Forecasting Disk Usage with Machine Learning – So easy, even a cave person can do it!

Steve Koelpin and Alicia Dale
TransUnion

splunk> .conf19

Steve Koelpin

Advisor - Splunk
TransUnion

Forward-Looking Statements

During the course of this presentation, we may make forward-looking statements regarding future events or plans of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results may differ materially. The forward-looking statements made in the this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, it may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements made herein.

In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only, and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionalities described or to include any such feature or functionality in a future release.

Splunk, Splunk>, Turn Data Into Doing, The Engine for Machine Data, Splunk Cloud, Splunk Light and SPL are trademarks and registered trademarks of Splunk Inc. in the United States and other countries. All other brand names, product names, or trademarks belong to their respective owners. © 2019 Splunk Inc. All rights reserved.

Agenda

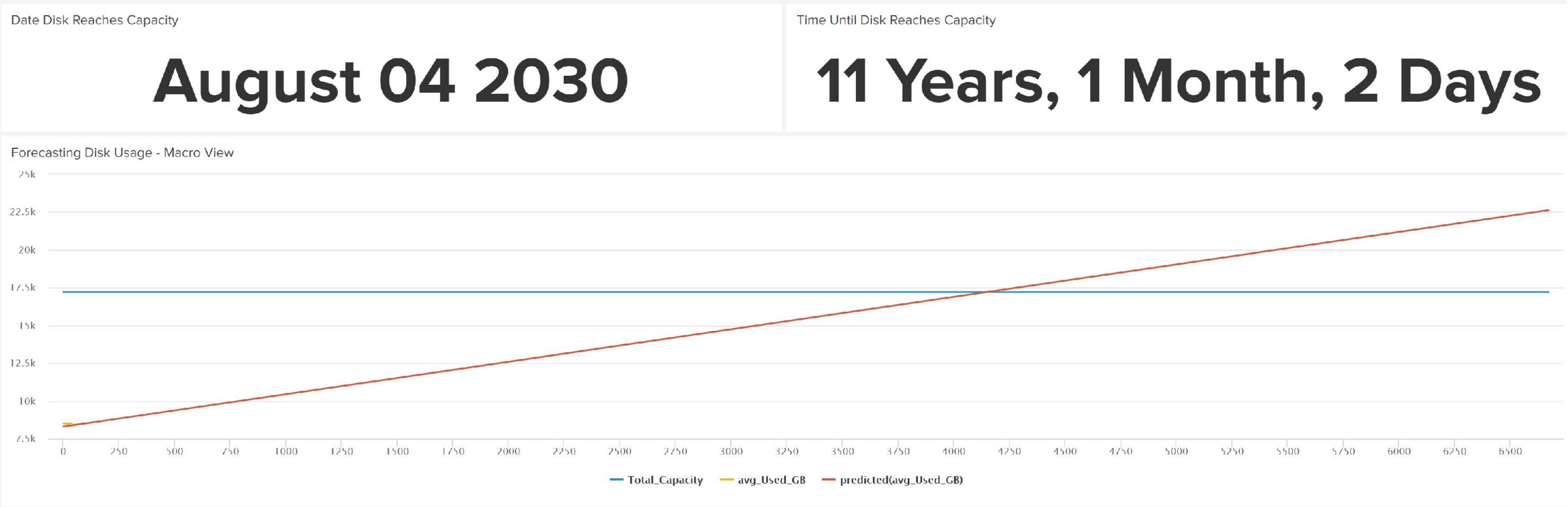
1. Objective
2. Common Challenges
3. Dark Data
4. Technical Deep Dive
5. Build a Score Mechanism
6. Scaling this Out
7. Apply this to your data with walk-through

Objective

In need of a solution to get **ahead** of capacity constraints

Objective

When will a server run out of disk?



Date Disk Reaches Capacity

August 04 2030

Time Until Disk Reaches Capacity

11 Years, 1 Month, 2 Days

Forecasting Disk Usage - Macro View

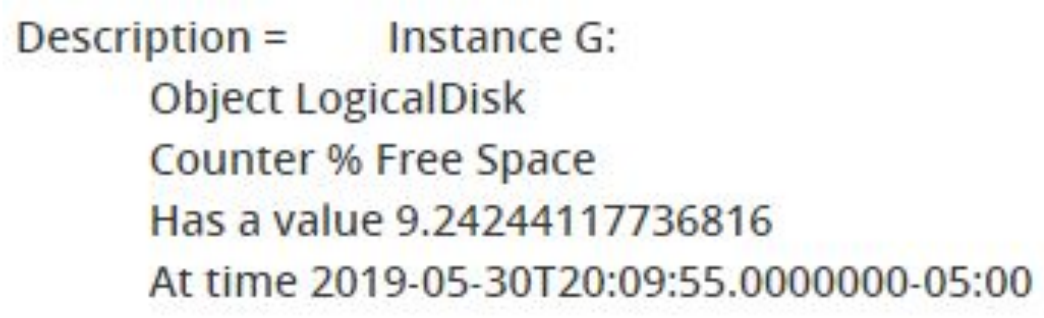
Go from **Guessing** to **Knowing**,
allowing your organization to run more **Proactively**.

Common Challenges

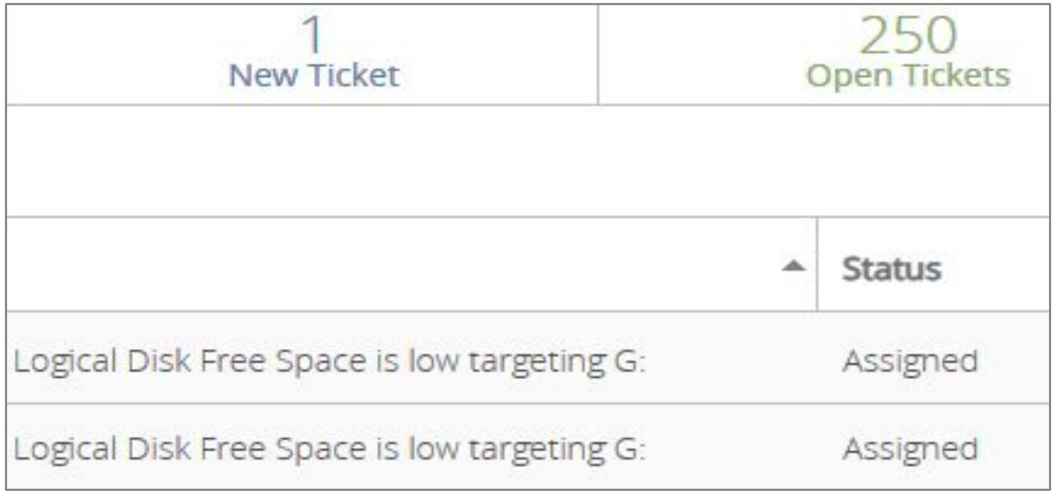
Identifying Inefficiencies
Current VS Desired Process Flows
Total Lead Time

Common Challenges

- Reacting to space issues
- Inefficient work flow stream
- 30% of incidents due to low disk space



```
Description =       Instance G:
        Object LogicalDisk
        Counter % Free Space
        Has a value 9.24244117736816
        At time 2019-05-30T20:09:55.0000000-05:00
```



1 New Ticket

250 Open Tickets

	Status
Logical Disk Free Space is low targeting G:	Assigned
Logical Disk Free Space is low targeting G:	Assigned

Identifying Inefficiencies

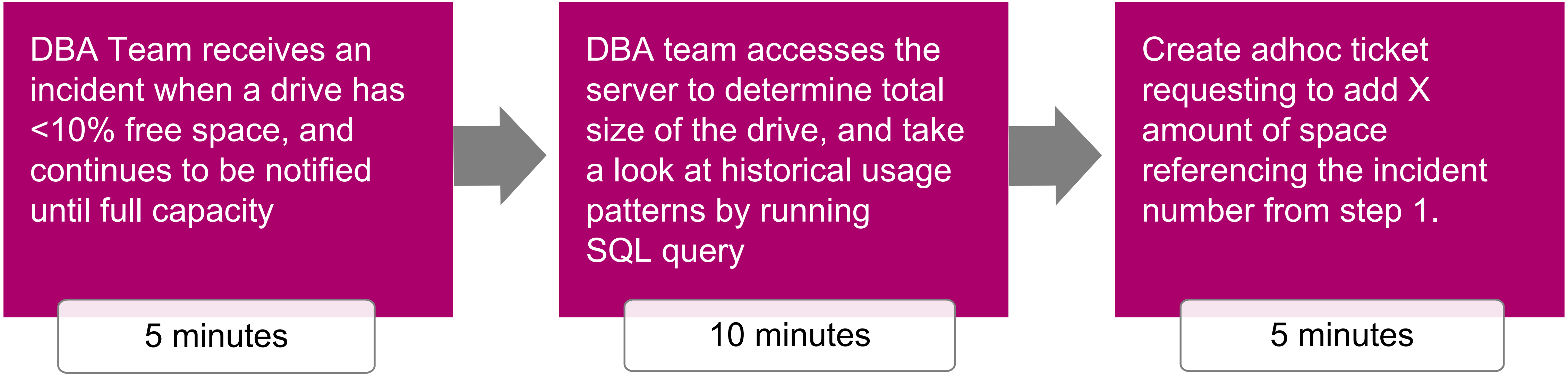
Inefficient Process

- DBA Team has, at times thousands of incident tickets
- **30%** of incidents are for low disk space

What Leads to Inefficiency

- Difficult to distinguish the critical issues from the noise
- Using % for when a disk is at risk is convoluting

Current Process Flow Chart



DBA Team receives an incident when a drive has <10% free space, and continues to be notified until full capacity

5 minutes

DBA team accesses the server to determine total size of the drive, and take a look at historical usage patterns by running SQL query

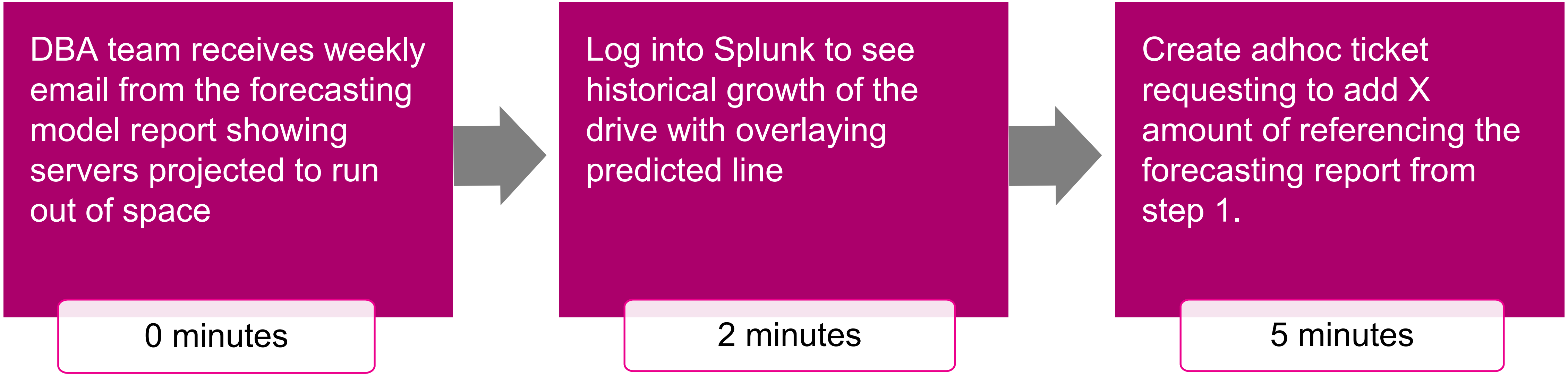
10 minutes

Create adhoc ticket requesting to add X amount of space referencing the incident number from step 1.

5 minutes

Total Time **20 Minutes**

Desired Process Flow Chart



DBA team receives weekly email from the forecasting model report showing servers projected to run out of space

0 minutes

Log into Splunk to see historical growth of the drive with overlaying predicted line

2 minutes

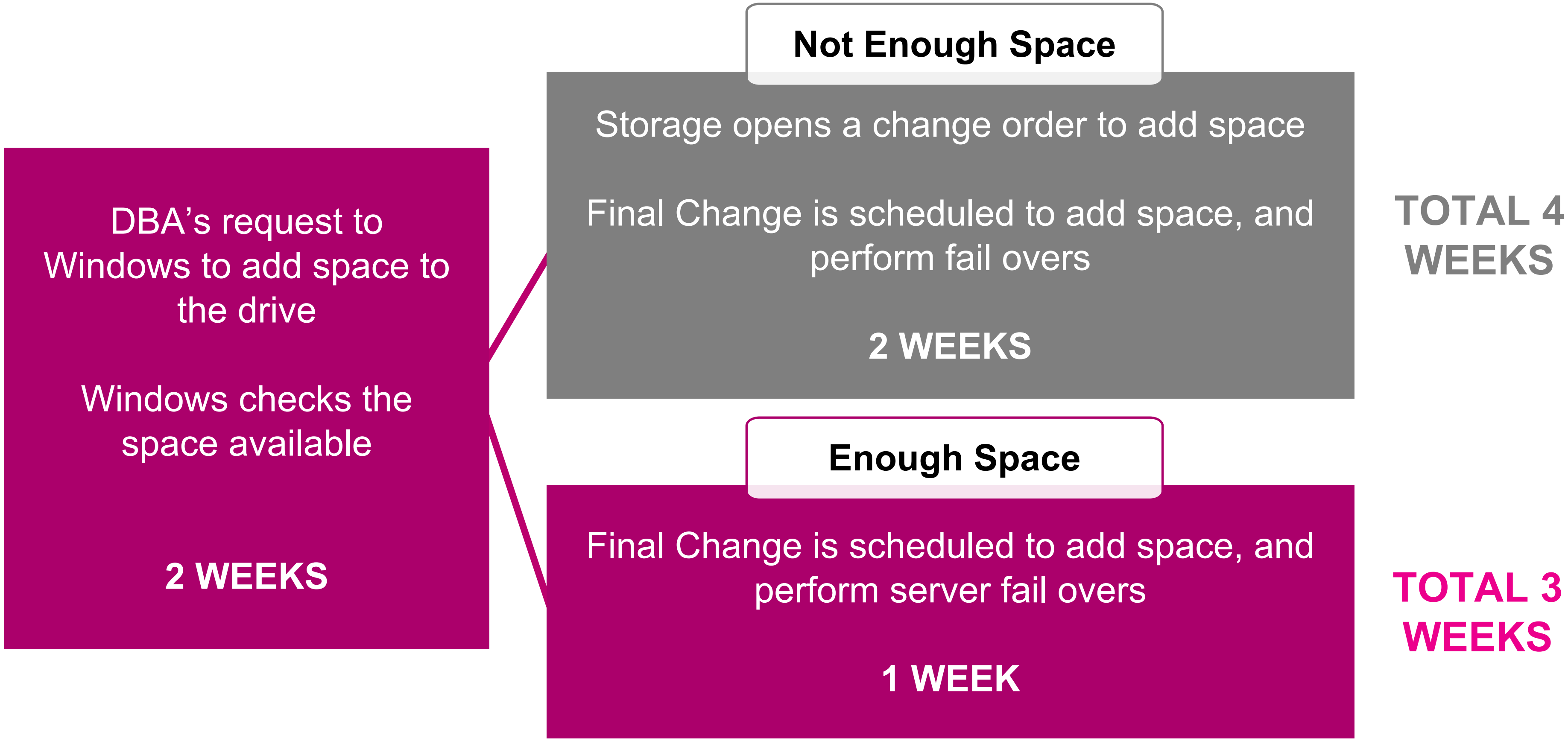
Create adhoc ticket requesting to add X amount of referencing the forecasting report from step 1.

5 minutes

Total Time **7 Minutes**

Total Lead Time

Process Flow showcasing adding space to a serve



DBA’s request to Windows to add space to the drive

Windows checks the space available

2 WEEKS

Not Enough Space

Storage opens a change order to add space

Final Change is scheduled to add space, and perform fail overs

2 WEEKS

TOTAL 4 WEEKS

Enough Space

Final Change is scheduled to add space, and perform server fail overs

1 WEEK

TOTAL 3 WEEKS

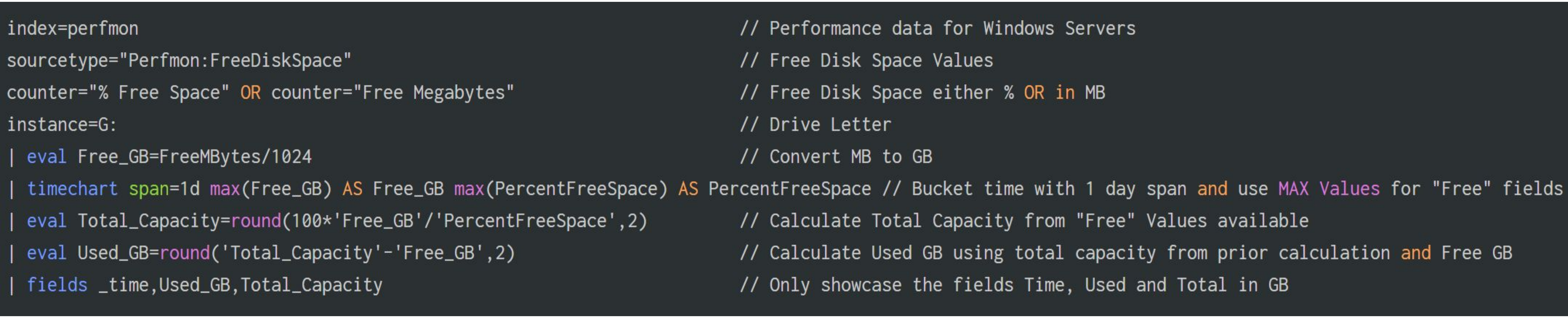
Dark Data

What you can do with your dark data

Dark Data

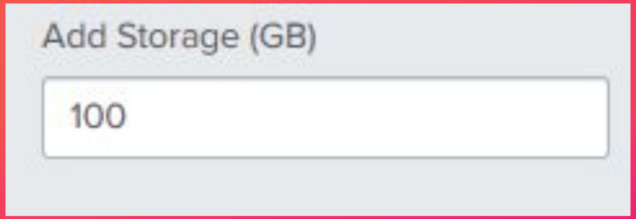
What you can do with your Dark Data

- How many of you have perfmon data available to you?
- How many have MLTK installed or the ability to get it installed?



```
index=perfmon                                                           // Performance data for Windows Servers
sourcetype="Perfmon:FreeDiskSpace"                                      // Free Disk Space Values
counter="% Free Space" OR counter="Free Megabytes"                      // Free Disk Space either % OR in MB
instance=G:                                                             // Drive Letter
| eval Free_GB=FreeMBytes/1024                                          // Convert MB to GB
| timechart span=1d max(Free_GB) AS Free_GB max(PercentFreeSpace) AS PercentFreeSpace // Bucket time with 1 day span and use MAX Values for "Free" fields
| eval Total_Capacity=round(100*'Free_GB'/'PercentFreeSpace',2)         // Calculate Total Capacity from "Free" Values available
| eval Used_GB=round('Total_Capacity'-'Free_GB',2)                      // Calculate Used GB using total capacity from prior calculation and Free GB
| fields _time,Used_GB,Total_Capacity                                   // Only showcase the fields Time, Used and Total in GB
```

A server is forecasted to run out of disk in one month, if I add 100GB, **how much time** will that buy me?

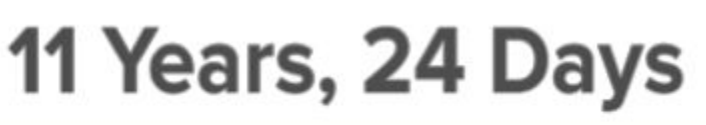


Add Storage (GB)

100

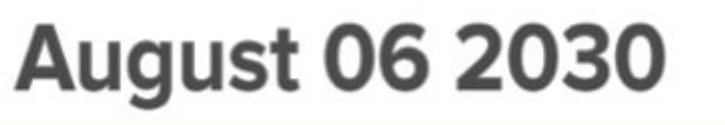
Solve Your Business Problems

How much time until full capacity is reached?



