Machine Learning Prediction: Classification

In this hands-on lab, you explore the Azure Machine Learning prediction functionality by using the Prediction Experiment for Dynamics 365 Business Central model and algorithm. This algorithm allows you to predict the classification or regression, based on a model trained on your sample data.

Contents

Create a new workspace

Before you start, create and configure a new workspace (if you are not sure how, check the “Instructions for the beginners” document).

Use the name “Classification Demo” for the new workspace. Use range 50130 to 50135 for this extension.

Download demo data

This hands-on lab requires you to have access to demo data that was prepared in advance. You can download the demo data from here:

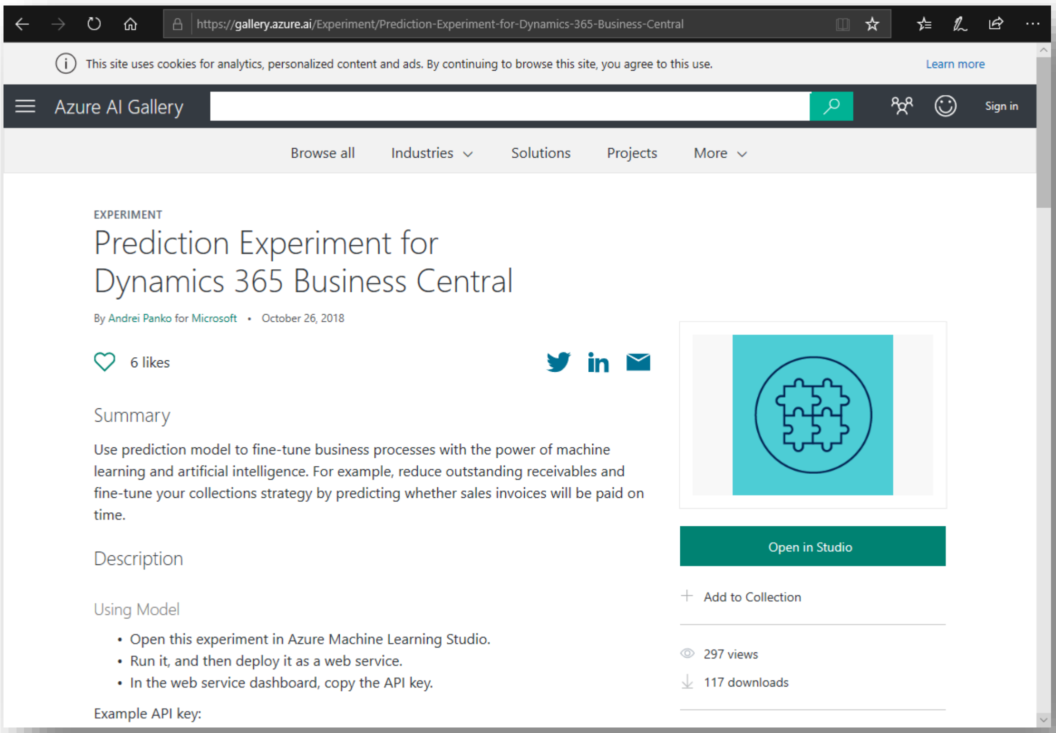
http://bit.ly/RedCarpetClassificationData

There is one Excel file, and two CSV files. You will need these files to complete the exercises in this lab. The exercises will instruct you when and how to use these files.

Obtain a machine learning endpoint and API key

In this exercise, you publish the prediction model as your own web service, and obtain an endpoint and API key to communicate with it form Business Central.

1. Open your browser and navigate to https://gallery.azure.ai/Experiment/Prediction-Experiment-for-Dynamics-365-Business-Central



https://gallery.azure.ai/Experiment/Prediction-Experiment-for-Dynamics-365-Business-Central

This site uses cookies for analytics, personalized content and ads. By continuing to browse this site, you agree to this use. Learn more

EXPERIMENT

Prediction Experiment for Dynamics 365 Business Central

By Andrei Panko for Microsoft • October 26, 2018

6 likes

Summary

Use prediction model to fine-tune business processes with the power of machine learning and artificial intelligence. For example, reduce outstanding receivables and fine-tune your collections strategy by predicting whether sales invoices will be paid on time.

Description

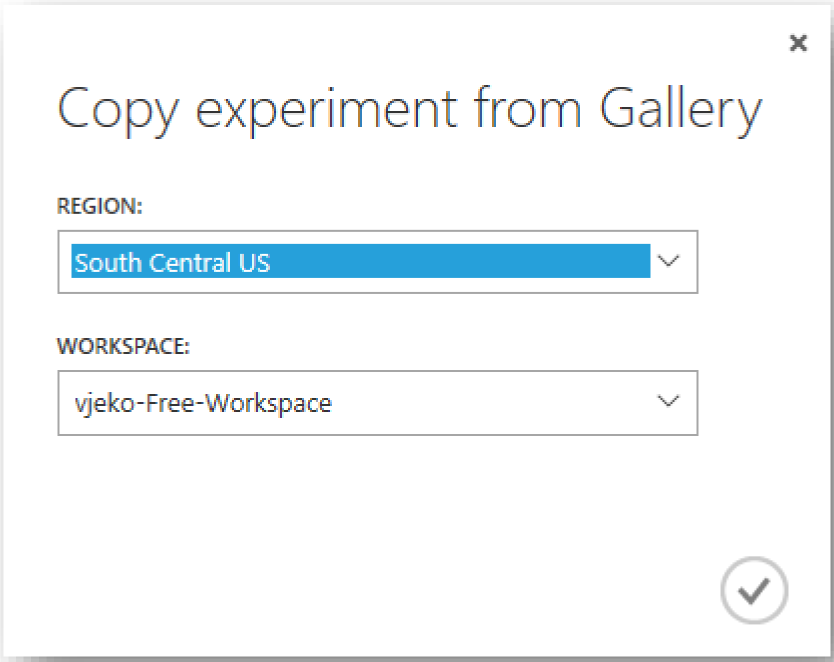
Using Model

- Open this experiment in Azure Machine Learning Studio.
- Run it, and then deploy it as a web service.
- In the web service dashboard, copy the API key.

Example API key:

2. Click **Open in Studio**.

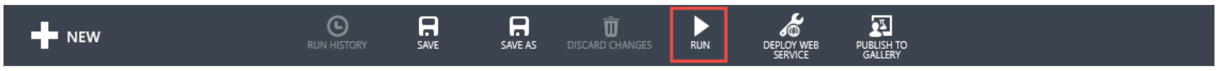
3. If asked for your account, sign in with your Microsoft or work account that has access to an Azure tenant. Otherwise, complete the sign up process, signing up for a free 7-day access.
4. When you sign in to Microsoft Azure Machine Learning Studio, you will be asked to copy the experiment from Gallery:


Copy experiment from Gallery

REGION: South Central US

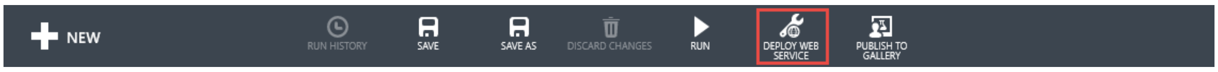
WORKSPACE: vjeko-Free-Workspace

5. Accept the defaults by clicking the Ok button (with the checkmark).
6. After the Prediction Experiment for Dynamics 365 Business Central opens in the Machine Learning Studio, in the bottom bar click **Run**.

