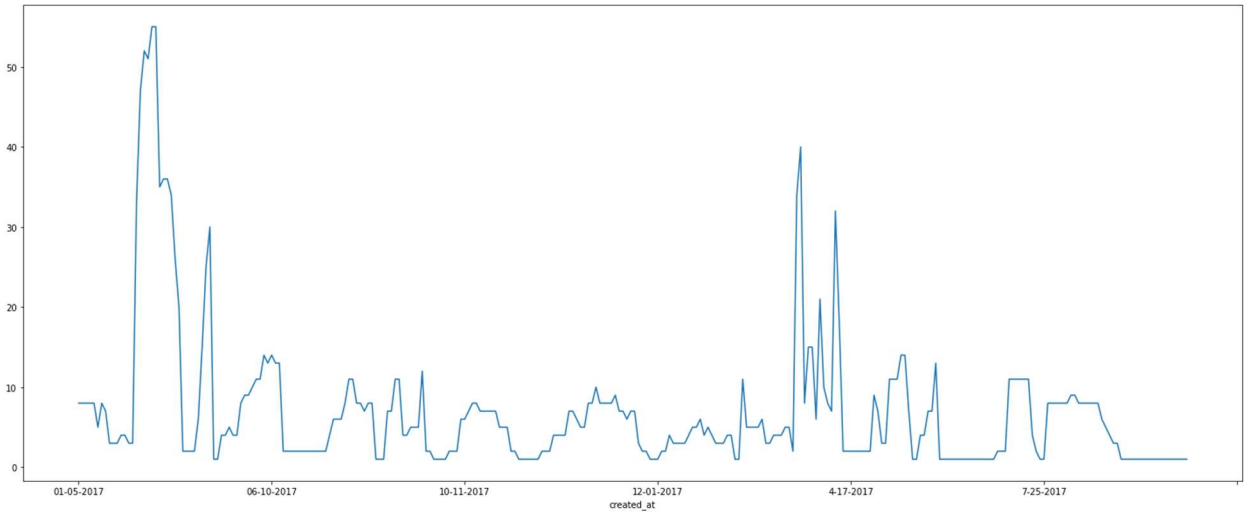
3. The month of the year that has maximum number of issues closed

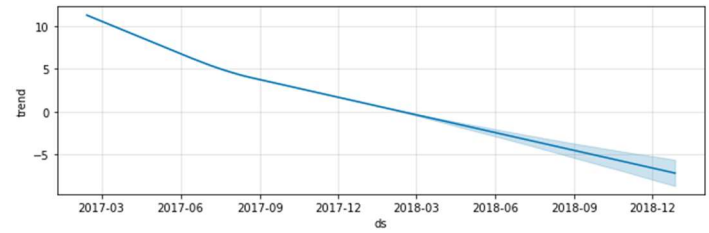
Maximum issues were closed in March - 2017.

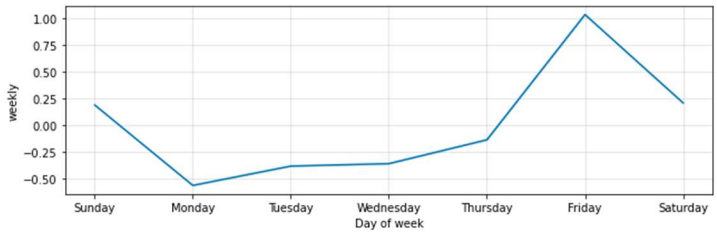
4. Plot the created issues forecast by calling the Prophet.plot method and passing in your forecast dataframe.



5. Plot the closed issues forecast; use the Prophet.plot_components method. By default you'll see the trend, yearly seasonality, and weekly seasonality of the time series. If you include holidays, you'll see those here, too.