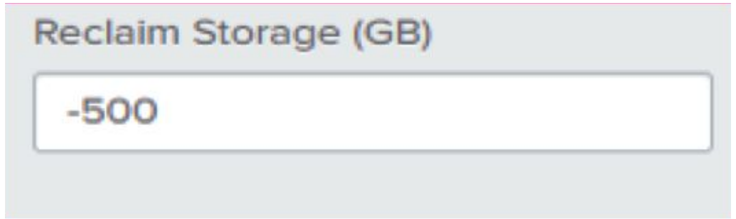
11 Years, 24 Days

Which day, month and year to reach capacity?



August 06 2030

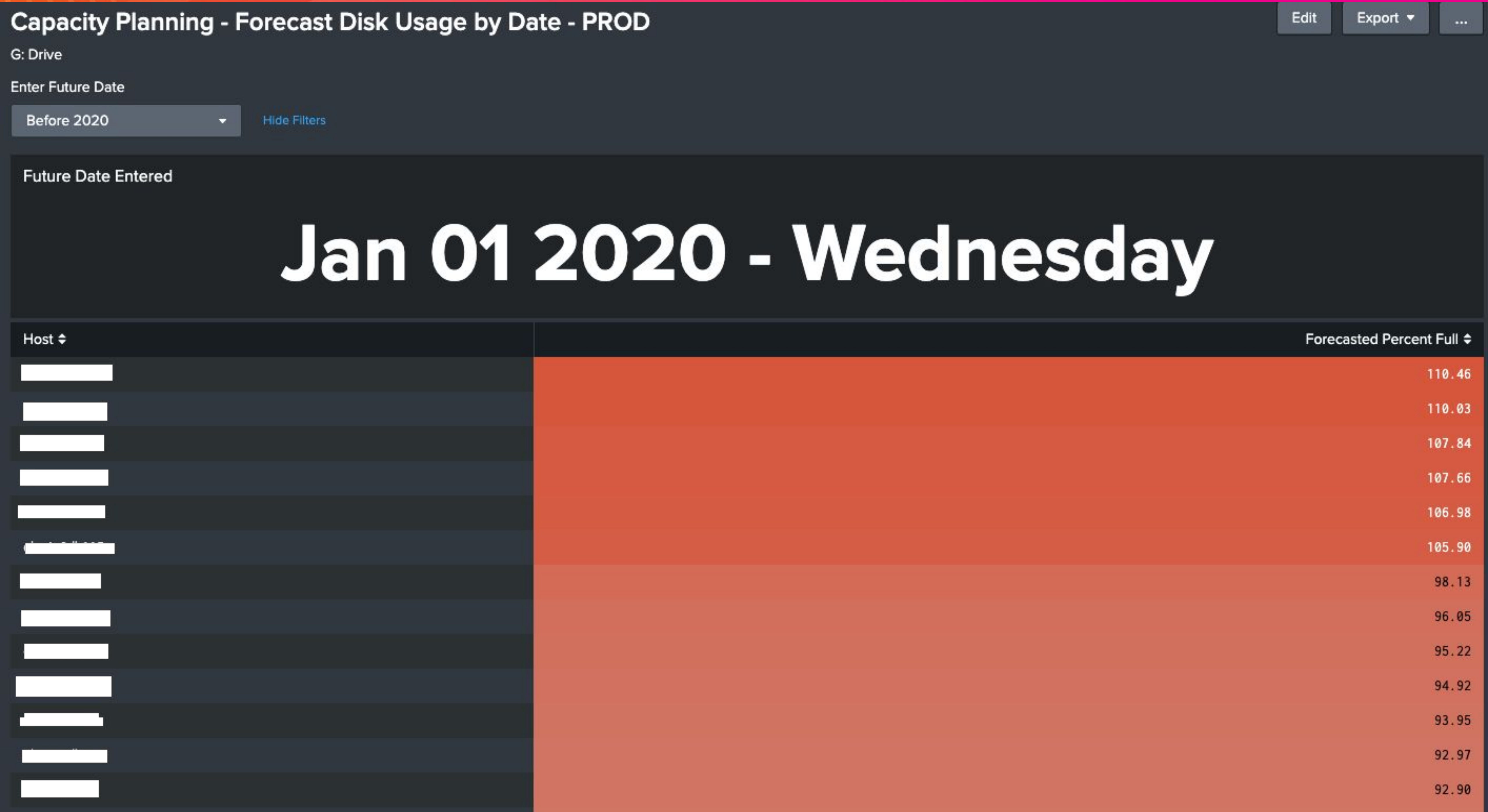
If we remove or add 500GB of space, how long will that last?



Reclaim Storage (GB): -500

Technical Deep Dive

Hands On



Capacity Planning - Forecast Disk Usage by Date - PROD

G: Drive

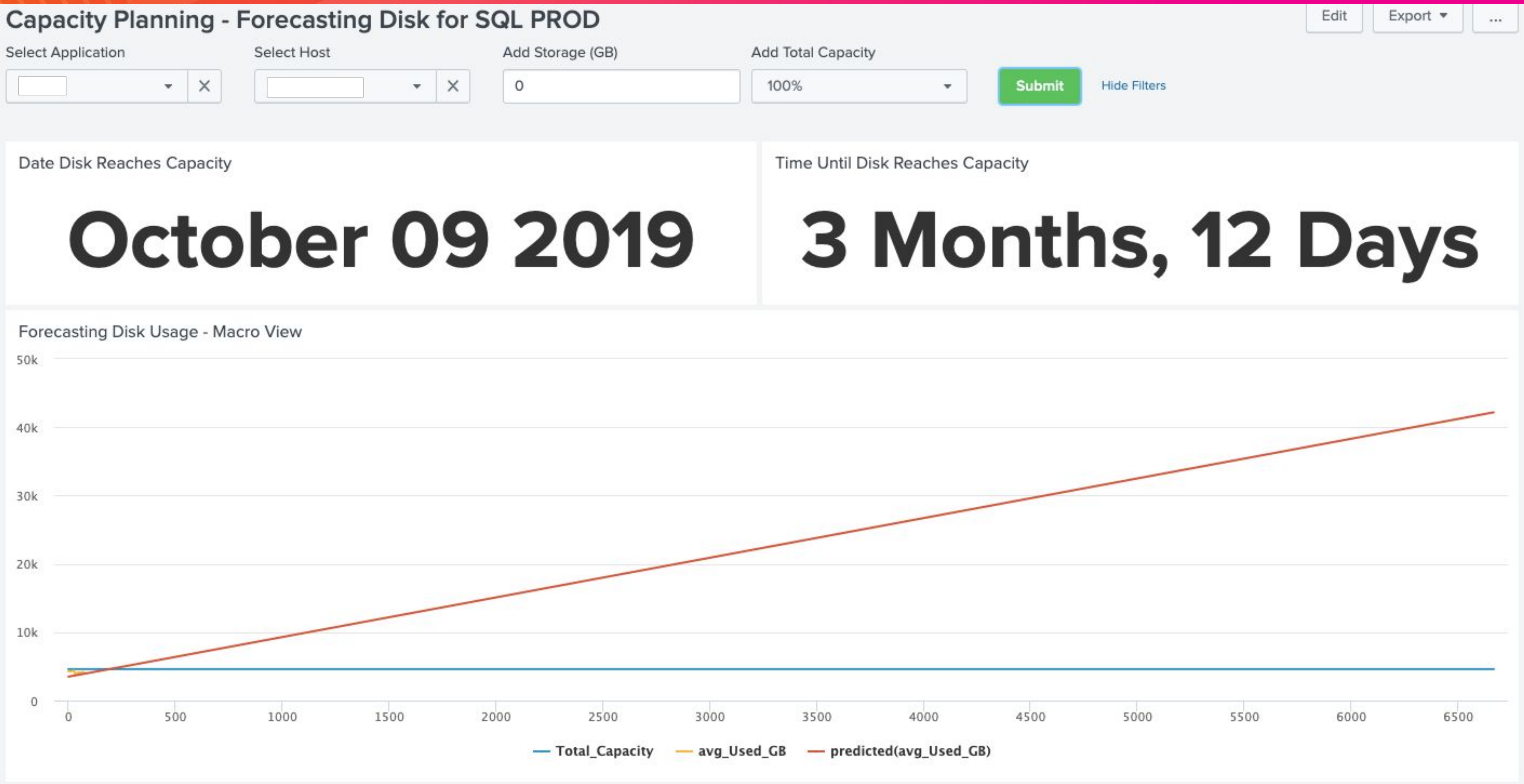
Enter Future Date

Before 2020 ▾

Future Date Entered

Jan 01 2020 - Wednesday

Host ⬍	Forecasted Percent Full ⬍
[redacted]	110.46
[redacted]	110.03
[redacted]	107.84
[redacted]	107.66
[redacted]	106.98
[redacted]	105.90
[redacted]	98.13
[redacted]	96.05
[redacted]	95.22
[redacted]	94.92
[redacted]	93.95
[redacted]	92.97
[redacted]	92.90



Capacity Planning - Forecasting Disk for SQL PROD

Select Application: ▾ ✕

Select Host: ▾ ✕

Add Storage (GB): 0

Add Total Capacity: 100% ▾

Date Disk Reaches Capacity

October 09 2019

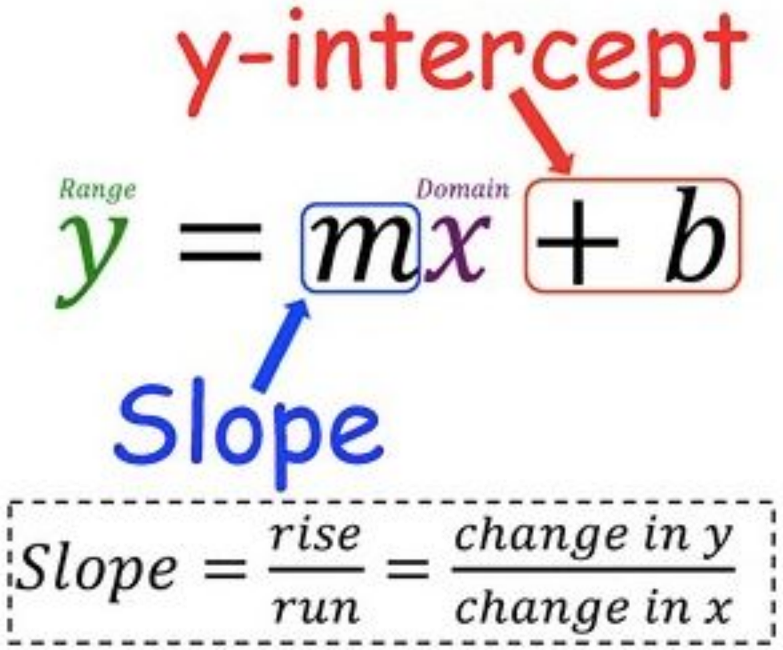
Time Until Disk Reaches Capacity

3 Months, 12 Days

Forecasting Disk Usage - Macro View

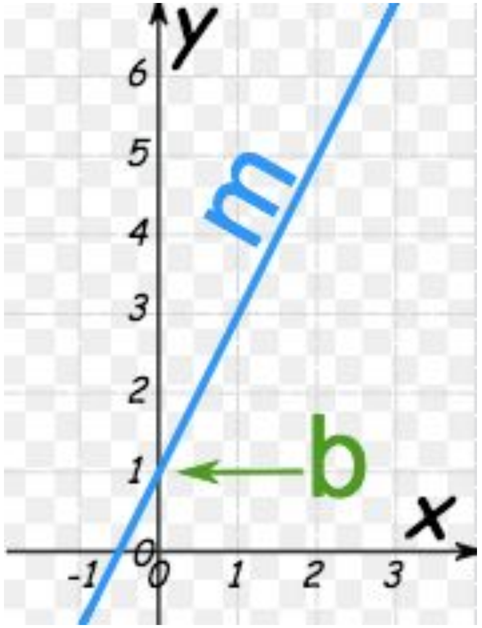
Remember This Formula?

