Azure Machine Learning Documentation

Written by Team 4- Shermaine Lim Si Hui

Contents

Introduction

Logic University faces several issues pertaining to stocks that requires inventory supply planning and control. With Machine Learning to automate the demand forecasting, predicting demands to fulfil orders is simplified, able to process larger datasets , and predicts with higher accuracy to prevent over/under-stocking compared to the traditional and tedious way to plan and control supply and demand orders.

Issues

- Although there is a pre-defined re-order stock level, items are often under-stocked and unable to fulfil demands from departments.

- The existing stocks in inventory deplete too fast before the reordered stocks arrive, and, as a result, many departments are unsatisfied and complain. This means the inventory is not sufficiently stocked and planned to bolster for current or future demands.

- The store clerk also takes too long to consolidate stationery requests, information and documents are stored in several places, making it difficult to retrieve and track information quickly.

Solution

- Machine Learning Demand Forecasting Automation to reduce forecasting errors by 70-90%

- Machine Learning Demand Forecasting Automation to reduce time taken to do demand forecasting and stock planning by 95-97% compared to traditional and manual methods of calculations which are also heavily influenced by human decision/bias.

- Technology Enhanced Supply Chain Management will lead to an improved accuracy that aims to predict and fulfil the demands of customers and ensure inventory is not over/under-stocked to facilitate efficient warehouse space management

- The computerisation of the stationery store inventory functions, consolidation of Orders from Departments and Purchase Orders to Suppliers

- Storing documents and information as e-copy on MSSQL and Cloud so it is easy and fast to retrieve and track

Demand Forecasting and Supply Chain Management

Demand forecasting is a basic component of inventory, demand and supply planning and control, impacting competitiveness and profitability, providing important information for purchasing decisions, stock levels, logistics, finance and warehouse inventory space and cost management.

Traditional Predictions

Biases and systematic errors in demand forecasting are frequent in supply chain decision-making process when human factors and personal judgement are involved (Arvan et al.,2018). In addition, human decisions may become difficult when complex forecasting models are used because they may need many variables to produce greater accuracy , calling for the need for support from automated tools(Puchalsky et al., 2018).

Levelling up Demand Forecasting with Machine Learning and Automated Technology

Demand forecasting is a field of predictive analytics that predicts the demands of customers through analysing statistical data and identifying patterns and correlations. Machine Learning takes the practice to a higher level, improving the accuracy and reducing forecasting errors by 70-90% in supply chain management compared to traditional predictions (Tarallo, E. et al, 2019).

Business Challenges and Demand Forecasting Importance (Logic University)

A context always exists around customer behaviour. It can be an upcoming event, holiday, or trend etc. As real product demand varies, businesses may face challenges in:

1) Income and profit loss when products are out of stock or a service is unavailable

2) Cash tied up in stock or reduced margins that come with getting it out of the warehouse

For Logic University, since real product demand varies, they face challenges in income, profit loss and customer unsatisfaction/complaints as products are frequently out of stock. They also have poor demand and supply planning which means products are always under-stocked or some products may be over-stocked, a consequence of poor inventory management.

Therefore, products that are always under-stocked, will become out of stock which incur profit losses , customer unsatisfaction because they are unable to fulfil customer demands, while over-stocked products unnecessarily increase warehouse space and costs.

Demand forecasting is statistics-heavy and data-rich, which is applicable for machine learning algorithms. The automation of data flows will help to manage logistics and optimize an organization's supply chain, inventory management and performance (Budek, 2018) .

Benefits of Demand Forecasting with Machine Learning for Logic University

According Taranenko (2020), Machine learning techniques allow predicting the amount/quantity of products to be purchased during a defined future period. The system will learn from the data for improved analysis and prediction. Compared to traditional demand forecasting methods, machine learning forecasting can:

- Accelerate and handle large data processing quickly
- Provide higher accuracy of forecast
- Adapt and update forecasts based on more recent data
- Improve Supplier Relation Management, by having the prediction of customer demands and quantity of orders, it improves decision making to plan with suppliers to facilitate Customer Relationship Management , ensuring customer demands are met and satisfied on time
- Improves Customer Relationship Management, because customers planning to order something may want the product to be available immediately. Demand forecasting allows predicting which products most likely need to be purchased in the next period, by creating optimal stock and safety stock in inventory for these products, there is immediate and sufficient stock availability for customers, increasing customer satisfaction

Azure Machine Learning Studio

Azure Machine Learning can build advanced analytical solutions and has great documentation to build, deploy and manage which is time and maintenance efficient with the scalability to manage and utilize big data (Azure Machine Learning documentation, 2020).