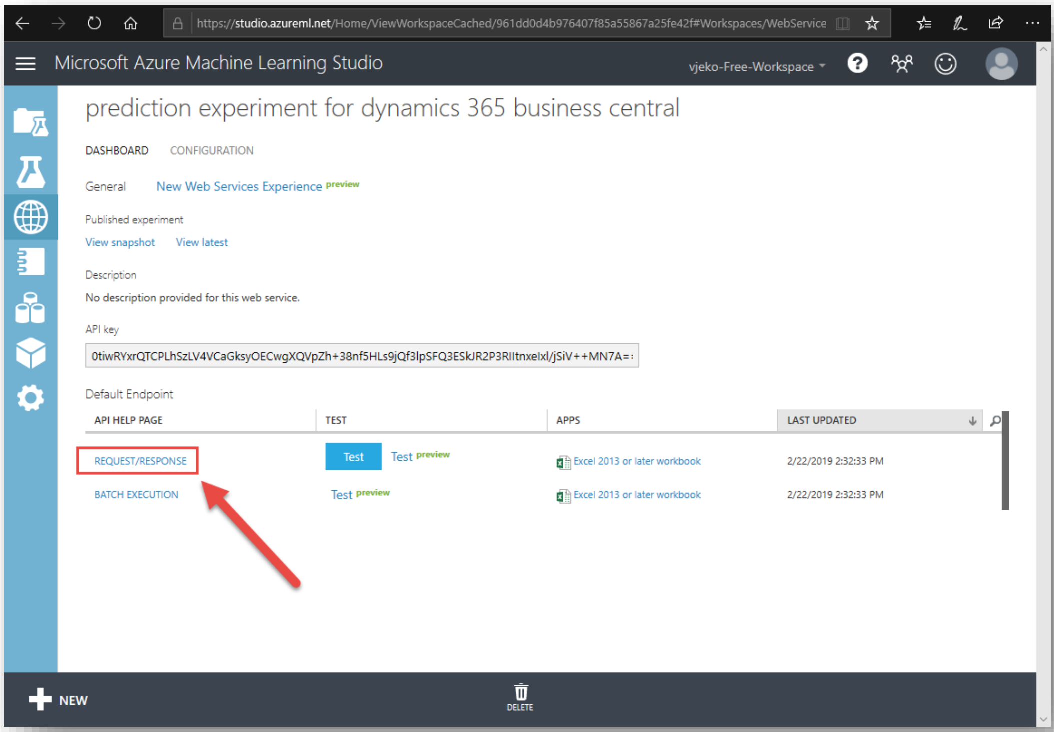


The experiment will start running and building its prediction model.

7. After the model finishes running, click **Deploy web service**.



8. Your prediction experiment is now deployed as web service, and you are presented with this page. Click Request/Response:


https://studio.azureml.net/Home/ViewWorkspaceCached/961dd0d4b976407f85a55867a25fe42f#Workspaces/WebService

Microsoft Azure Machine Learning Studio

vjeko-Free-Workspace

prediction experiment for dynamics 365 business central

DASHBOARD CONFIGURATION

General New Web Services Experience preview

Published experiment

View snapshot View latest

Description

No description provided for this web service.

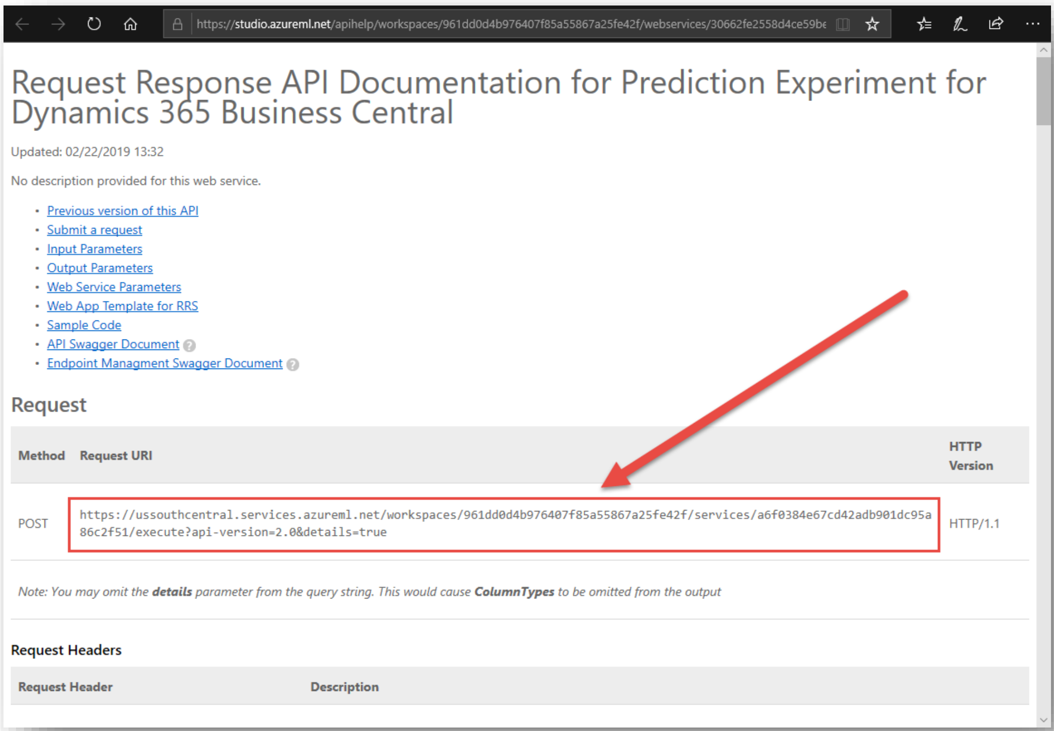
API key

0tiwRYxrQTCPLhSzLV4VCaGksyOECwgXQVpZh+38nf5HLs9jQf3lpSFQ3ESkJR2P3RIItnxelxl/jSiV++MN7A==

Default Endpoint

API HELP PAGE	TEST	APPS	LAST UPDATED
REQUEST/RESPONSE	Test Test preview	Excel 2013 or later workbook	2/22/2019 2:32:33 PM
BATCH EXECUTION	Test preview	Excel 2013 or later workbook	2/22/2019 2:32:33 PM

9. A new tab opens with the Request Response API Documentation. Select and copy the **Request URI**:


https://studio.azureml.net/apihelp/workspaces/961dd0d4b976407f85a55867a25fe42f/webservices/30662fe2558d4ce59be

Request Response API Documentation for Prediction Experiment for Dynamics 365 Business Central

Updated: 02/22/2019 13:32

No description provided for this web service.

- Previous version of this API
- Submit a request
- Input Parameters
- Output Parameters
- Web Service Parameters
- Web App Template for RRS
- Sample Code
- API Swagger Document
- Endpoint Managment Swagger Document

Request

Method	Request URI	HTTP Version
POST	https://ussouthcentral.services.azureml.net/workspaces/961dd0d4b976407f85a55867a25fe42f/services/a6f0384e67cd42adb901dc95a86c2f51/execute?api-version=2.0&details=true	HTTP/1.1

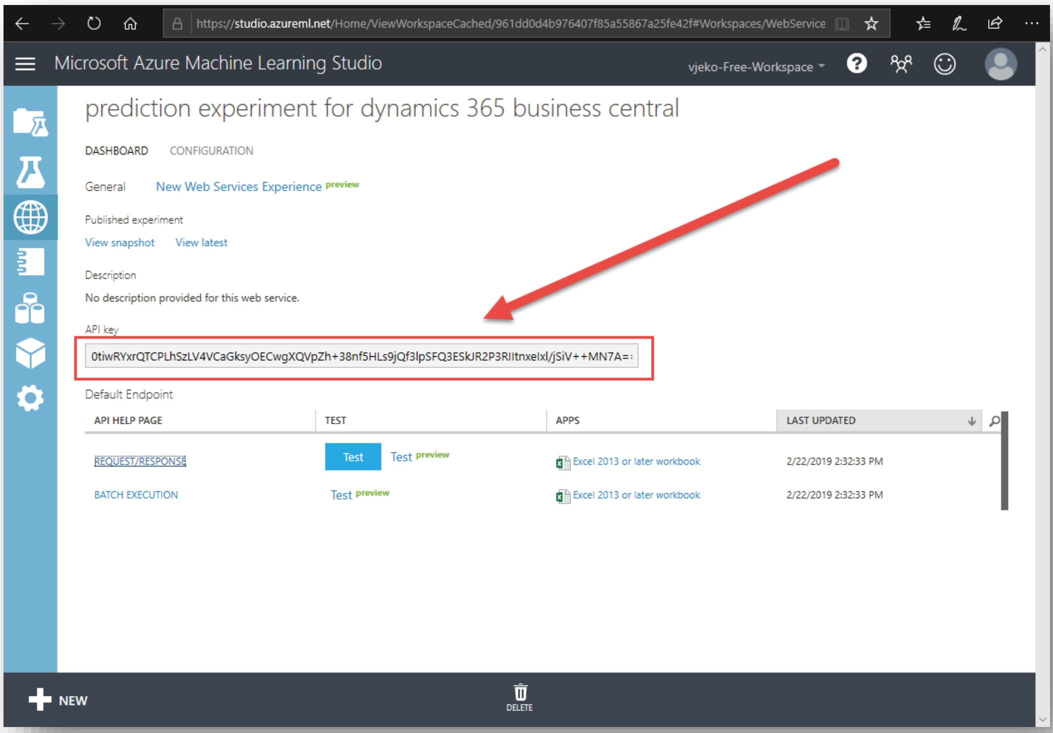
Note: You may omit the ***details*** *parameter from the query string. This would cause* ***ColumnTypes*** *to be omitted from the output*

Request Headers

Request Header	Description

Store this URL in a safe place. This is your URL specific to this web service.

