Hands-on: Time Series API

Using Azure Machine Learning

In this hands-on lab, you explore the Azure Machine Learning prediction functionality by using the Forecasting Model experiment for Dynamics 365 Business Central. This algorithm allows you to make time series predictions that can assist you in identifying future trends based on historic data.

Contents

Install git

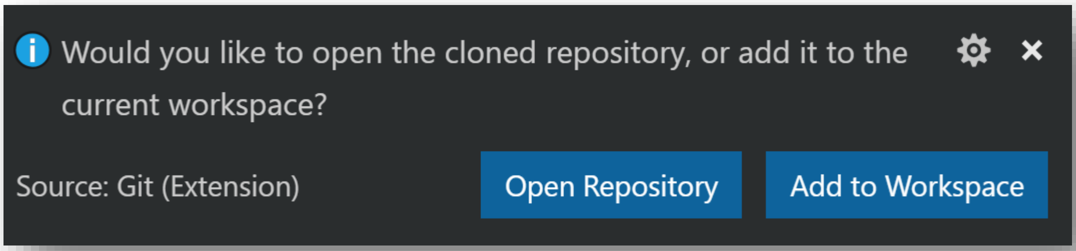
In this exercise, you prepare your development environment by installing and configuring Git. Open your web browser. If you already have Git installed, skip to the next exercise.

1. Navigate to https://git-scm.com/download
2. Select the download for Windows.
3. Save the incoming installation file to a desired location on your computer.
4. After download completes, locate the file you downloaded in the previous step, and run it. Accept any security warnings if shown.
5. Complete the installation by accepting all the default options.

Clone the hands-on repository

In this step you clone the git repository containing exercise code for this hands-on lab.

1. Start Visual Studio Code (or restart it if it was already started).
2. Press Ctrl+Shift+P to access the command palette.
3. In the command palette prompt, enter “Git: Clone”
4. In the Repository URL prompt, enter: https://github.com/vjekob/TimeSeriesAPI.git
5. Select a target folder where you want to clone this repository.
6. When cloning completes, Visual Studio Code asks the following question:



Would you like to open the cloned repository, or add it to the current workspace?

Source: Git (Extension)

7. Click **Open Repository**.

Prepare demo data

In this step you prepare your Business Central sandbox environment with the data necessary for meaningful simulations of time series predictions.

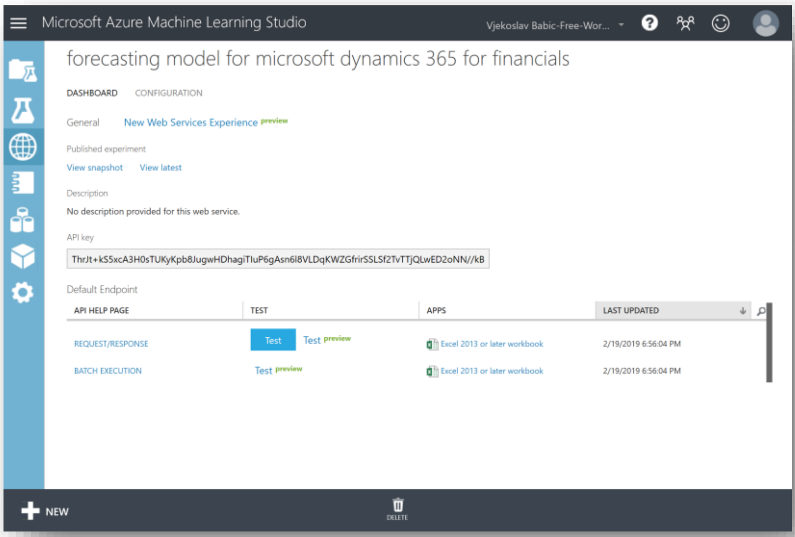
1. Press Ctrl+Shift+P to access the command palette.
2. In the command palette prompt, enter “Git: Checkout to…”
3. From the list of branches, select “Preparation”
4. Press Ctrl+P to access the **Go to file** feature.
5. In the prompt, enter “launch.json” and then press Enter to open launch.json file in the editor.
6. Make the necessary configuration changes to be able to publish and run the extension in your Business Central sandbox instance. If you are unsure how to do this, ask your instructor for help.
7. Save and close launch.json file.
8. Access the command palette again, and then run “AL: Download Symbols”
9. Press F5 to run your extension.
10. In the Data Preparation page, click **Actions > Prepare Data**.
11. When the **Item Journals** page opens, click **Process > Post** to post the journal lines.
12. In Visual Studio Code, use the “Git: Checkout to…” command to select the master branch.

Publish an endpoint from a machine learning model

Time Series prediction functionality in Dynamics 365 Business Central requires a published endpoint. There are no publicly available endpoints, because they depend on each customer's individual Azure plans. Therefore, you must publish your own endpoint.

In this exercise, you'll select an existing machine learning model from Cortana Intelligence Gallery and create an endpoint from it. The model you will select is prepared by the Microsoft ERP team and is aimed at time-series predictions.

1. Open your web browser and navigate to: https://gallery.cortanaintelligence.com/Experiment/Forecasting-Model-for-Microsoft-Dynamics-365-for-Financials-1
2. Click **Open in Studio**. The browser will open a new tab.
3. If you haven't already been signed in to Azure, the browser will ask you to sign in into your Azure account. Enter your credentials and sign in.
4. When Microsoft Azure Machine Learning Studio opens, the **Copy experiment from Gallery** dialog pops up.
5. Click the **OK** button ✓ to accept the defaults. Machine Learning Studio will set up your workspace.
6. Click the **Run** button in the bottom of the experiment canvas to run and validate the experiment. This step is required before you can deploy the model as a web-service endpoint.
7. After the execution completes, the status message at the upper-right point of the screen (to the right of the model title) will indicate that the experiment has finished successfully:
   Finished running ✓
   The **Deploy Web service** button becomes enabled.
8. Click **Deploy Web service**. The system deploys the Azure Machine Learning experiment as web service and provides a REST API that can be consumed by a wide range of devices and platforms, including Dynamics 365 Business Central.
9. When the deployment finishes, the web service dashboard opens. Here, you can see the API key and two available API endpoints: **Request/Response** and **Batch Execution**. You will need the Request/Response API endpoint, because it is the only API that the current version of the Time Series functionality in Business Central supports.


Microsoft Azure Machine Learning Studio
Vjekoslav Babic-Free-Wor...
NEW
DELETE

forecasting model for microsoft dynamics 365 for financials

DASHBOARD CONFIGURATION

General New Web Services Experience preview

Published experiment

View snapshot View latest

Description

No description provided for this web service.

API key

ThrJt+kS5xcA3H0sTUKyKpb8JugwHDhagiTIuP6gAsn6l8VLDqKWZGfrirSSLSf2TvTTjQLwED2oNN//kB

Default Endpoint

API HELP PAGE	TEST	APPS	LAST UPDATED
REQUEST/RESPONSE	Test Test preview	Excel 2013 or later workbook	2/19/2019 6:56:04 PM
BATCH EXECUTION	Test preview	Excel 2013 or later workbook	2/19/2019 6:56:04 PM

10. Copy the API Key value and store it somewhere for later use. This is your personal **API Key** that you must not share with anyone.
11. Click **Request/Response**. This opens the API help page where you can find the input and output definitions, code samples, and explanations. The only thing you need here is the endpoint URL. You can find it in the **Request URI** column of the table in the **Request** section.
12. Copy the **Request URI** value and store it somewhere for later use.

Use the Time Series API to get predictions

In this exercise you'll write code that calls the Time Series API from your Business Central sandbox environment. You will use different data that is available in the database. You will also check the quality of the predictions programmatically before displaying them to the end user.

Create setup table and page

To access the endpoint created in the previous exercise, Business Central will require a correct URI and API key. Since these values can change between implementations, it's best practice to store these values in a setup table. Create a setup table to hold these values, and a card-type page to allow users to configure these values.

1. Start Visual Studio Code. Make sure that the Time Series workspace is open.
2. Press Ctrl+N to create a new object. Save it as "Table 50110 ML Forecast Setup.al".
3. In the editor, define the new object **table 50110 "ML Forecast Setup"**.
4. In the table, define the following fields:

No.	Name	Type
1	Primary Key	Code[10]
2	Endpoint URI	Text[250]
3	API Key	Text[250]

5. Make the **Primary Key** field non-editable.
6. Define the caption property for each field.
7. Define the table primary key, and set it to include only the field **Primary Key**.
8. When you are done editing, don't forget to save the object.
9. Press Ctrl+N to create a new object. Save it as "Page 50110 ML Forecast Setup.al".
10. In the editor, define the new object **page 50110 "ML Forecast Setup"**.
11. Define the page to be of **Card** type, set the page caption to 'Machine Learning Forecast Setup', make the page show data from table **ML Forecast Setup**, make sure that the page shows in the **Tell me** feature in the **Administration** group, and define the **ApplicationArea** property.

```
PageType = Card;
Caption = 'Machine Learning Forecast Setup';
SourceTable = "ML Forecast Setup";
UsageCategory = Administration;
ApplicationArea = All;
```

12. Define the page layout and include the **Content** area. Inside the **Content** area, define a single group named **General**. For the **General** group, set the caption to 'General'.

```
layout
{
    area(Content)
    {
        group(General)
        {
            Caption = 'General';
        }
    }
}
```