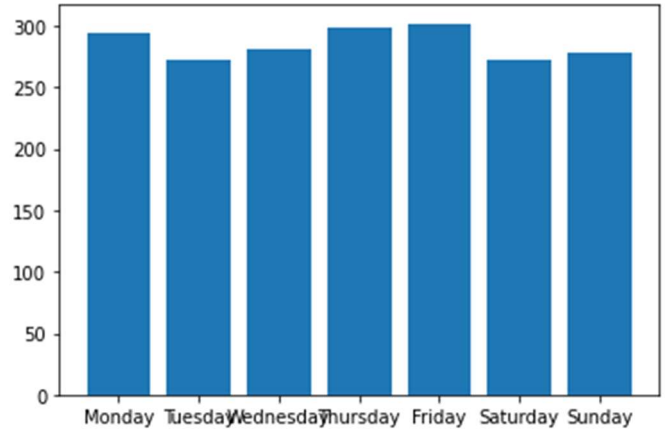




3) Re-implement the above requirements using TensorFlow Time Series (TFTS

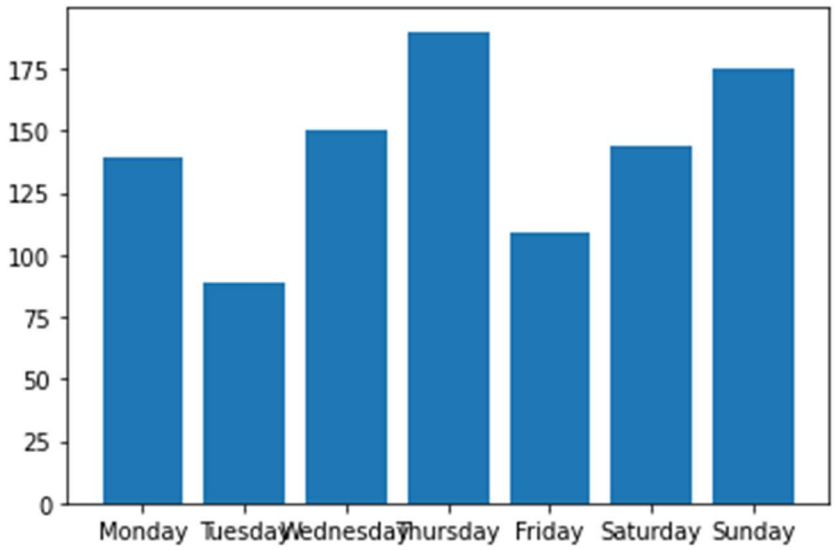
1. The day of the week maximum number of issues created

The day of the week maximum number of issues created : Friday



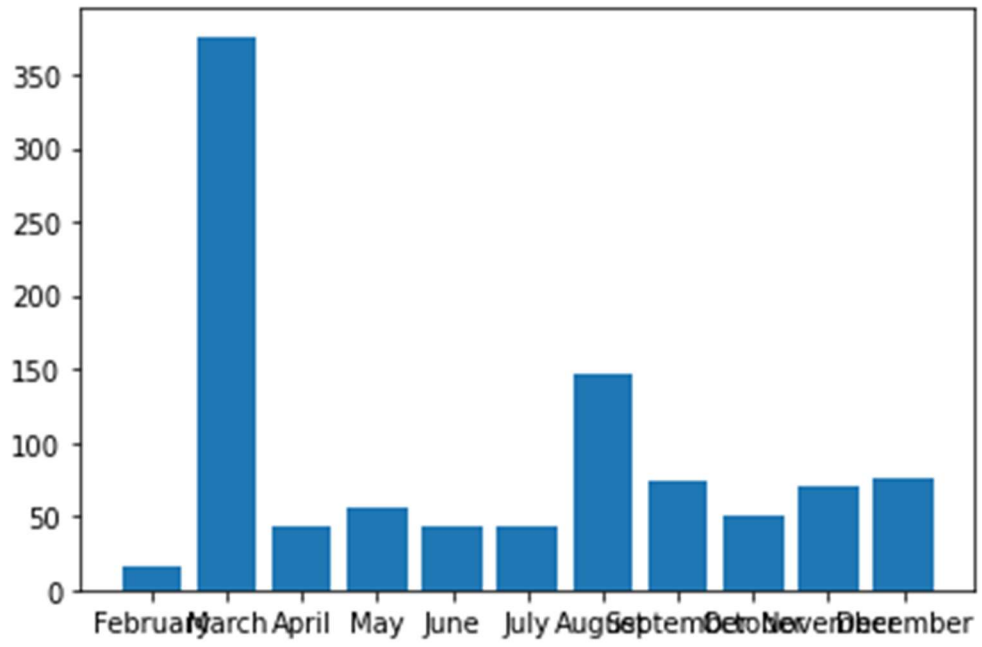
2. The day of the week maximum number of issues closed