10. Go back to the **Web services** tab (in the browser, that is the tab where you clicked **Request/Response**), and select and copy the **API key**:


https://studio.azureml.net/Home/ViewWorkspaceCached/961dd0d4b976407f85a55867a25fe42f#Workspaces/WebService

Microsoft Azure Machine Learning Studio

vjeko-Free-Workspace

prediction experiment for dynamics 365 business central

DASHBOARD CONFIGURATION

General New Web Services Experience preview

Published experiment

View snapshot View latest

Description

No description provided for this web service.

API key

0tiwRYxrQTCPLhSzLV4VCaGksyOECwgXQVpZh+38nf5HLs9jQf3lpSFQ3ESkJR2P3RIItnxelxl/jSiV++MN7A=:

Default Endpoint

API HELP PAGE	TEST	APPS	LAST UPDATED
REQUEST/RESPONSE	Test Test preview	Excel 2013 or later workbook	2/22/2019 2:32:33 PM
BATCH EXECUTION	Test preview	Excel 2013 or later workbook	2/22/2019 2:32:33 PM

Store this key in a safe place. You will need this key whenever invoking your web service.

11. Close the browser.

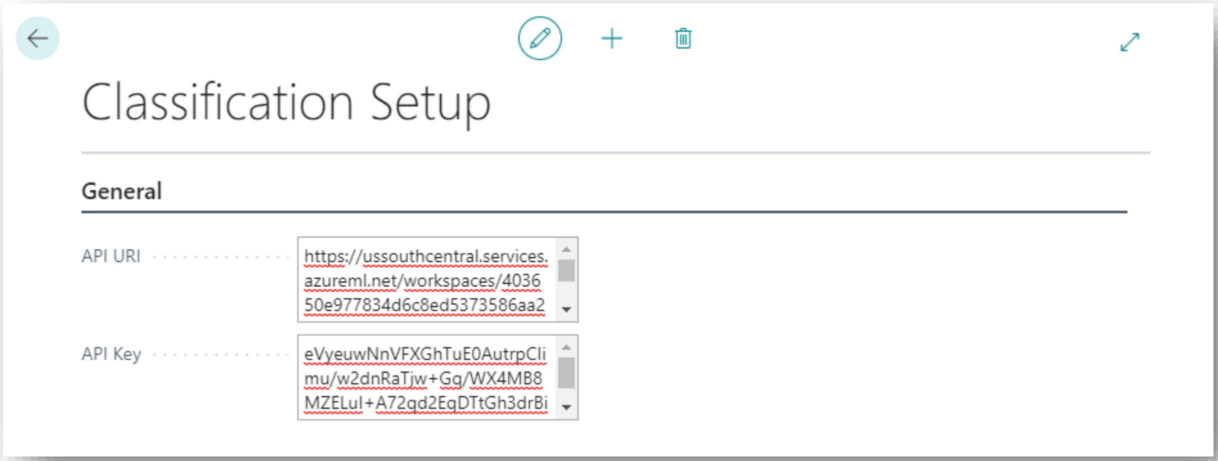
Create table and page for setup

In this exercise, you create a table and a page to hold the necessary extension configuration. Also, you create a page to allow users to modify this configuration.

1. Create a new table 50130 Classification Setup, with the following fields:

ID	Field	Type
1	Primary Key	Code[10]
2	API URI	Text[250]
3	API Key	Text[250]

2. Make the **Primary Key** field non-editable.
3. Make the **Primary Key** field the primary key of the table.
4. Create a new page 50130 Classification Setup, with the following specification:
    a. It's of card type.
    b. It uses table **Classification Setup** as its source.
    c. It does not allow deleting.
    d. It contains one FastTab named **General**, and in there it shows the **API URI** and **API Key** fields from the source table. Make these fields multi-line.
    e. Make the page available in the **Tell me** feature in the **Administration** usage category.
    f. When page is opened, if no record is found, create one.
5. Build, deploy, and run your extension.
6. In the **Tell me** box, search for and then run **Classification Setup**.
7. In the Classification Setup page, put your API URI and API Key that you obtained in the previous exercise, into the corresponding text boxes.



API URI: https://ussouthcentral.services.azureml.net/workspaces/403650e977834d6c8ed5373586aa2

API Key: eVyeuwNnVFXGhTuE0AutrpClimu/w2dnRaTjw+Gq/WX4MB8MZELul+A72qd2EqDTtGh3drBi

Solution: Table 50130 Classification Setup.al

```
table 50130 "Classification Setup"
{
    Caption = 'Classification Setup';

    fields
    {
        field(1; "Primary Key"; Code[10])
        {
            Caption = 'Primary Key';
            Editable = false;
        }

        field(2; "API URI"; Text[250])
        {
            Caption = 'API URI';
        }

        field(3; "API Key"; Text[250])
        {
            Caption = 'API Key';
        }
    }
}
```

Solution: Page 50130 Classification Setup.al

```
page 50130 "Classification Setup"
{
    Caption = 'Classification Setup';
    PageType = Card;
    SourceTable = "Classification Setup";
    DeleteAllowed = false;
    UsageCategory = Administration;
    ApplicationArea = All;

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

                field("API URI"; "API URI")
                {
                    MultiLine = true;
                    ApplicationArea = All;
                }
                field("API Key"; "API Key")
                {
                    MultiLine = true;
                    ApplicationArea = All;
                }
            }
        }
    }

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

Classification prediction: Sales volume

In this exercise, you'll create a table to contain your sales volume data. You'll build a user interface that will allow you to view and modify the data, as well as an XMLport object to import the demo data into the table. Also, you'll extend the setup table you created in the previous exercise with an additional blob field to store the sales volume prediction model.

Then, you'll build a codeunit that will prepare a model based on the sales volume data, store that model in the setup table, and then use that model to invoke the prediction web service. To help you with this, you'll use the standard codeunit 2003 ML Prediction Management.

Finally, you'll create a page that will allow you to enter all the data except the Volume field. From this page you'll allow invoking the prediction model that will predict **Volume** for all the records in the page.

Sales Volume table

1. Create a new table 50131 Sales Volume with the following fields:

ID	Field	Type
1	Entry No.	Integer Set the AutoIncrement property.
2	Item No.	Code[20]
3	Posting Date	Date
4	Volume	Option: None, VeryLow, Low, Medium, High, VeryHigh, Extraordinary
5	Color	Option: Red, Green, Blue
6	Size	Option: S, M, L, XL
7	Price	Decimal
8	Confidence	Decimal

2. Make the **Entry No.** field the primary key of the table.

Solution: Table 50131 Sales Volume.al

```
table 50131 "Sales Volume"
{
    Caption = 'Sales Volume';

    fields
    {
        field(1; "Entry No."; Integer)
        {
            Caption = 'Entry No.';
            AutoIncrement = true;
        }
        field(2; "Item No."; Code[20])
        {
            Caption = 'Item No.';
            TableRelation = Item;
        }

        field(3; "Posting Date"; Date)
        {
            Caption = 'Posting Date';
        }

        field(4; Volume; Option)
        {
            Caption = 'Volume';
            OptionMembers =
None,VeryLow,Low,Medium,High,VeryHigh,Extraordinary;
            OptionCaption =
'None,VeryLow,Low,Medium,High,VeryHigh,Extraordinary';
        }

        field(5; Color; Option)
        {
            Caption = 'Color';
            OptionMembers = Red,Green,Blue;
            OptionCaption = 'Red,Green,Blue';
        }

        field(6; Size; Option)
        {
            Caption = 'Size';
            OptionMembers = S,M,L,XL;
            OptionCaption = 'S,M,L,XL';
        }

        field(7; Price; Decimal)
        {
            Caption = 'Price';
        }

        field(8; Confidence; Decimal)
        {
            Caption = 'Confidence';
        }
    }

    keys
    {
        key(Primary; "Entry No.") { }
    }
```

```
}
```

Import Sales Volume XMLport

1. Create a new XMLport 50131 Import Sales Volume with the following specification:
    a. It uses VariableText format.
    b. It uses “;” as its field separator.
    c. It supports only import direction.
    d. It contains schema with one root text element, that contains one table element that will contain fields from the Sales Volume table: Item No., Posting Date, Volume, Color, Size, and Price.

Solution: XmlPort 50131 Import Sales Volume.al