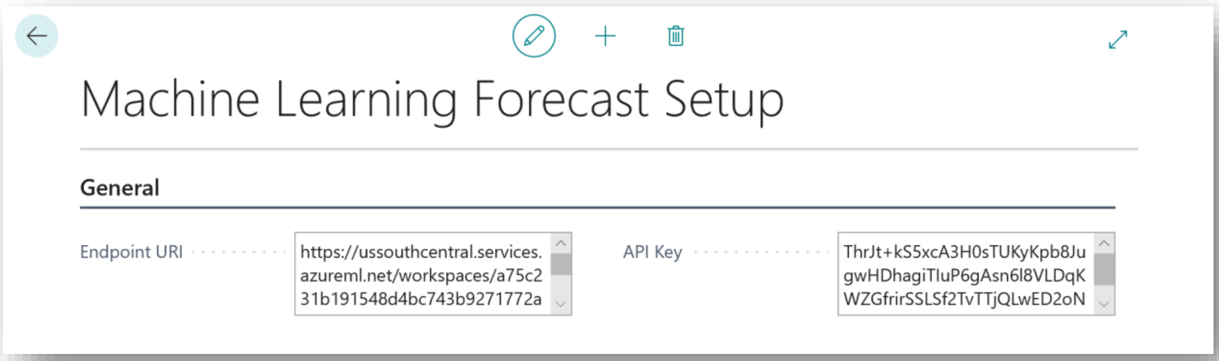
13. Inside the **General** group, define two fields for the **Endpoint URI** and **API Key** table fields respectively. Define their Caption and ApplicationArea properties. Also, make them multi-line to display as much text as possible (the values they will hold are long).

```
field("Endpoint URI"; "Endpoint URI")
{
    Caption = 'Endpoint URI';
    MultiLine = true;
    ApplicationArea = All;
}
field("API Key"; "API Key")
{
    Caption = 'API Key';
    MultiLine = true;
    ApplicationArea = All;
}
```

14. Define the OnOpenPage trigger, and inside it write the standard Setup page pattern boilerplate code: reset any filters, attempt to retrieve the current record, and if none was found, initialize and insert the record:

```
trigger OnOpenPage();
begin
    Reset();
    if not Get() then begin
        Init();
        Insert();
    end;
end;
```

15. Build, deploy, and run the extension.
16. In the Business Central web client, click the **Tell me** button (💡).
17. Search for the “Machine Learning Forecast Setup” page.
18. In the list of results, click **Machine Learning Forecast Setup**.
19. In the **Machine Learning Forecast Setup** page, populate the Endpoint URI and API Key fields with the values generated by the Azure platform for the API web service you created earlier in this hands-on lab. The page should look somewhat like this.


Machine Learning Forecast Setup

General

Endpoint URI: https://ussouthcentral.services.azureml.net/workspaces/a75c231b191548d4bc743b9271772a

API Key: ThrJt+kS5xcA3H0sTUKyKpb8JugwHDhagiTluP6gAsn6l8VLDqKWZGfrirSSLSf2TvTTjQLwED2oN

Solution: Table 50110 ML Forecast Setup.ml

```
table 50110 "ML Forecast Setup"
{
    fields
    {
        field(1; "Primary Key"; Code[10])
        {
            Caption = 'Primary Key';
            Editable = false;
        }
        field(2; "Endpoint URI"; Text[250])
        {
            Caption = 'Endpoint URI';
        }
        field(3; "API Key"; Text[250])
        {
            Caption = 'API Key';
        }
    }

    keys
    {
        key(PrimaryKey; "Primary Key") { }
    }
}
```

Solution: Page 50110 Machine Learning Setup.al

```
page 50110 "ML Forecast Setup"
{
    PageType = Card;
    Caption = 'Machine Learning Forecast Setup';
    SourceTable = "ML Forecast Setup";
    UsageCategory = Administration;
    ApplicationArea = All;

    layout
    {
        area(Content)
        {
            group(General)
            {
                Caption = 'General';
                field("Endpoint URI"; "Endpoint URI")
                {
                    Caption = 'Endpoint URI';
                    MultiLine = true;
                    ApplicationArea = All;
                }
                field("API Key"; "API Key")
                {
                    Caption = 'API Key';
                    MultiLine = true;
                    ApplicationArea = All;
                }
            }
        }
    }

    trigger OnOpenPage();
    begin
        Reset();
        if not Get() then begin
            Init();
            Insert();
        end;
    end;
}
```

Create the codeunit to invoke API

1. In Visual Studio Code, press Ctrl+N to create a new object, and save it as “Codeunit 50110 Machine Learning Forecast.al”.
2. In the editor define the new object **codeunit 50110 “Machine Learning Forecast”**.
3. In the codeunit, define a function CalculateForecast that receives a single parameter of type Record Item, and returns a Decimal.

```
    procedure CalculateForecast(Item: Record Item): Decimal;
    begin

    end;
```

4. For the function, declare the following variables:

```
        MLForecastSetup: Record "ML Forecast Setup";
        ItemLedgerEntry: Record "Item Ledger Entry";
        Date: Record Date;
        TempTimeSeriesForecast: Record "Time Series Forecast" temporary;
        TimeSeriesManagement: Codeunit "Time Series Management";
```

5. In the function body, write code to unconditionally retrieve the ML Forecast Setup record, and then initialize the TimeSeriesManagement codeunit by passing the **Endpoint URI** and **API Key** field values:

```
        TimeSeriesManagement.Initialize(
            MLForecastSetup."Endpoint URI", MLForecastSetup."API Key",
            0, false);
```

6. Write code to filter the item ledger entries where **Entry Type** is **Sale**, that belong to the item passed into the function through its Item parameter:

```
        ItemLedgerEntry.SetRange("Item No.", Item."No.");
        ItemLedgerEntry.SetRange("Entry Type", ItemLedgerEntry."Entry
Type"::Sale);
```

7. Write code that invokes the PrepareData function. This function transforms any table data into a dataset that is ready for submission. You will pass the filtered Item Ledger Entry table, field numbers for **Item No.**, **Posting Date**, and **Quantity** fields. Configure the forecast period of length of a month, starting from the current work date, by taking into account past 12 months of data.

```
        TimeSeriesManagement.PrepareData(
            ItemLedgerEntry,
            ItemLedgerEntry.FieldNo("Item No."),
            ItemLedgerEntry.FieldNo("Posting Date"),
            ItemLedgerEntry.FieldNo(Quantity),
            Date."Period Type"::Month,
            WorkDate,
            12);
```

8. Write code to invoke the Azure Machine Learning endpoint by requesting the forecast for one month, then retrieve the results from the forecast invocation into the TempTimeSeriesForecast temporary record variable. If there are any records returned from the forecast invocation, return the value of the Value field of the first record from the function.

```
        TimeSeriesManagement.Forecast(1, 0, 0);
        TimeSeriesManagement.GetForecast(TempTimeSeriesForecast);
        if TempTimeSeriesForecast.FindFirst() then
            exit(TempTimeSeriesForecast.Value);
```

Solution: Codeunit 50110 Machine Learning Forecast.al

```
codeunit 50110 "Machine Learning Forecast"
{
    procedure CalculateForecast(Item: Record Item): Decimal;
    var
        MLForecastSetup: Record "ML Forecast Setup";
        ItemLedgerEntry: Record "Item Ledger Entry";
        Date: Record Date;
        TempTimeSeriesForecast: Record "Time Series Forecast" temporary;
        TimeSeriesManagement: Codeunit "Time Series Management";
    begin
        MLForecastSetup.Get();
        TimeSeriesManagement.Initialize(
            MLForecastSetup."Endpoint URI", MLForecastSetup."API Key",
            0, false);

        ItemLedgerEntry.SetRange("Item No.", Item."No.");
        ItemLedgerEntry.SetRange("Entry Type", ItemLedgerEntry."Entry
Type"::Sale);

        TimeSeriesManagement.PrepareData(
            ItemLedgerEntry,
            ItemLedgerEntry.FieldNo("Item No."),
            ItemLedgerEntry.FieldNo("Posting Date"),
            ItemLedgerEntry.FieldNo(Quantity),
            Date."Period Type"::Month,
            WorkDate,
            12);

        TimeSeriesManagement.Forecast(1, 0, 0);
        TimeSeriesManagement.GetForecast(TempTimeSeriesForecast);
        if TempTimeSeriesForecast.FindFirst() then
            exit(TempTimeSeriesForecast.Value);
    end;
}
```

Create page extension for the Item List page

Your next task is to provide additional user interface that allows users to invoke the function you have just written. The most intuitive and logical place for such user interface is a new action in the Item List page.

1. Press Ctrl+N to create a new object, and save it as “PageExtension 50110 Item List Extension.al”.
2. In the editor define the new object **pageextension 50110 "Item List Extension" extends "Item List"**.
3. In the object, define an action as last action of the **Item** action group. Set the following properties on the action:

```
                Caption = 'Get Monthly Sales Forecast';
                Image = Forecast;
                Promoted = true;
                PromotedCategory = Category4;
                ApplicationArea = All;
```

4. Define the **OnAction** trigger. From the trigger invoke the **CalculateForecast** function of the **Machine Learning Forecast** codeunit, and show the monthly sales forecast result in a message.

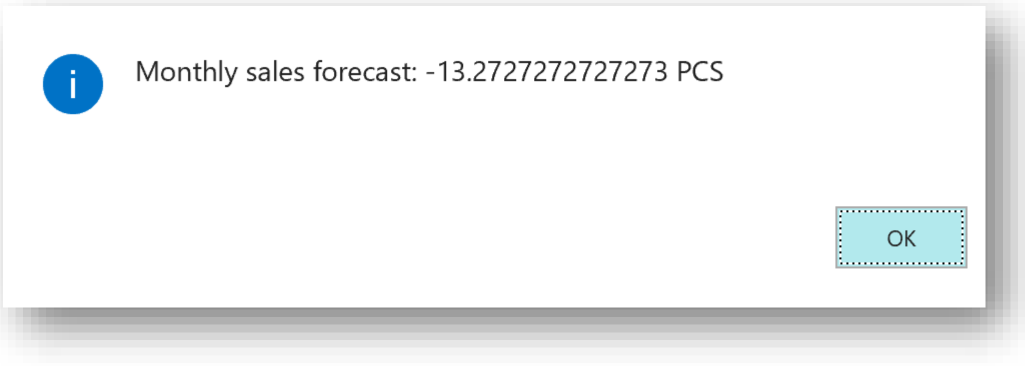
```
                trigger OnAction();
                var
                    MLForecast: Codeunit "Machine Learning Forecast";
                begin
                    Message('Monthly sales forecast: %1 %2',
                        MLForecast.CalculateForecast(Rec),
                        Rec."Base Unit of Measure");
                end;
```

5. Configure the launch.json configuration file to start the debugger on object Page 31.

```
            "startupObjectType": "Page",
            "startupObjectId": 31
```

6. Build, deploy, and run the extension.
7. In the **Item List**, make sure item 1896-S is selected.

8. Click **Item > Get Monthly Sales Forecast**. The application shows the following message on screen:


Monthly sales forecast: -13.2727272727273 PCS

Note: The system stores sales data as negative amounts. That's because each sale is a decrease of the inventory. Therefore, predicted sales forecast will show the expected decrease of the inventory. However, the users will intuitively expect sales figures to be expressed as positive values. Therefore, you must make sure that the dataset passed to the Azure Machine Learning algorithm contains positive, rather than negative values.