Azure Machine Learning Studio is a user interface layer with tools for authoring experiments with a palette of available modules, uploading and saving models, datasets user assets, sharing experiments and converting experiments to publish web services and consume at back-end development (Azure Machine Learning documentation, 2020).

For Logic University , **Figure 1** shows the experimental graph and the module palette experiment consisting of a directed acyclic graph connecting several modules. Modules encapsulate the data, machine learning algorithms, data transformation routines, saved models and user-defined code. Modules are divided into various categories like machine learning and data manipulation etc. Such modules can be referenced from multiple experiments and web services. **Figure 2** shows the predictive experiment graph with added Web Service input and Web Service output to make predictions with Elastic APIs.

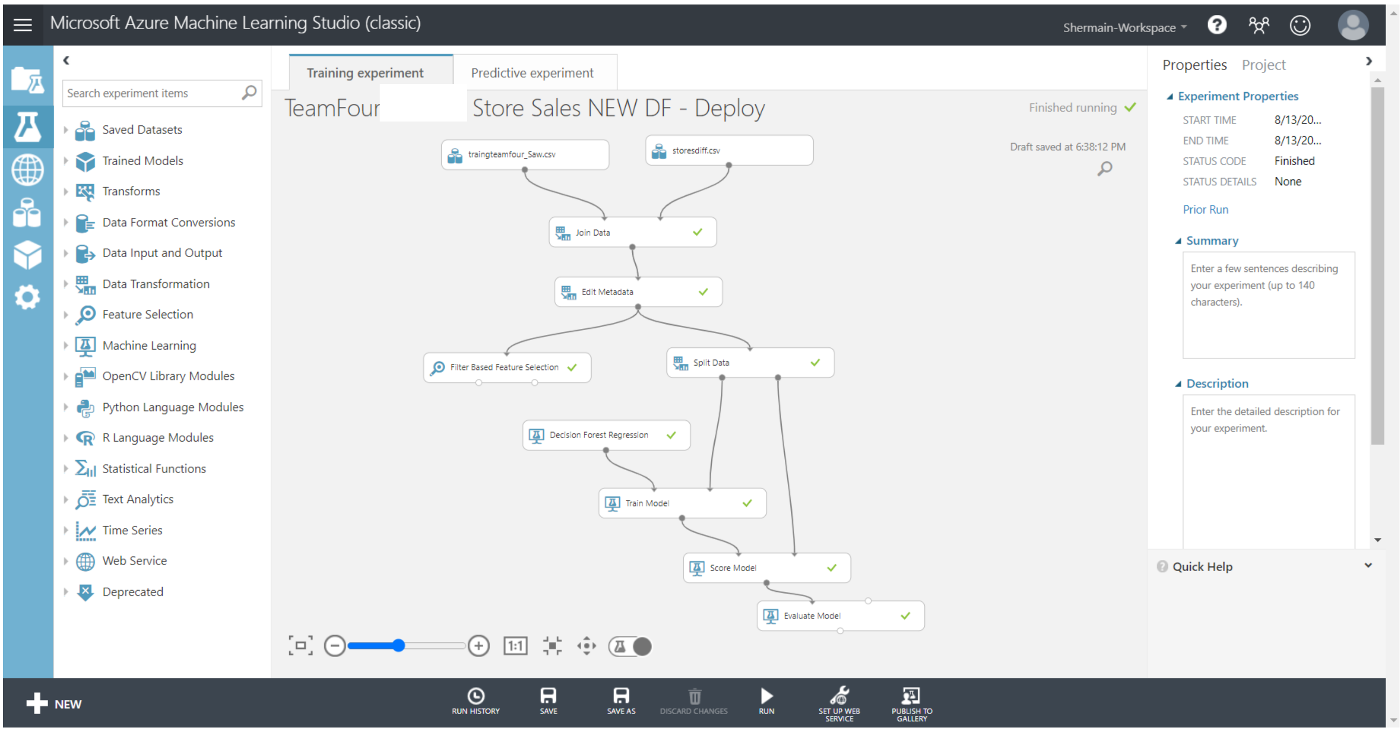


Figure 1: Azure Machine Learning Studio with the Training experimental graph for Demand Forecasting of Logic University