```
xmlport 50131 "Import Sales Volume"
{
    Format = VariableText;
    FieldSeparator = ';';
    Direction = Import;

    schema
    {
        textelement(root)
        {
            tableelement(Volume; "Sales Volume")
            {
                fieldelement(Price; Volume."Item No.") { }
                fieldelement(Gender; Volume."Posting Date") { }
                fieldelement(Material; Volume.Volume) { }
                fieldelement(SleeveLength; Volume.Color) { }
                fieldelement(IsChristmas; Volume.Size) { }
                fieldelement(DecemberSales; Volume.Price) { }
            }
        }
    }
}
```

Classification Setup table and page modifications

1. Open Table 50130 Classification Setup.al
2. Define the following new field:

ID	Field	Type
4	Sales Volume Model	Blob
5	Sales Volume Model Quality	Decimal The field must not be editable

Hint: it’s the following new code:

```
        field(4; "Sales Volume Model"; Blob)
        {
            Caption = 'Sales Volume Model';
        }

        field(5; "Sales Volume Model Quality"; Decimal)
        {
            Caption = 'Sales Volume Model Quality';
            Editable = false;
        }
```

3. Define a global function **GetModel** that reads the model stored in the Sales Volume Model blob field and returns it as text as function’s return value. The function uses the standard codeunit **Type Helper** to convert from blob to text.

   Hint: it’s the following new code:

```
    procedure GetModel(): Text
    var
        TypeHelper: Codeunit "Type Helper";
        BlobRef: FieldRef;
        RecRef: RecordRef;
    begin
        RecRef.GetTable(Rec);
        BlobRef := RecRef.Field(FieldNo("Sales Volume Model"));
        exit(TypeHelper.ReadBlob(BlobRef));
    end;
```

4. Define a global function **SetModel** that receives one text parameter that represents the model to be stored in the **Sales Volume Model** field. Inside this function write code that makes sure that there is always a record present in the table, and then stores the text into the blob field, by using the standard codeunit **Type Helper**.

   Hint: it’s the following new code:

```
    procedure SetModel(Model: Text)
    var
        TypeHelper: Codeunit "Type Helper";
        RecRef: RecordRef;
    begin
        if not Rec.Get() then
            Rec.Insert();

        RecRef.GetTable(Rec);
        TypeHelper.SetBlobString(
            RecRef,
            FieldNo("Sales Volume Model"),
            Model);
        RecRef.SetTable(Rec);
    end;
```

5. Open Page 50130 Classification Setup.al
6. After the **General** group, define a new group and name it **Models**, and inside it a subgroup called **Sales Volume Model**.
7. In the **Sales Volume Model** group, define a field that shows the following source expression: `"Sales Volume Model".HasValue()`, and a field that shows the **Sales Volume Model Quality** field from the table.

   Hint: it's the following new code:

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

    group(SalesVolume)
    {
        Caption = 'Sales Volume Model';

        field("Sales Volume Model"; "Sales Volume Model".HasValue())
        {
            Caption = 'Sales Volume Model Exists';
            ApplicationArea = All;
        }

        field("Sales Volume Model Quality"; "Sales Volume Model
Quality")
        {
            ApplicationArea = All;
        }
    }
}
```

Sales Volume Classif. Mgt. Codeunit

1. Create a new codeunit 50131 "Sales Volume Classif. Mgt.".
2. In the codeunit, declare:
   a. A local function called **GetSetup** that receives a **Classification Setup** record variable by reference. This function retrieves the setup record and makes sure its **API URI** and **API Key** fields are defined.
   b. A global function **Train**. This function receives no parameters and returns no value.
   c. A global function **Predict**. This function receives one parameter of type temporary record **Sales Volume** and returns no value.
3. In the **Train** function declare the following local variables:

Variable	Type
Setup	Record “Classification Setup”

SalesVolume	Record “Sales Volume”
Prediction	Codeunit “ML Prediction Management”
Model	Text
ModelQuality	Decimal

4. In the Train function, write the following logic:
    a. Invoke the **GetSetup** local function.
    b. Invoke the Prediction.Initialize function, and pass the configured URL and API key, and zero as timeout.
    Hint: if this sounds too complicated, simply investigate the arguments of the Initialize function to see which values you need to pass into it. Do this for all other functions you’ll invoke.
    c. Invoke the Prediction.SetRecord function and pass the SalesVolume record variable into it.
    d. Invoke the Prediction.AddFeature method for each of the **Color**, **Size**, and **Price** fields and pass their respective field numbers.
    e. Invoke the Prediction.SetLabel method for the Volume field and pass its field number.
    f. Invoke the Prediction.Train method, and pass the Model and ModelQuality variable references.
    g. Store the Model and ModelQuality values in the **Classification Setup** table.
    Hint: Use the SetModel function.
    h. Show message that informs the user of the model quality.
    Hint: model quality is measured in percentages, and is calculated as a value between 0 and 1.
5. In the **Predict** function declare the following local variables:

Variable	Type
Setup	Record “Classification Setup”
Prediction	Codeunit “ML Prediction Management”

6. In the **Predict** function, write the following logic:
    a. Invoke the **GetSetup** local function.
    b. Makes sure that the Sales Volume Model is defined in the setup table.
    c. Invoke the Prediction.Initialize function, and pass the configured URL and API key, and zero as timeout.
    d. Invoke the Prediction.SetRecord function and pass the SalesVolumeTemp record parameter into it.
    e. Invoke the Prediction.AddFeature function for each of the **Color**, **Size**, and **Price** fields and pass their respective field numbers.
    f. Invoke the Prediction.SetConfidence function for the **Confidence** field and pass its field number.
    g. Invoke the Prediction.SetLabel function for the **Volume** field and pass its field number.
    h. Invoke the Prediction.Predict function and pass the model into it.
    Hint: Read the model from the setup table using the GetModel function you created earlier.

Solution: Sales Volume Classif. Mgt.

```
codeunit 50131 "Sales Volume Classif. Mgt."
{
    local procedure GetSetup(var Setup: Record "Classification Setup");
    begin
        Setup.Get();
        Setup.TestField("API URI");
        Setup.TestField("API Key");
    end;

    procedure Train();
    var
        Setup: Record "Classification Setup";
        SalesVolume: Record "Sales Volume";
        Prediction: Codeunit "ML Prediction Management";
        Model: Text;
        ModelQuality: Decimal;
    begin
        GetSetup(Setup);
        Prediction.Initialize(Setup."API URI", Setup."API Key", 0);

        Prediction.SetRecord(SalesVolume);
        Prediction.AddFeature(SalesVolume.FieldNo(Color));
        Prediction.AddFeature(SalesVolume.FieldNo(Size));
        Prediction.AddFeature(SalesVolume.FieldNo(Price));
        Prediction.SetLabel(SalesVolume.FieldNo(Volume));

        Prediction.Train(Model, ModelQuality);

        Setup.SetModel(Model);
        Setup."Sales Volume Model Quality" := ModelQuality;
        Setup.Modify(true);

        Message('Model is trained. Quality is %1%',
            Round(ModelQuality * 100, 1));
    end;

    procedure Predict(var SalesVolumeTemp: Record "Sales Volume" temporary);
    var
        Setup: Record "Classification Setup";
        Prediction: Codeunit "ML Prediction Management";
    begin
        GetSetup(Setup);
        Setup.TestField("Sales Volume Model");
        Prediction.Initialize(Setup."API URI", Setup."API Key", 0);

        Prediction.SetRecord(SalesVolumeTemp);
        Prediction.AddFeature(SalesVolumeTemp.FieldNo(Color));
        Prediction.AddFeature(SalesVolumeTemp.FieldNo(Size));
        Prediction.AddFeature(SalesVolumeTemp.FieldNo(Price));
        Prediction.SetConfidence(SalesVolumeTemp.FieldNo(Confidence));
        Prediction.SetLabel(SalesVolumeTemp.FieldNo(Volume));

        Prediction.Predict(Setup.GetModel());
    end;
}
```

Sales Volume page

1. Create a new page 50131 Sales Volume with the following specification:
    a. It is a list type page.
    b. It uses table **Sales Volume** as its source.
    c. Make the page available in the **Tell me** feature in the **Administration** usage category.
    d. Define the layout section with content area and one repeater.
    e. In the repeater, show fields: **Entry No.**, **Item No.**, **Posting Date**, **Volume**, **Color**, **Size**, and **Price**.
    f. Define the actions section with Processing area.
    g. Define the **Import** action with the following logic:
        i. If there are any records in the Sales Volume table, ask the user if the user wants to empty the **Sales Volume** table first.
        ii. If the user confirms, delete all records from the **Sales Volume** table.
        iii. Run the **Import Sales Volume** XMLport.
    h. Define the **Train** action that invokes the **Train** method of the **Sales Volume Classif. Mgt.** codeunit.

