The `predict` operator uses a series of time-stamped numerical values to predict future values. The `predict` operator is useful in the following cases:

- As an early warning system, alerting you when a threshold is about to be reached.
- For resource and capacity planning, helpful for determining seasonal impacts, like a Cyber Monday rush on an ecommerce site.
- Improved risk calculation.

For example, you could use `predict` to take your current disk space capacity numbers, and predict when your system might run out of disk space. In these cases, the sooner an operations manager is informed that a key threshold is about to be reached the more effectively he or she can plan to avoid service degradation.

The `predict` operator supports two predictive models:

- Auto-regressive. Uses an advanced auto-regressive (AR) algorithm to learn patterns in the data. It automatically detects the cyclical patterns in the data and uses the cycles in its prediction.
- Linear regression. Uses existing data over the query time range as a training set to generate a linear model, and then extrapolates future values using this model.

If a missing data point is encountered in the generated time series, Sumo uses a zero value and issues a warning to alert you. If you would like to use customized values, you can do so with the `fillmissing` operator.

**Syntax**

The syntax for `predict` varies depending on whether you use the linear regression model or the auto-regressive model. In either case, the following requirements apply:

- The query must contain an aggregate operator for example, `count`, `min`, `max`, `sum` and so on. Aggregation must be by timeslice, for example, `count by _timeslice`.
- The query must contain the `timeslice` operator.
- Both the aggregate operator and the `timeslice` operator must precede the `predict` operator.

**Syntax for the linear regression model**

For the linear regression model:
The linear regression algorithm produces the following fields in the output:

- **_count** The number of matches per minute for the currently selected time range.
- **_count_predicted** Value predicted by the simple linear model.
- **(absolute value)_count_error** Value predicted by the simple linear model, minus the actual number.

### Syntax for the auto-regressive model

```
... | timeslice 1m | count by _timeslice | predict _count by 1m model=ar, ar.window=n, forecast=n
```

The table below defines the parameters for running `predict` using the AR model.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>model=ar</code></td>
<td>Required. Use this parameter to use the AR model, rather than a simple linear regression.</td>
</tr>
<tr>
<td><code>ar.window</code></td>
<td>Optional. Use this parameter to specify the number of consecutive data points on which to base the prediction.</td>
</tr>
<tr>
<td><code>forecast</code></td>
<td>Optional. Use this option to specify the duration of the prediction, either in terms of data points or minutes. The <code>forecast</code> parameter uses the same syntax as the <code>timeslice</code> operator and accepts either:</td>
</tr>
<tr>
<td>- A simple number, such as <code>forecast=5</code>, which predicts five data points into the future.</td>
<td></td>
</tr>
<tr>
<td>- A time granularity, such as <code>forecast=5m</code>, which predicts five minutes into the future.</td>
<td></td>
</tr>
</tbody>
</table>

The auto-regressive model requires more data points than the linear regression option to train a proper model. We recommended that you generate more than 100 data points for the AR model. If the number of data points is less than 100, Sumo Logic displays a warning stating that the learned model may not be optimal.

If you want to customize the window size, we recommend that you set the window size to be larger than 40% of the data points in the time series. However, don't make the window size too large. The maximum would be 60%. Otherwise, the model will not be trained properly.

Optional. Use this option to specify the duration of the prediction, either in terms of data points or minutes. The `forecast` parameter uses the same syntax as the `timeslice` operator and accepts either:

- A simple number, such as `forecast=5`, which predicts five data points into the future.
- A time granularity, such as `forecast=5m`, which predicts five minutes into the future.

If you don’t specify a value for the `forecast` parameter, the operator will default to three data points into the future, or `forecast=3`.

In the following query, the first three lines count the number of messages that contain an error term for every half minute. The last line uses the auto-regressive model to predict 100 data points in the future, based on 50 data points.
The auto-regressive algorithm produces the following fields in the output:

- `_count_` The number of errors per 30-second timeslice.
- `_count_predicted_` Value predicted by the auto-regressive algorithm.
- `_count_linear_` Value predicted by the simple linear regression.
- `(absolute value) _count_error_` Value predicted by the simple linear regression minus the actual number.

**Limitations**

These internal limitations are meant to provide "speed bumps" to ensure the best performance.