Solution: PageExtension 50110 Item List Extension.al

```
pageextension 50110 "Item List Extension" extends "Item List"
{
    actions
    {
        addlast(Item)
        {
            action(MonthlyForecast)
            {
                Caption = 'Get Monthly Sales Forecast';
                Image = Forecast;
                Promoted = true;
                PromotedCategory = Category4;
                ApplicationArea = All;

                trigger OnAction();
                var
                    MLForecast: Codeunit "Machine Learning Forecast";
                begin
                    Message('Monthly sales forecast: %1 %2',
                        MLForecast.CalculateForecast(Rec),
                        Rec."Base Unit of Measure");
                end;
            }
        }
    }
}
```

Correct the data passed to the Machine Learning algorithm

Since the value returned from the Machine Learning algorithm shows a negative value, where users expect a positive one, invert the sign for each entry passed to the algorithm before invoking it.

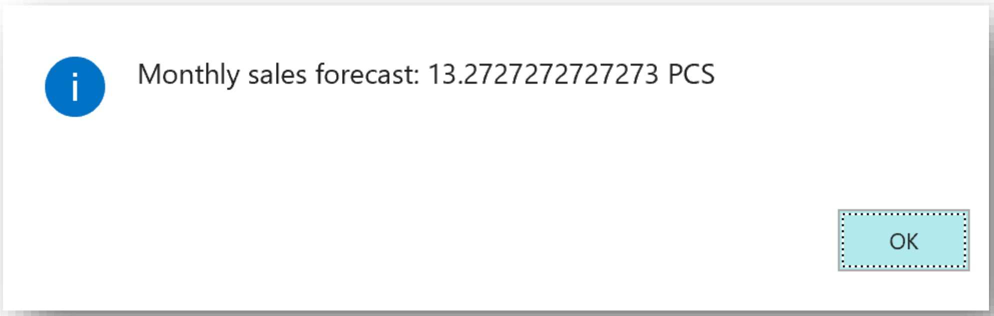
1. Open Codeunit 50110 Machine Learning Forecast.al.
2. In the CalculateForecast function, declare an additional temporary record variable of subtype **Time Series Buffer**.

```
        TempTimeSeriesBuffer: Record "Time Series Buffer" temporary;
```

3. Before invoking the TimeSeriesManagement.Forecast function, write code that gets prepared data from the TimeSeriesManagement codeunit and store that data in the TempTimeSeriesBuffer temporary table. Then, iterate through all records in this temporary table and invert the sign. Make sure to modify each record.

```
        TimeSeriesManagement.GetPreparedData(TempTimeSeriesBuffer);
        if TempTimeSeriesBuffer.FindSet() then
            repeat
                TempTimeSeriesBuffer.Value := -TempTimeSeriesBuffer.Value;
                TempTimeSeriesBuffer.Modify();
            until TempTimeSeriesBuffer.Next() = 0;
```

4. Build, deploy, and run the extension.
5. Test the monthly sales forecast function for item 1896-S once again. This time, the message will show this:


Monthly sales forecast: 13.2727272727273 PCS

Solution: Codeunit 50110 Machine Learning Forecast.al after modification

```
codeunit 50110 "Machine Learning Forecast"
{
    procedure CalculateForecast(Item: Record Item): Decimal;
    var
        MLForecastSetup: Record "ML Forecast Setup";
        ItemLedgerEntry: Record "Item Ledger Entry";
        TempTimeSeriesBuffer: Record "Time Series Buffer" temporary;
        Date: Record Date;
        TempTimeSeriesForecast: Record "Time Series Forecast" temporary;
        TimeSeriesManagement: Codeunit "Time Series Management";
    begin
        MLForecastSetup.Get();
        TimeSeriesManagement.Initialize(
            MLForecastSetup."Endpoint URI", MLForecastSetup."API Key",
            0, false);

        ItemLedgerEntry.SetRange("Item No.", Item."No.");
        ItemLedgerEntry.SetRange("Entry Type", ItemLedgerEntry."Entry
Type"::Sale);

        TimeSeriesManagement.PrepareData(
            ItemLedgerEntry,
            ItemLedgerEntry.FieldNo("Item No."),
            ItemLedgerEntry.FieldNo("Posting Date"),
            ItemLedgerEntry.FieldNo(Quantity),
            Date."Period Type"::Month,
            WorkDate,
            12);

        TimeSeriesManagement.GetPreparedData(TempTimeSeriesBuffer);
        if TempTimeSeriesBuffer.FindSet() then
            repeat
                TempTimeSeriesBuffer.Value := -TempTimeSeriesBuffer.Value;
                TempTimeSeriesBuffer.Modify();
            until TempTimeSeriesBuffer.Next() = 0;

        TimeSeriesManagement.Forecast(1, 0, 0);
        TimeSeriesManagement.GetForecast(TempTimeSeriesForecast);
        if TempTimeSeriesForecast.FindFirst() then
            exit(TempTimeSeriesForecast.Value);
    end;
}
```

Analyze the quality of sales predictions

In the previous exercise you have developed an extension for Business Central that invokes an Azure Machine Learning model to predict monthly sales forecasts based on historical data. However, the only number you have ever seen was a singular value representing predicted sales quantity for the next month.

In this exercise you will gain a deeper insight into the quality of sales predictions and will experiment with different prediction parameters and algorithms.

Extend the Machine Learning Forecast codeunit

Add a new function to the **Machine Learning Forecast** codeunit, that calculates the sales forecast for all items within the specified filter pattern. The function receives a temporary instance of the **Time Series Forecast** table, and it populates this table with forecast data after invoking the Machine Learning algorithm.

1. Open codeunit 50110 Machine Learning Forecast.
2. Create a new function **CalculateForecastBulk**, that receives two parameters, a text parameter for filtering the item ledger entries, and a temporary record for the **Time Series Forecast** table.

```
    procedure CalculateForecastBulk(ItemNoFilter: Text; var 
TempTimeSeriesForecast: Record "Time Series Forecast" temporary);
    begin

    end;
```

3. Declare the following variables for the function:

```
    MLForecastSetup: Record "ML Forecast Setup";
    ItemLedgerEntry: Record "Item Ledger Entry";
    Date: Record Date;
    TempTimeSeriesForecast2: Record "Time Series Forecast" temporary;
    TimeSeriesManagement: Codeunit "Time Series Management";
```

4. In the body of the function, check if the TempTimeSeriesForecast variable indeed represents a temporary table. If not, throw an error.

```
        if not TempTimeSeriesForecast.IsTemporary() then
            Error('TempTimeSeriesForecast must be temporary.');
```

5. Use the same code to initialize the **TimeSeriesManagement** codeunit variable that you used in the **CalculateForecast** function.
6. Write code that filters the ItemLedgerEntry record to include only items having **Entry Type** of **Sale**, that match the **ItemNoFilter** parameter.

```
        ItemLedgerEntry.SetRange("Entry Type", ItemLedgerEntry."Entry 
Type"::Sale);
        ItemLedgerEntry.SetFilter("Item No.", ItemNoFilter);
```

7. Write code that prepares data for the Machine Learning algorithm, and then invoke the forecast web service exactly as you wrote it for the **CalculateForecast** function. Do not write code to invert the quantity sign.
8. Get the forecast data into the **TempTimeSeriesForecast2** variable.

```
        TimeSeriesManagement.GetForecast(TempTimeSeriesForecast2);
```

Note: You must not use the **TempTimeSeriesForecast** variable here! You will copy data into that table in the next step.

9. Write code that empties **TempTimeSeriesForecast** table and then copies all the result retrieved from the forecast web service into the **TempTimeSeriesForecast** table.

```
        TempTimeSeriesForecast.Reset();
        TempTimeSeriesForecast.DeleteAll();
        if TempTimeSeriesForecast2.FindSet() then
            repeat
                TempTimeSeriesForecast := TempTimeSeriesForecast2;
                TempTimeSeriesForecast.Insert();
            until TempTimeSeriesForecast2.Next() = 0;
```

Solution: CalculateForecastBulk function in codeunit 50110 "Machine Learning Forecast"

```
    procedure CalculateForecastBulk(ItemNoFilter: Text; var
TempTimeSeriesForecast: Record "Time Series Forecast" temporary);
    var
        MLForecastSetup: Record "ML Forecast Setup";
        ItemLedgerEntry: Record "Item Ledger Entry";
        Date: Record Date;
        TempTimeSeriesForecast2: Record "Time Series Forecast" temporary;
        TimeSeriesManagement: Codeunit "Time Series Management";
    begin
        if not TempTimeSeriesForecast.IsTemporary() then
            Error('TempTimeSeriesForecast must be temporary.');

        MLForecastSetup.Get();
        TimeSeriesManagement.Initialize(
            MLForecastSetup."Endpoint URI", MLForecastSetup."API Key",
            0, false);

        ItemLedgerEntry.SetRange("Entry Type", ItemLedgerEntry."Entry
Type"::Sale);
        ItemLedgerEntry.SetFilter("Item No.", ItemNoFilter);

        TimeSeriesManagement.PrepareData(
           ItemLedgerEntry,
           ItemLedgerEntry.FieldNo("Item No."),
           ItemLedgerEntry.FieldNo("Posting Date"),
           ItemLedgerEntry.FieldNo(Quantity),
           Date."Period Type"::Month,
           WorkDate,
           12);

        TimeSeriesManagement.Forecast(1, 0, 0);
        TimeSeriesManagement.GetForecast(TempTimeSeriesForecast2);

        TempTimeSeriesForecast.Reset();
        TempTimeSeriesForecast.DeleteAll();
        if TempTimeSeriesForecast2.FindSet() then
            repeat
                TempTimeSeriesForecast := TempTimeSeriesForecast2;
                TempTimeSeriesForecast.Insert();
            until TempTimeSeriesForecast2.Next() = 0;
    end;
```

Create the Prediction Overview page

You start by creating a list page over the **Time Series Forecast** table. This is a standard table that Business Central uses to store prediction data.