Slope Intercept Form



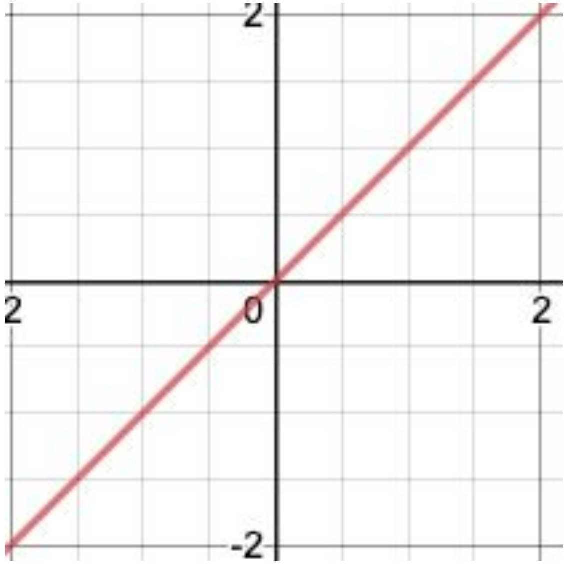
$$y = \boxed{m}x \boxed{+ b}$$

$$Slope = \frac{rise}{run} = \frac{change\ in\ y}{change\ in\ x}$$

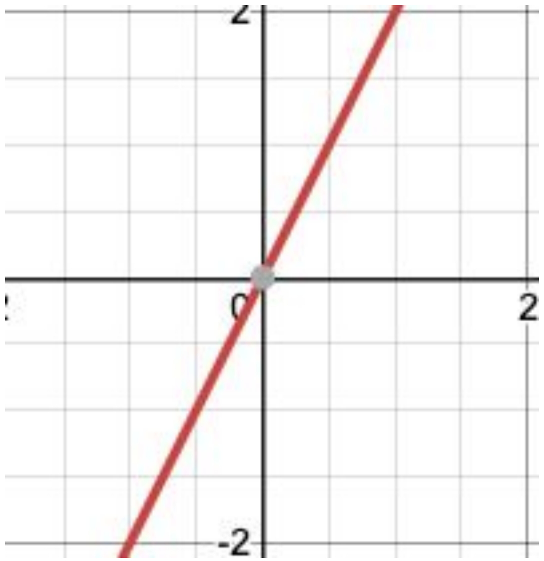


Slope

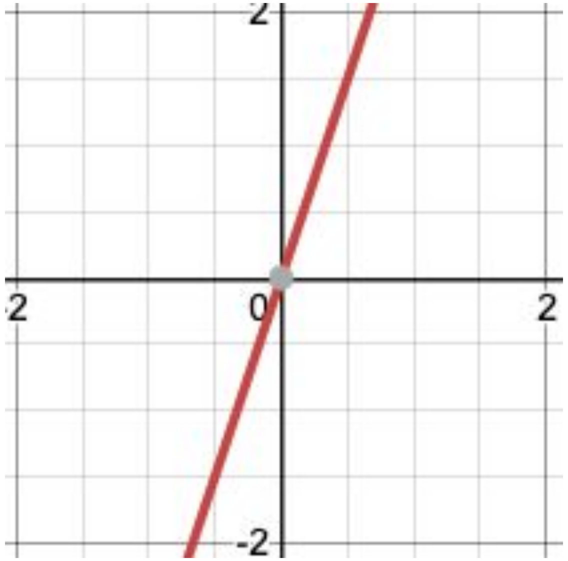
Rise Over Run



$y = 1x + 0$



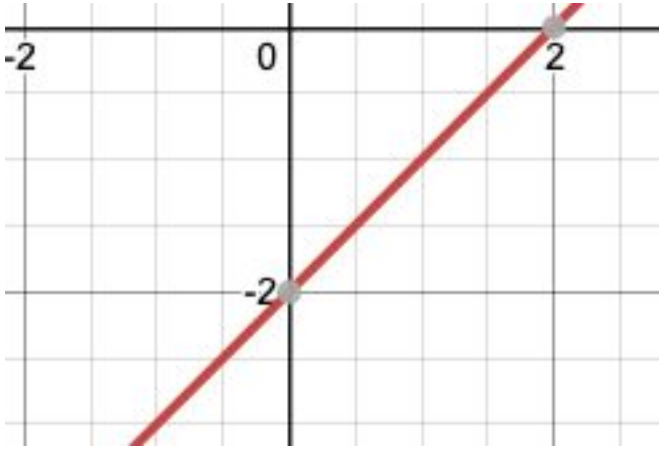
$y = 2x + 0$



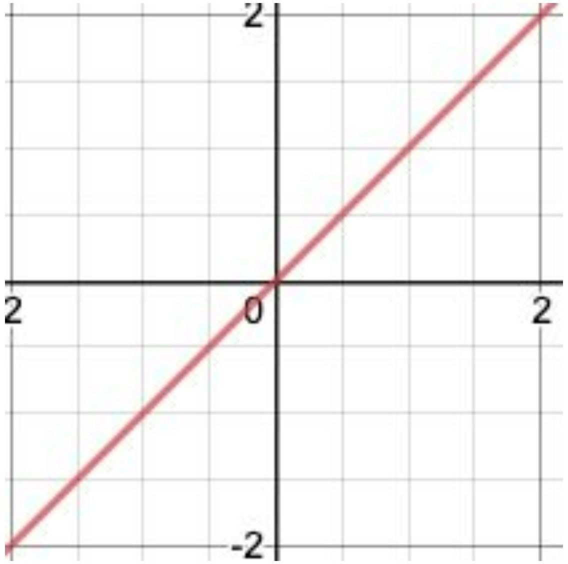
$y = 3x + 0$

Y - Intercept

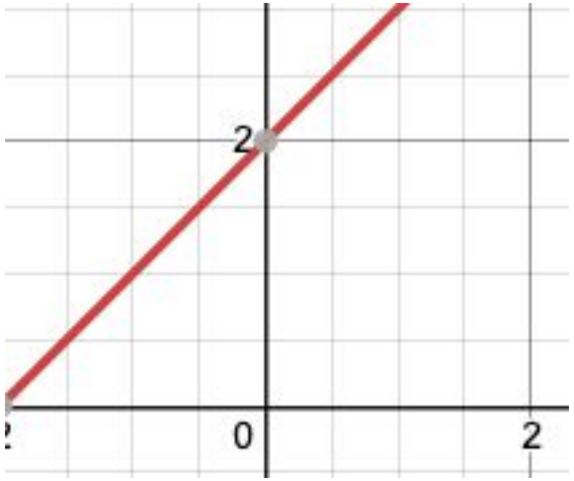
Where it Crosses the Y-Axis



$y = 1x - 2$

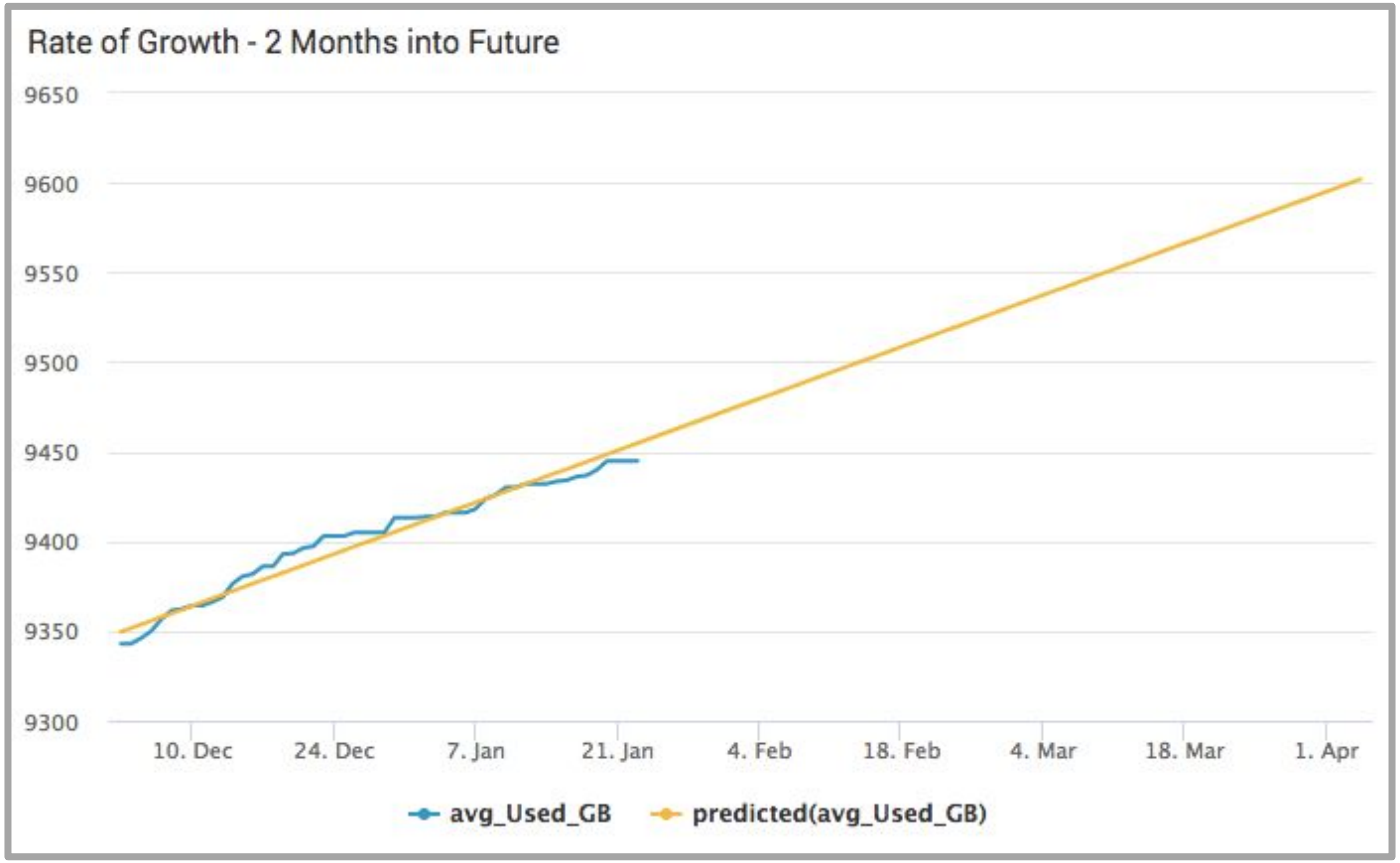


$y = 1x + 0$

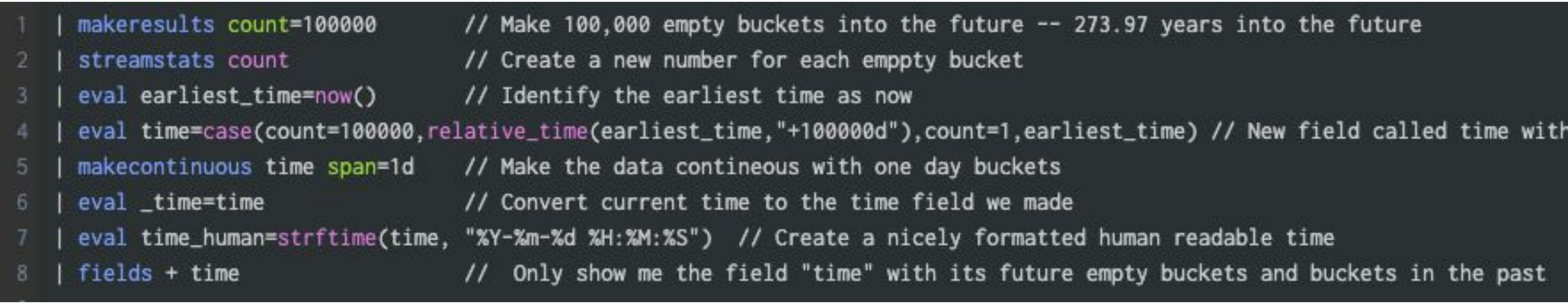


$y = 1x + 2$

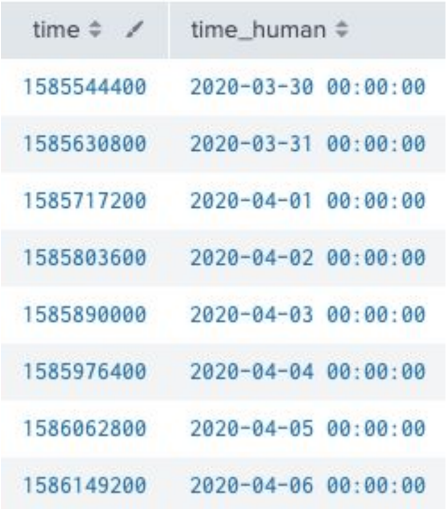
Train The Model



Create Future Empty Buckets



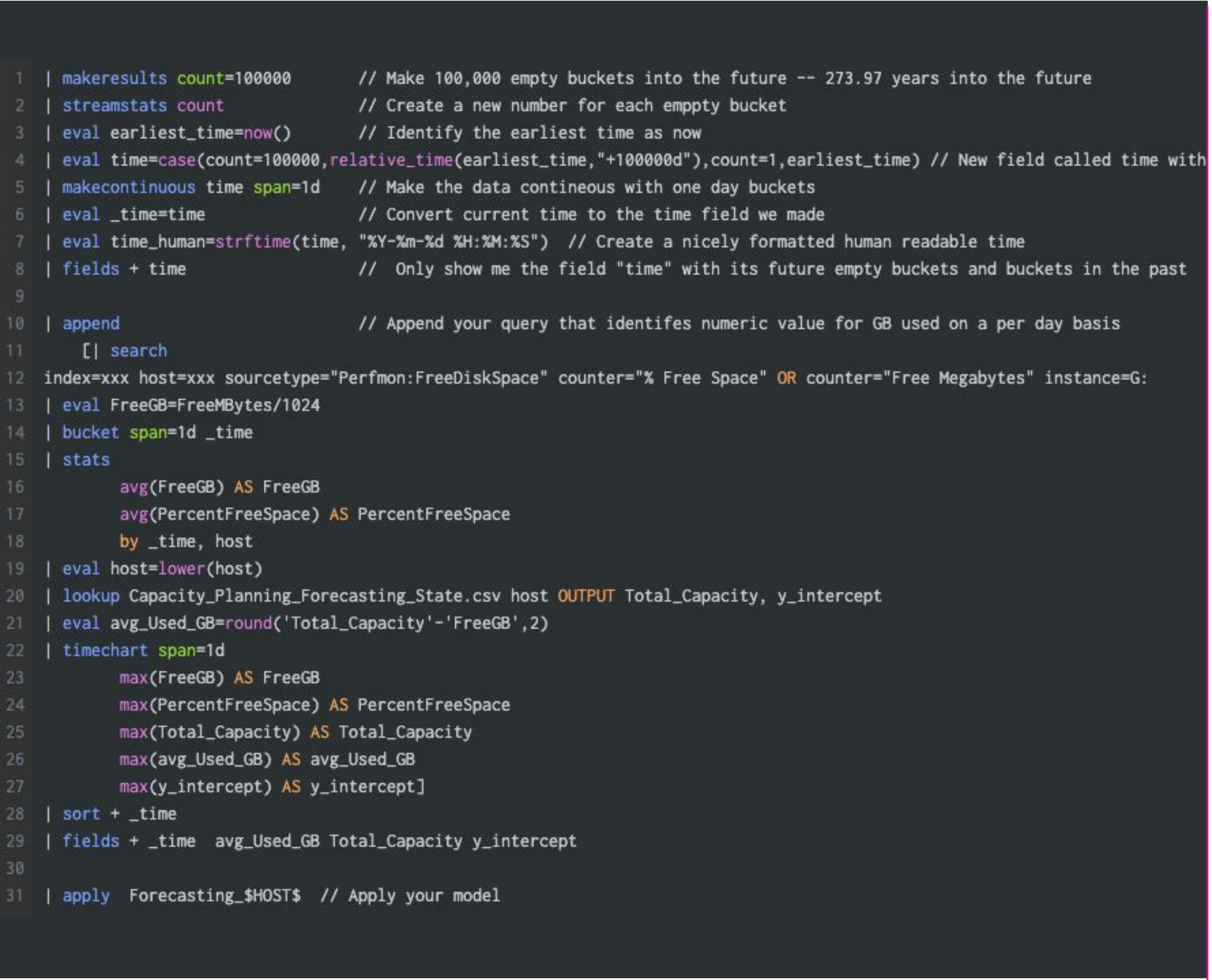
```
| makeresults count=100000       // Make 100,000 empty buckets into the future -- 273.97 years into the future
| streamstats count              // Create a new number for each emppty bucket
| eval earliest_time=now()       // Identify the earliest time as now
| eval time=case(count=100000,relative_time(earliest_time,"+100000d"),count=1,earliest_time) // New field called time wit
| makecontinuous time span=1d    // Make the data contineous with one day buckets
| eval _time=time                // Convert current time to the time field we made
| eval time_human=strftime(time, "%Y-%m-%d %H:%M:%S")  // Create a nicely formatted human readable time
| fields + time                  //  Only show me the field "time" with its future empty buckets and buckets in the past
```



time	time_human
1585544400	2020-03-30 00:00:00
1585630800	2020-03-31 00:00:00
1585717200	2020-04-01 00:00:00
1585803600	2020-04-02 00:00:00
1585890000	2020-04-03 00:00:00
1585976400	2020-04-04 00:00:00
1586062800	2020-04-05 00:00:00
1586149200	2020-04-06 00:00:00

splunk> .conf19

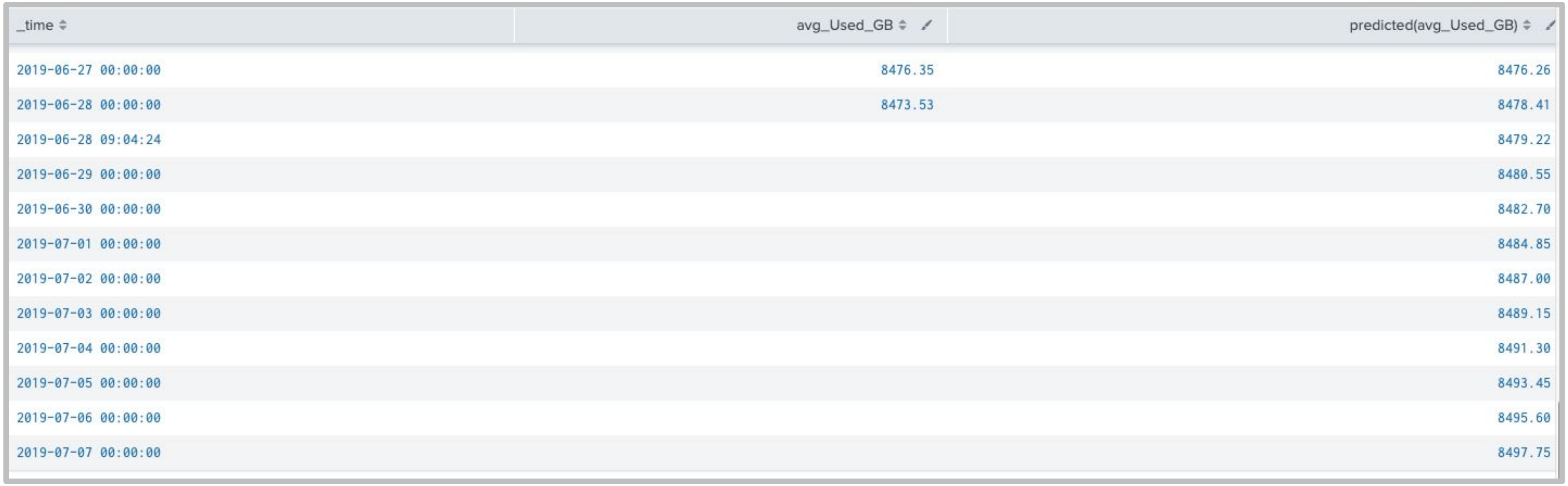
Apply Model onto Future Empty Buckets



```
| makeresults count=100000       // Make 100,000 empty buckets into the future -- 273.97 years into the future
| streamstats count              // Create a new number for each emppty bucket
| eval earliest_time=now()       // Identify the earliest time as now
| eval time=case(count=100000,relative_time(earliest_time,"+100000d"),count=1,earliest_time) // New field called time with
| makecontinuous time span=1d    // Make the data contineous with one day buckets
| eval _time=time                // Convert current time to the time field we made
| eval time_human=strftime(time, "%Y-%m-%d %H:%M:%S")  // Create a nicely formatted human readable time
| fields + time                  //  Only show me the field "time" with its future empty buckets and buckets in the past

| append                         // Append your query that identifes numeric value for GB used on a per day basis
    [| search
index=xxx host=xxx sourcetype="Perfmon:FreeDiskSpace" counter="% Free Space" OR counter="Free Megabytes" instance=G:
| eval FreeGB=FreeMBytes/1024
| bucket span=1d _time
| stats
        avg(FreeGB) AS FreeGB
        avg(PercentFreeSpace) AS PercentFreeSpace
        by _time, host
| eval host=lower(host)
| lookup Capacity_Planning_Forecasting_State.csv host OUTPUT Total_Capacity, y_intercept
| eval avg_Used_GB=round('Total_Capacity'-'FreeGB',2)
| timechart span=1d
        max(FreeGB) AS FreeGB
        max(PercentFreeSpace) AS PercentFreeSpace
        max(Total_Capacity) AS Total_Capacity
        max(avg_Used_GB) AS avg_Used_GB
        max(y_intercept) AS y_intercept]
| sort + _time
| fields + _time  avg_Used_GB Total_Capacity y_intercept

| apply  Forecasting_$HOST$  // Apply your model
```

Forecast Into The Future

Fill those empty buckets using the apply command



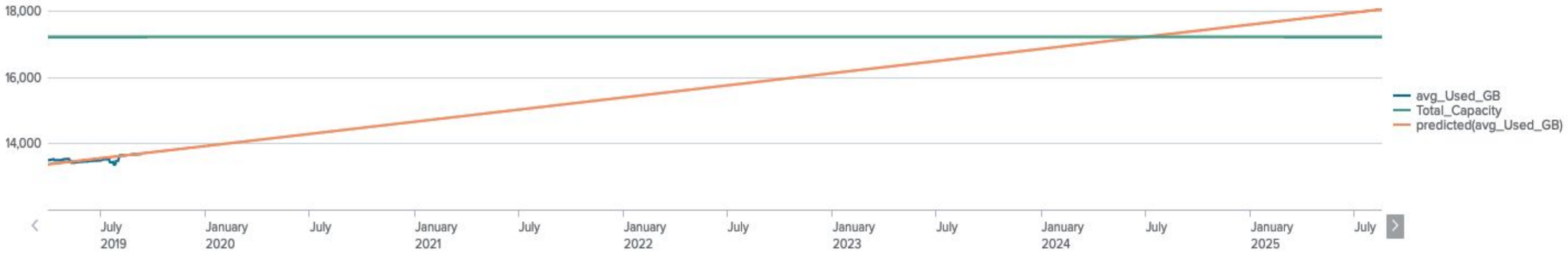
_time	avg_Used_GB	predicted(avg_Used_GB)
2019-06-27 00:00:00	8476.35	8476.26
2019-06-28 00:00:00	8473.53	8478.41
2019-06-28 09:04:24		8479.22
2019-06-29 00:00:00		8480.55
2019-06-30 00:00:00		8482.70
2019-07-01 00:00:00		8484.85
2019-07-02 00:00:00		8487.00
2019-07-03 00:00:00		8489.15
2019-07-04 00:00:00		8491.30
2019-07-05 00:00:00		8493.45
2019-07-06 00:00:00		8495.60
2019-07-07 00:00:00		8497.75

Convert Disk Usage to Time

Find intersection between forecasted disk usage and total capacity

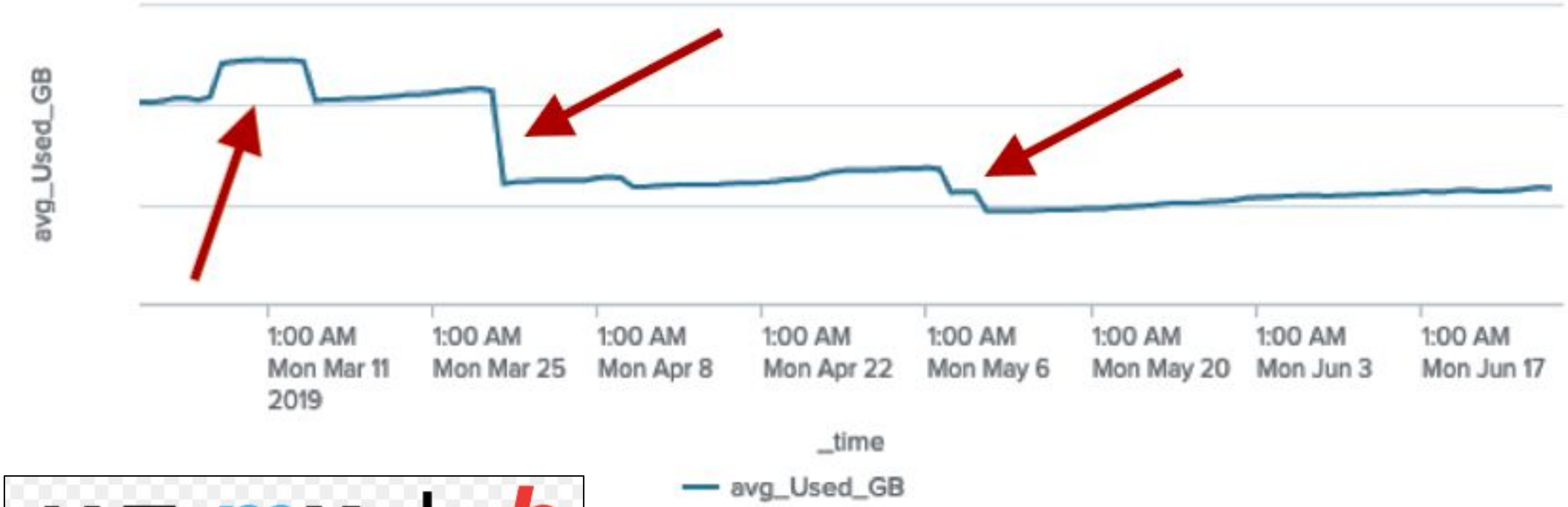
This intersection will represent when forecasted disk reaches capacity Take the _time of that intersection

Convert number of days from “today” to calculate the exact Year, Month, and Day disk will run out



External Forces That Could Skew the Results

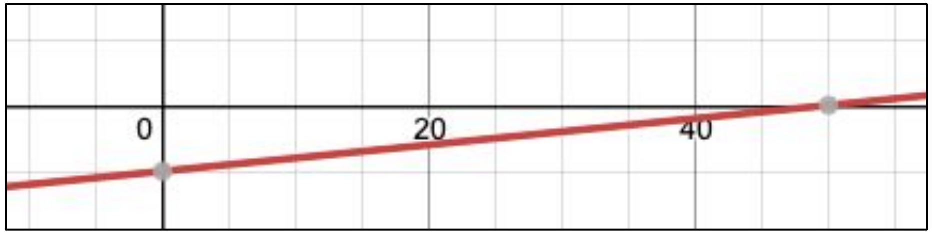
Historical Disk Usage

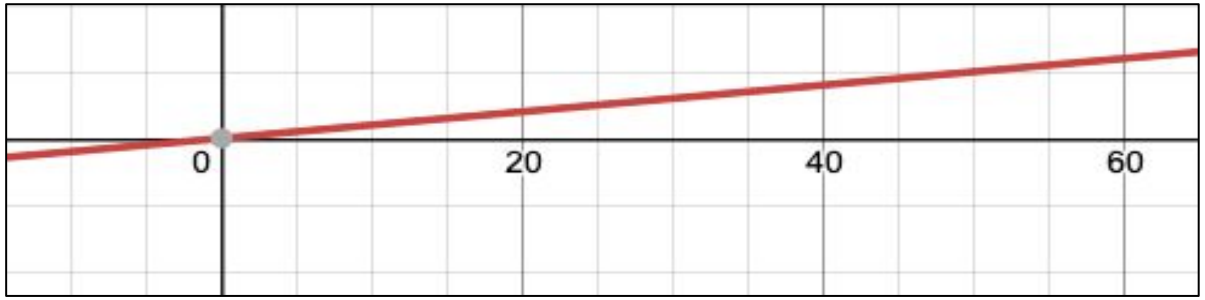


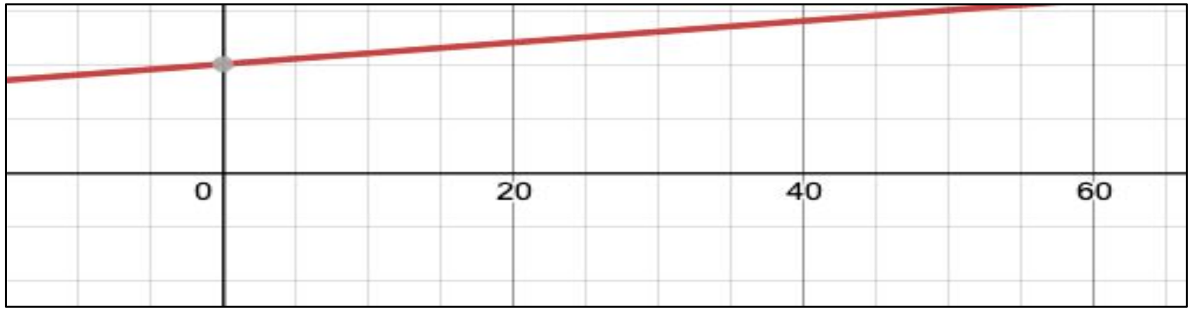
$$y = mx + b$$

Y – Intercept

What is This and Why Do We Care?