Solution: Page 50131 Sales Volume.al

```
page 50131 "Sales Volume"
{
    Caption = 'Sales Volume';
    PageType = List;
    SourceTable = "Sales Volume";
    ApplicationArea = All;
    UsageCategory = Lists;

    layout
    {
        area(Content)
        {
            repeater(Data)
            {
                field("Entry No."; "Entry No.")
                {
                    ApplicationArea = All;
                }

                field("Item No."; "Item No.")
                {
                    ApplicationArea = All;
                }

                field("Posting Date"; "Posting Date")
                {
                    ApplicationArea = All;
                }

                field(Volume; Volume)
                {
                    ApplicationArea = All;
                }

                field(Color; Color)
                {
                    ApplicationArea = All;
                }

                field(Size; Size)
                {
                    ApplicationArea = All;
                }

                field(Price; Price)
                {
                    ApplicationArea = All;
                }
            }
        }
    }

    actions
    {
        area(Processing)
        {
            action(Import)
            {
                Caption = 'Import data';
                Ellipsis = true;
                Image = ImportExcel;
                ApplicationArea = All;

                trigger OnAction();
                var
                    SalesVolume: Record "Sales Volume";
                    Confirmation: Label 'Do you want to empty the %1 table
first?';
                begin
```

```
                    if not SalesVolume.IsEmpty() then
                        if (Confirm(Confirmation, false,
SalesVolume.TableCaption())) then
                            SalesVolume.DeleteAll();

                    Xmlport.Run(Xmlport::"Import Sales Volume", false, true);
                end;
            }

            action(Train)
            {
                Caption = 'Train Model';
                Image = CalculatePlan;
                ApplicationArea = All;

                trigger OnAction();
                var
                    Classification: Codeunit "Sales Volume Classif. Mgt.";
                begin
                    Classification.Train();
                end;
            }
        }
    }
}
```

Sales Volume Prediction Page

1. Create a new page 50132 Sales Volume Prediction by copying page 50131.
2. Declare the following global variables:

Variable	Type
MinimumConfidence	Decimal
EntryNo	Integer

3. Do the following layout changes:
    a. Change name to **Sales Volume Prediction**.
    b. Set the **SourceTableTemporary** property.
    c. Change the **Volume** field by making it non-editable, and applying the `StrongAccent` style.
    d. Add the **Confidence** field as the last field in the repeater. Make this field non-editable. Set the style to `Unfavorable`, but only Confidence is higher than 0 and lower than the value of MinimumConfidence variable.
4. Do the following action changes:
    a. Remove the **Import** action.
    b. Remove the **Train** action.
    c. Add action **Predict** that invokes the **Predict** function of the **Sales Volume Classif. Mgt.** codeunit by passing current record as its by-reference parameter.

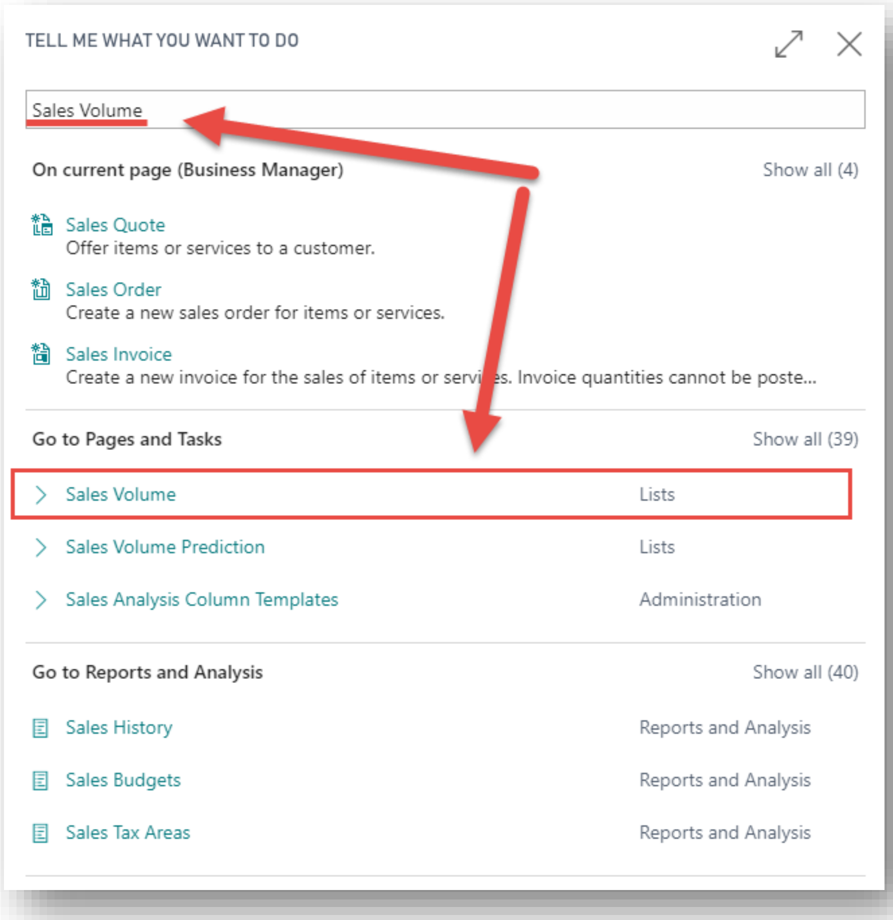
5. Define the **OnOpenPage** trigger. It must retrieve the **Classification Setup** record and then make sure the **Sales Volume Model** is defined. Then, it sets the **MinimumConfidence** variable to the value of the **Sales Volume Model Quality** field from the **Classification Setup** table.

Solution: Page 50132 Sales Volume Prediction.al

```
page 50132 "Sales Volume Prediction"
{
    Caption = 'Sales Volume Prediction';
    PageType = List;
    SourceTable = "Sales Volume";
    SourceTableTemporary = true;
    ApplicationArea = All;
    UsageCategory = Lists;

    layout
    {
        area(Content)
        {
            repeater(Data)
            {
                field("Item No."; "Item No.")
                {
                    ApplicationArea = All;
                }

                field("Posting Date"; "Posting Date")
                {
                    ApplicationArea = All;
                }

                field(Volume; Volume)
                {
                    Editable = false;
                    Style = StrongAccent;
                    ApplicationArea = All;
                }

                field(Color; Color)
                {
                    ApplicationArea = All;
                }

                field(Size; Size)
                {
                    ApplicationArea = All;
                }
                field(Price; Price)
                {
                    ApplicationArea = All;
                }

                field(Confidence; Confidence)
                {
                    Editable = false;
                    Style = Unfavorable;
                    StyleExpr = (Confidence > 0) and (Confidence <=
MinimumConfidence);
                    ApplicationArea = All;
                }
            }
        }
    }

    actions
    {
        area(Processing)
        {
            action(Predict)
            {
                Caption = 'Predict';
                Image = CalculatePlan;
                ApplicationArea = All;

                trigger OnAction();
                var
                    Classification: Codeunit "Sales Volume Classif. Mgt.";
                begin
                    Classification.Predict(Rec);
                end;
            }
        }
    }

    var
        MinimumConfidence: Decimal;
        EntryNo: Integer;

    trigger OnNewRecord(BelowxRec: Boolean);
    begin
        EntryNo += 1;
        "Entry No." := EntryNo;
    end;

    trigger OnOpenPage();
    var
        Setup: Record "Classification Setup";
    begin
        Setup.Get();
        Setup.TestField("Sales Volume Model");
        MinimumConfidence := Setup."Sales Volume Model Quality";
    end;
}
```

Test Sales Volume prediction functionality

1. Build, deploy, and run the extension.
2. In the **Tell me** box, search for and run the **Sales Volume** page.



TELL ME WHAT YOU WANT TO DO

Sales Volume

On current page (Business Manager) — Show all (4)

Sales Quote
Offer items or services to a customer.

Sales Order
Create a new sales order for items or services.

Sales Invoice
Create a new invoice for the sales of items or services. Invoice quantities cannot be poste...