1. Press Ctrl+N to create a new object, and save it as "Page 50111 Prediction Overview.al".
2. In the editor define the new object **page 50111 "Prediction Overview"**.
3. Define the page to be of **List** type, set the page caption to 'Prediction Overview', make the page show data from table **Time Series Forecast**, make sure that the page shows in the **Tell me** feature in the **Tasks** group, make sure that the page shows data from the temporary table, make sure that insert, modify, and delete are not allowed, and define the **ApplicationArea** property.

```
    PageType = List;
    Caption = 'Prediction Overview';
    SourceTable = "Time Series Forecast";
    UsageCategory = Tasks;
    SourceTableTemporary = true;
    InsertAllowed = false;
    ModifyAllowed = false;
    DeleteAllowed = false;
    ApplicationArea = All;
```

4. Define the page layout to contain one repeater in the Content area. Make sure that users cannot edit data in the repeater.

```
    layout
    {
        area(Content)
        {
            repeater(Lines)
            {
                Editable = false;
            }
        }
    }
```

5. Inside the repeater, define field controls for the following fields: **Group ID**, **Period No.**, **Period Start Date**, **Value**, **Delta**, and **Delta %**. For each of these fields, set the **ApplicationArea** property.
6. Declare a text variable to hold the item number filter.

```
    var
        ItemNoFilter: Text;
```

7. Inside the Content area, but above the repeater, create a field control for the ItemNoFilter variable. Define its caption, and Application Area properties.

```
            field(ItemNoFilter; ItemNoFilter)
            {
                Caption = 'Item No. Filter';
                ApplicationArea = All;
            }
```

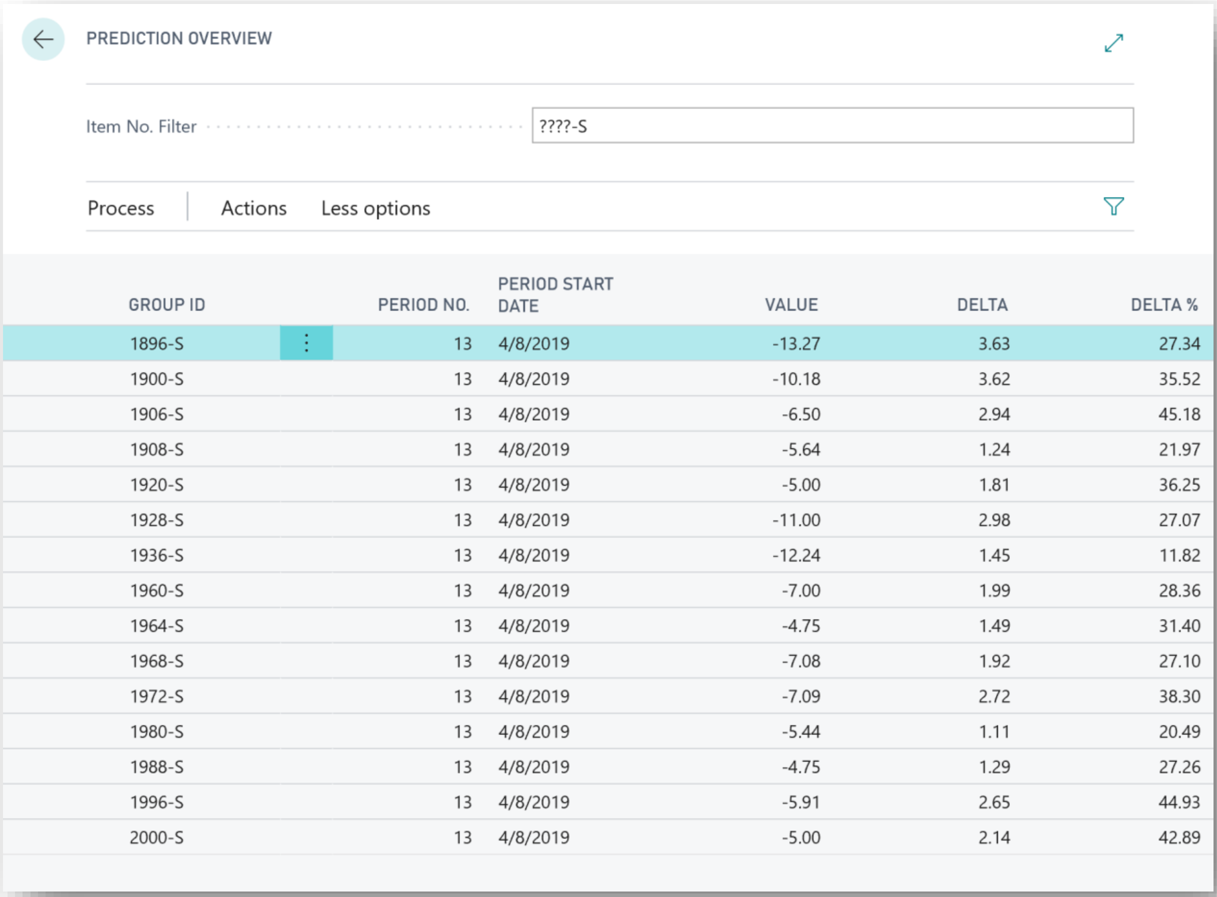
8. Define the page actions section to contain one action in the Processing area. On the action, define the following properties:

```
                Caption = 'Calculate Forecast';
                Image = CalculatePlan;
                Promoted = true;
                PromotedCategory = Process;
                ApplicationArea = All;
```

9. Define the OnAction trigger. In it, declare a local variable for the Machine Learning Forecast codeunit, and then invoke the CalculateForecastBulk function by passing the ItemNoFilter and Rec as its parameters. At the end, move the position to the first row in the Rec record if possible.

```
                trigger OnAction();
                var
                    MLForecast: Codeunit "Machine Learning Forecast";
                begin
                    MLForecast.CalculateForecastBulk(ItemNoFilter, Rec);
                    if Rec.FindFirst() then;
                end;
```

10. Build, deploy, and run the extension.
11. From the **Tell me** feature, search for the **Prediction Overview** page and then run it.
12. In the **Prediction Overview** page, set the **Item No. Filter** to '????-S' and then click **Actions > Calculate Forecast**. After a short while, the page will display the forecast data:


PREDICTION OVERVIEW

Item No. Filter ????-S

Process | Actions | Less options

GROUP ID	PERIOD NO.	PERIOD START DATE	VALUE	DELTA	DELTA %
1896-S	13	4/8/2019	-13.27	3.63	27.34
1900-S	13	4/8/2019	-10.18	3.62	35.52
1906-S	13	4/8/2019	-6.50	2.94	45.18
1908-S	13	4/8/2019	-5.64	1.24	21.97
1920-S	13	4/8/2019	-5.00	1.81	36.25
1928-S	13	4/8/2019	-11.00	2.98	27.07
1936-S	13	4/8/2019	-12.24	1.45	11.82
1960-S	13	4/8/2019	-7.00	1.99	28.36
1964-S	13	4/8/2019	-4.75	1.49	31.40
1968-S	13	4/8/2019	-7.08	1.92	27.10
1972-S	13	4/8/2019	-7.09	2.72	38.30
1980-S	13	4/8/2019	-5.44	1.11	20.49
1988-S	13	4/8/2019	-4.75	1.29	27.26
1996-S	13	4/8/2019	-5.91	2.65	44.93
2000-S	13	4/8/2019	-5.00	2.14	42.89

Group ID contains the item number, **Period No.** is the predicted period number with respect to the number of past periods taken into account. Since you used 12 past periods and 1 period of forecasting, the Period No. shows 13. If you call `TimeSeriesManagement.Forecast(2, 0, 0)`, there would be two periods (13 and 14) displayed for each item. **Value** is the predicted quantity (in this example you didn't change the sign, so this shows the predicted decrease). **Delta** shows the range in which the predicted value will fall with the default confidence level of 80%, and **Delta %** shows the magnitude of the **Delta** as compared to **Value**.

In the example above, for item 1896-S, there is 80% probability that the sales will be 13.27 ±3.63 (between 9.64 and 16.9).

Solution: Page 50111 Prediction Overview.al

```
page 50111 "Prediction Overview"
{
    PageType = Worksheet;
    Caption = 'Prediction Overview';
    SourceTable = "Time Series Forecast";
    UsageCategory = Tasks;
    SourceTableTemporary = true;
    InsertAllowed = false;
    ModifyAllowed = false;
    DeleteAllowed = false;
    ApplicationArea = All;

    layout
    {
        area(Content)
        {
            field(ItemNoFilter; ItemNoFilter)
            {
                Caption = 'Item No. Filter';
                ApplicationArea = All;
            }

            repeater(Lines)
            {
                Editable = false;
                field("Group ID"; "Group ID")
                {
                    ApplicationArea = All;
                }
                field("Period No."; "Period No.")
                {
                    ApplicationArea = All;
                }
                field("Period Start Date"; "Period Start Date")
                {
                    ApplicationArea = All;
                }
                field(Value; Value)
                {
                    ApplicationArea = All;
                }
                field(Delta; Delta)
                {
                    ApplicationArea = All;
                }
                field("Delta %"; "Delta %")
                {
                    ApplicationArea = All;
                }
            }
        }
    }

    actions
    {
        area(Processing)
        {
            action(CalculateForecastBulk)
            {
                Caption = 'Calculate Forecast';
                Image = CalculatePlan;
                Promoted = true;
                PromotedCategory = Process;
                ApplicationArea = All;

                trigger OnAction();
                var
                    MLForecast: Codeunit "Machine Learning Forecast";
                begin
                    MLForecast.CalculateForecastBulk(ItemNoFilter, Rec);
                    if Rec.FindFirst() then;
                end;
            }
        }
    }

    var
        ItemNoFilter: Text;
}
```

Allow experimenting with predictions

So far, you have always invoked the Machine Learning algorithm with default or hardcoded parameters. However, the algorithm behind your web service endpoint allows a lot more flexibility. In this exercise, you'll allow the end users to tweak the results by modifying the parameters of the Machine Learning algorithm.