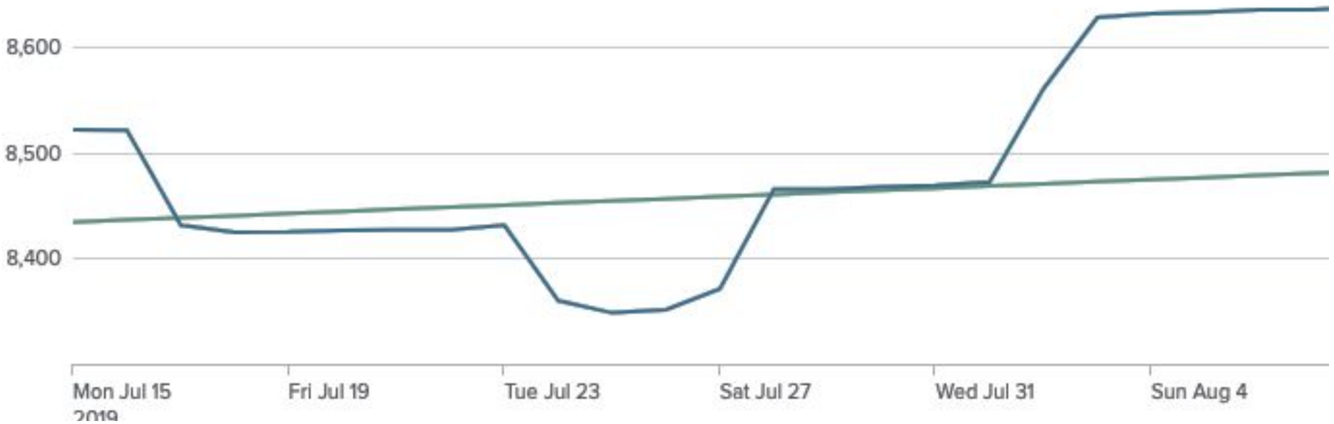




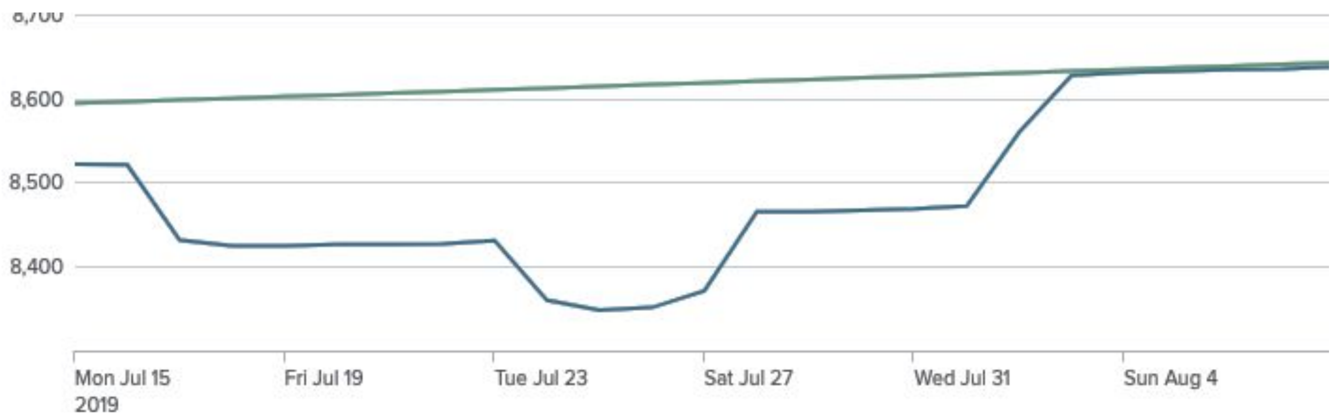


Scheduled Report to Adjust the Y-Intercept

Before



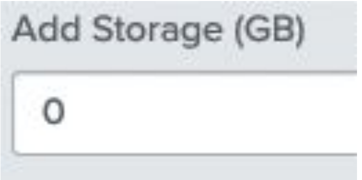
After



With auto-adjusting the Y-Intercept, the predicted values are now using the latest Y-Intercept to forecast.

What If Machine

Current Situation
without changing total capacity



Add Storage (GB)

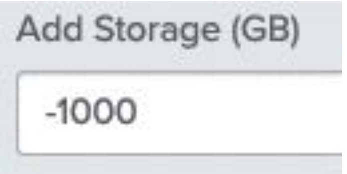
0

January 03

2032

12 Years, 3 Months, 25 Days

If we ***Reclaim 1TB of Capacity***
what will that look like?



Add Storage (GB)

-1000

August 21,

2030

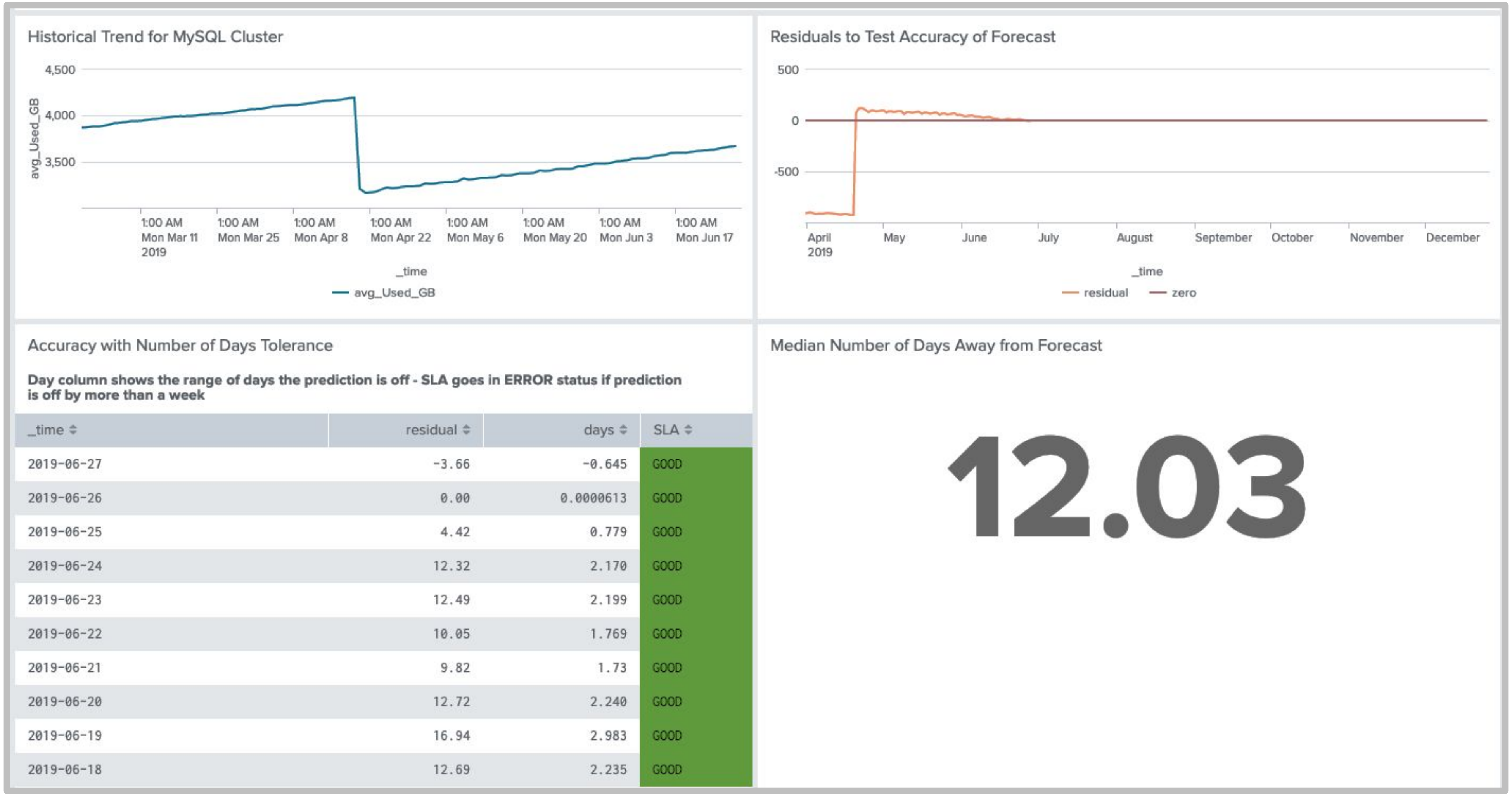
10 Years, 11 Months, 15 Days

Build a Scoring Mechanism

Shorten the Feedback Loops

splunk> .conf19

Build a Scoring Mechanism to Test Accuracy



Historical Trend for MySQL Cluster

Residuals to Test Accuracy of Forecast

Accuracy with Number of Days Tolerance

Day column shows the range of days the prediction is off - SLA goes in ERROR status if prediction is off by more than a week

_time ⇕	residual ⇕	days ⇕	SLA ⇕
2019-06-27	-3.66	-0.645	GOOD
2019-06-26	0.00	0.0000613	GOOD
2019-06-25	4.42	0.779	GOOD
2019-06-24	12.32	2.170	GOOD
2019-06-23	12.49	2.199	GOOD
2019-06-22	10.05	1.769	GOOD
2019-06-21	9.82	1.73	GOOD
2019-06-20	12.72	2.240	GOOD
2019-06-19	16.94	2.983	GOOD
2019-06-18	12.69	2.235	GOOD

Median Number of Days Away from Forecast

12.03

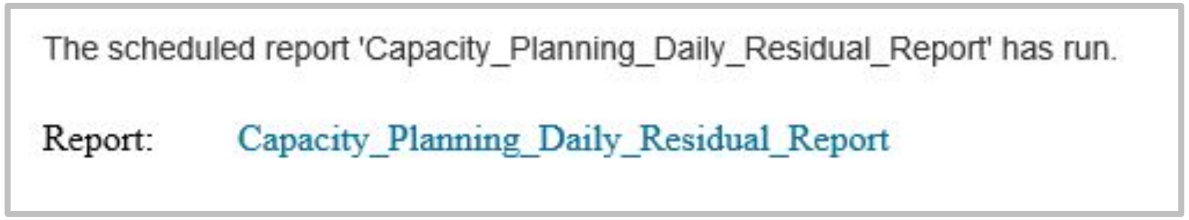
Scaling This Out

Monitor Accuracy with Scheduled Reports
Efficiency Techniques Used
Common Challenges
Overfitting the Data

Scale your ML Project

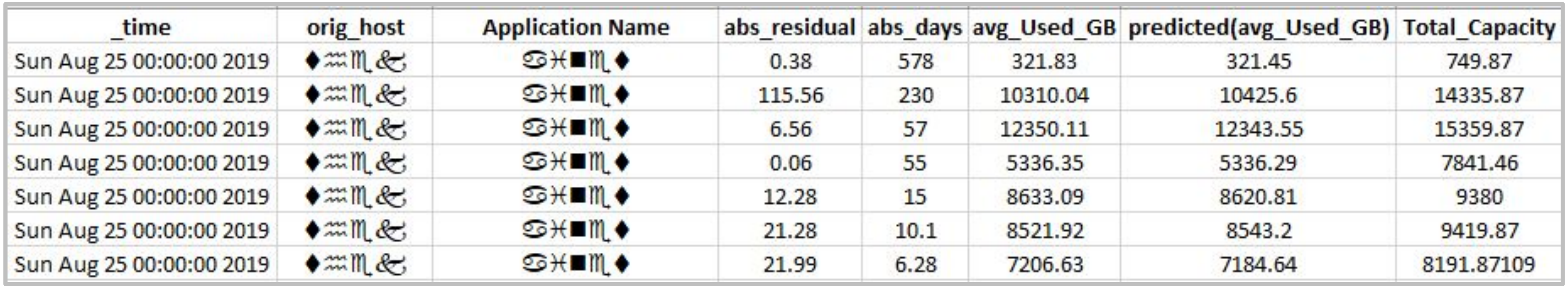
Scaling Touch-Points

Schedule **Daily** Residual Reports to ensure accuracy



The scheduled report 'Capacity_Planning_Daily_Residual_Report' has run.

Report: Capacity_Planning_Daily_Residual_Report

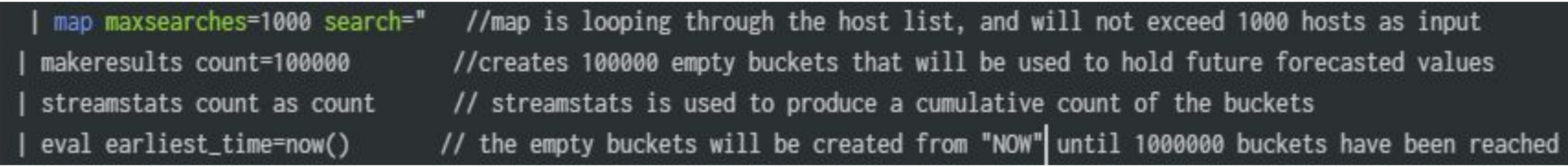


_time	orig_host	Application Name	abs_residual	abs_days	avg_Used_GB	predicted(avg_Used_GB)	Total_Capacity
Sun Aug 25 00:00:00 2019	♦♒♏&	♋♓■♏♦	0.38	578	321.83	321.45	749.87
Sun Aug 25 00:00:00 2019	♦♒♏&	♋♓■♏♦	115.56	230	10310.04	10425.6	14335.87
Sun Aug 25 00:00:00 2019	♦♒♏&	♋♓■♏♦	6.56	57	12350.11	12343.55	15359.87
Sun Aug 25 00:00:00 2019	♦♒♏&	♋♓■♏♦	0.06	55	5336.35	5336.29	7841.46
Sun Aug 25 00:00:00 2019	♦♒♏&	♋♓■♏♦	12.28	15	8633.09	8620.81	9380
Sun Aug 25 00:00:00 2019	♦♒♏&	♋♓■♏♦	21.28	10.1	8521.92	8543.2	9419.87
Sun Aug 25 00:00:00 2019	♦♒♏&	♋♓■♏♦	21.99	6.28	7206.63	7184.64	8191.87109

Scale your ML Project

Scaling Touch-Points

Get familiar with the map command



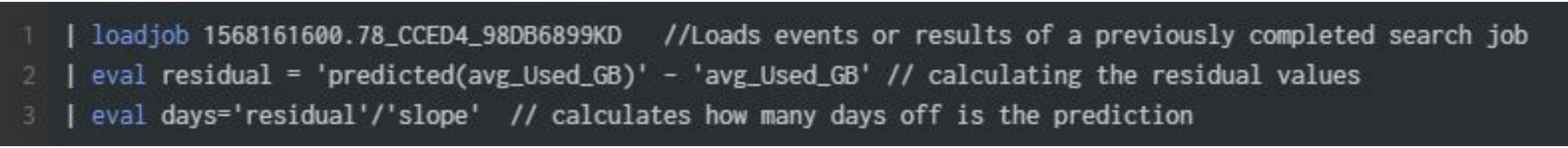
```
 | map maxsearches=1000 search="   //map is looping through the host list, and will not exceed 1000 hosts as input
| makeresults count=100000         //creates 100000 empty buckets that will be used to hold future forecasted values
| streamstats count as count       // streamstats is used to produce a cumulative count of the buckets
| eval earliest_time=now()        // the empty buckets will be created from "NOW" until 1000000 buckets have been reached
```

Efficiency Techniques

Train during a change freeze

Use | loadjob for post-processing

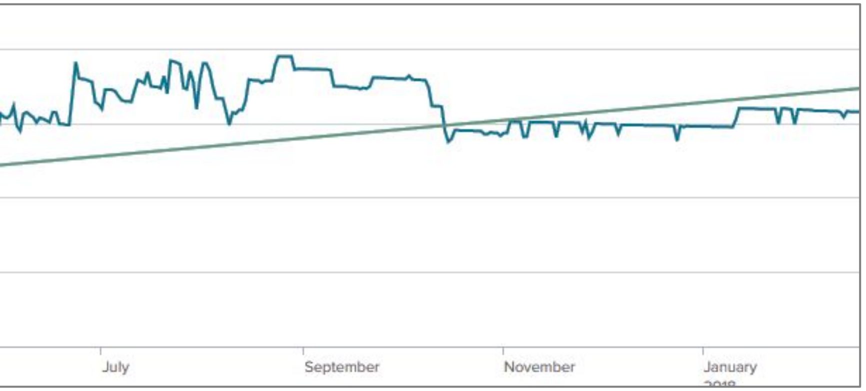
Summary Indexing to spread out the load

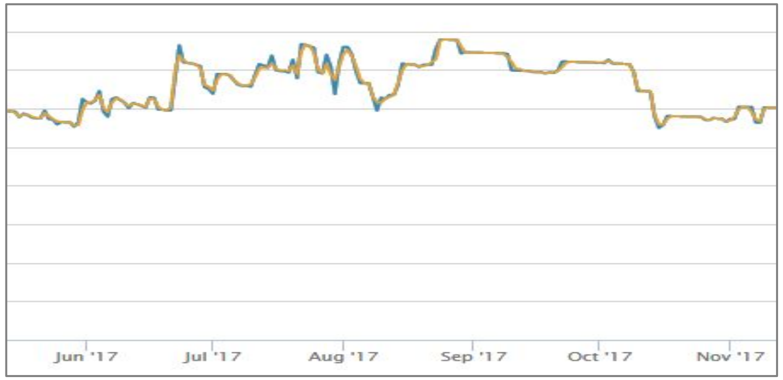


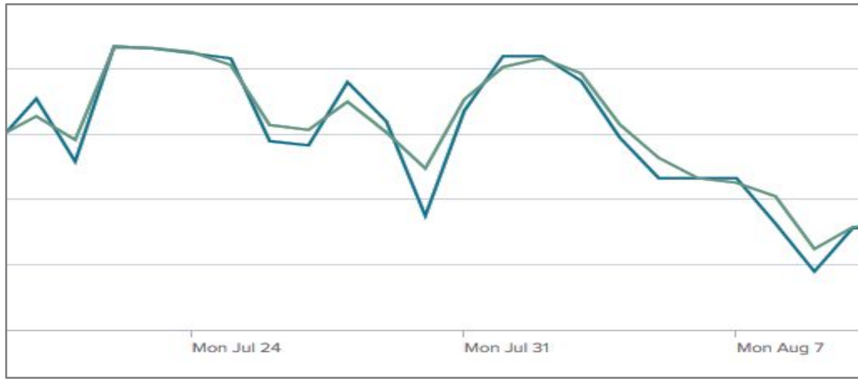
```
| loadjob 1568161600.78_CCED4_98DB6899KD   //Loads events or results of a previously completed search job
| eval residual = 'predicted(avg_Used_GB)' - 'avg_Used_GB' // calculating the residual values
| eval days='residual'/'slope'  // calculates how many days off is the prediction
```