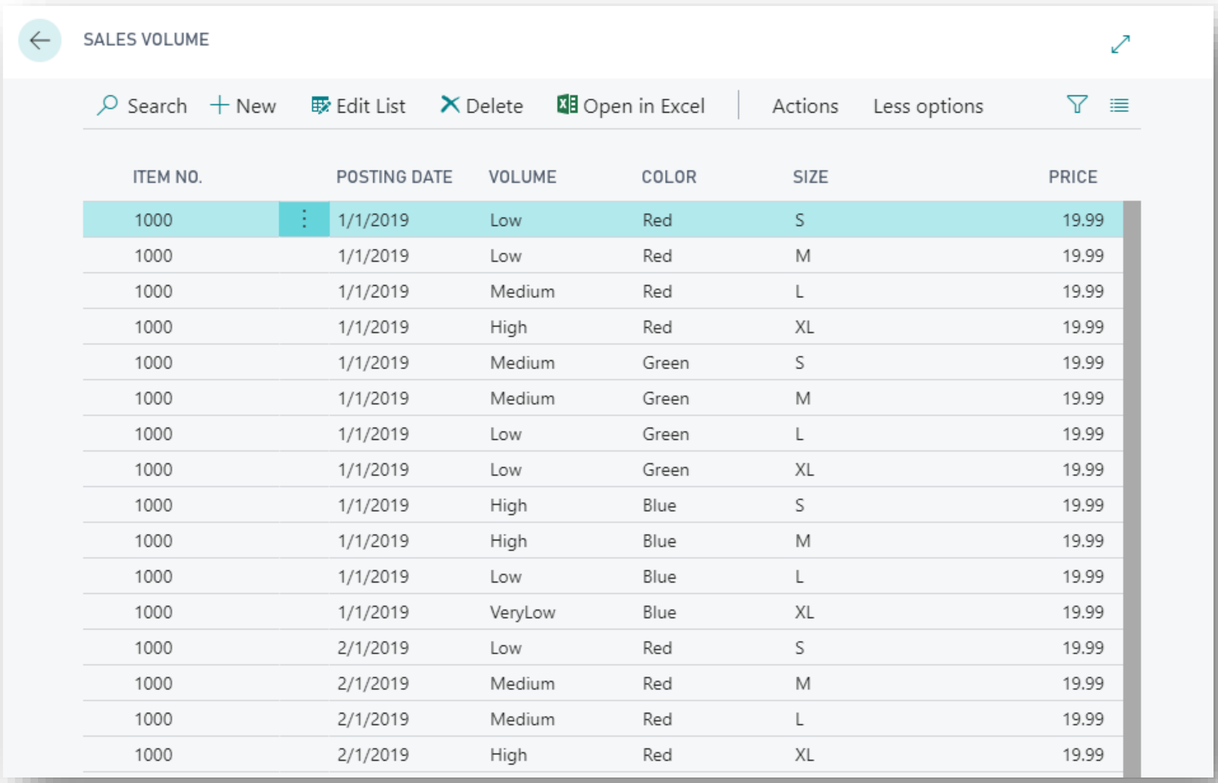
Go to Pages and Tasks — Show all (39)

Sales Volume	Lists
Sales Volume Prediction	Lists
Sales Analysis Column Templates	Administration

Go to Reports and Analysis — Show all (40)

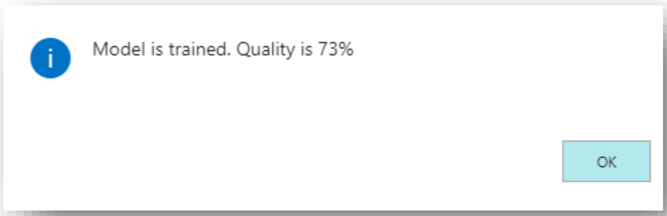
Sales History	Reports and Analysis
Sales Budgets	Reports and Analysis
Sales Tax Areas	Reports and Analysis

3. In the Sales Volume page, click **Actions > Import data…**
4. In the **Import** dialog, click **Choose…**
5. Locate and select the **Sales Volume.csv** file, and then click **Open**. The Sales Volume page populates with data:

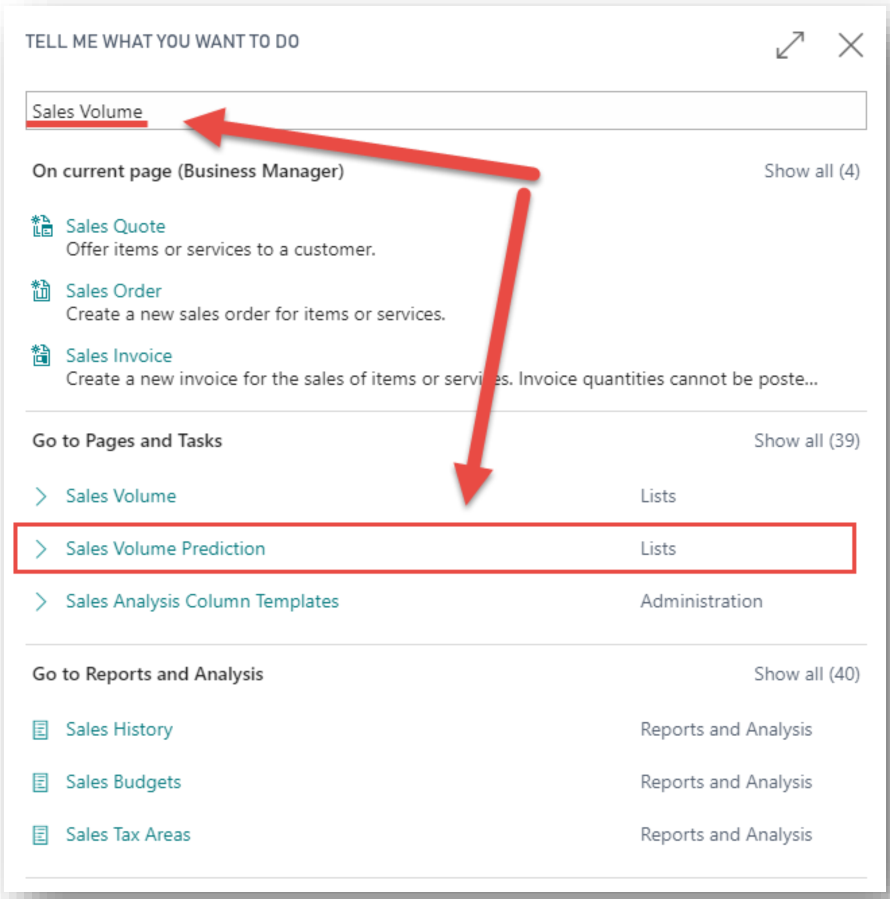


ITEM NO.	POSTING DATE	VOLUME	COLOR	SIZE	PRICE
1000	1/1/2019	Low	Red	S	19.99
1000	1/1/2019	Low	Red	M	19.99
1000	1/1/2019	Medium	Red	L	19.99
1000	1/1/2019	High	Red	XL	19.99
1000	1/1/2019	Medium	Green	S	19.99
1000	1/1/2019	Medium	Green	M	19.99
1000	1/1/2019	Low	Green	L	19.99
1000	1/1/2019	Low	Green	XL	19.99
1000	1/1/2019	High	Blue	S	19.99
1000	1/1/2019	High	Blue	M	19.99
1000	1/1/2019	Low	Blue	L	19.99
1000	1/1/2019	VeryLow	Blue	XL	19.99
1000	2/1/2019	Low	Red	S	19.99
1000	2/1/2019	Medium	Red	M	19.99
1000	2/1/2019	Medium	Red	L	19.99
1000	2/1/2019	High	Red	XL	19.99

6. Click **Actions > Train Model**.
7. If everything is correctly set up, after a few seconds a message will tell you that the model is trained and inform you about the model quality:


Model is trained. Quality is 73%

8. Click **OK**.
9. In the **Tell me** box, search for and run **Sales Volume Prediction**.



TELL ME WHAT YOU WANT TO DO

Sales Volume

On current page (Business Manager) Show all (4)

Sales Quote
Offer items or services to a customer.

Sales Order
Create a new sales order for items or services.

Sales Invoice
Create a new invoice for the sales of items or services. Invoice quantities cannot be poste...