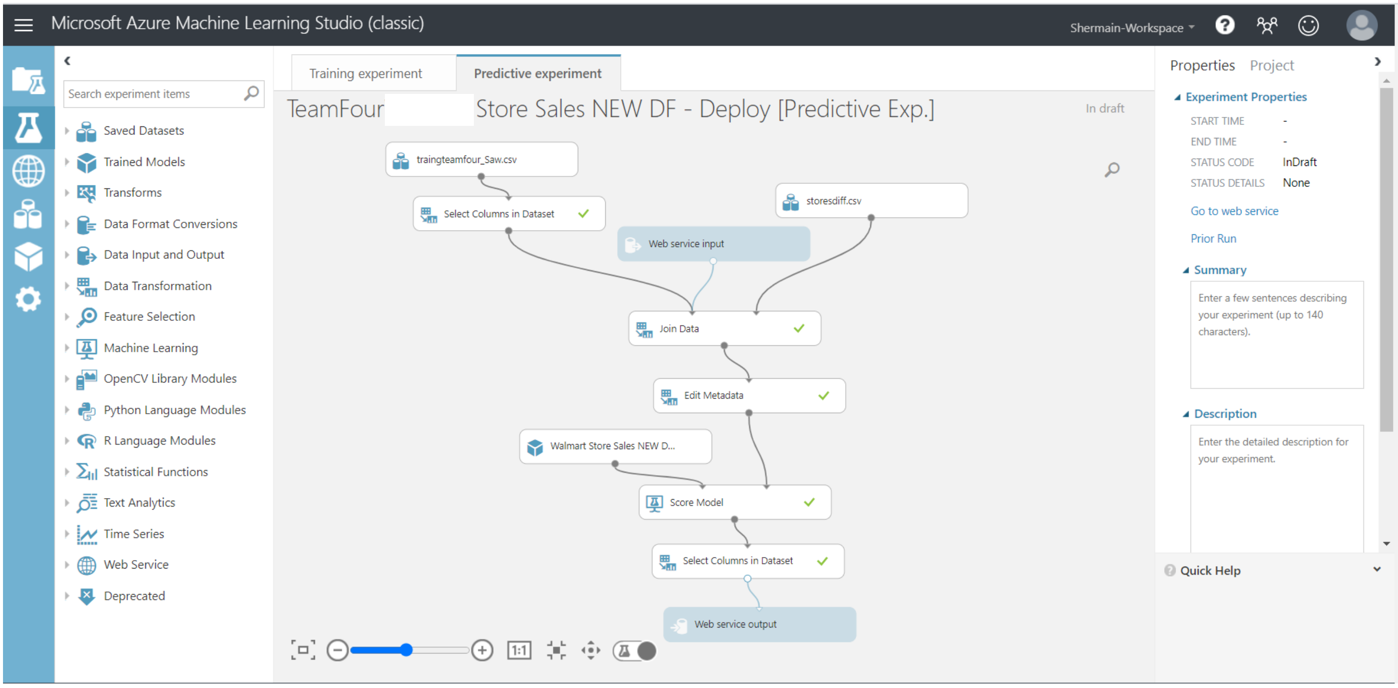


Figure 2: Azure Machine Learning Studio with the Predictive experimental graph for Web Service of Logic University

Process of Building Training Model and Predictive Web Service

Findings and Observations

The lack of data to train test model will affect the accuracy. Accuracy is improved through Data prep processes such as selecting only relevant columns and using Filter Based Feature Selection. Experimenting with different regression algorithms and using Decision forest regression algorithm achieved the best accuracy for the model.

