**Table of Contents**

[1. Understanding Net Error and Abs Error …2](#_Toc107756766)

[2. Build Net Error and Net Error % (Start – 5:45) ….3](#_Toc107756767)

[3. Build ABS Error, ABS Error % and Forecast Accuracy % Calculation (5:46 – End) ….5](#_Toc107756768)

# 1. Understanding Net Error and Abs Error

Before building the measures, it is recommended to build some understanding on the concepts of Net Error and ABS Error. Ideally, every business would want their prediction to become 100 % true – which means they want no gap between sales and forecast. This gap is technically an error which can be both positive and negative. In this case study, Atliq calculates the difference as Forecast – Actuals, since they anticipate the forecast to be more than sales and their intention is always to keep some extra stock. Therefore, the error is positive if Forecast > Actuals and negative if Actuals > Forecast.

**Note:** The difference can also be calculated as Actuals – Forecast if more sales are anticipated than forecast.  
  
Now, this error itself is broadly classified into two types as below

1. **Net Error** – consolidating all positive and negative errors to a **net** value.

Consider the below example

|  |  |  |  |
| --- | --- | --- | --- |
| Product | Sales | Forecast | Error |
| A | 50 | 40 | -10 |
| B | 100 | 130 | +30 |

The net error, in this case, is -10 for product A + 30 for product B = 20.

Net Error = 20 provides you with the sense of error at the total product level.

But, consider the example below

|  |  |  |  |
| --- | --- | --- | --- |
| Product | Sales | Forecast | Error |
| A | 50 | 40 | -10 |
| B | 100 | 110 | +10 |

In this case, the net error is -10 for product A +10 for product B = 0. This number looks great on paper at the total product level but ignores the error values at each product level. Hence, we need to calculate ABS error (Absolute error) which will provide a true sense of total error.

1. **ABS Error** – aggregating the absolute value from granular level to top.

The abs error value is calculated by taking the absolute value of each error at a granular level as shown below

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Product | Sales | Forecast | Error | ABS Error |
| A | 50 | 40 | -10 | 10 |
| B | 100 | 110 | +10 | 10 |

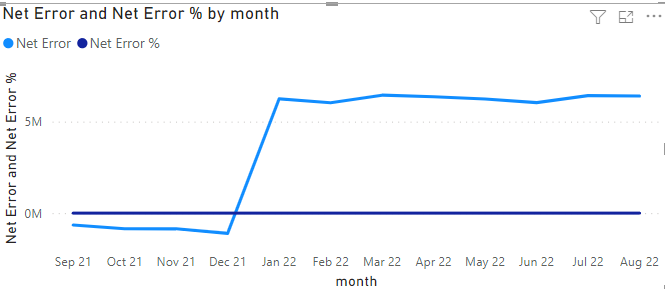
Now the total ABS Error is 10 for product A + 10 for product B = 20. This 20 is directionless, it does not say positive error or negative error – but helps to understand the true magnitude of total error.

# 2. Build Net Error and Net Error % (Start – 5:42)

**Creating Net Error**

This is a straightforward formula, just getting the difference between forecast and sales.  
  
Net Error = [Forecast Qty]- 'Key Measures'[Sales Qty]

**Creating Net Error %**  
  
Again, a straightforward one - find the error value to the total forecasted value

Net Error % = DIVIDE([Net Error], [Forecast Qty],0)  
  
**Limiting the formula to the last sales date**  
  
If you add the Net Error or Net Error % formula to the chart with the month on the x-axis and select FY 2022, you can see that it is providing results beyond December 2021 i.e. beyond the Last Sales Date. Is this correct?   
  
  
  
Ask yourself and take few seconds to think.  
.  
.

.

.

.

.

Congratulations, if you said ‘No’.   
  
Net Error displaying values beyond last sales date is incorrect as there are no sales to compare with the forecast.