1. Open Page 50111 Prediction Overview.al.
2. Declare the following global variables:

```
    PeriodType: Option Date,Week,Month,Quarter,Year;
    NumberOfForecastPeriods: Integer;
    NumberOfPastPeriods: Integer;
    ConfidenceLevel: Integer;
    ForecastAlgorithm: Option ARIMA,ETS,STL,"ETS+ARIMA","ETS+STL",ALL;
```

3. Just above the repeater and below the **Item No. Filter** field, declare field controls for each of the variables defined in the previous step. Make sure to define Caption and ApplicationArea properties for each of them.

```
            field(PeriodType; PeriodType)
            {
                Caption = 'Period Type';
                ApplicationArea = All;
            }
            field(NoForecastPeriods; NumberOfForecastPeriods)
            {
                Caption = 'No. of Forecast Periods';
                ApplicationArea = All;
            }
            field(NoPastPeriods; NumberOfPastPeriods)
            {
                Caption = 'No. of Past Periods';
                ApplicationArea = All;
            }
            field(ConfidenceLevel; ConfidenceLevel)
            {
                Caption = 'Confidence Level';
                ApplicationArea = All;
            }
            field(Algorithm; ForecastAlgorithm)
            {
                Caption = 'Forecast Algorithm';
                ApplicationArea = All;
            }
```

4. Define the OnInit trigger to initialize the variables to their default values that match the parameters used by the forecast algorithm so far:

```
    trigger OnInit();
    begin
        PeriodType := PeriodType::Month;
        NumberOfForecastPeriods := 1;
        NumberOfPastPeriods := 12;
        ConfidenceLevel := 80;
    end;
```

5. Open Codeunit 50110 Machine Learning Forecast.al.
6. Change the signature of the **CalculateForecastBulk** function by adding parameters with the same name and type as the additional variables you just declared in the **Prediction Overview** page.

```
procedure CalculateForecastBulk(
    ItemNoFilter: Text;
    PeriodType: Option Date,Week,Month,Quarter,Year;
    NumberOfForecastPeriods: Integer;
    NumberOfPastPeriods: Integer;
    ConfidenceLevel: Integer;
    ForecastAlgorithm: Option ARIMA,ETS,STL,"ETS+ARIMA","ETS+STL",ALL;
    var TempTimeSeriesForecast: Record "Time Series Forecast"
temporary);
```

7. Modify the **PrepareData** invocation so that instead of hardcoded period of 12 months, you pass the period type and number of past periods provided as function arguments. Also, remove the **Date** local variable as the code no longer needs it.

```
        TimeSeriesManagement.PrepareData(
           ItemLedgerEntry,
           ItemLedgerEntry.FieldNo("Item No."),
           ItemLedgerEntry.FieldNo("Posting Date"),
           ItemLedgerEntry.FieldNo(Quantity),
           PeriodType,
           WorkDate,
            NumberOfPastPeriods);
```

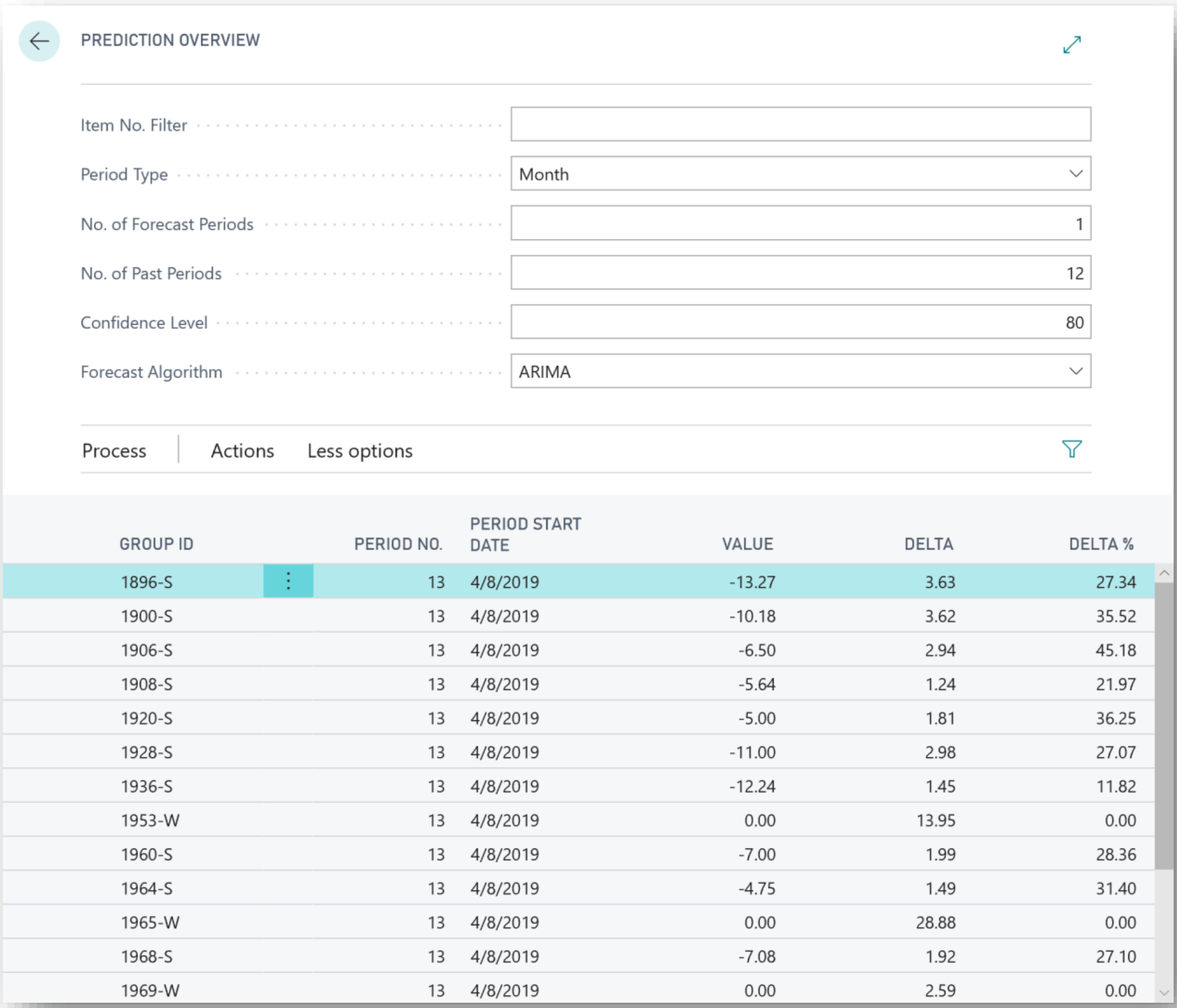
8. Modify the **Forecast** function invocation by removing the hardcoded values and replacing them with the values passed as arguments.

```
        TimeSeriesManagement.Forecast(
            NumberOfForecastPeriods,
            ConfidenceLevel,
            ForecastAlgorithm);
```

9. Open Page 50111 Prediction Overview.al.
10. The object does not compile, because of the error inside the **OnAction** trigger. Modify the **OnAction** trigger by providing the correct parameters to the CalculateForecastBulk function.

```
                    MLForecast.CalculateForecastBulk(
                        ItemNoFilter,
                        PeriodType,
                        NumberOfForecastPeriods,
                        NumberOfPastPeriods,
                        ConfidenceLevel,
                        ForecastAlgorithm,
                        Rec);
```

11. Build, deploy, and run the extension.
12. Run page **Prediction Overview** (again by using the **Tell me** feature), then invoke the **Calculate Forecast** action with the default values.



PREDICTION OVERVIEW

Item No. Filter	
Period Type	Month
No. of Forecast Periods	1
No. of Past Periods	12
Confidence Level	80
Forecast Algorithm	ARIMA

GROUP ID	PERIOD NO.	PERIOD START DATE	VALUE	DELTA	DELTA %
1896-S	13	4/8/2019	-13.27	3.63	27.34
1900-S	13	4/8/2019	-10.18	3.62	35.52
1906-S	13	4/8/2019	-6.50	2.94	45.18
1908-S	13	4/8/2019	-5.64	1.24	21.97
1920-S	13	4/8/2019	-5.00	1.81	36.25
1928-S	13	4/8/2019	-11.00	2.98	27.07
1936-S	13	4/8/2019	-12.24	1.45	11.82
1953-W	13	4/8/2019	0.00	13.95	0.00
1960-S	13	4/8/2019	-7.00	1.99	28.36
1964-S	13	4/8/2019	-4.75	1.49	31.40
1965-W	13	4/8/2019	0.00	28.88	0.00
1968-S	13	4/8/2019	-7.08	1.92	27.10
1969-W	13	4/8/2019	0.00	2.59	0.00

13. Change a few parameters, and invoke the forecast again, to see how the values change. For example, try forecasting three months ahead, or increasing confidence level to 85.

Note: The quality of the forecast data depends on the quality of the historical data based on which the predictions are being made. Since our example is using only a small number of historical transactions, always with exactly one month between each two, it makes no sense to predict on periods shorter than months.

Increasing confidence level necessarily increases the delta, because to predict with higher confidence, the algorithm needs more margin for error. That’s why increasing confidence too high may result in useless numbers. The higher the Delta %, the less meaningful the result.

There are three different algorithms that you can use, plus their combinations. When you choose combinations, Azure will run all selected algorithms, and then select values that provide the most accurate predictions (lowest Delta %). However, running more algorithms requires more processing power, and costs more. Choosing ALL runs all three algorithms, but provides the most accurate predictions.