Experimented Regression Algorithms:

- Bayesian Linear Regression
- Boosted Decision Tree Regression
- Linear Regression
- Decision Forest Regression (final)
- Random Forest Regression
- Neural Network Regression

An experiment comprises of datasets that provide data to analytical modules, which are connected to construct the predictive analysis model. One can use data from one or more sources, transform and analyse data through various data manipulation and statistical functions to generate result sets. The process is iterative. As one modifies the various functions and parameters, the results converge until a trained and effective model has been built. Then the training experiment can be converted to a predictive experiment which is then published as a web service so the model and Elastic APIs can be accessed (*ML Studio (classic): Deploy a web service - Azure* , 2017).

Diagram 1 displays a Capabilities Overview of Azure Machine Learning Studio.

Figures 3 and **4** show the flow of the experiments with the modules performing different functions, deploy web service and make predictions with Elastic APIs.

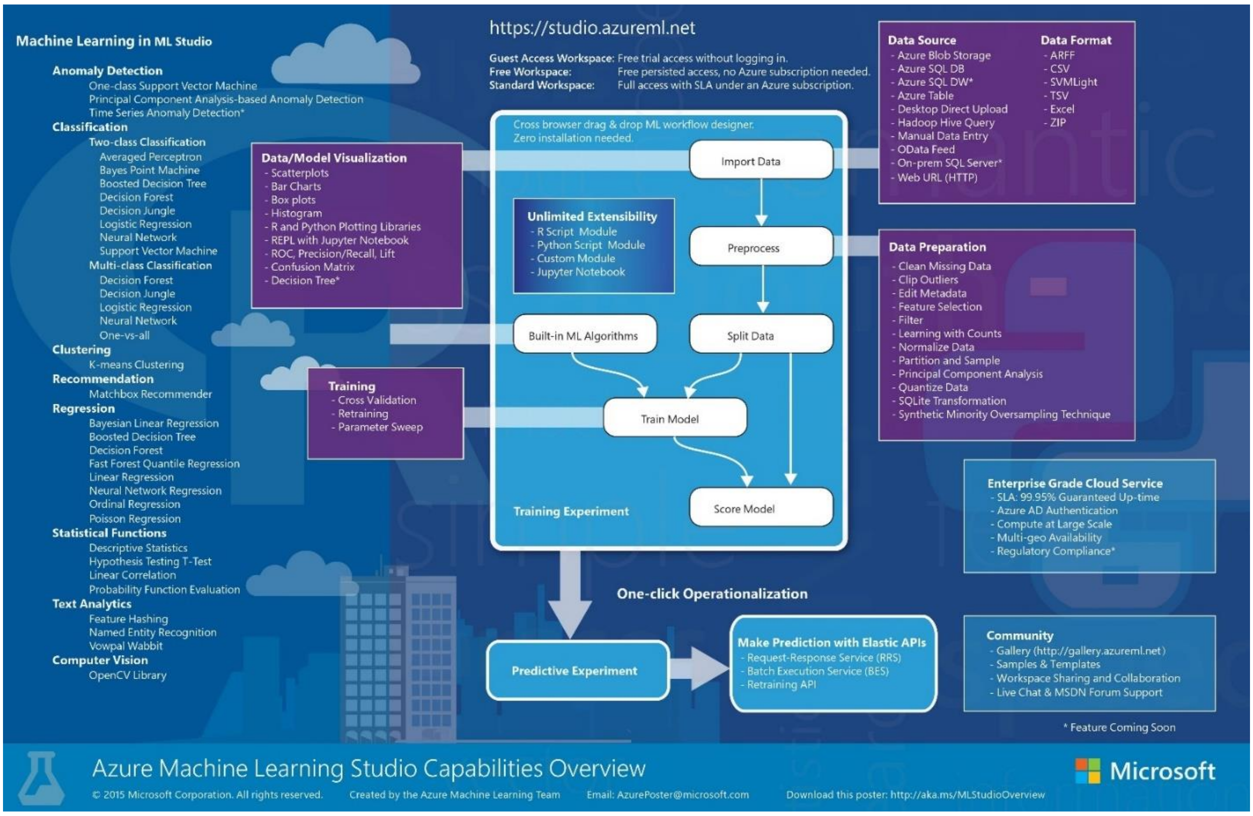


Diagram 1: Capabilities Overview of Azure Machine Learning Studio. Taken from: https://docs.microsoft.com/en-us/azure/machine-learning/studio/deploy-a-machine-learning-web-service

