Assignment #5

Due Date: 5/2/21 by 11:59pm

Deliverable:

Post your homework as a SINGLE ZIP file on Blackboard with the name "HW5_YourLastName_ FirstName"

Important Notes:

- Do NOT communicate or share your assignment with others
- Do NOT share your personal laptop with your classmates
- There are 2 parts for this Assignment and both are required to be submitted in a single ZIP file

Required Readings

Before you start working on this assignment you must:

1. Review the tutorial and ipynb script for forecasting and prediction using StatsModels, Fbprophet, and TensorFlow/Keras LSTM discussed in the class lecture.

2. Read the following articles:

    a. https://towardsdatascience.com/time-series-in-python-exponential-smoothing-and-arima-processes-2c67f2a52788
    b. https://machinelearningmastery.com/time-series-forecasting-methods-in-python-cheat-sheet/
    c. https://towardsdatascience.com/an-end-to-end-project-on-time-series-analysis-and-forecasting-with-python-4835e6bf050b

PART I - Requirements:

Consider the data listed in the following matrix for a product of size 120KLOC:

1. Calculate the defect removal rate for every phase
2. Calculate the defect injection rate for every phase
3. Calculate the defect escape rate for every phase
4. Calculate the overall defect removal effectiveness.
5. Which phase is the most effective in removing defects? Explain.
6. Do you think reviews and inspections were effective? Explain.
7. If the number of defects originated in design phase increased by 15% and defects detected in design review increased by 60%, would these changes increase or decrease the defects **escaped** to the coding phase? Explain your answer in detail (present data to support your answer).

		Defect Origin							
		Requirement	Analysis	Design	Coding	Unit Testing	Integration Testing	System Testing	Field
Where Found	Requirement	223							
	Analysis	102	34						
	Design	56	45	210					
	Coding	97	71	62	278				
	Unit Testing	43	31	43	123	4			
	Integration Testing	12	21	34	37	-	7		
	System Testing	7	4	21	29	-	-	6	
	Field	2	1	3	5	-	-	-	7

PART II - Requirements:

General Description:

Different tasks in the software development process like testing, inspections, and reviews are the means by which we control and manage the quality of the software products when we plan, execute, manage, and monitor the software projects. It is common to have outliers in datasets of software metrics, change requests, and quality records, however, outliers in these datasets are often the most interesting metrics that often used to detect potentially useful patterns of hidden problem. The S-curve and Bell-curve are the tools used to track and monitor the healthy progress of the project while executing the software project plan and compare the planned numbers to the actual numbers of tasks executed, effort spent, change requests, issues created, and defects detected. However, analyzing the outlyingness of issues and defects reported with respect to the average expectations might be indicative of hidden and potentially useful patterns to detect problematic areas in the workflow of the software development process lifecycle or the software project plan. Also, datasets of historical data gathered from prior projects can be utilized to make reliable forecast for future tasks and resource requirements.

General Instructions:

1. Write a technical report that presents and analyzes the experimental results that can help in detecting outliers in the provided dataset of issues that might indicate hidden problems in the given data set of issues for project X

2. Modify the provided python ipynb script to run your experiments to obtain results to use in your analysis

3. Only a tested and runnable ipynb/python script can get a credit; NO partial credit for code that doesn't produce output

4. Leverage all knowledge and skills you have learned in the assignments and lecture notes when writing your report and discussing the experimental results

Requirements:

The default labels that GitHub provides us with are very primitive and very insufficient for hierarchical indexing and filtering. Your task for this assignment is to modify the provided ipynb script to answer simple and complex queries, chart the data, forecast issues, and provide a report that analyzes the experimental results to detect outliers that might be used to indicate patterns of potentially hidden problems in the given dataset.

Use the provided CSV file for the issues dataset to obtain the experimental results

1) **Issues Tracking and Report Design**: Provide your Python ipynb script that will allow project managers and quality managers to answer and chart all types of queries related to issues/tickets/change requests/modification requests; the following queries are only a sample of queries that your python script must be able to answer and provide the output report/chart
    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)

2) **Labeling Scheme**: The Labeling Scheme follows the **key:value** format "**LabelName:*LabelValue***" for every label

3) **Types of Labels**: There are different types of labels, the following is the list of labels used for issues listed in the provided dataset:

   i. **OriginationPhase**: Could have one of the values{Requirements, Design, Coding, Testing, Documentation, Field}
   ii. **DetectionPhase**: Could have one of the values{Requirements, Design, Coding, Testing, Documentation, Field}
   iii. **Priority**: Could have one of the values{Critical, Major, High, Low, Medium}
   iv. **Status**: Could have one of the values{Approved, Rejected, Completed, inProgress, pendingReview}
   v. **Category**: Could have one of the values{Bug, Enhancement, Inquiry}