Solution: Codeunit 50110 Machine Learning Forecast.al after modification

```
codeunit 50110 "Machine Learning Forecast"
{
    procedure CalculateForecast(Item: Record Item): Decimal;
    var
        MLForecastSetup: Record "ML Forecast Setup";
        ItemLedgerEntry: Record "Item Ledger Entry";
        TempTimeSeriesBuffer: Record "Time Series Buffer" temporary;
        Date: Record Date;
        TempTimeSeriesForecast: Record "Time Series Forecast" temporary;
        TimeSeriesManagement: Codeunit "Time Series Management";
    begin
        MLForecastSetup.Get();
        TimeSeriesManagement.Initialize(
            MLForecastSetup."Endpoint URI", MLForecastSetup."API Key",
            0, false);

        ItemLedgerEntry.SetRange("Item No.", Item."No.");
        ItemLedgerEntry.SetRange("Entry Type", ItemLedgerEntry."Entry
Type"::Sale);

        TimeSeriesManagement.PrepareData(
            ItemLedgerEntry,
            ItemLedgerEntry.FieldNo("Item No."),
            ItemLedgerEntry.FieldNo("Posting Date"),
            ItemLedgerEntry.FieldNo(Quantity),
            Date."Period Type"::Month,
            WorkDate,
            12);

        TimeSeriesManagement.GetPreparedData(TempTimeSeriesBuffer);
        if TempTimeSeriesBuffer.FindSet() then
            repeat
                TempTimeSeriesBuffer.Value := -TempTimeSeriesBuffer.Value;
                TempTimeSeriesBuffer.Modify();
            until TempTimeSeriesBuffer.Next() = 0;

        TimeSeriesManagement.Forecast(1, 0, 0);
        TimeSeriesManagement.GetForecast(TempTimeSeriesForecast);
        if TempTimeSeriesForecast.FindFirst() then
            exit(TempTimeSeriesForecast.Value);
    end;

    procedure CalculateForecastBulk(
        ItemNoFilter: Text;
        PeriodType: Option Date,Week,Month,Quarter,Year;
        NumberOfForecastPeriods: Integer;
        NumberOfPastPeriods: Integer;
        ConfidenceLevel: Integer;
        ForecastAlgorithm: Option ARIMA,ETS,STL,"ETS+ARIMA","ETS+STL",ALL;
        var TempTimeSeriesForecast: Record "Time Series Forecast" temporary);
    var
        MLForecastSetup: Record "ML Forecast Setup";
        ItemLedgerEntry: Record "Item Ledger Entry";
        TempTimeSeriesForecast2: Record "Time Series Forecast" temporary;
        TimeSeriesManagement: Codeunit "Time Series Management";
    begin
        if not TempTimeSeriesForecast.IsTemporary() then
            Error('TempTimeSeriesForecast must be temporary.');

        MLForecastSetup.Get();
        TimeSeriesManagement.Initialize(
            MLForecastSetup."Endpoint URI", MLForecastSetup."API Key",
            0, false);

        ItemLedgerEntry.SetRange("Entry Type", ItemLedgerEntry."Entry
Type"::Sale);
        ItemLedgerEntry.SetFilter("Item No.", ItemNoFilter);

        TimeSeriesManagement.PrepareData(
           ItemLedgerEntry,
           ItemLedgerEntry.FieldNo("Item No."),
           ItemLedgerEntry.FieldNo("Posting Date"),
           ItemLedgerEntry.FieldNo(Quantity),
           PeriodType,
           WorkDate,
           NumberOfPastPeriods);

        TimeSeriesManagement.Forecast(
            NumberOfForecastPeriods,
            ConfidenceLevel,
            ForecastAlgorithm);
        TimeSeriesManagement.GetForecast(TempTimeSeriesForecast2);

        TempTimeSeriesForecast.Reset();
        TempTimeSeriesForecast.DeleteAll();
        if TempTimeSeriesForecast2.FindSet() then
            repeat
                TempTimeSeriesForecast := TempTimeSeriesForecast2;
                TempTimeSeriesForecast.Insert();
            until TempTimeSeriesForecast2.Next() = 0;
    end;
}
```

Solution: Page 50111 Prediction Overview.al after modification

```
page 50111 "Prediction Overview"
{
    PageType = Worksheet;
    Caption = 'Prediction Overview';
    SourceTable = "Time Series Forecast";
    UsageCategory = Tasks;
    SourceTableTemporary = true;
    InsertAllowed = false;
    ModifyAllowed = false;
    DeleteAllowed = false;
    ApplicationArea = All;

    layout
    {
        area(Content)
        {
            field(ItemNoFilter; ItemNoFilter)
            {
                Caption = 'Item No. Filter';
                ApplicationArea = All;
            }
            field(PeriodType; PeriodType)
            {
                Caption = 'Period Type';
                ApplicationArea = All;
            }
            field(NoForecastPeriods; NumberOfForecastPeriods)
            {
                Caption = 'No. of Forecast Periods';
                ApplicationArea = All;
            }
            field(NoPastPeriods; NumberOfPastPeriods)
            {
                Caption = 'No. of Past Periods';
                ApplicationArea = All;
            }
            field(ConfidenceLevel; ConfidenceLevel)
            {
                Caption = 'Confidence Level';
                ApplicationArea = All;
            }
            field(Algorithm; ForecastAlgorithm)
            {
                Caption = 'Forecast Algorithm';
                ApplicationArea = All;
            }

            repeater(Lines)
            {
                Editable = false;
                field("Group ID"; "Group ID")
                {
                    ApplicationArea = All;
                }
                field("Period No."; "Period No.")
                {
                    ApplicationArea = All;
                }
                field("Period Start Date"; "Period Start Date")
                {
                    ApplicationArea = All;
                }
                field(Value; Value)
                {
                    ApplicationArea = All;
                }
                field(Delta; Delta)
                {
                    ApplicationArea = All;
                }
                field("Delta %"; "Delta %")
                {
                    ApplicationArea = All;
                }
            }
        }
    }

    actions
    {
        area(Processing)
        {
            action(CalculateForecastBulk)
            {
                Caption = 'Calculate Forecast';
                Image = CalculatePlan;
                Promoted = true;
                PromotedCategory = Process;
                ApplicationArea = All;

                trigger OnAction();
                var
                    MLForecast: Codeunit "Machine Learning Forecast";
                begin
                    MLForecast.CalculateForecastBulk(
                        ItemNoFilter,
                        PeriodType,
                        NumberOfForecastPeriods,
                        NumberOfPastPeriods,
                        ConfidenceLevel,
                        ForecastAlgorithm,
                        Rec);
                    if Rec.FindFirst() then;
                end;
            }
        }
    }

    var
        ItemNoFilter: Text;
        PeriodType: Option Date,Week,Month,Quarter,Year;
        NumberOfForecastPeriods: Integer;
        NumberOfPastPeriods: Integer;
        ConfidenceLevel: Integer;
        ForecastAlgorithm: Option ARIMA,ETS,STL,"ETS+ARIMA","ETS+STL",ALL;

    trigger OnInit();
    begin
        PeriodType := PeriodType::Month;
        NumberOfForecastPeriods := 1;
        NumberOfPastPeriods := 12;
        ConfidenceLevel := 80;
    end;
}
```

Take action based on forecast results

You have just completed introducing a new machine learning algorithm into your business planning. At this point it allows you to detect patterns in your past sales trends and forecast future sales based on those trends. However, the only result you have is numbers presented on screen. Your last task is to enable creating a purchase order based on sales forecast, so that you can make sure to satisfy the expected sales demand.

1. Open Codeunit 50110 Machine Learning Forecast.al.
2. Create a new function called **CreatePurchaseOrder** that receives three parameters. The first is a by-reference temporary record variable of subtype Time Series Forecast; the second is a decimal variable to indicate the quality bar that limits which items will be selected for purchase (predictions above the quality bar should be rejected); and the third represents Vendor No.

```
    procedure CreatePurchaseOrder(var TempTimeSeriesForecast: Record "Time
Series Forecast"; QualityBar: Decimal; VendorNo: Code[20]);
    begin

    end;
```

3. Declare two local variables for the **Purchase Header** and **Purchase Line** tables.

```
        PurchHeader: Record "Purchase Header";
        PurchLine: Record "Purchase Line";
```

4. In the body of the function, filter the TempTimeSeriesForecast record to include only those lines with non-zero Value.
5. If there are any lines in the filtered set, create a new purchase order from the specified vendor. Then, for each line that has Delta % less than or equal to the quality bar, or there was no quality bar provided, insert a new purchase line with corresponding quantity.

```
TempTimeSeriesForecast.SetFilter(Value, '<>0');
if TempTimeSeriesForecast.FindSet() then begin
    PurchHeader.Validate("Document Type", PurchHeader."Document
Type"::Order);
    PurchHeader.Validate("Buy-from Vendor No.", VendorNo);
    PurchHeader.Insert(true);
    repeat
        if (TempTimeSeriesForecast."Delta %" <= QualityBar) or
(QualityBar = 0) then begin
            PurchLine.Init();
            PurchLine.Validate("Document Type", PurchHeader."Document
Type");
            PurchLine.Validate("Document No.", PurchHeader."No.");
            PurchLine.Validate("Line No.", PurchLine."Line No." +
10000);
            PurchLine.Validate(Type, PurchLine.Type::Item);
            PurchLine.Validate("No.", TempTimeSeriesForecast."Group
ID");
            PurchLine.Validate(Quantity,
                Round(-TempTimeSeriesForecast.Value, 1));
            PurchLine.Insert(true);
        end;
    until TempTimeSeriesForecast.Next() = 0;
    Page.Run(Page::"Purchase Order", TempTimeSeriesForecast);
end;
```

6. Open Page 50111 Prediction Overview.al.
7. Declare the following two variables:

```
        VendorNo: Code[20];
        QualityBar: Decimal;
```

8. Group all existing fields above the repeater into a new group named **Prediction**.

```
            group(Prediction)
            {
                Caption = 'Prediction';

                // Move fields here

            }
```

9. Below the **Prediction** group and above the repeater, create a new group, and call it Purchase Order.

```
            group(PurchaseOrder)
            {
                Caption = 'Purchase Order';
            }
```

10. Inside the **Purchase Order** group, create field controls for the VendorNo and QualityBar variables. Make sure to set the correct **TableRelation** property for the VendorNo field.

```
                field(VendorNo; VendorNo)
                {
                    Caption = 'Vendor No.';
                    TableRelation = Vendor;
                    ApplicationArea = All;
                }

                field(QualityBar; QualityBar)
                {
                    Caption = 'Quality Bar';
                    ApplicationArea = All;
                }
```