- `predict` will not use more than 10,000 input points to estimate the model.
- `predict` will not forecast more than 100 points into the future.
- `predict` will not interpolate more than 20,000 input points. (Predict adds "phantom" input points where there should be a timeslice, but no data point is present.)

**Cyclical patterns and the auto-regressive model**

If there are cyclical patterns that fit within the `ar.window`, the auto-regressive algorithm will learn the cyclical pattern and use that in prediction.

For example, if there is an hourly cyclical pattern, the following query will learn that cycle:

```plaintext
... | timeslice 5m
| <aggregate function> by _timeslice as _val
| predict _val by 5m model=ar, ar.window=15
```

In this query, the window size (15 consecutive data points) covers more than 1 hour (15 data points * 5m interval = 75 minutes). So if there are cyclical patterns with a period of less than 75 minutes, the model will discover them.

**Examples**

**predict using linear regression**

This query predicts the count of 404 errors per minute using linear regression.

```plaintext
_sourceCategory=Labs/Apache/Access status_code=404 | timeslice 1m | count(status_code) as error_count by _timeslice | predict error_count by 1m
```
The query returns an aggregation table with columns for `error_count`, `error_count_predicted`, and `error_count_error`.

<table>
<thead>
<tr>
<th>#</th>
<th>Time</th>
<th>error_count</th>
<th>error_count_predicted</th>
<th>error_count_error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11/13/2017 1:35:00 PM - 0800</td>
<td>16</td>
<td>68.10492</td>
<td>52.10492</td>
</tr>
<tr>
<td>2</td>
<td>11/13/2017 1:36:00 PM - 0800</td>
<td>67</td>
<td>68.19458</td>
<td>1.19458</td>
</tr>
<tr>
<td>3</td>
<td>11/13/2017 1:37:00 PM - 0800</td>
<td>70</td>
<td>68.28425</td>
<td>-1.71575</td>
</tr>
<tr>
<td>4</td>
<td>11/13/2017 1:38:00 PM - 0800</td>
<td>94</td>
<td>68.37391</td>
<td>-25.62509</td>
</tr>
<tr>
<td>5</td>
<td>11/13/2017 1:39:00 PM - 0800</td>
<td>74</td>
<td>68.46357</td>
<td>-5.53643</td>
</tr>
<tr>
<td>6</td>
<td>11/13/2017 1:40:00 PM - 0800</td>
<td>70</td>
<td>68.55324</td>
<td>-1.44676</td>
</tr>
<tr>
<td>7</td>
<td>11/13/2017 1:41:00 PM - 0800</td>
<td>60</td>
<td>68.64290</td>
<td>8.64290</td>
</tr>
<tr>
<td>8</td>
<td>11/13/2017 1:42:00 PM - 0800</td>
<td>63</td>
<td>68.73256</td>
<td>5.73256</td>
</tr>
<tr>
<td>9</td>
<td>11/13/2017 1:43:00 PM - 0800</td>
<td>73</td>
<td>68.82223</td>
<td>-4.17777</td>
</tr>
<tr>
<td>10</td>
<td>11/13/2017 1:44:00 PM - 0800</td>
<td>56</td>
<td>68.91189</td>
<td>12.91189</td>
</tr>
</tbody>
</table>

From here, you can select the **Line Chart** icon, and automatically create a Combo Chart that represents the `error_count_error` as a column chart, and the `error_count` and `error_count_predicted` mapped on top of that with separate lines. Note that the (absolute value)_count_error series is toggled off by default. Click it in the legend to display the column chart.

**predict using auto-regressive model**

This query predicts the count of 404 errors per minute using the auto-regressive model.

```
_sourceCategory=Labs/Apache/Access status_code=404 | timeslice 1m | count(status_code) as error_count by _timeslice | predict error_count by 1m model=ar
```

The query returns an aggregation table with columns for `error_count`, `error_count_predicted`, `error_count_linear`, and `error_count_error`. 
From here, you can select the **Line Chart** icon, and automatically create a Combo Chart that represents the `error_count_error` as a column chart, and the `error_count` and `error_count_predicted` mapped on top of that with separate lines. Note that the (absolute value) `count_error` series is toggled off by default. Click it in the legend to display the column chart.

Note that, if desired, you can display the `count_linear` series, to see the value predicted by the simple linear regression model by clicking it in the legend.