day when maximum issues were closed is Thursday

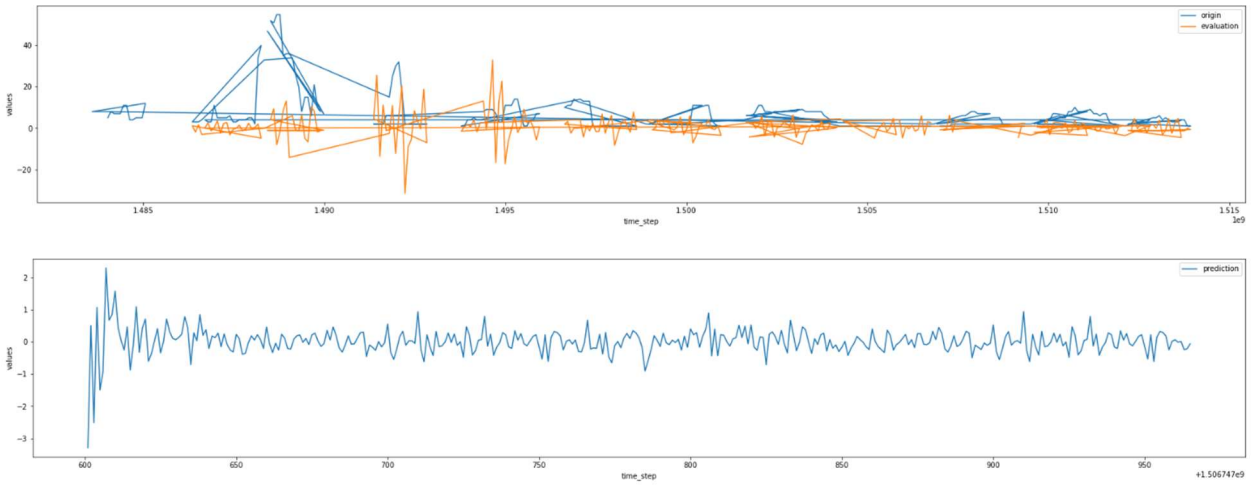


3. The month of the year that has maximum number of issues closed

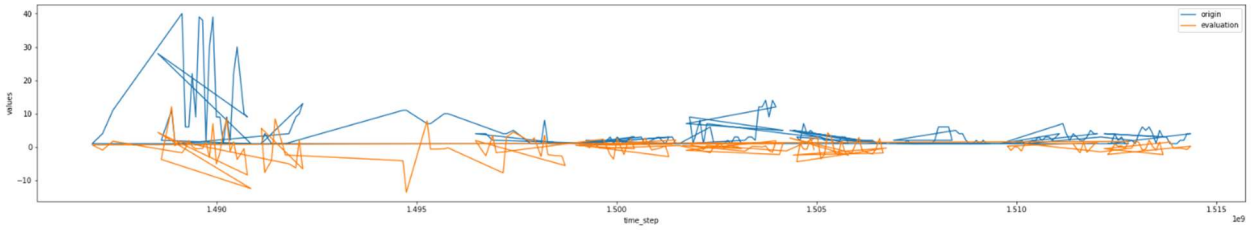
Maximum issues were closed in March - 2017.