Challenges

- **25 failed prototype dashboards** before coming to the conclusion of using the linear regression
- Data didn’t express **complete** linearity..
- Searches took longer than expected to run

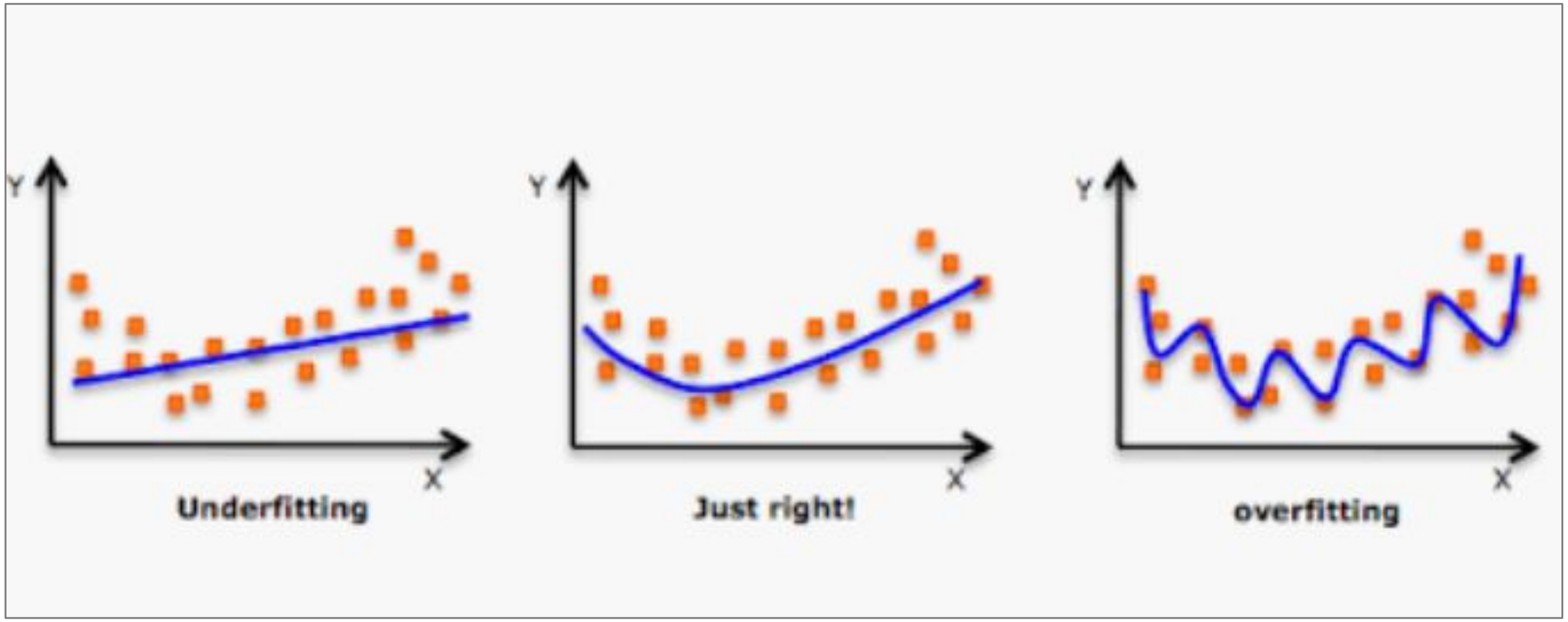






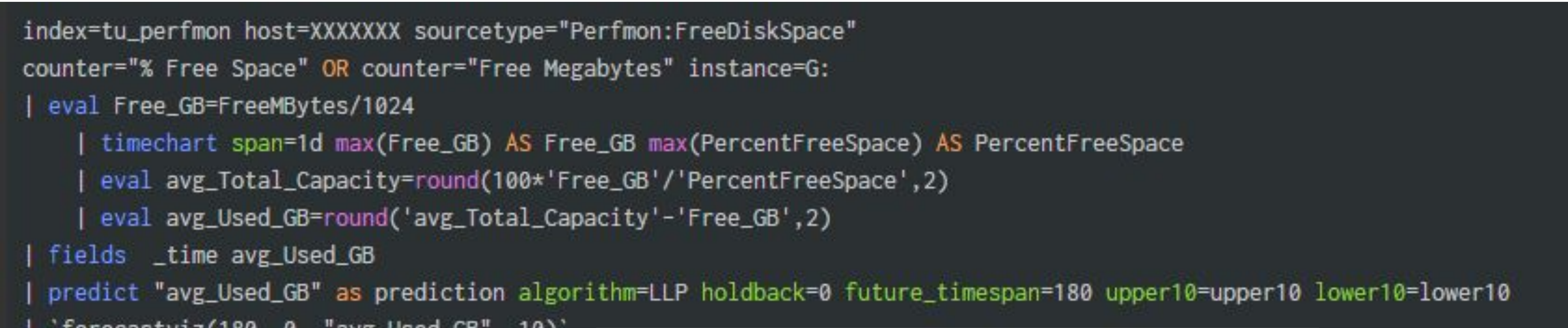
Overfitting

1. Using the predict command resulted in a really good prediction…almost too good to be true.
2. The best solution to an overfitting problem is avoidance.
3. Identify overfitting, then find ways to tackle overfitting and learn from the mistake.

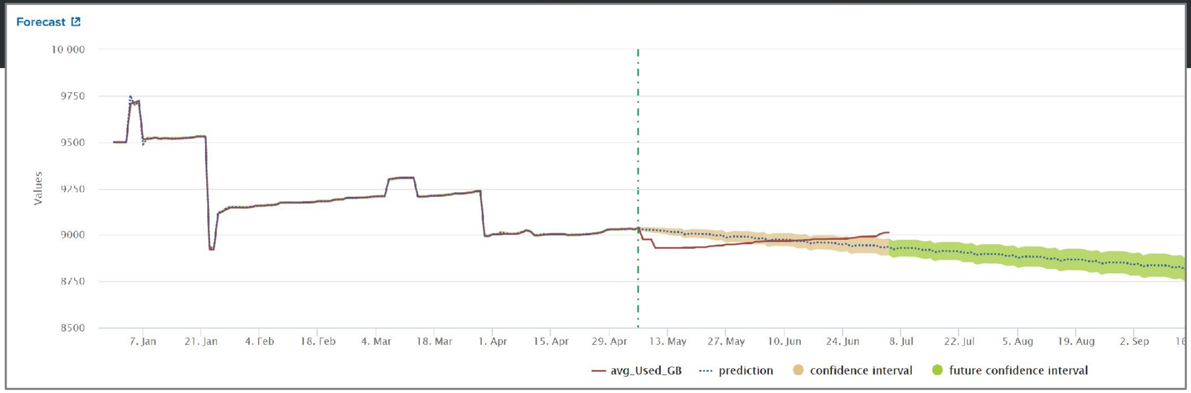


Predict Command

KALMAN FILTERING – when using the predict command we were actually overfitting the data



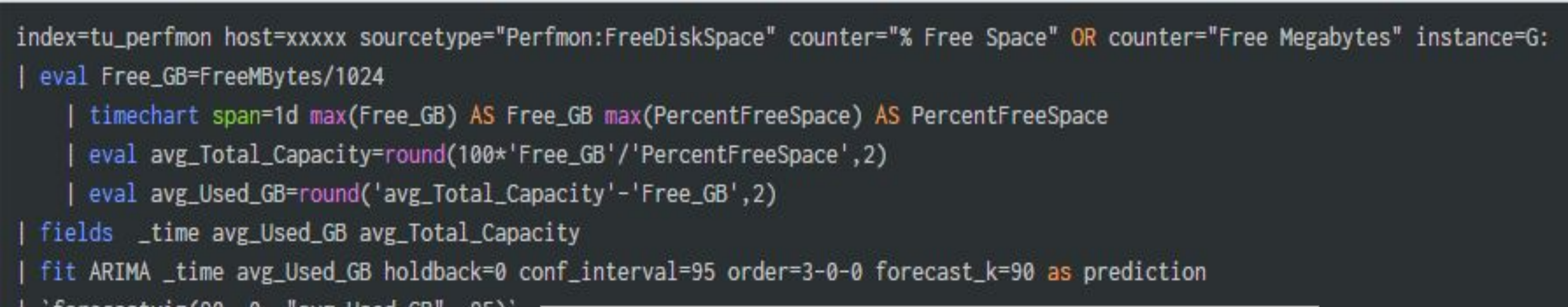
```
index=tu_perfmon host=XXXXXXX sourcetype="Perfmon:FreeDiskSpace"
counter="% Free Space" OR counter="Free Megabytes" instance=G:
| eval Free_GB=FreeMBytes/1024
    | timechart span=1d max(Free_GB) AS Free_GB max(PercentFreeSpace) AS PercentFreeSpace
    | eval avg_Total_Capacity=round(100*'Free_GB'/'PercentFreeSpace',2)
    | eval avg_Used_GB=round('avg_Total_Capacity'-'Free_GB',2)
| fields  _time avg_Used_GB
| predict "avg_Used_GB" as prediction algorithm=LLP holdback=0 future_timespan=180 upper10=upper10 lower10=lower10
| `forecastviz(180, 0, "avg_Used_GB", 10)`
```



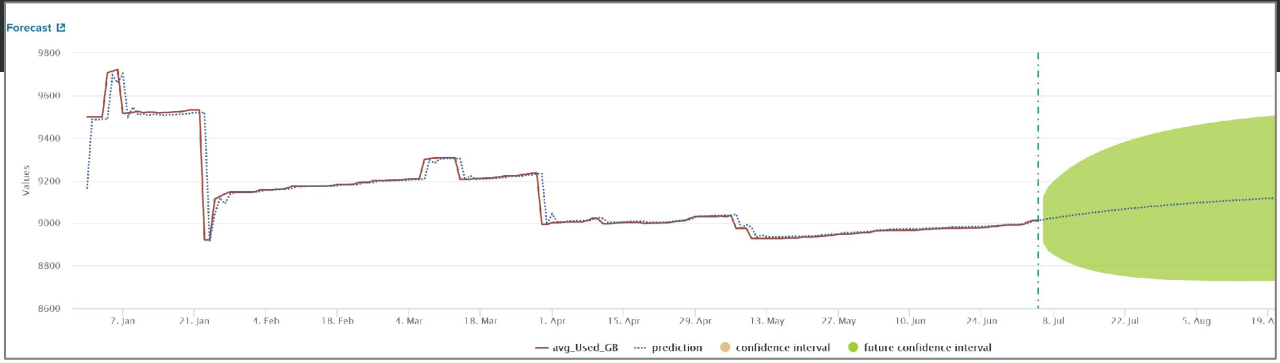
splunk> .conf19

ARIMA

ARIMA – gave large confidence interval (we want a smaller interval in order to trust the accuracy of the forecast)



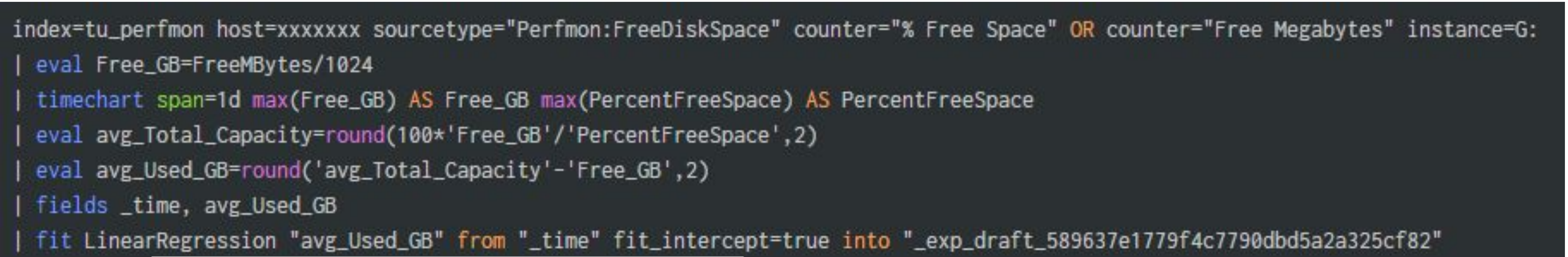
```
index=tu_perfmon host=xxxxx sourcetype="Perfmon:FreeDiskSpace" counter="% Free Space" OR counter="Free Megabytes" instance=G:
| eval Free_GB=FreeMBytes/1024
    | timechart span=1d max(Free_GB) AS Free_GB max(PercentFreeSpace) AS PercentFreeSpace
    | eval avg_Total_Capacity=round(100*'Free_GB'/'PercentFreeSpace',2)
    | eval avg_Used_GB=round('avg_Total_Capacity'-'Free_GB',2)
| fields  _time avg_Used_GB avg_Total_Capacity
| fit ARIMA _time avg_Used_GB holdback=0 conf_interval=95 order=3-0-0 forecast_k=90 as prediction
| `forecastviz(90, 0, "avg_Used_GB", 95)`
```



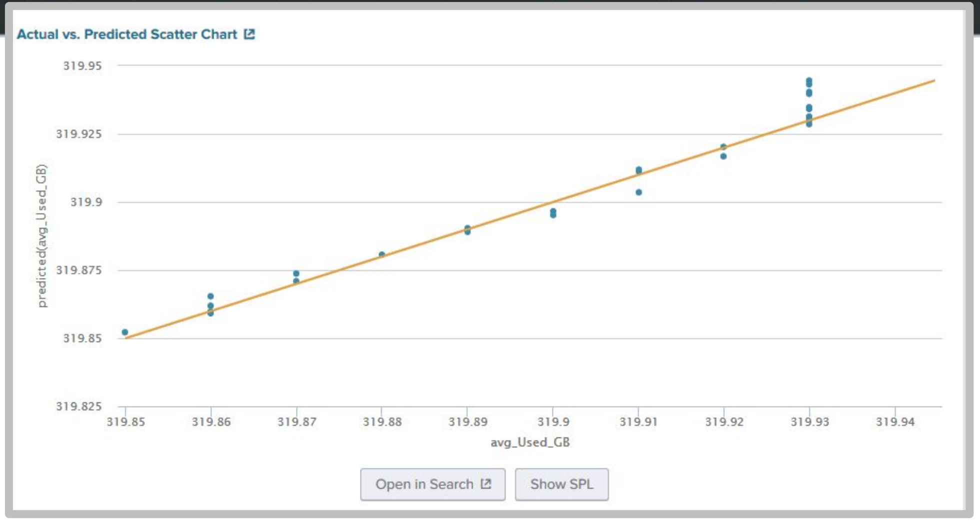
splunk> .conf19

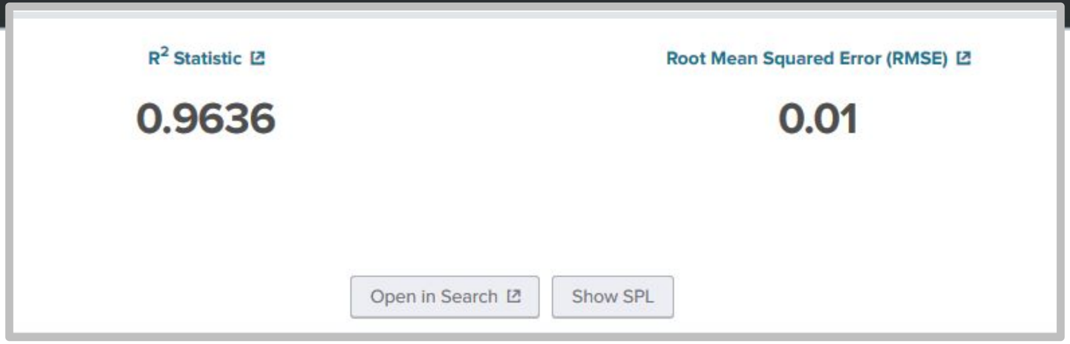
Linear Regression

Linear Regression- showed incredible accuracy and didn’t over fit the data like the predict command



```
index=tu_perfmon host=xxxxxxx sourcetype="Perfmon:FreeDiskSpace" counter="% Free Space" OR counter="Free Megabytes" instance=G:
| eval Free_GB=FreeMBytes/1024
| timechart span=1d max(Free_GB) AS Free_GB max(PercentFreeSpace) AS PercentFreeSpace
| eval avg_Total_Capacity=round(100*'Free_GB'/'PercentFreeSpace',2)
| eval avg_Used_GB=round('avg_Total_Capacity'-'Free_GB',2)
| fields _time, avg_Used_GB
| fit LinearRegression "avg_Used_GB" from "_time" fit_intercept=true into "_exp_draft_589637e1779f4c7790dbd5a2a325cf82"
```





R^2 Statistic

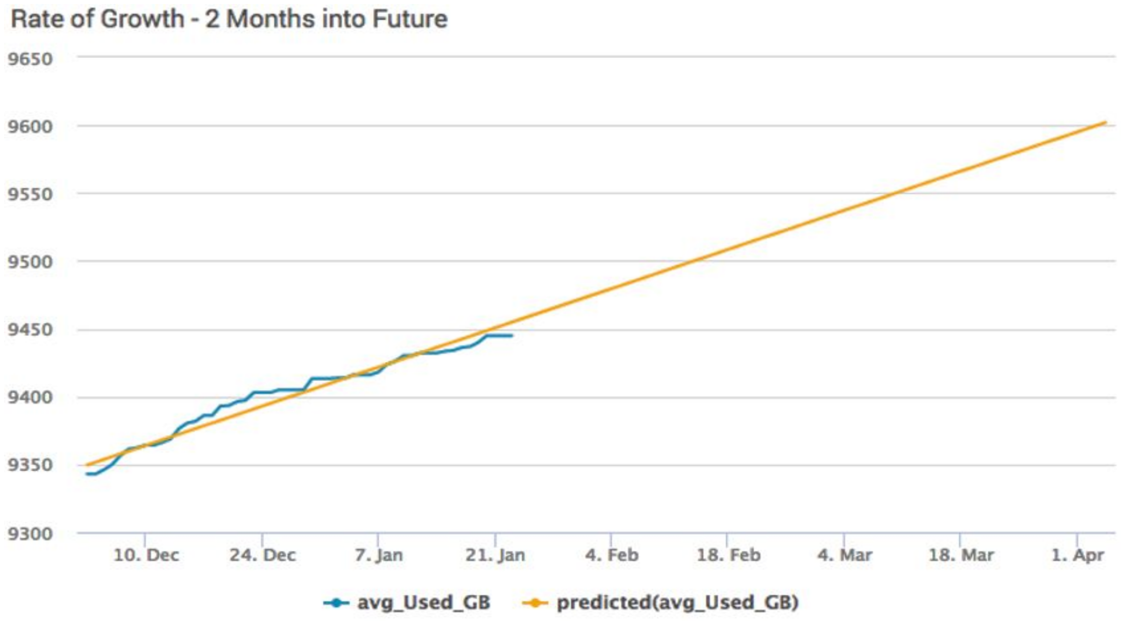
0.9636

Root Mean Squared Error (RMSE)