Go to Pages and Tasks Show all (39)

Sales Volume	Lists
Sales Volume Prediction	Lists
Sales Analysis Column Templates	Administration

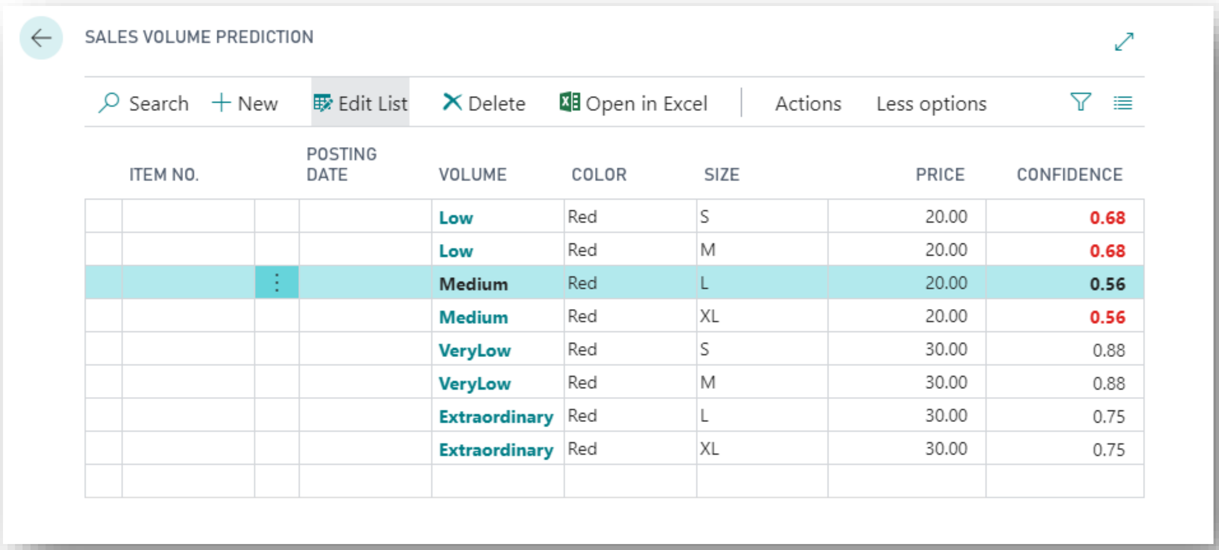
Go to Reports and Analysis Show all (40)

Sales History	Reports and Analysis
Sales Budgets	Reports and Analysis
Sales Tax Areas	Reports and Analysis

10. In the **Sales Volume Prediction** page, click **Edit List**.
11. Populate the table as follows:

Color	Size	Price
Red	S	20.00
Red	M	20.00
Red	L	20.00
Red	XL	20.00
Red	S	30.00
Red	M	30.00
Red	L	30.00
Red	XL	30.00

    You can leave other fields empty. Item No. and Posting Date play no role in prediction.
12. Click **Actions > Predict**.
13. After a few seconds, the codeunit you invoked will populate the predictions from the machine learning web service. The **Volume** column will indicate the expected sales volume, and the **Confidence** column will indicate how confident the model is in its prediction. Any values lower than the calculated model quality will be shown in red.


SALES VOLUME PREDICTION

ITEM NO.	POSTING DATE	VOLUME	COLOR	SIZE	PRICE	CONFIDENCE
		Low	Red	S	20.00	**0.68**
		Low	Red	M	20.00	**0.68**
		Medium	Red	L	20.00	**0.56**
		Medium	Red	XL	20.00	**0.56**
		VeryLow	Red	S	30.00	0.88
		VeryLow	Red	M	30.00	0.88
		Extraordinary	Red	L	30.00	0.75
		Extraordinary	Red	XL	30.00	0.75

Challenge Yourself: Titanic Survival Prediction

In this exercise you will train a classification prediction model with Titanic passenger data set. Titanic survival prediction is a “Hello, World!” of machine learning, and is often used to test prediction algorithms.

Your task is to create the necessary objects to import Titanic dataset, import the dataset, train the prediction model, and finally test the model by providing random passenger data, and verify whether those hypothetical passengers would survive or perish on Titanic.

Extend the setup

1. Modify table 50130 **Classification Setup**, and add the following new fields:

ID	Field	Type
6	Titanic Prediction Model	Blob
7	Titanic Predict. Model Quality	Decimal The field must not be editable.

2. Copy the GetModel and SetModel functions, and rename them to GetTitanicModel and SetTitanicModel, respectively.
3. Inside these two functions, modify field references to use **Titanic Prediction Model** instead of **Sales Volume Model**.
4. Modify page 50130 **Classification Setup**:
    a. Inside the **Models** group, and after the **Sales Volume Model** subgroup, create a new group named **Titanic Prediction Model**.
    b. In this group, add the two additional fields that you have added to the **Classification Setup** table in the previous step.

Create the data model

1. Create a new file and name it “Table 50133 Titanic Passenger.al”.
2. In the file, declare table 50133 "Titanic Passenger" with the following fields:

ID	Field	Type
1	Id	Integer AutoIncrement = true
2	Name	Text[250]
3	Class	Option OptionMembers: 1st, 2nd, 3rd
4	Age	Integer BlankZero = true
5	Age is Known	Boolean
6	Sex	Option OptionMembers: Male, Female
7	Survived	Boolean

8	Confidence	Decimal
		BlankZero = true
		Editable = false

3. Declare the keys section and in it declare the primary key field over the Id column. Make the primary key clustered.
4. For the **Age** field, define the OnValidate trigger. From the trigger, set the **Age is Known** field depending on the value entered in the **Age** field. Age is deemed unknown if it is 0.
5. For the **Age is Known** field, define the OnValidate trigger. From the trigger, if **Age** is higher than 0, and **Age is Known** is false, set **Age** to 0. Also, if Age is equal to 0, and **Age is Known** is true, set **Age is Known** to false.

Solution: Table 50133 Titanic Passenger.al

```
table 50133 "Titanic Passenger"
{
    Caption = 'Titanic Passenger';

    fields
    {
        field(1; Id; Integer)
        {
            Caption = 'Id';
            AutoIncrement = true;
        }

        field(2; Name; Text[250])
        {
            Caption = 'Name';
        }

        field(3; Class; Option)
        {
            Caption = 'Class';
            OptionMembers = "1st","2nd","3rd";
            OptionCaption = '1st,2nd,3rd';
        }

        field(4; Age; Integer)
        {
            Caption = 'Age';
            BlankZero = true;

            trigger OnValidate();
            begin
                "Age is Known" := Age > 0;
            end;
        }

        field(5; "Age is Known"; Boolean)
        {
            Caption = 'Age is Known';
            trigger OnValidate();
            begin
                if (Age > 0) and (not "Age is Known") then
                    Age := 0;
                if (Age = 0) and "Age is Known" then
                    "Age is Known" := false;
            end;
        }

        field(6; Sex; Option)
        {
            Caption = 'Sex';
            OptionMembers = Male,Female;
            OptionCaption = 'Male,Female';
        }

        field(7; Survived; Boolean)
        {
            Caption = 'Survived';
        }

        field(8; Confidence; Decimal)
        {
            Caption = 'Confidence';
            BlankZero = true;
            Editable = false;
        }
    }

    keys
    {
        key(Primary; Id) { Clustered = true; }
    }
}
```

Create the XmlPort

1. Create a new file and name it XmlPort 50133 Import Titanic Passengers.al.
2. Inside the file declare xmlport 50133 "Import Titanic Passengers" with the following specification:
    a. It uses VariableText format.
    b. It uses “;” as its field separator.
    c. It supports only import direction.
2. It contains schema with one root text element, that contains one table element that will contain fields from the **Titanic Passenger** table.
3. Inside the table element, declare the following fields
    a. Name, maps to “Titanic Passenger”.Name
    b. Class, maps to “Titanic Passenger”.Class

   c. Age, a text element (variable), does not map directly to a field
   d. Sex, maps to “Titanic Pasenger”.Sex
   e. Survived, maps to “Titanic Passenger”.Survived
4. For the **Age** text element, define the OnAfterAssignVariable trigger. In the trigger, set the **Age** field on the **Titanic Passenger** record to 0 and set the **Age is Known** field to false. Then, if the **Age** variable is not blank, evaluate its content into the **Age** field, and set the **Age is Known** field to true.