1. Data Analysis & Statistical Control Process:

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

Examples of what you will consider in your analysis report
Are there any useful patterns of outliers in the dataset?
What are the UCL (Upper Control Limit) and LCL (Lower Control Limit) for certain issue metrics?
Can the outliers detect hidden problems in the given dataset?
What is the correlation between outliers and hidden problems in the given dataset?
How to detect if there is problem hidden in the given dataset?
How to detect if certain engineer deliberately creates issues with Priority Critical?
How to detect if certain origination phase causes majority of the in progress-Critical-Bug issues?
Can you chart the patterns of outliers in the dataset?
Can you create the right pivot stackedbar chart?
How to group multi-levels? Group by Origination phase or Category for example.
How many Levels of indexing? Should the Priority be displayed in a pivotchart of DetectionPhase for example
What is the avg number of issues opened per DetectionPhase?
What is the avg turn around time per issue (time from the day the issue created till it got closed)?
What is the avg number of rejected issues opened per eningeer?
What is the avg number of critical issues opened per eningeer?
What is the avg number of rejected issues per OriginationPhase?
What is the avg number of critical issues per OriginationPhase?
What is the avg number of created issues per OriginationPhase?
What is the avg number of rejected critical issues per OriginationPhase?
What is the ratio of total number of critical to medium issues per OriginationPhase?
Which month got the maximum number of Critical issues created?
Which week got the minimum number of issues created?

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
   1. The day of the week maximum number of issues created
   2. The day of the week maximum number of issues closed
   3. The month of the year that has maximum number of issues closed
   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 2/Keras LSTM**

4) Re-implement the above requirements using **StatsModel**

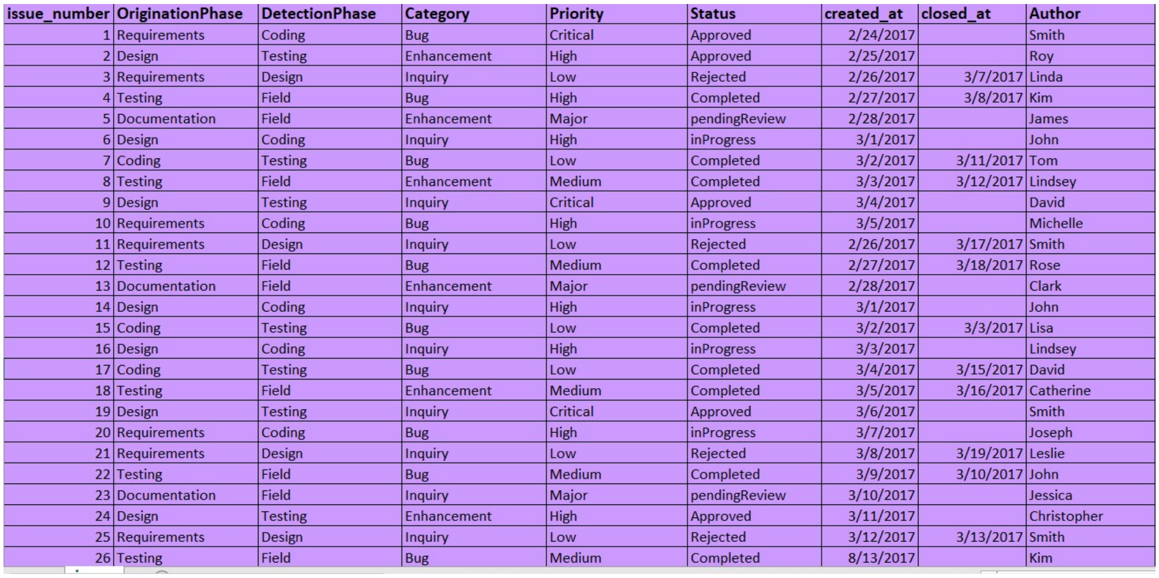
Deliverables:

You are required to submit in the ZIP file the following deliverables are:

1. Your Python ipynb script Source Code and Output
2. Your Technical report in PDF format that presents and analyzes the experimental results of the provided data set

Appendix A:

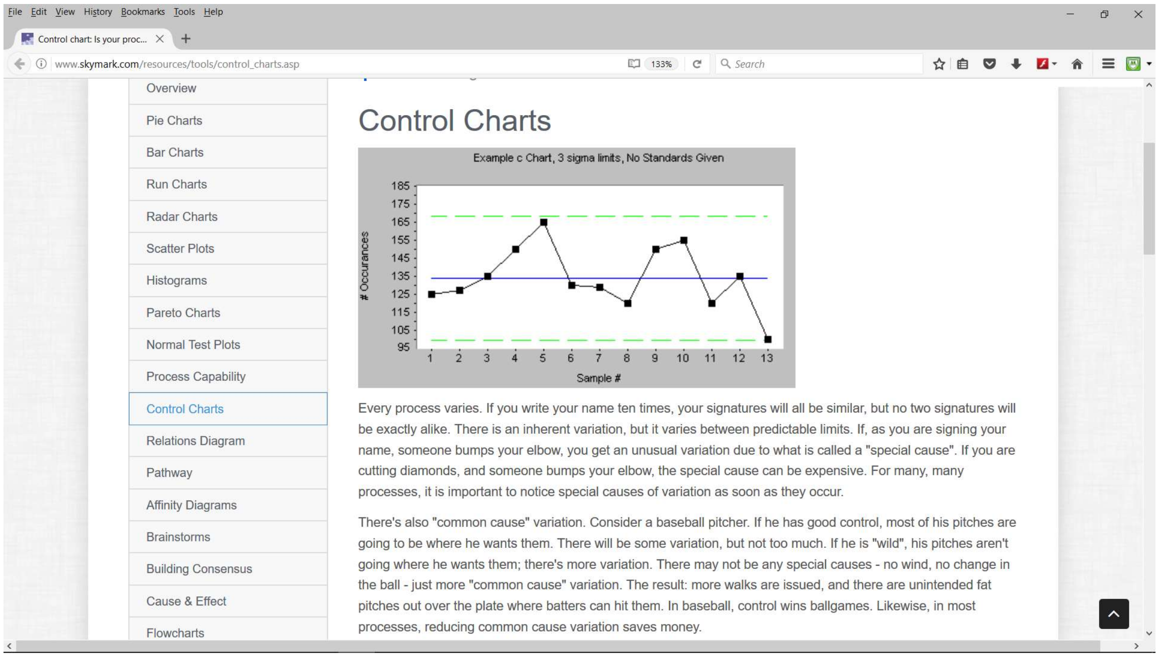
Issues Dataset. The issues.csv file has the following layout(you could open the CSV in excel or notepad).



issue_number	OriginationPhase	DetectionPhase	Category	Priority	Status	created_at	closed_at	Author
1	Requirements	Coding	Bug	Critical	Approved	2/24/2017		Smith
2	Design	Testing	Enhancement	High	Approved	2/25/2017		Roy
3	Requirements	Design	Inquiry	Low	Rejected	2/26/2017	3/7/2017	Linda
4	Testing	Field	Bug	High	Completed	2/27/2017	3/8/2017	Kim
5	Documentation	Field	Enhancement	Major	pendingReview	2/28/2017		James
6	Design	Coding	Inquiry	High	inProgress	3/1/2017		John
7	Coding	Testing	Bug	Low	Completed	3/2/2017	3/11/2017	Tom
8	Testing	Field	Enhancement	Medium	Completed	3/3/2017	3/12/2017	Lindsey
9	Design	Testing	Inquiry	Critical	Approved	3/4/2017		David
10	Requirements	Coding	Bug	High	inProgress	3/5/2017		Michelle
11	Requirements	Design	Inquiry	Low	Rejected	2/26/2017	3/17/2017	Smith
12	Testing	Field	Bug	Medium	Completed	2/27/2017	3/18/2017	Rose
13	Documentation	Field	Enhancement	Major	pendingReview	2/28/2017		Clark
14	Design	Coding	Inquiry	High	inProgress	3/1/2017		John
15	Coding	Testing	Bug	Low	Completed	3/2/2017	3/3/2017	Lisa
16	Design	Coding	Inquiry	High	inProgress	3/3/2017		Lindsey
17	Coding	Testing	Bug	Low	Completed	3/4/2017	3/15/2017	David
18	Testing	Field	Enhancement	Medium	Completed	3/5/2017	3/16/2017	Catherine
19	Design	Testing	Inquiry	Critical	Approved	3/6/2017		Smith
20	Requirements	Coding	Bug	High	inProgress	3/7/2017		Joseph
21	Requirements	Design	Inquiry	Low	Rejected	3/8/2017	3/19/2017	Leslie
22	Testing	Field	Bug	Medium	Completed	3/9/2017	3/10/2017	John
23	Documentation	Field	Inquiry	Major	pendingReview	3/10/2017		Jessica
24	Design	Testing	Enhancement	High	Approved	3/11/2017		Christopher
25	Requirements	Design	Inquiry	Low	Rejected	3/12/2017	3/13/2017	Smith
26	Testing	Field	Bug	Medium	Completed	8/13/2017		Kim

Appendix B:

https://en.wikipedia.org/wiki/Control_chart
https://en.wikipedia.org/wiki/Six_Sigma
http://www.skymark.com/resources/tools/control_charts.asp



Control Charts

Every process varies. If you write your name ten times, your signatures will all be similar, but no two signatures will be exactly alike. There is an inherent variation, but it varies between predictable limits. If, as you are signing your name, someone bumps your elbow, you get an unusual variation due to what is called a "special cause". If you are cutting diamonds, and someone bumps your elbow, the special cause can be expensive. For many, many processes, it is important to notice special causes of variation as soon as they occur.

There's also "common cause" variation. Consider a baseball pitcher. If he has good control, most of his pitches are going to be where he wants them. There will be some variation, but not too much. If he is "wild", his pitches aren't going where he wants them; there's more variation. There may not be any special causes - no wind, no change in the ball - just more "common cause" variation. The result: more walks are issued, and there are unintended fat pitches out over the plate where batters can hit them. In baseball, control wins ballgames. Likewise, in most processes, reducing common cause variation saves money.