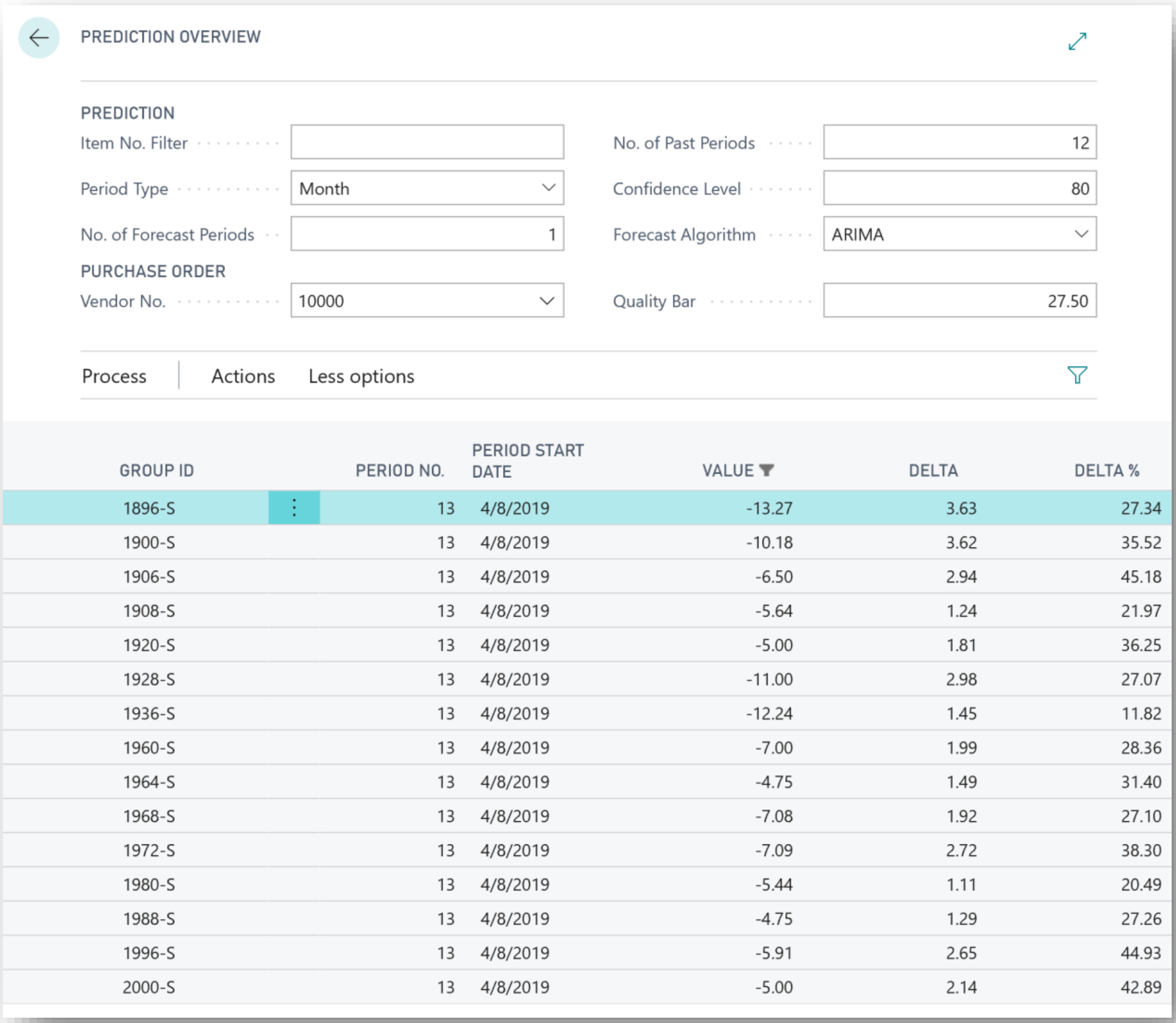
11. Create a new action and name it **Create Purchase Order**. Define its **Image**, **Promoted**, **PromotedCategory**, and **ApplicationArea** properties.

```
            action(CreatePO)
            {
                Caption = 'Create Purchase Order';
                Image = NewPurchaseInvoice;
                Promoted = true;
                PromotedCategory = Process;
                ApplicationArea = All;
            }
```

12. Define the **OnAction** trigger for this action. In the trigger, declare a local variable for the **Machine Learning Forecast** codeunit. Inside the trigger, invoke the CreatePurchaseOrder function with appropriate parameters.

```
        trigger OnAction();
        var
            MLForecast: Codeunit "Machine Learning Forecast";
        begin
            MLForecast.CreatePurchaseOrder(Rec, QualityBar, VendorNo);
        end;
```

13. Build, deploy, and run the extension.
14. Search for the **Prediction Overview** page using the **Tell me** feature.
15. In the **Prediction Overview** page, click **Process > Calculate Forecast**, to calculate the sales forecast based on default values.


PREDICTION OVERVIEW

PREDICTION

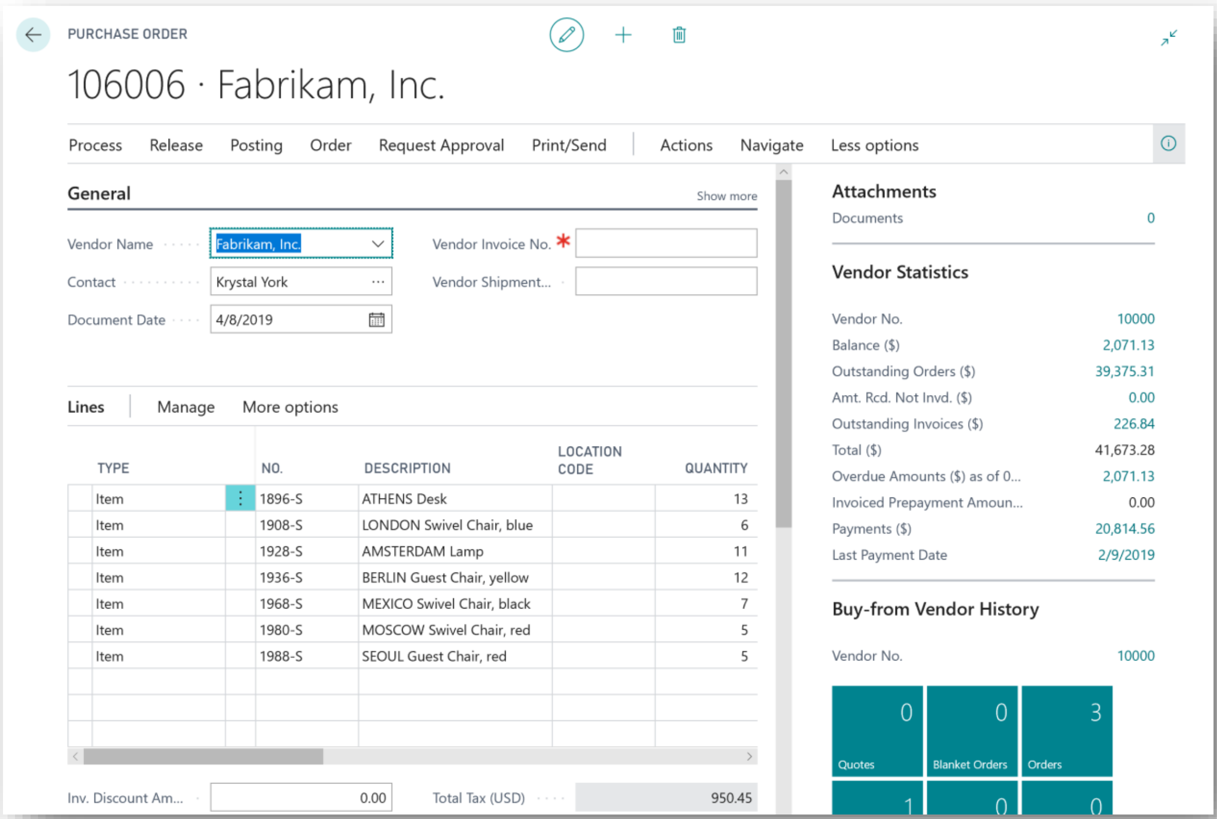
Item No. Filter		No. of Past Periods	12
Period Type	Month	Confidence Level	80
No. of Forecast Periods	1	Forecast Algorithm	ARIMA

PURCHASE ORDER

Vendor No.	10000	Quality Bar	27.50

GROUP ID	PERIOD NO.	PERIOD START DATE	VALUE	DELTA	DELTA %
1896-S	13	4/8/2019	-13.27	3.63	27.34
1900-S	13	4/8/2019	-10.18	3.62	35.52
1906-S	13	4/8/2019	-6.50	2.94	45.18
1908-S	13	4/8/2019	-5.64	1.24	21.97
1920-S	13	4/8/2019	-5.00	1.81	36.25
1928-S	13	4/8/2019	-11.00	2.98	27.07
1936-S	13	4/8/2019	-12.24	1.45	11.82
1960-S	13	4/8/2019	-7.00	1.99	28.36
1964-S	13	4/8/2019	-4.75	1.49	31.40
1968-S	13	4/8/2019	-7.08	1.92	27.10
1972-S	13	4/8/2019	-7.09	2.72	38.30
1980-S	13	4/8/2019	-5.44	1.11	20.49
1988-S	13	4/8/2019	-4.75	1.29	27.26
1996-S	13	4/8/2019	-5.91	2.65	44.93
2000-S	13	4/8/2019	-5.00	2.14	42.89

16. In the **Vendor No.** field, look up vendor 10000. In the **Quality Bar** field, enter “27.5”.
17. Click **Process > Create Purchase Order**. The application creates and opens the purchase order:


PURCHASE ORDER

106006 · Fabrikam, Inc.