0.01

What This *Will* Do

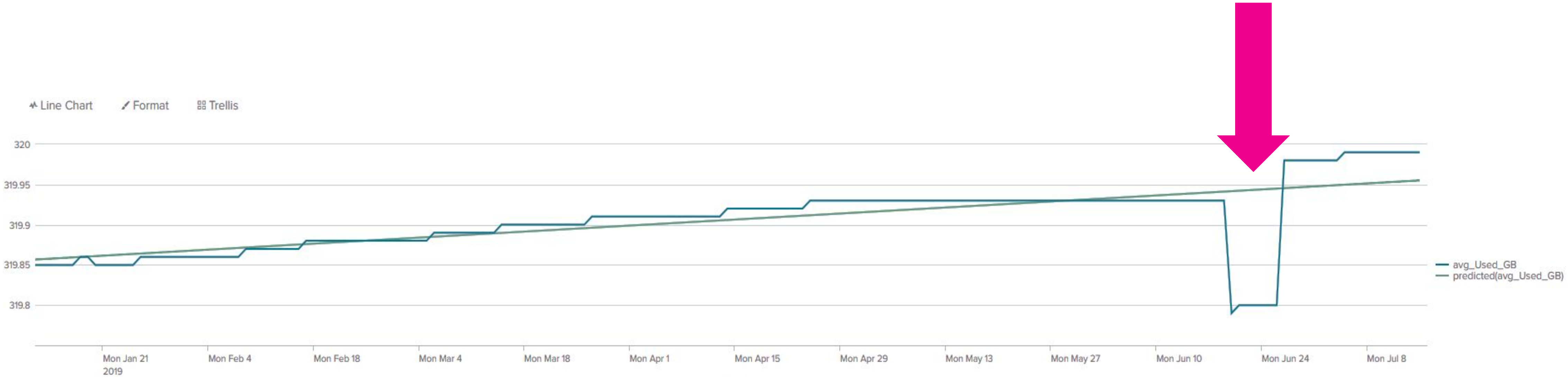
1. Assist with Prevention of Outages, and Budgeting
2. Reduce time when addressing capacity concerns
3. Allow for Re-usability within the organization



splunk> .conf19

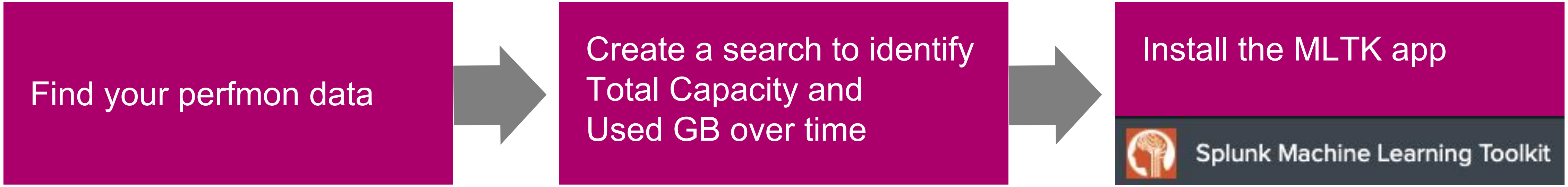
What This Will *NOT* Do

1. Business logic needs to be accounted for when looking at the visualization.
2. This will not account for external changes in growth that do not have an established pattern



Walk-through

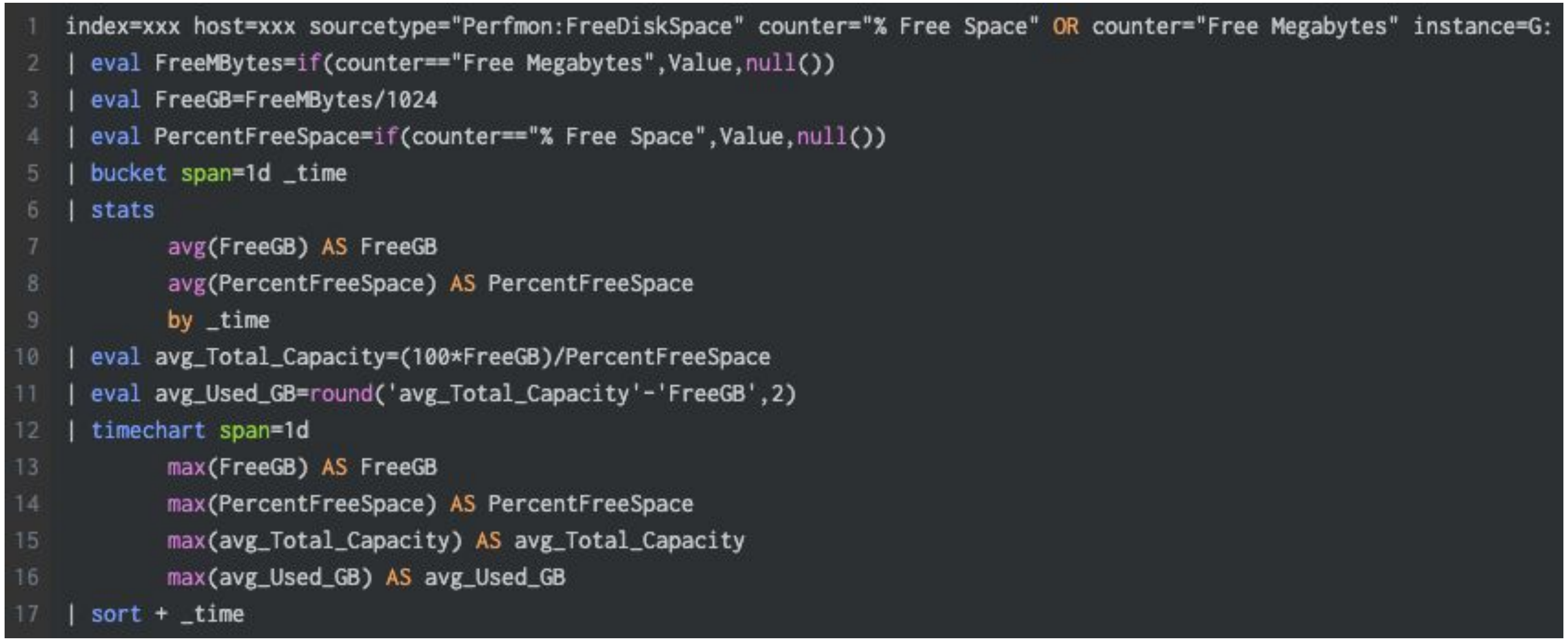
Where to Start?



Find your perfmon data

Create a search to identify Total Capacity and Used GB over time

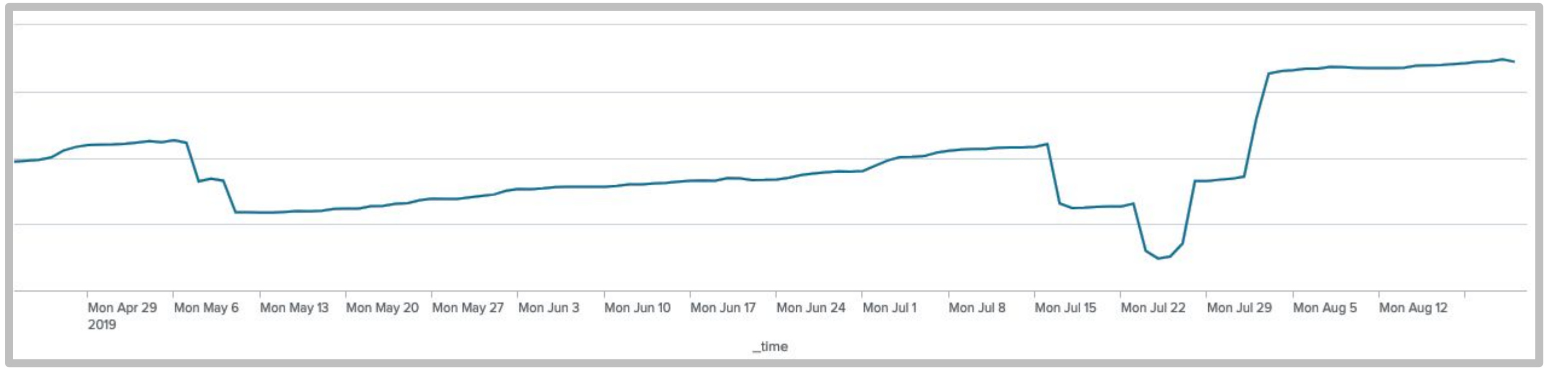
Install the MLTK app

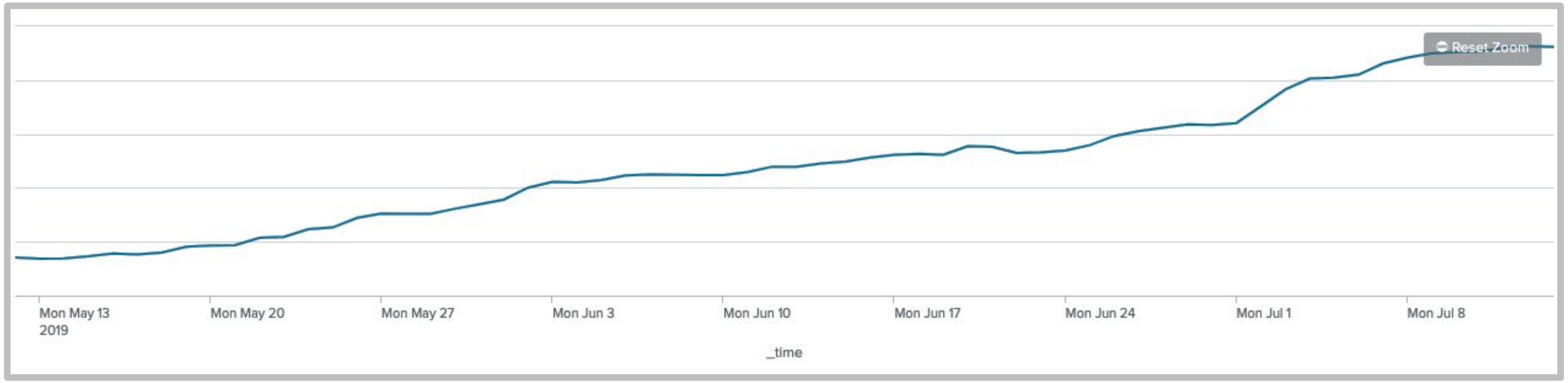


```
index=xxx host=xxx sourcetype="Perfmon:FreeDiskSpace" counter="% Free Space" OR counter="Free Megabytes" instance=G:
| eval FreeMBytes=if(counter=="Free Megabytes",Value,null())
| eval FreeGB=FreeMBytes/1024
| eval PercentFreeSpace=if(counter=="% Free Space",Value,null())
| bucket span=1d _time
| stats
    avg(FreeGB) AS FreeGB
    avg(PercentFreeSpace) AS PercentFreeSpace
    by _time
| eval avg_Total_Capacity=(100*FreeGB)/PercentFreeSpace
| eval avg_Used_GB=round('avg_Total_Capacity'-'FreeGB',2)
| timechart span=1d
    max(FreeGB) AS FreeGB
    max(PercentFreeSpace) AS PercentFreeSpace
    max(avg_Total_Capacity) AS avg_Total_Capacity
    max(avg_Used_GB) AS avg_Used_GB
| sort + _time
```

Find Target Host

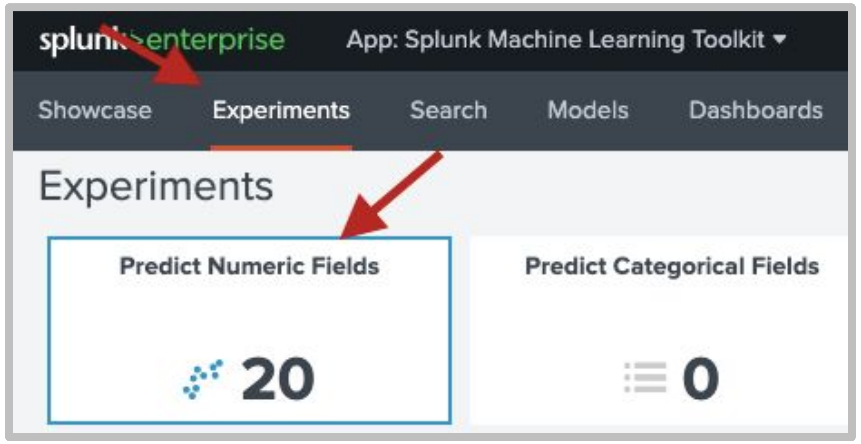
1. Identify a single server and instance to test against
2. Validate that your growth has a good constant linear slope. Ex.(find a time during a change freeze)

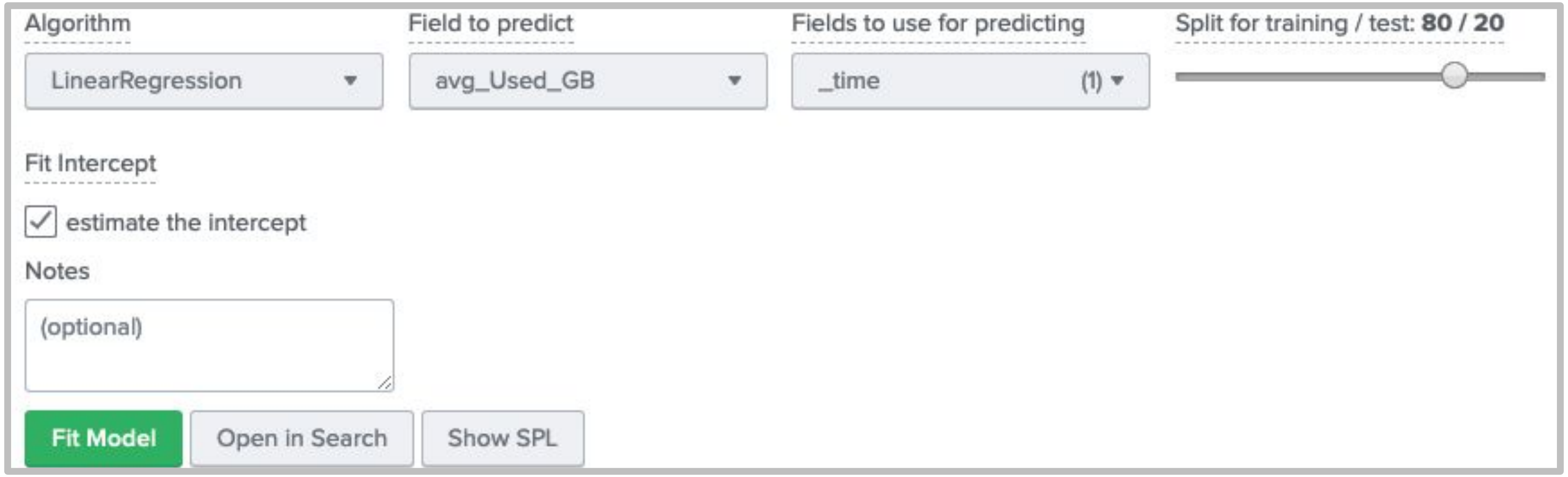




Using the MLTK

1. Navigate to the MLTK app > Experiments > Predict Numerical Fields
2. Enter your SPL into the search
3. Select "Linear Regression" as the algorithm
4. Select field to predict and single _time feature





Algorithm: LinearRegression ▾

Field to predict: avg_Used_GB ▾

Fields to use for predicting: _time (1) ▾

Split for training / test: **80 / 20**

Fit Intercept

☑ estimate the intercept

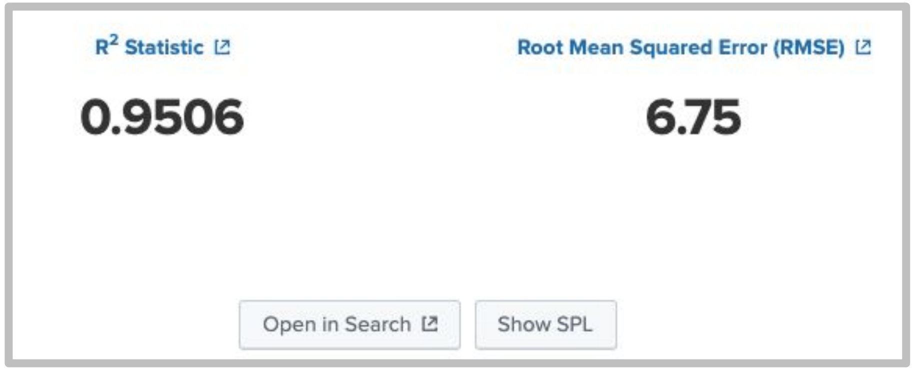
Notes

(optional)

Interpret Results

R^2 > how much the estimates deviate from the actual values in the data set on average.

- * Better Models have R^2 closer to 1.



R^2 Statistic

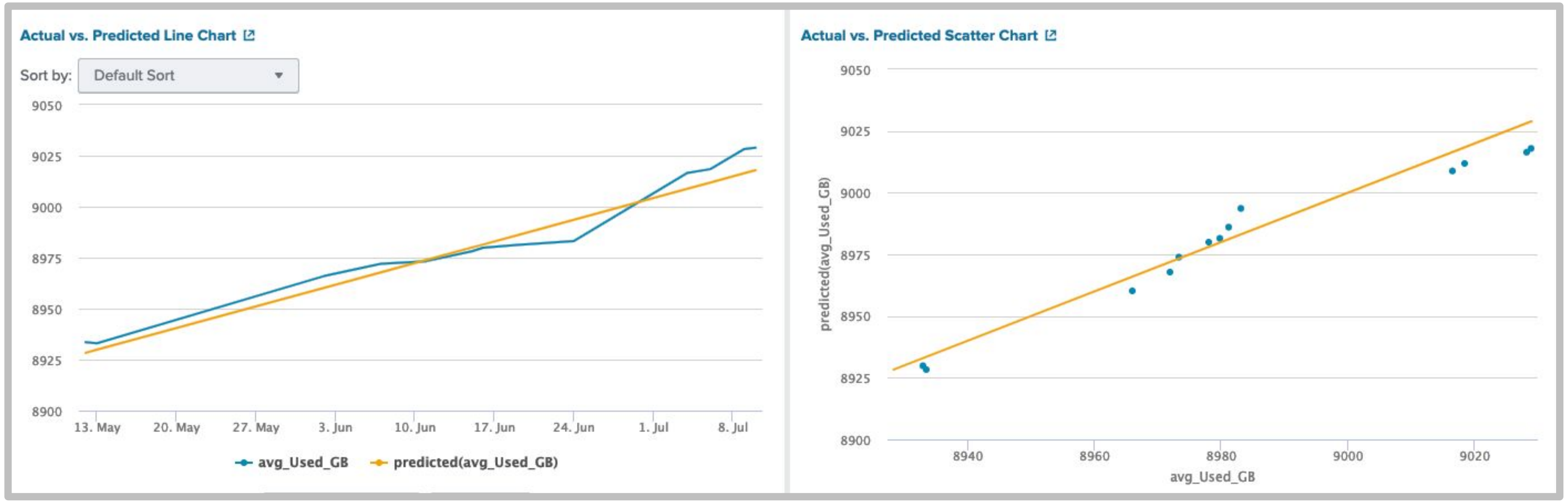
0.9506

Root Mean Squared Error (RMSE)

6.75

RMSE > indicates how close the observed data points are to the model's predicted values.

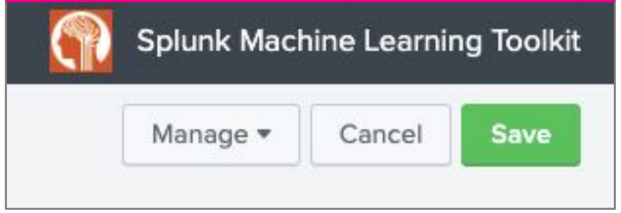
- * Better models have a smaller RMSE.