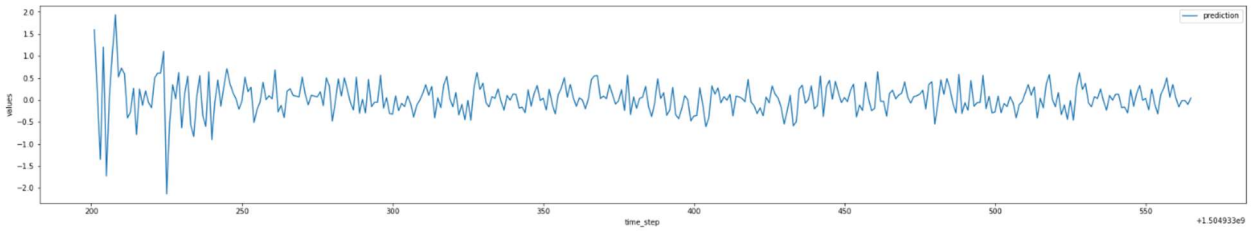


4. Plot the created issues forecast



5. Plot the closed issues forecast

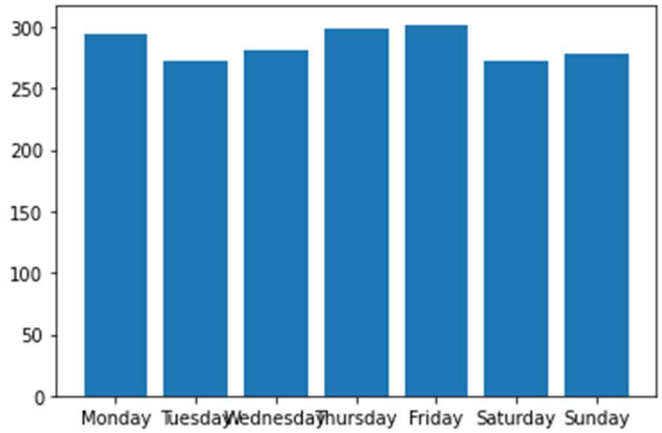




4) Re-implement the above requirements using StatsModel

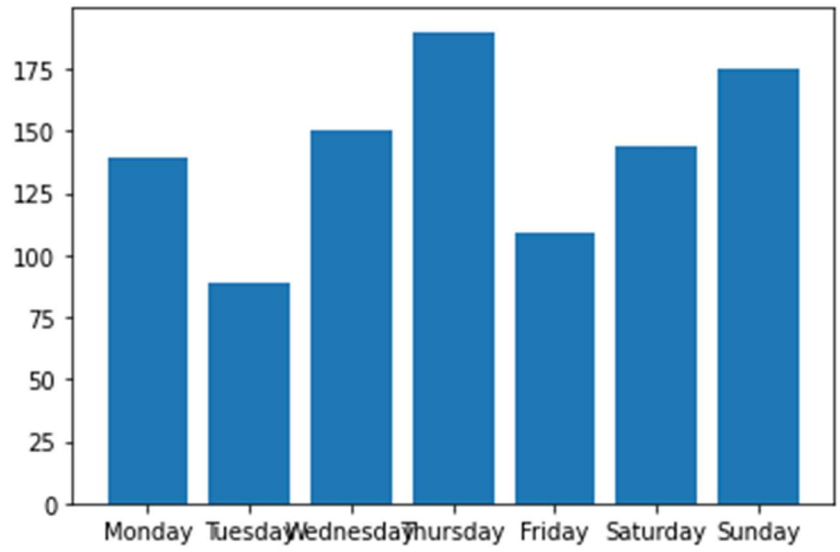
1. The day of the week maximum number of issues created