Solution: XmlPort 50133 Import Titanic Passengers.al

```
xmlport 50133 "Import Titanic Passengers"
{
    Format = VariableText;
    FieldSeparator = ';';
    Direction = Import;

    schema
    {
        textelement(root)
        {
            tableelement(TitanicPassenger; "Titanic Passenger")
            {
                fieldelement(Name; TitanicPassenger.Name) { }
                fieldelement(Class; TitanicPassenger.Class) { }
                textelement(Age)
                {
                    trigger OnAfterAssignVariable();
                    begin
                        TitanicPassenger.Age := 0;
                        TitanicPassenger."Age is Known" := false;
                        if Age <> '' then begin
                            Evaluate(TitanicPassenger.Age, Age);
                            TitanicPassenger."Age is Known" := true;
                        end;
                    end;
                }
                fieldelement(Sex; TitanicPassenger.Sex) { }
                fieldelement(Survived; TitanicPassenger.Survived) { }
            }
        }
    }
}
```

Create the codeunit stub

1. Create a new file and name it Codeunit 50133 Titanic Passenger Pred. Mgt.al.
2. In the file, declare codeunit 50133 "Titanic Passenger Pred. Mgt.".
3. Declare two public functions:

    a. Function **Train**, receives no parameters and returns no value.
    b. Function **Predict**, receives a by-reference temporary **Titanic Passenger** record as a parameter, and returns no value.
4. Do not put any code yet into these functions.

Create the list page

1. Create a new file and name it Page 50133 Titanic Passengers.al.
2. Inside the file, declare page 50133 "Titanic Passengers". It must be a page of type list over the **Titanic Passenger** source table. Define **UsageCategory** of **Lists** and set the **ApplicationArea** to **All**.
3. Declare the layout section.
4. Inside the layout section, declare the Content area.
5. Inside the Content area, declare a repeater, and name it **Passengers**.
6. Inside the **Passengers** repeater, add all the fields from the **Titanic Passenger** table, except for fields **Id** and **Confidence**.
7. Define the actions section with Processing area.
8. Define the **Import** action with the following logic:
    a. Empty the **Titanic Passenger** table.
    b. Run the **Import Titanic Passengers** XMLport.
9. Define the **Train** action that invokes the **Train** function from the **Titanic Passenger Pred. Mgt.** codeunit.

Solution: Page 50133 Titanic Passengers.al

```
page 50133 "Titanic Passengers"
{
    Caption = 'Titanic Passengers';
    PageType = List;
    SourceTable = "Titanic Passenger";
    UsageCategory = Lists;
    ApplicationArea = All;

    layout
    {
        area(Content)
        {
            repeater(Passengers)
            {
                field(Name; Name) { ApplicationArea = All; }
                field(Class; Class) { ApplicationArea = All; }
                field(Age; Age) { ApplicationArea = All; }
                field("Age is Known"; "Age is Known") { ApplicationArea = All; 
}
                field(Sex; Sex) { ApplicationArea = All; }
                field(Survived; Survived) { ApplicationArea = All; }
            }
        }
    }

    actions
    {
        area(Processing)
        {
            action(Import)
            {
                Caption = 'Import';
                Ellipsis = true;
                Image = ImportExcel;
                ApplicationArea = All;

                trigger OnAction();
                var
                    TitanicPassenger: Record "Titanic Passenger";
                begin
                    TitanicPassenger.DeleteAll();
                    Xmlport.Run(Xmlport::"Import Titanic Passengers", false,
true);
                end;
            }

            action(Train)
            {
                Caption = 'Train Model';
                Image = CalculatePlan;
                ApplicationArea = All;

                trigger OnAction();
                var
                    Prediction: Codeunit "Titanic Passenger Pred. Mgt.";
                begin
                    Prediction.Train();
                end;
            }
        }
    }
}
```

Create the prediction page

1. Create a new file and name it Page 50134 Titanic Prediction.al.
2. In the file, declare page 50134 "Titanic Prediction".
3. Copy the entire content of page 50133 **Titanic Passengers** and paste it into the page **Titanic Prediction**.
4. Set the **SourceTableTemporary** page property to true.
5. Make the **Survived** field not editable.
6. Add the **Confidence** field after the **Survived** field. Make it not editable.

7. Modify the **Predict** action. Make it invoke the Predict method of the **Titanic Passenger Pred. Mgt.** codeunit.
8. Rename the **EntryNo** global variable to **NextId**.
9. Remove the **MinimumConfidence** global variable.
10. Modify the definition of the OnOpenPage trigger:
    a. Remove the assignment of the **MinimumConfidence** variable.
    b. Modify the TestField line by replacing the **Sales Volume Model** with **Titanic Prediction Model**.

Solution: Page 50134 Titanic Prediction.al

```
page 50134 "Titanic Prediction"
{
    Caption = 'Titanic Prediction';
    PageType = List;
    SourceTable = "Titanic Passenger";
    SourceTableTemporary = true;
    UsageCategory = Lists;
    ApplicationArea = All;

    layout
    {
        area(Content)
        {
            repeater(Passengers)
            {
                field(Name; Name) { ApplicationArea = All; }
                field(Class; Class) { ApplicationArea = All; }
                field(Age; Age) { ApplicationArea = All; }
                field("Age is Known"; "Age is Known") { ApplicationArea = All;
}
                field(Sex; Sex) { ApplicationArea = All; }
                field(Survived; Survived)
                {
                    Editable = false;
                    ApplicationArea = All;
                }
                field(Confidence; Confidence)
                {
                    Editable = false;
                    ApplicationArea = All;
                }
            }
        }
    }

    actions
    {
        area(Processing)
        {
            action(Predict)
            {
                Caption = 'Predict';
                Image = CalculatePlan;
                ApplicationArea = All;

                trigger OnAction();
                var
                    Prediction: Codeunit "Titanic Passenger Pred. Mgt.";
                begin
                    Prediction.Predict(Rec);
                end;
            }
        }
    }

    var
        NextId: Integer;

    trigger OnNewRecord(BelowxRec: Boolean);
    begin
        NextId += 1;
        Id := NextId;
    end;

    trigger OnOpenPage();
    var
        Setup: Record "Classification Setup";
    begin
        Setup.Get();
        Setup.TestField("Titanic Prediction Model");
    end;
}
```

Final challenge: business logic

Your task here is to complete the business logic in the **Titanic Passenger Pred. Mgt.** codeunit. Use the **Sales Volume Classif. Mgt.** codeunit as your template.

Try not to consult the solution code, unless absolutely necessary.

After you have modified the business logic, run your extension, import the Titanic dataset, train the model, enter data for a few hypothetical passengers, and then run the prediction.

Solution: Codeunit 50133 Titanic Passenger Pred. Mgt.al

```
codeunit 50133 "Titanic Passenger Pred. Mgt."
{
    local procedure GetSetup(var Setup: Record "Classification Setup");
    begin
        Setup.Get();
        Setup.TestField("API URI");
        Setup.TestField("API Key");
    end;

    procedure Train();
    var
        Setup: Record "Classification Setup";
        TitanicPassenger: Record "Titanic Passenger";
        Prediction: Codeunit "ML Prediction Management";
        Model: Text;
        ModelQuality: Decimal;
    begin
        GetSetup(Setup);
        Prediction.Initialize(Setup."API URI", Setup."API Key", 0);

        Prediction.SetRecord(TitanicPassenger);
        Prediction.AddFeature(TitanicPassenger.FieldNo(Class));
        Prediction.AddFeature(TitanicPassenger.FieldNo(Age));
        Prediction.AddFeature(TitanicPassenger.FieldNo("Age is Known"));
        Prediction.AddFeature(TitanicPassenger.FieldNo(Sex));
        Prediction.SetLabel(TitanicPassenger.FieldNo(Survived));

        Prediction.Train(Model, ModelQuality);

        Setup.SetTitanicModel(Model);
        Setup."Titanic Predict. Model Quality" := ModelQuality;
        Setup.Modify(true);

        Message('Model is trained. Quality is %1%',
            Round(ModelQuality * 100, 1));
    end;

    procedure Predict(var TitanicPassenger: Record "Titanic Passenger"
temporary);
    var
        Setup: Record "Classification Setup";
        Prediction: Codeunit "ML Prediction Management";
    begin
        GetSetup(Setup);
        Setup.TestField("Titanic Prediction Model");
        Prediction.Initialize(Setup."API URI", Setup."API Key", 0);

        Prediction.SetRecord(TitanicPassenger);
        Prediction.AddFeature(TitanicPassenger.FieldNo(Class));
        Prediction.AddFeature(TitanicPassenger.FieldNo(Age));
        Prediction.AddFeature(TitanicPassenger.FieldNo("Age is Known"));
        Prediction.AddFeature(TitanicPassenger.FieldNo(Sex));
        Prediction.SetConfidence(TitanicPassenger.FieldNo(Confidence));
        Prediction.SetLabel(TitanicPassenger.FieldNo(Survived));

        Prediction.Predict(Setup.GetTitanicModel());
    end;
}
```