Process Release Posting Order Request Approval Print/Send Actions Navigate Less options

General

Vendor Name	Fabrikam, Inc.	Vendor Invoice No. *	
Contact	Krystal York	Vendor Shipment...	
Document Date	4/8/2019		

Lines Manage More options

TYPE	NO.	DESCRIPTION	LOCATION CODE	QUANTITY
Item	1896-S	ATHENS Desk		13
Item	1908-S	LONDON Swivel Chair, blue		6
Item	1928-S	AMSTERDAM Lamp		11
Item	1936-S	BERLIN Guest Chair, yellow		12
Item	1968-S	MEXICO Swivel Chair, black		7
Item	1980-S	MOSCOW Swivel Chair, red		5
Item	1988-S	SEOUL Guest Chair, red		5

Inv. Discount Am... 0.00 Total Tax (USD) 950.45

Attachments

Documents	0

Vendor Statistics

Vendor No.	10000
Balance ($)	2,071.13
Outstanding Orders ($)	39,375.31
Amt. Rcd. Not Invd. ($)	0.00
Outstanding Invoices ($)	226.84
Total ($)	41,673.28
Overdue Amounts ($) as of 0...	2,071.13
Invoiced Prepayment Amoun...	0.00
Payments ($)	20,814.56
Last Payment Date	2/9/2019

Buy-from Vendor History

Vendor No.	10000

Quotes	Blanket Orders	Orders
0	0	3
1	0	0

Solution: Codeunit 50110 Machine Learning Forecast.al after modification

```
codeunit 50110 "Machine Learning Forecast"
{
    procedure CalculateForecast(Item: Record Item): Decimal;
    var
        MLForecastSetup: Record "ML Forecast Setup";
        ItemLedgerEntry: Record "Item Ledger Entry";
        TempTimeSeriesBuffer: Record "Time Series Buffer" temporary;
        Date: Record Date;
        TempTimeSeriesForecast: Record "Time Series Forecast" temporary;
        TimeSeriesManagement: Codeunit "Time Series Management";
    begin
        MLForecastSetup.Get();
        TimeSeriesManagement.Initialize(
            MLForecastSetup."Endpoint URI", MLForecastSetup."API Key",
            0, false);

        ItemLedgerEntry.SetRange("Item No.", Item."No.");
        ItemLedgerEntry.SetRange("Entry Type", ItemLedgerEntry."Entry
Type"::Sale);

        TimeSeriesManagement.PrepareData(
            ItemLedgerEntry,
            ItemLedgerEntry.FieldNo("Item No."),
            ItemLedgerEntry.FieldNo("Posting Date"),
            ItemLedgerEntry.FieldNo(Quantity),
            Date."Period Type"::Month,
            WorkDate,
            12);

        TimeSeriesManagement.GetPreparedData(TempTimeSeriesBuffer);
        if TempTimeSeriesBuffer.FindSet() then
            repeat
                TempTimeSeriesBuffer.Value := -TempTimeSeriesBuffer.Value;
                TempTimeSeriesBuffer.Modify();
            until TempTimeSeriesBuffer.Next() = 0;

        TimeSeriesManagement.Forecast(1, 0, 0);
        TimeSeriesManagement.GetForecast(TempTimeSeriesForecast);
        if TempTimeSeriesForecast.FindFirst() then
            exit(TempTimeSeriesForecast.Value);
    end;

    procedure CalculateForecastBulk(
        ItemNoFilter: Text;
        PeriodType: Option Date,Week,Month,Quarter,Year;
        NumberOfForecastPeriods: Integer;
        NumberOfPastPeriods: Integer;
        ConfidenceLevel: Integer;
        ForecastAlgorithm: Option ARIMA,ETS,STL,"ETS+ARIMA","ETS+STL",ALL;
        var TempTimeSeriesForecast: Record "Time Series Forecast" temporary);
    var
        MLForecastSetup: Record "ML Forecast Setup";
        ItemLedgerEntry: Record "Item Ledger Entry";
        TempTimeSeriesForecast2: Record "Time Series Forecast" temporary;
        TimeSeriesManagement: Codeunit "Time Series Management";
    begin
        if not TempTimeSeriesForecast.IsTemporary() then
            Error('TempTimeSeriesForecast must be temporary.');

        MLForecastSetup.Get();
        TimeSeriesManagement.Initialize(
            MLForecastSetup."Endpoint URI", MLForecastSetup."API Key",
            0, false);

        ItemLedgerEntry.SetRange("Entry Type", ItemLedgerEntry."Entry
Type"::Sale);
        ItemLedgerEntry.SetFilter("Item No.", ItemNoFilter);

        TimeSeriesManagement.PrepareData(
           ItemLedgerEntry,
           ItemLedgerEntry.FieldNo("Item No."),
           ItemLedgerEntry.FieldNo("Posting Date"),
           ItemLedgerEntry.FieldNo(Quantity),
           PeriodType,
           WorkDate,
           NumberOfPastPeriods);

        TimeSeriesManagement.Forecast(
            NumberOfForecastPeriods,
            ConfidenceLevel,
            ForecastAlgorithm);
        TimeSeriesManagement.GetForecast(TempTimeSeriesForecast2);

        TempTimeSeriesForecast.Reset();
        TempTimeSeriesForecast.DeleteAll();
        if TempTimeSeriesForecast2.FindSet() then
            repeat
                TempTimeSeriesForecast := TempTimeSeriesForecast2;
                TempTimeSeriesForecast.Insert();
            until TempTimeSeriesForecast2.Next() = 0;
    end;

    procedure CreatePurchaseOrder(var TempTimeSeriesForecast: Record "Time
Series Forecast"; QualityBar: Decimal; VendorNo: Code[20]);
    var
        PurchHeader: Record "Purchase Header";
        PurchLine: Record "Purchase Line";
    begin
        TempTimeSeriesForecast.SetFilter(Value, '<>0');
        if TempTimeSeriesForecast.FindSet() then begin
            PurchHeader.Validate("Document Type", PurchHeader."Document
Type"::Order);
            PurchHeader.Validate("Buy-from Vendor No.", VendorNo);
            PurchHeader.Insert(true);

            repeat
                if (TempTimeSeriesForecast."Delta %" <= QualityBar) or
(QualityBar = 0) then begin
                    PurchLine.Init();
                    PurchLine.Validate("Document Type", PurchHeader."Document
Type");
                    PurchLine.Validate("Document No.", PurchHeader."No.");
                    PurchLine.Validate("Line No.", PurchLine."Line No." +
10000);
                    PurchLine.Validate(Type, PurchLine.Type::Item);
                    PurchLine.Validate("No.", TempTimeSeriesForecast."Group
ID");
                    PurchLine.Validate(Quantity,
                        Round(-TempTimeSeriesForecast.Value, 1));
                    PurchLine.Insert(true);
                end;
            until TempTimeSeriesForecast.Next() = 0;
            Page.Run(Page::"Purchase Order", PurchHeader);
        end;
    end;
}
```

Solution: Page 50111 Prediction Overview.al after modification

```
page 50111 "Prediction Overview"
{
    PageType = Worksheet;
    Caption = 'Prediction Overview';
    SourceTable = "Time Series Forecast";
    UsageCategory = Tasks;
    SourceTableTemporary = true;
    InsertAllowed = false;
    ModifyAllowed = false;
    DeleteAllowed = false;
    ApplicationArea = All;

    layout
    {
        area(Content)
        {
            group(Prediction)
            {
                Caption = 'Prediction';

                field(ItemNoFilter; ItemNoFilter)
                {
                    Caption = 'Item No. Filter';
                    ApplicationArea = All;
                }
                field(PeriodType; PeriodType)
                {
                    Caption = 'Period Type';
                    ApplicationArea = All;
                }
                field(NoForecastPeriods; NumberOfForecastPeriods)
                {
                    Caption = 'No. of Forecast Periods';
                    ApplicationArea = All;
                }
                field(NoPastPeriods; NumberOfPastPeriods)
                {
                    Caption = 'No. of Past Periods';
                    ApplicationArea = All;
                }
                field(ConfidenceLevel; ConfidenceLevel)
                {
                    Caption = 'Confidence Level';
                    ApplicationArea = All;
                }
                field(Algorithm; ForecastAlgorithm)
                {
                    Caption = 'Forecast Algorithm';
                    ApplicationArea = All;
                }
            }

            group(PurchaseOrder)
            {
                Caption = 'Purchase Order';

                field(VendorNo; VendorNo)
                {
                    Caption = 'Vendor No.';
                    TableRelation = Vendor;
                    ApplicationArea = All;
                }

                field(QualityBar; QualityBar)
                {
                    Caption = 'Quality Bar';
                    ApplicationArea = All;
                }
            }

            repeater(Lines)
            {
                Editable = false;
                field("Group ID"; "Group ID")
                {
                    ApplicationArea = All;
                }
                field("Period No."; "Period No.")
                {
                    ApplicationArea = All;
                }
                field("Period Start Date"; "Period Start Date")
                {
                    ApplicationArea = All;
                }
                field(Value; Value)
                {
                    ApplicationArea = All;
                }
                field(Delta; Delta)
                {
                    ApplicationArea = All;
                }
                field("Delta %"; "Delta %")
                {
                    ApplicationArea = All;
                }
            }
        }
    }

    actions
    {
        area(Processing)
        {
            action(CalculateForecastBulk)
            {
                Caption = 'Calculate Forecast';
                Image = CalculatePlan;
                Promoted = true;
                PromotedCategory = Process;
                ApplicationArea = All;

                trigger OnAction();
                var
                    MLForecast: Codeunit "Machine Learning Forecast";
                begin
                    MLForecast.CalculateForecastBulk(
                        ItemNoFilter,
                        PeriodType,
                        NumberOfForecastPeriods,
                        NumberOfPastPeriods,
                        ConfidenceLevel,
                        ForecastAlgorithm,
                        Rec);
                    if Rec.FindFirst() then;
                end;
            }

            action(CreatePO)
            {
                Caption = 'Create Purchase Order';
                Image = NewPurchaseInvoice;
                Promoted = true;
                PromotedCategory = Process;
                ApplicationArea = All;
                trigger OnAction();
                var
                    MLForecast: Codeunit "Machine Learning Forecast";
                begin
                    MLForecast.CreatePurchaseOrder(Rec, QualityBar, VendorNo);
                end;
            }
        }
    }

    var
        ItemNoFilter: Text;
        PeriodType: Option Date,Week,Month,Quarter,Year;
        NumberOfForecastPeriods: Integer;
        NumberOfPastPeriods: Integer;
        ConfidenceLevel: Integer;
        ForecastAlgorithm: Option ARIMA,ETS,STL,"ETS+ARIMA","ETS+STL",ALL;
        VendorNo: Code[20];
        QualityBar: Decimal;

    trigger OnInit();
    begin
        PeriodType := PeriodType::Month;
        NumberOfForecastPeriods := 1;
        NumberOfPastPeriods := 12;
        ConfidenceLevel := 80;
    end;
}
```