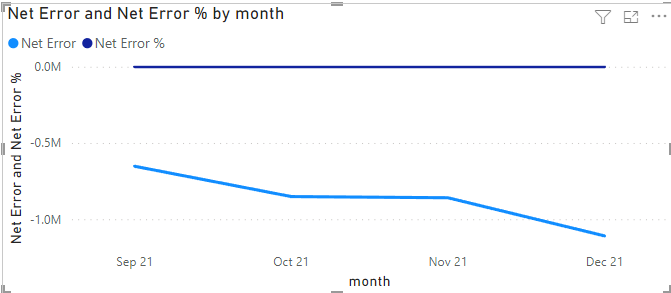
One way you can solve this issue is by limiting the forecast value only until the last sales date.

Hence, change your forecast formula as follows  
  
Forecast Qty =

var lsalesdate = MAX(LastSalesMonth[LastSalesMonth]) //last sales date

return

CALCULATE(SUM(fact\_forecast\_monthly[forecast\_quantity]),  
fact\_forecast\_monthly[date]<=lsalesdate) //limiting forecast to last sales date

Now, when the Net Error is executed, it will return blanks for both Sales and Forecast value for Jan, Feb … Sep 2022 and therefore the Net Error becomes blank. As blank values are not displayed by default in Power BI, you can see them disappearing in the chart as shown below.   
  


# 3. Build ABS Error, ABS Error % and Forecast Accuracy % Calculation (5:55 – 18:00)

Okay, now this concept needs more attention from you. Take a break if you need to – I want you to be completely relaxed when you are consuming this.

Now - before going further, revise the concept on page 1 once. The following explanation is entirely built on it.

Okay - this is the formula for calculating ABS Error. Read this once from the bottom.  
  
ABS Error =

SUMX( // **5** Iterate this formula for all months and sum them up  
DISTINCT(dim\_date[month]), // **4** convert net error to Abs error by taking absolute value for each month

SUMX( // **3** Iterate for all products and sum them up

DISTINCT(dim\_product[product\_code]), // **2** convert net error to Abs error by taking absolute value  
at each product

ABS([Net Error]) // **1** convert net error to Abs error by taking absolute value

)

)

Does this formula make sense?

If not, don’t worry - we are going to break this down.

Let’s start with ABS([Net Error])

This expression simply converts the Net Error value to ABS Error. For example, if a product has a -10 error it will convert that to 10.

If you are asking, why we need to convert net error to abs error – please go back to page 1. It is critical you understand this clearly.

This formula is good and should solve the problem, right?

I hear you; you are asking why are we using Distinct and Sumx, that too twice?!

Before going further, you need to understand what’s a Distinct and what’s Sumx.

[**Distinct:**](https://docs.microsoft.com/en-us/dax/distinct-function-dax)Distinct returns a column of unique values. In the above formula, you are using this to generate a distinct value for months and products.

[**Sumx**](https://docs.microsoft.com/en-us/dax/sumx-function-dax)**:** Sumx is an iterator. It simply **iterates** an expression for each row in a **table**. The expression it iterates in the above formula is ABS([Net Error]) and you are using the table generated by distinct in above step as the table rows to be iterated.

Let’s deal with a practical example to understand this better. Below table shows Net Error for two products at the Month level.

|  |  |  |  |
| --- | --- | --- | --- |
| **Product** | **Month** | **Net Error** | **ABS Error** |
| Mouse M1 | Oct 21 | -30 | 30 |
| Mouse M1 | Nov 21 | +25 | 25 |
| Mouse M1 | Dec 21 | -80 | 80 |
| Keyboard K1 | Oct 21 | +30 | 30 |
| Keyboard K1 | Nov 21 | -20 | 20 |
| Keyboard K1 | Dec 21 | -10 | 10 |

But why Product and Month – you may ask or ‘you need to ask’!

Because, in Atliq as per the supply chain team’s requirement - the ABS Error needs to be measured for each **product** at the **monthly** level. If there is a requirement to measure Abs Error at the segment and at the weekly level, you would have to change the approach and formula accordingly.