The day of the week maximum number of issues created : Friday



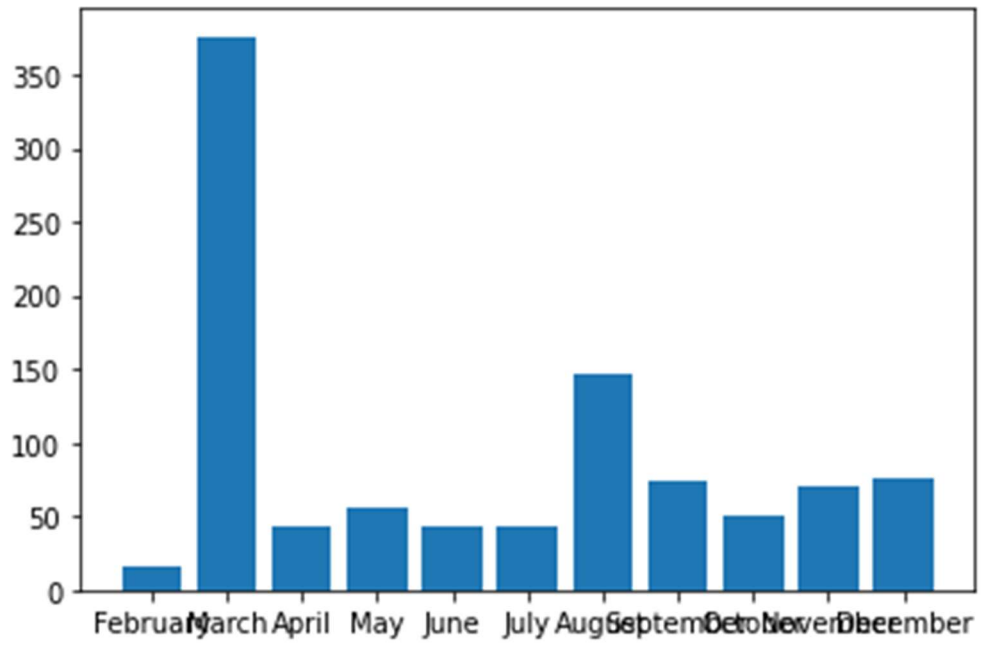
2. The day of the week maximum number of issues closed