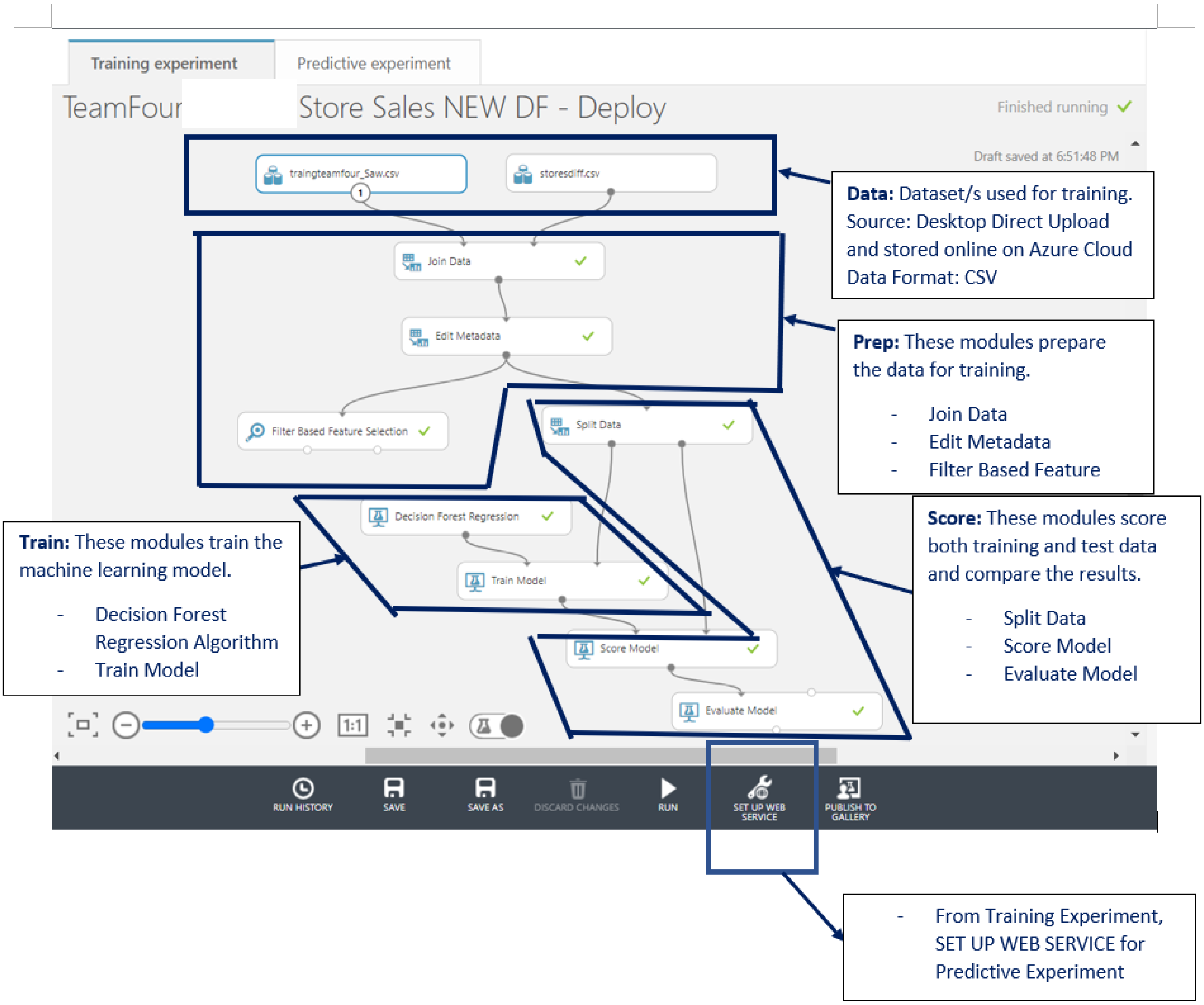


Figure 3: The flow of the Training Experiment modules that perform different functions