Okay, getting back to the above table. You know that you need to convert the Net error to ABS Error at the product and month level. In other words, you need to apply the ABS([Net Error]) for each row in the above table to get the highlighted last column and sum it up.

And how would you apply the formula to Products and Months? You need a list of products and Months for that. Hence DISTINCT(dim\_product[product\_code]) and DISTINCT(dim\_date[month]) are used.

Now, you need to iterate the ABS([Net Error]) for each product and sum them up. Hence, Sumx is required and formula develops as below

SUMX(   
DISTINCT(dim\_product[product\_code]),  
ABS([Net Error])  
  
But remember, you need to iterate the same at both Product **and** Month levels at the same time – i.e. you need to create a formula that will treat as if product and month are in the same row in the same table. However, you know that month and product are at a different tables in our dataset.  
  
Therefore, you take an approach to repeat the Sumx twice to iterate the ABS([Net Error]) both at the Product **and** Month level.

There you go, this is how you get your final formula

ABS Error =

SUMX( // **5** Iterate this formula for all months and sum them up  
DISTINCT(dim\_date[month]), // **4** convert net error to Abs error by taking absolute value for each month

SUMX( // **3** Iterate for all products and sum them up

DISTINCT(dim\_product[product\_code]), // **2** convert net error to Abs error by taking absolute value  
at each product

ABS([Net Error]) // **1** convert net error to Abs error by taking absolute value

)

)

**ABS Error %**

Now, we got the ABS Error, ABS Error % is going to be a straightforward calculation. It is the % of total ABS Error to the forecast Quantity.

ABS Error % = DIVIDE ('Key Measures'[ABS Error], [Forecast Qty],0)

**Forecast Accuracy %**

What is Forecast Accuracy?

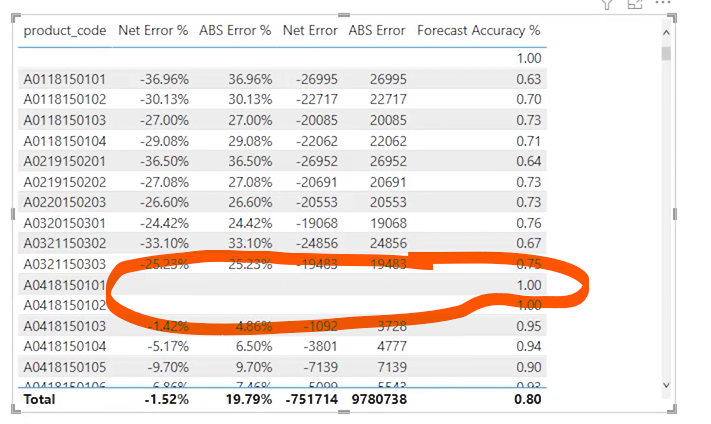
It is the accuracy with which we forecast the sales. In other words, it is the % of non-error forecasts.

This means, that if the ABS error is 20 % the non-error is 80 % or in other words, the Forecast Accuracy is 80%.

Hence, the forecast accuracy formula will be  
  
Forecast Accuracy % = 1- [ABS Error %]

Great job, we are done!

Maybe not yet, you will find that forecast accuracy is 1 (100 %) for some of the products as shown below, which is not correct.

****

This happens due to the fact that products without sales or forecast is included in this calculation.

How it is getting included? Because for products without forecast or sales, the ABS Error % is blank. By that logic,  
  
Forecast Accuracy = 1- Blank()  
  
which returns 1.  
  
Hence, we need to add an additional filter to attribute to blank value from[ABS Error %]

Forecast Accuracy % = IF(  
'Key Measures'[ABS Error %]<>BLANK(),  
1-'Key Measures'[ABS Error %],  
BLANK())

By the above logic, if the [ABS Error %] is blank the formula will also return blank and hence it won’t be displayed in the table.

Yayy!! You have completed the metrics required for the supply chain view and going to build the visuals in the next chapter.Happy learning **😊**