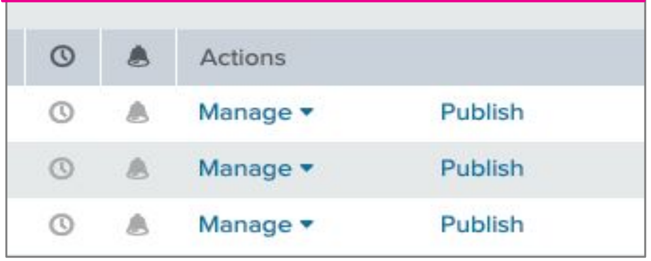
splunk> .conf19

Save and Publish

Save Your Experiment

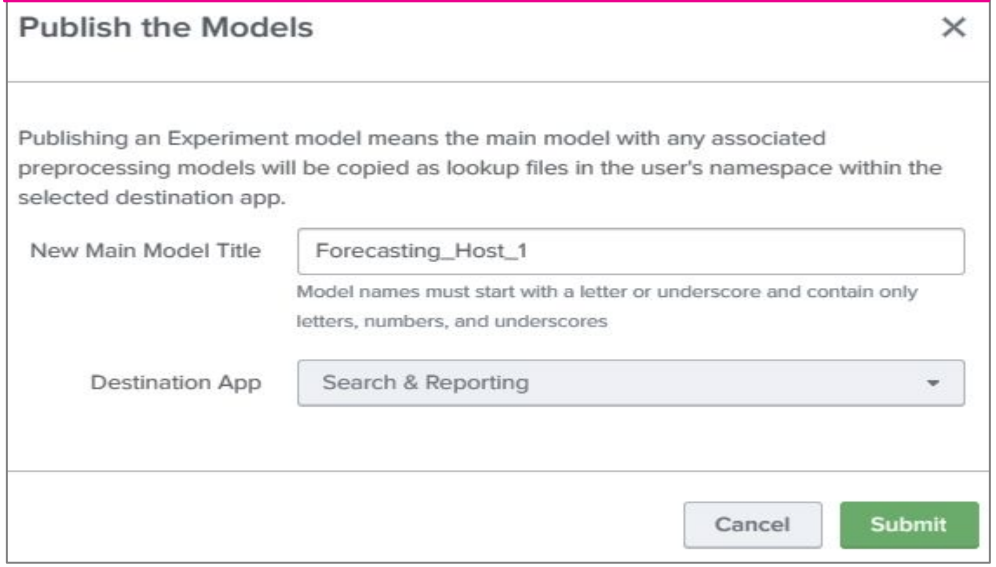


Publish Your Model



🕓	🔔	Actions	
🕓	🔔	Manage ▾	Publish
🕓	🔔	Manage ▾	Publish
🕓	🔔	Manage ▾	Publish

Name Your Model



Publish the Models

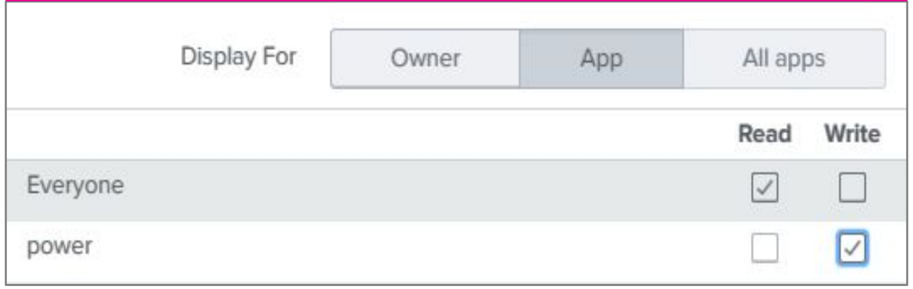
Publishing an Experiment model means the main model with any associated preprocessing models will be copied as lookup files in the user's namespace within the selected destination app.

New Main Model Title: Forecasting_Host_1

Model names must start with a letter or underscore and contain only letters, numbers, and underscores

Destination App: Search & Reporting ▾

Open Permissions

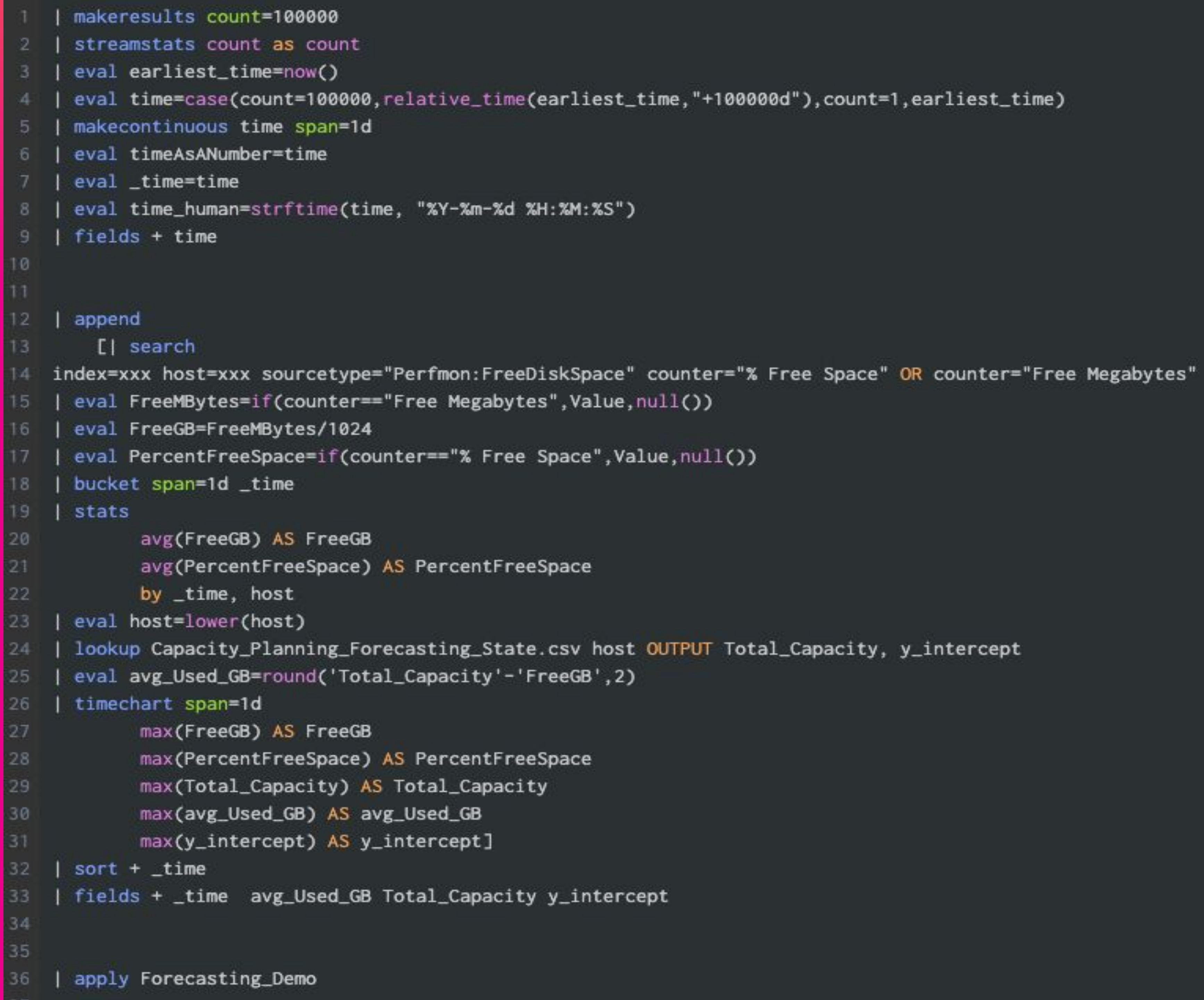


Display For: Owner | [✓ App] | All apps

	Read	Write
Everyone	☑	☐
power	☐	☑

Apply

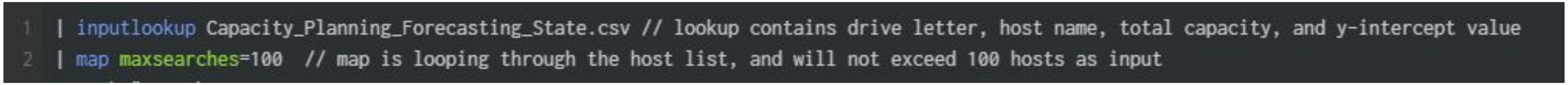
Apply your saved model onto your data



```
| makeresults count=100000
| streamstats count as count
| eval earliest_time=now()
| eval time=case(count=100000,relative_time(earliest_time,"+100000d"),count=1,earliest_time)
| makecontinuous time span=1d
| eval timeAsANumber=time
| eval _time=time
| eval time_human=strftime(time, "%Y-%m-%d %H:%M:%S")
| fields + time


| append
    [| search
index=xxx host=xxx sourcetype="Perfmon:FreeDiskSpace" counter="% Free Space" OR counter="Free Megabytes"
| eval FreeMBytes=if(counter=="Free Megabytes",Value,null())
| eval FreeGB=FreeMBytes/1024
| eval PercentFreeSpace=if(counter=="% Free Space",Value,null())
| bucket span=1d _time
| stats
        avg(FreeGB) AS FreeGB
        avg(PercentFreeSpace) AS PercentFreeSpace
        by _time, host
| eval host=lower(host)
| lookup Capacity_Planning_Forecasting_State.csv host OUTPUT Total_Capacity, y_intercept
| eval avg_Used_GB=round('Total_Capacity'-'FreeGB',2)
| timechart span=1d
        max(FreeGB) AS FreeGB
        max(PercentFreeSpace) AS PercentFreeSpace
        max(Total_Capacity) AS Total_Capacity
        max(avg_Used_GB) AS avg_Used_GB
        max(y_intercept) AS y_intercept]
| sort + _time
| fields + _time  avg_Used_GB Total_Capacity y_intercept


| apply Forecasting_Demo
```

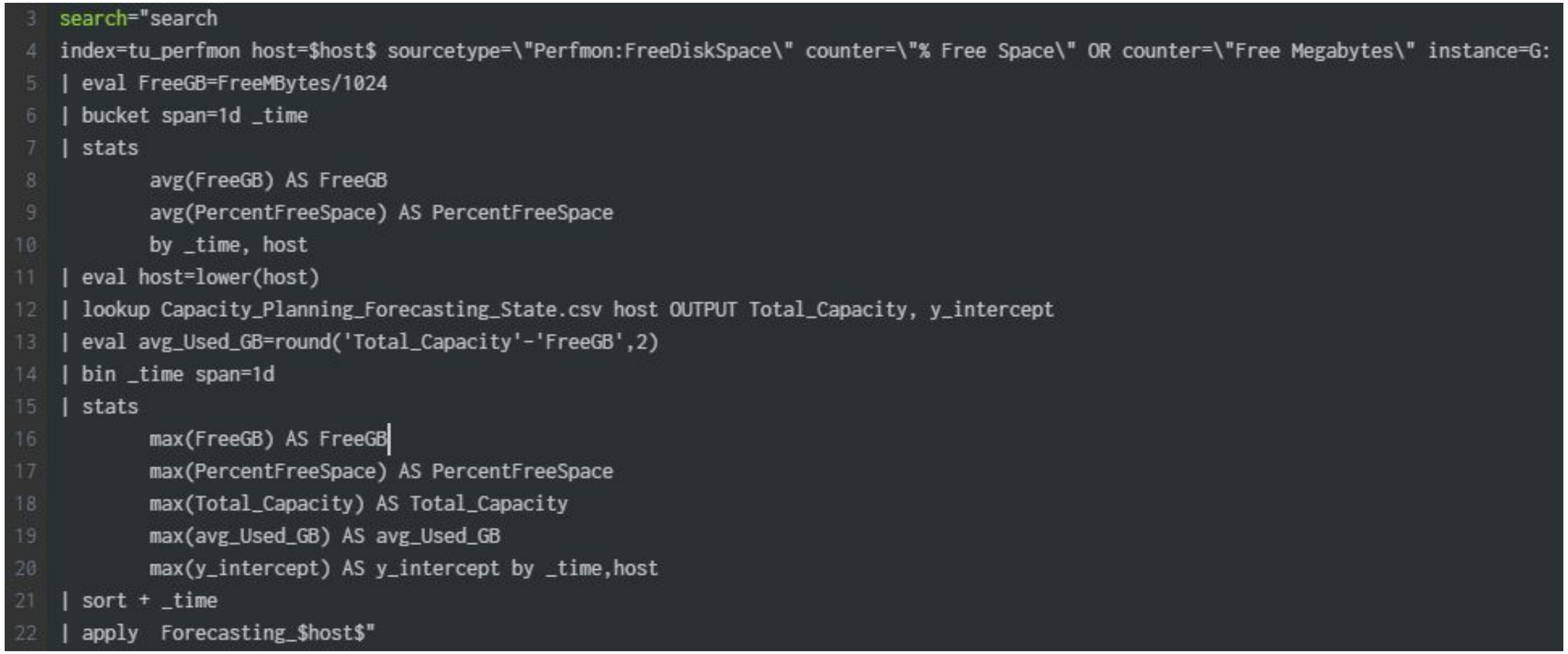
Auto-Adjust the Y-Intercept

Start with gathering the data from a lookup file



```
| inputlookup Capacity_Planning_Forecasting_State.csv // lookup contains drive letter, host name, total capacity, and y-intercept value
| map maxsearches=100  // map is looping through the host list, and will not exceed 100 hosts as input
```

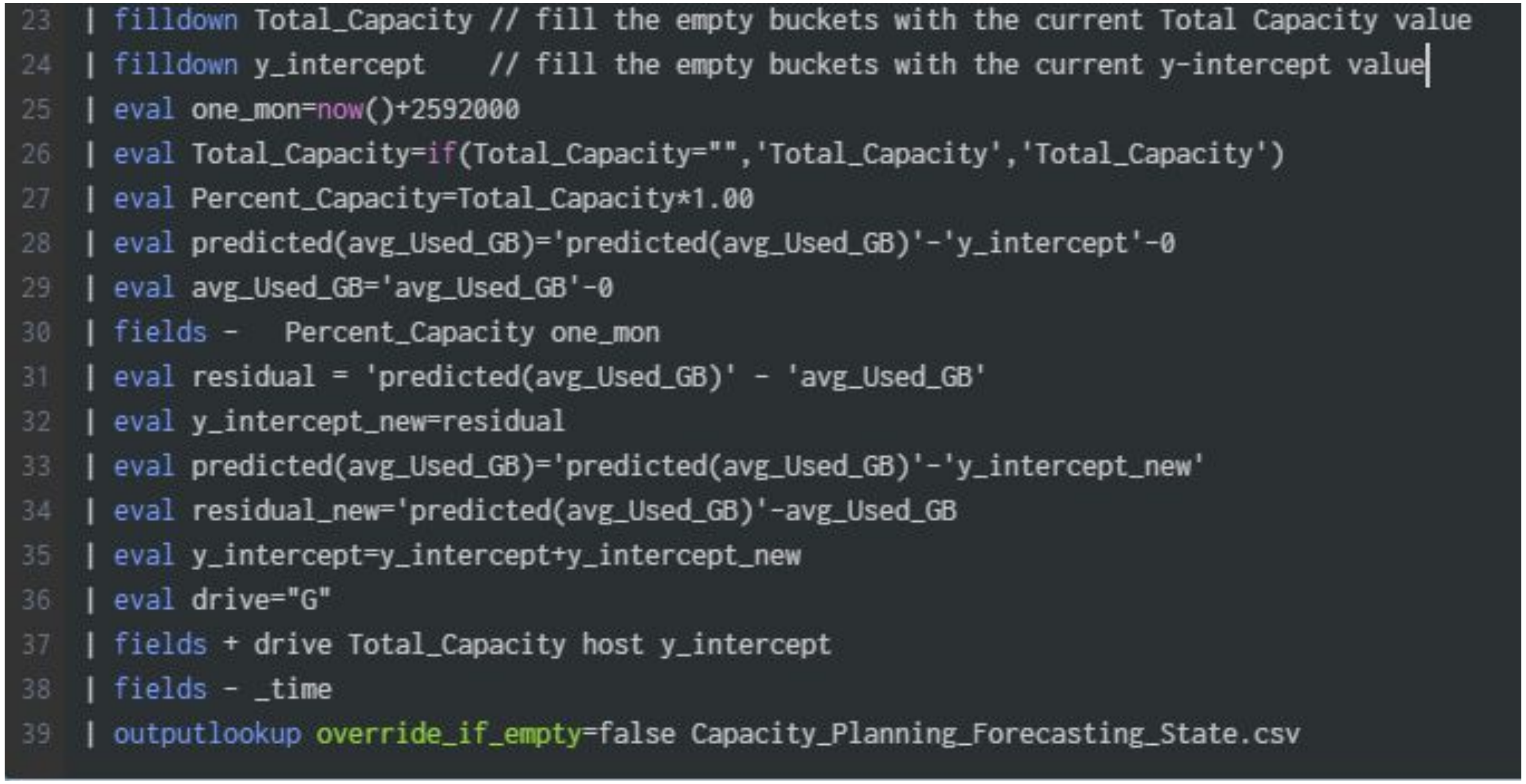
Use the same search string that we have been using to gather Total Capacity as well as Used GB



```
search="search
index=tu_perfmon host=$host$ sourcetype=\"Perfmon:FreeDiskSpace\" counter=\"% Free Space\" OR counter=\"Free Megabytes\" instance=G:
| eval FreeGB=FreeMBytes/1024
| bucket span=1d _time
| stats
        avg(FreeGB) AS FreeGB
        avg(PercentFreeSpace) AS PercentFreeSpace
        by _time, host
| eval host=lower(host)
| lookup Capacity_Planning_Forecasting_State.csv host OUTPUT Total_Capacity, y_intercept
| eval avg_Used_GB=round('Total_Capacity'-'FreeGB',2)
| bin _time span=1d
| stats
        max(FreeGB) AS FreeGB
        max(PercentFreeSpace) AS PercentFreeSpace
        max(Total_Capacity) AS Total_Capacity
        max(avg_Used_GB) AS avg_Used_GB
        max(y_intercept) AS y_intercept by _time,host
| sort + _time
| apply  Forecasting_$host$"
```

Auto-Adjust the Y-Intercept cont.…

- Filling the future buckets with the current total capacity, and y-intercept value
- Calculating % Capacity
- Outputlookup conducts the y-intercept adjustment in the lookup file.