day when maximum issues were closed is Thursday

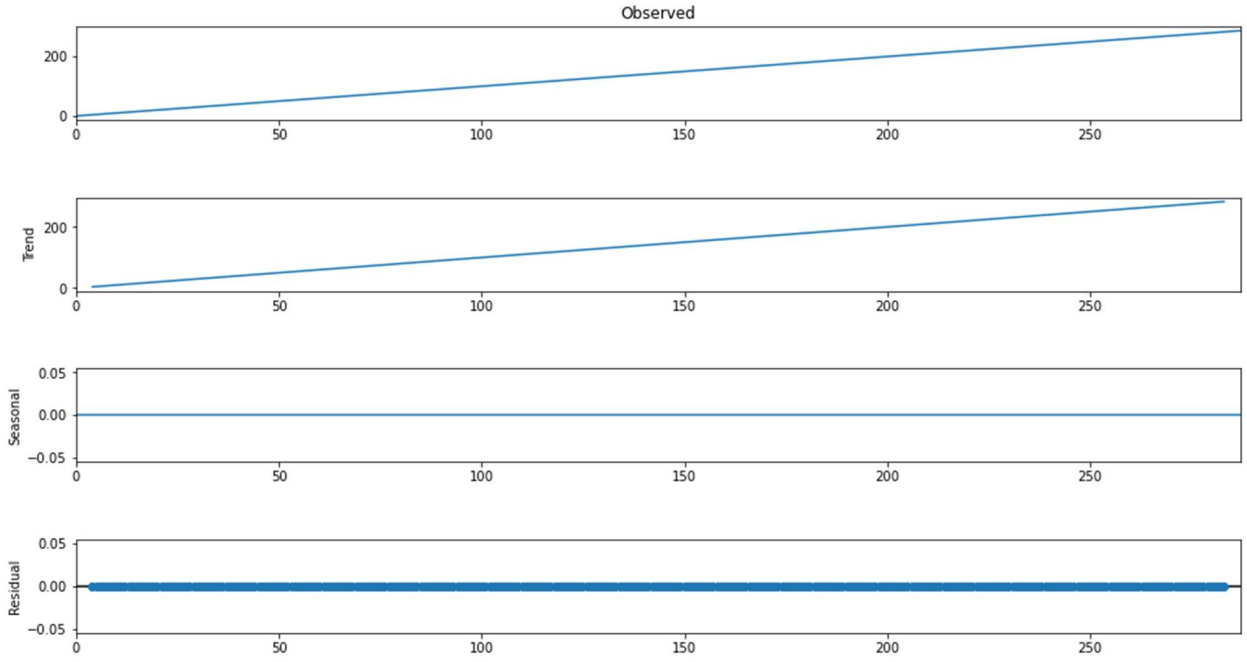


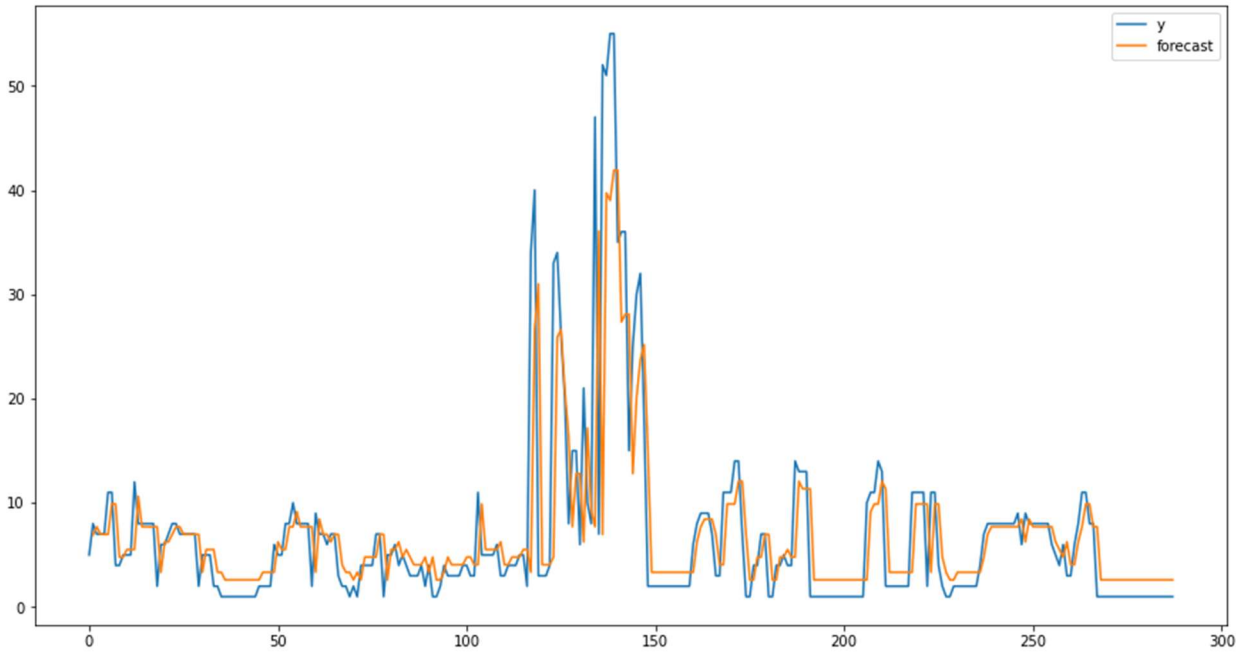
3. The month of the year that has maximum number of issues closed

Maximum issues were closed in March - 2017.

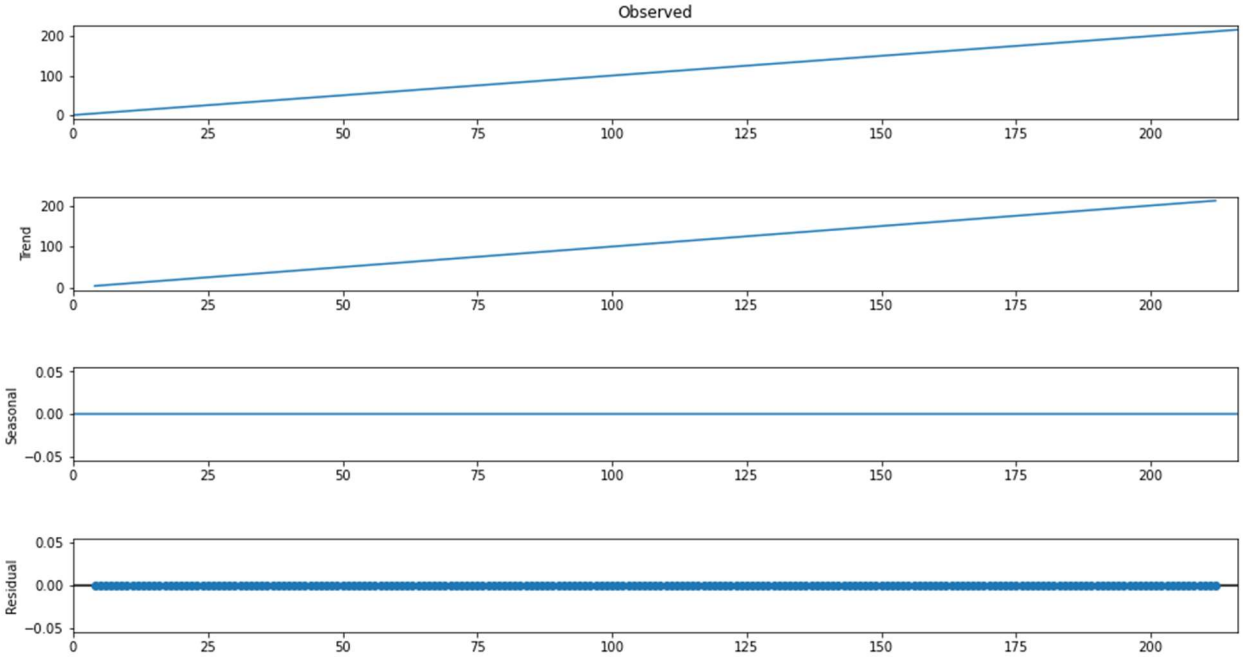


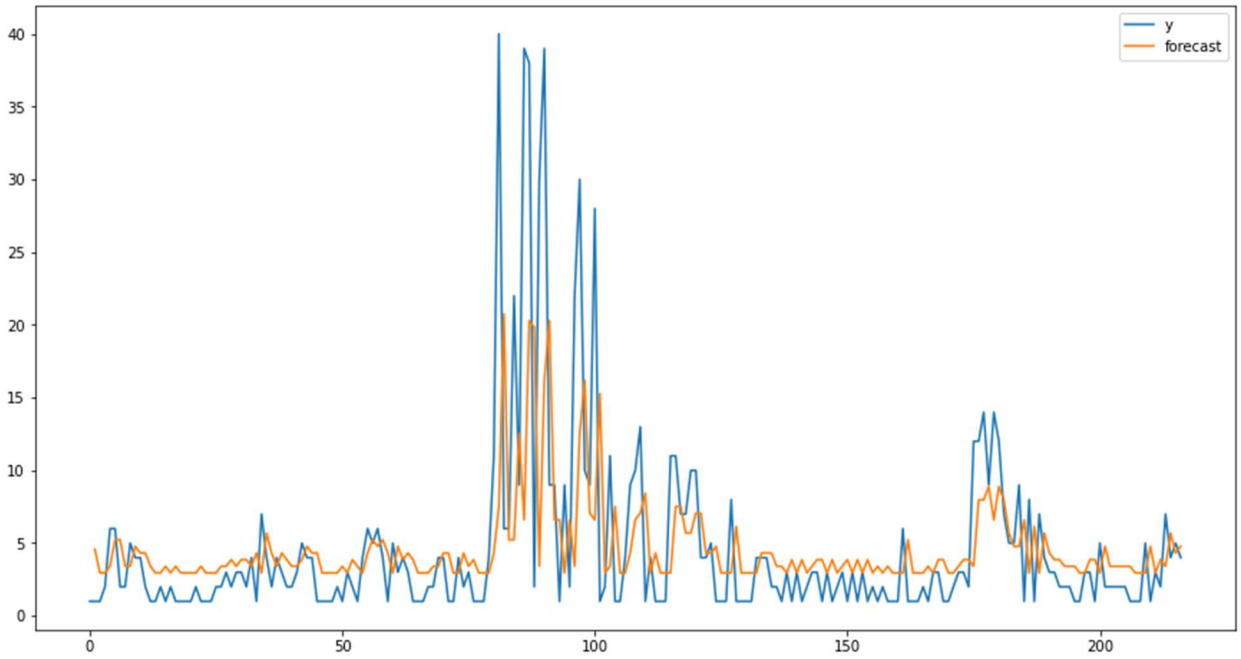
4. Plot the created issues forecast





5. Plot the closed issues forecast




y
forecast
40
35
30
25
20
15
10
5
0
0
50
100
150
200