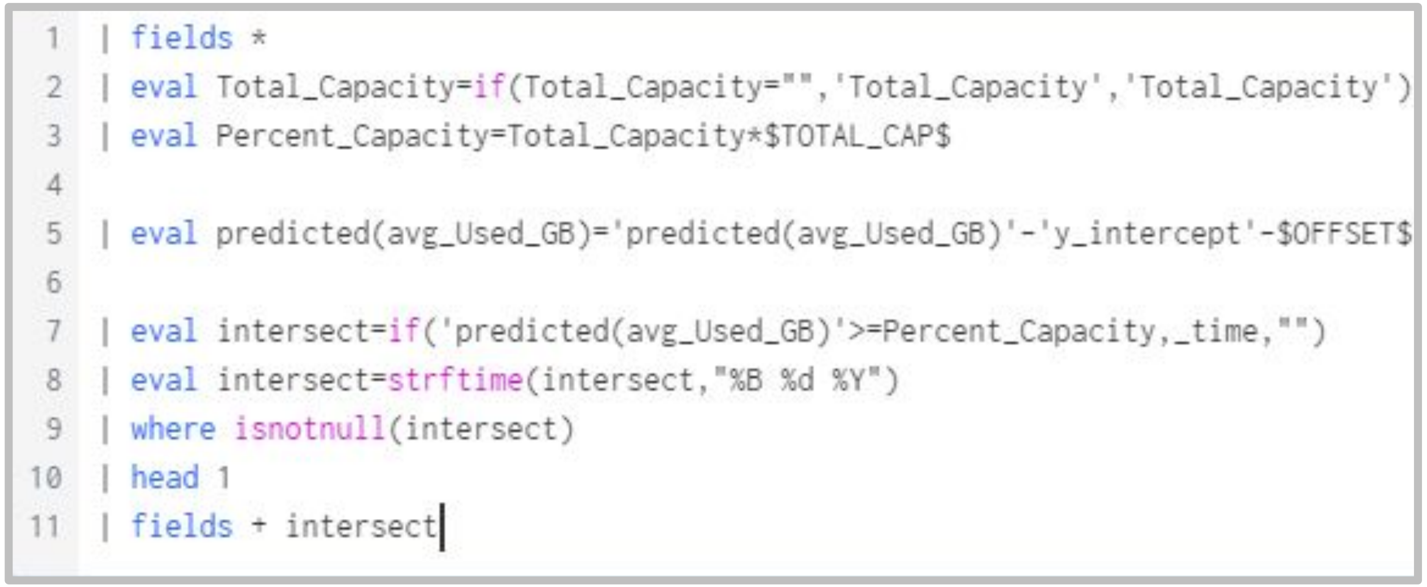
```
| filldown Total_Capacity // fill the empty buckets with the current Total Capacity value
| filldown y_intercept    // fill the empty buckets with the current y-intercept value
| eval one_mon=now()+2592000
| eval Total_Capacity=if(Total_Capacity="",'Total_Capacity','Total_Capacity')
| eval Percent_Capacity=Total_Capacity*1.00
| eval predicted(avg_Used_GB)='predicted(avg_Used_GB)'-'y_intercept'-0
| eval avg_Used_GB='avg_Used_GB'-0
| fields -   Percent_Capacity one_mon
| eval residual = 'predicted(avg_Used_GB)' - 'avg_Used_GB'
| eval y_intercept_new=residual
| eval predicted(avg_Used_GB)='predicted(avg_Used_GB)'-'y_intercept_new'
| eval residual_new='predicted(avg_Used_GB)'-avg_Used_GB
| eval y_intercept=y_intercept+y_intercept_new
| eval drive="G"
| fields + drive Total_Capacity host y_intercept
| fields - _time
| outputlookup override_if_empty=false Capacity_Planning_Forecasting_State.csv
```

Build a Forecasting Dashboard

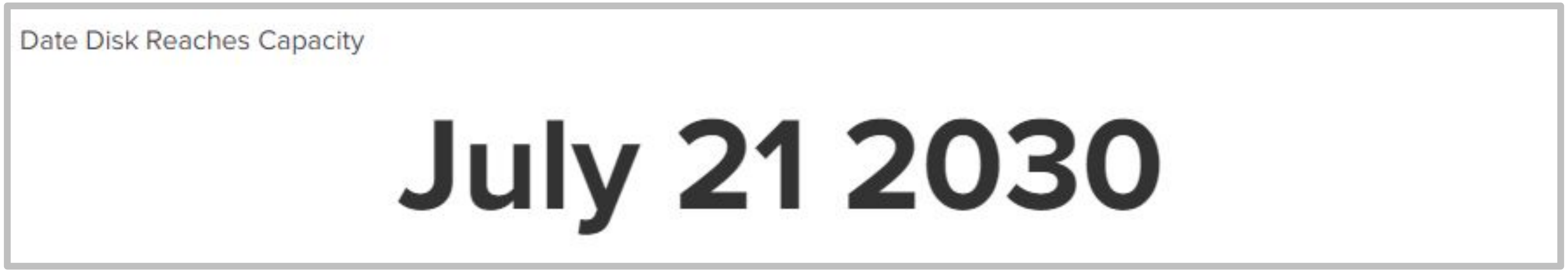
Base Search

```
| makeresults count=100000
| streamstats count as count
| eval earliest_time=now()
| eval time=case(count=100000,relative_time(earliest_time,"+100000d"),count=1,earliest_time)
| makecontinuous time span=1d
| eval _time=time
| eval time_human=strftime(time, "%Y-%m-%d %H:%M:%S")
| fields + time
| append
[| search
index=tu_perfmon host=$HOST$ sourcetype="Perfmon:FreeDiskSpace" counter="% Free Space" OR counter="Free Megabytes"
instance=G:
| eval FreeGB=FreeMBytes/1024
| bucket span=1d _time
| stats avg(FreeGB) AS FreeGB  avg(PercentFreeSpace) AS PercentFreeSpace by _time, host
| eval host=lower(host)
| lookup Capacity_Planning_Forecasting_State.csv host OUTPUT Total_Capacity, y_intercept|
eval avg_Used_GB=round('Total_Capacity'-'FreeGB',2)
| timechart span=1d max(FreeGB) AS FreeGB  max(PercentFreeSpace) AS PercentFreeSpace  max(Total_Capacity) AS
Total_Capacity        max(avg_Used_GB) AS avg_Used_GB  max(y_intercept) AS y_intercept]
| sort + _time
| fields + _time  avg_Used_GB Total_Capacity y_intercept
| apply  Forecasting_$HOST$
| filldown Total_Capacity| filldown y_intercept
```

Build a Forecasting Dashboard



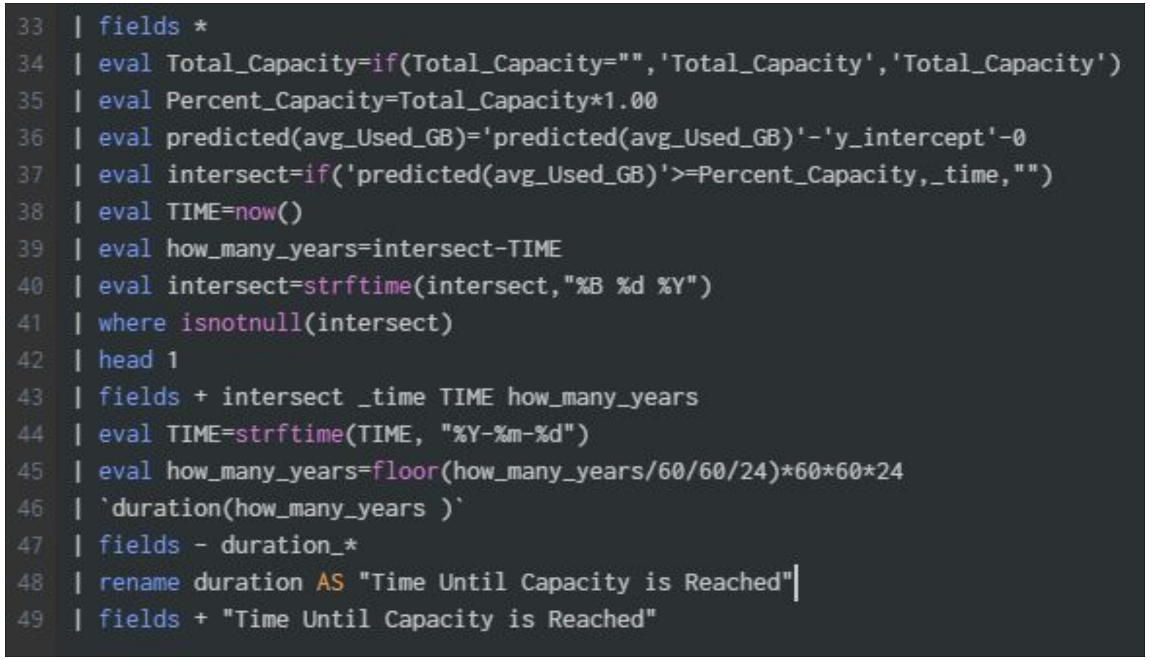
```
| fields *
| eval Total_Capacity=if(Total_Capacity="",'Total_Capacity','Total_Capacity')
| eval Percent_Capacity=Total_Capacity*$TOTAL_CAP$

| eval predicted(avg_Used_GB)='predicted(avg_Used_GB)'-'y_intercept'-$OFFSET$

| eval intersect=if('predicted(avg_Used_GB)'>=Percent_Capacity,_time,"")
| eval intersect=strftime(intersect,"%B %d %Y")
| where isnotnull(intersect)
| head 1
| fields + intersect
```



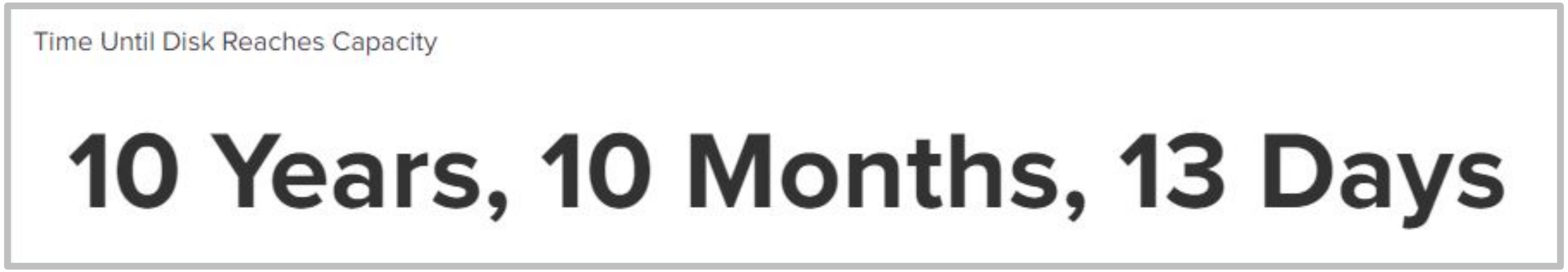
Date Disk Reaches Capacity

July 21 2030

Build a Forecasting Dashboard



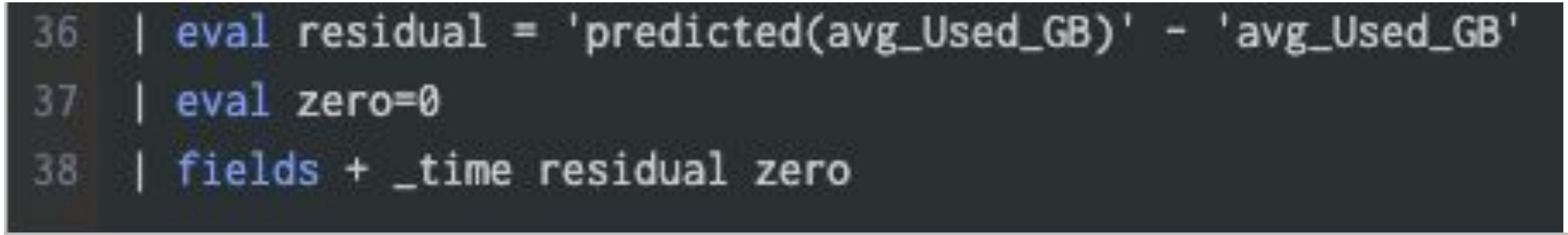
```
| fields *
| eval Total_Capacity=if(Total_Capacity="",'Total_Capacity','Total_Capacity')
| eval Percent_Capacity=Total_Capacity*1.00
| eval predicted(avg_Used_GB)='predicted(avg_Used_GB)'-'y_intercept'-0
| eval intersect=if('predicted(avg_Used_GB)'>=Percent_Capacity,_time,"")
| eval TIME=now()
| eval how_many_years=intersect-TIME
| eval intersect=strftime(intersect,"%B %d %Y")
| where isnotnull(intersect)
| head 1
| fields + intersect _time TIME how_many_years
| eval TIME=strftime(TIME, "%Y-%m-%d")
| eval how_many_years=floor(how_many_years/60/60/24)*60*60*24
| `duration(how_many_years )`
| fields - duration_*
| rename duration AS "Time Until Capacity is Reached"
| fields + "Time Until Capacity is Reached"
```



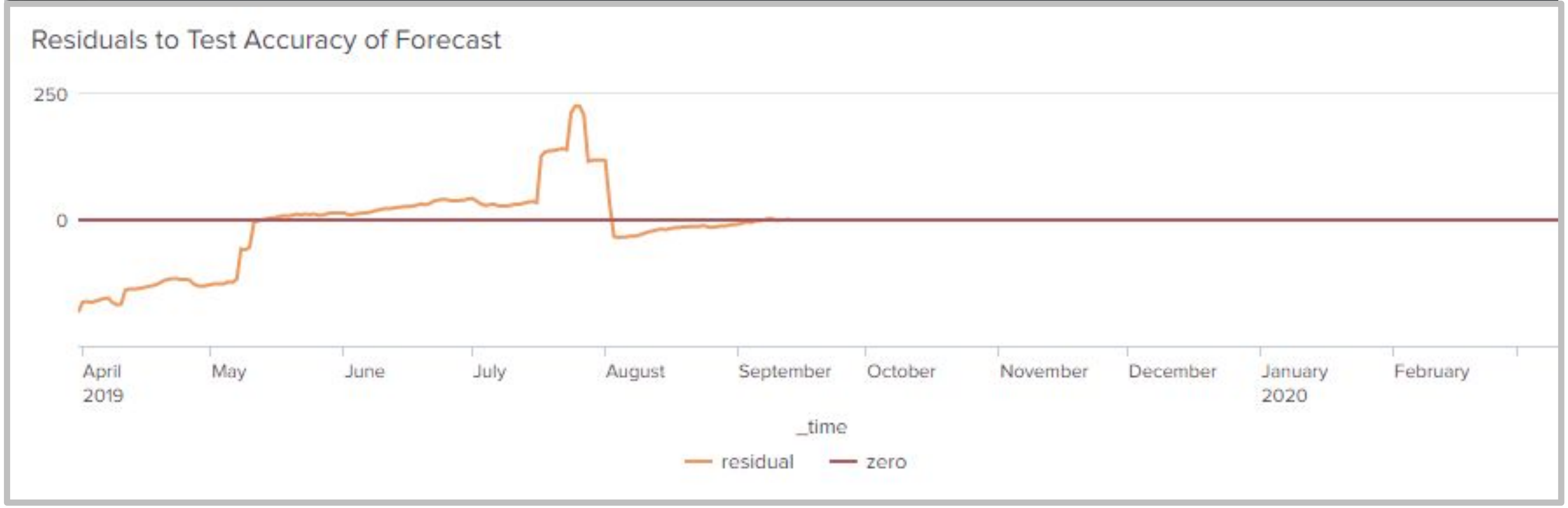
Time Until Disk Reaches Capacity

10 Years, 10 Months, 13 Days

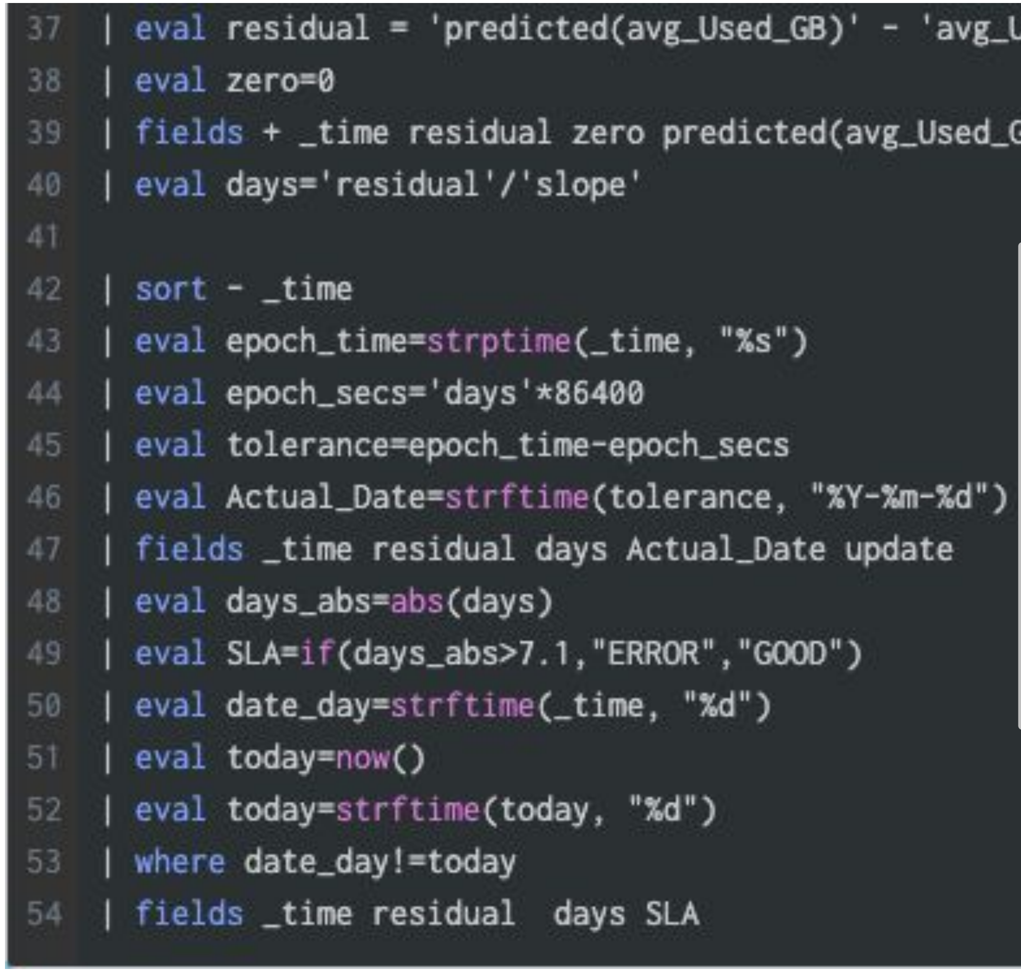
Build a Forecasting Dashboard



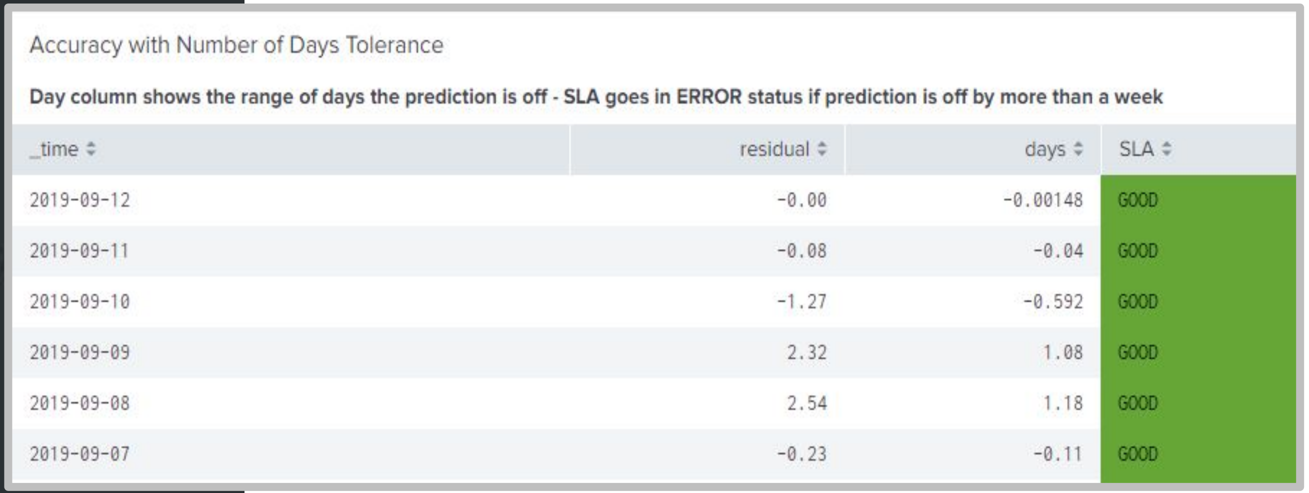
```
| eval residual = 'predicted(avg_Used_GB)' - 'avg_Used_GB'
| eval zero=0
| fields + _time residual zero
```



Build a Forecasting Dashboard



```
| eval residual = 'predicted(avg_Used_GB)' - 'avg_Used_GB'
| eval zero=0
| fields + _time residual zero predicted(avg_Used_GB) slope
| eval days='residual'/'slope'

| sort - _time
| eval epoch_time=strptime(_time, "%s")
| eval epoch_secs='days'*86400
| eval tolerance=epoch_time-epoch_secs
| eval Actual_Date=strftime(tolerance, "%Y-%m-%d")
| fields _time residual days Actual_Date update
| eval days_abs=abs(days)
| eval SLA=if(days_abs>7.1,"ERROR","GOOD")
| eval date_day=strftime(_time, "%d")
| eval today=now()
| eval today=strftime(today, "%d")
| where date_day!=today
| fields _time residual  days SLA
```



Accuracy with Number of Days Tolerance

Day column shows the range of days the prediction is off - SLA goes in ERROR status if prediction is off by more than a week

_time	residual	days	SLA
2019-09-12	-0.00	-0.00148	GOOD
2019-09-11	-0.08	-0.04	GOOD
2019-09-10	-1.27	-0.592	GOOD
2019-09-09	2.32	1.08	GOOD
2019-09-08	2.54	1.18	GOOD
2019-09-07	-0.23	-0.11	GOOD

splunk> .conf19

Q&A

Steve Koelpin | Splunk Advisor

splunk> .conf19

.conf19

splunk>

Thank You !

Go to the .conf19 mobile app to

RATE THIS SESSION