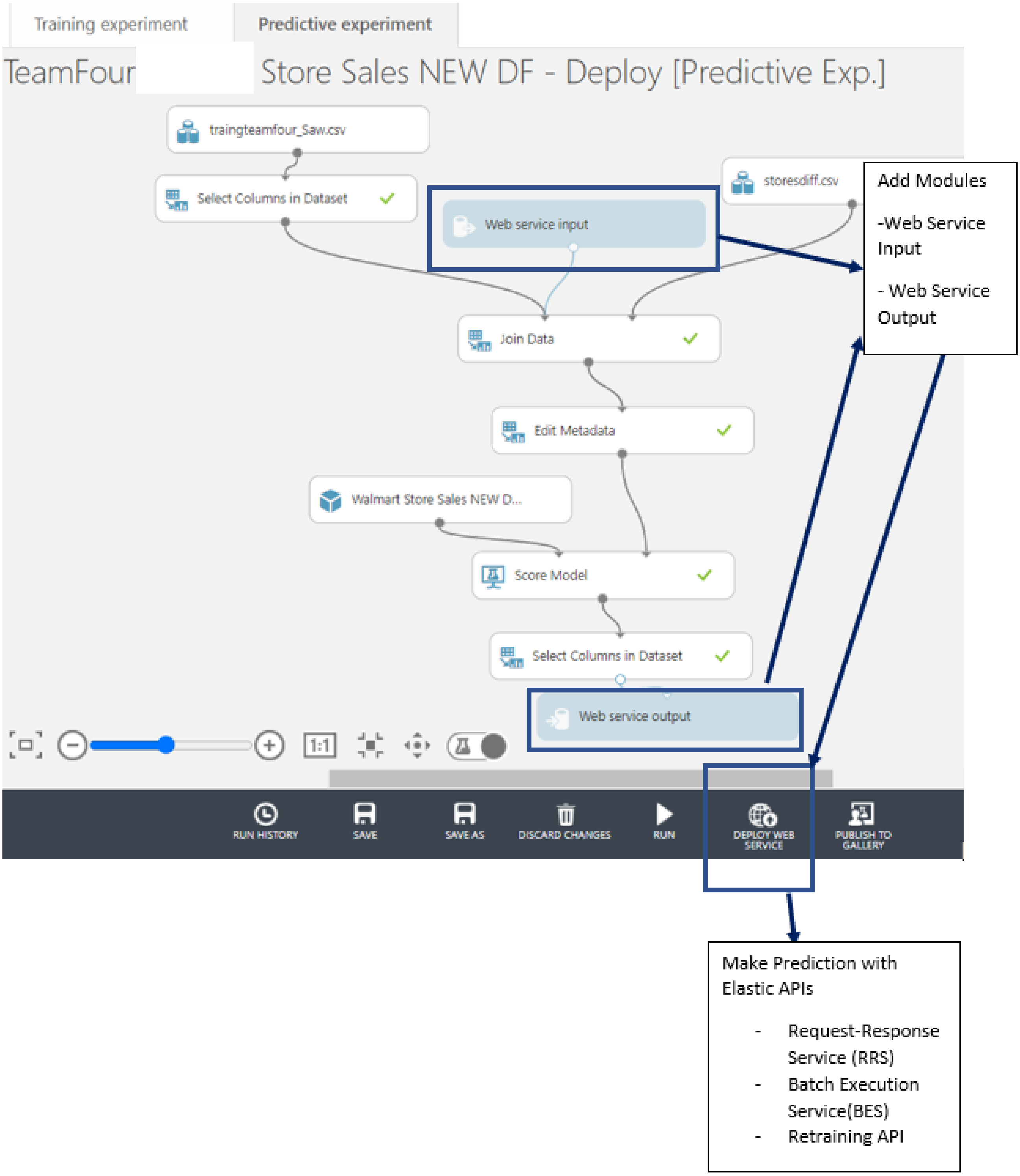


Figure 4: The flow of the Predictive Experiment modules that perform different functions and deploy web service

Test Web Service

After creating a web service. From here, you can easily access both the Test page and Consume page as shown in **Figure 5**. In Logic University ASP.NET project, Request-Response web service was consumed with C# and apiKey is unique in **Figure 6** and **7**.

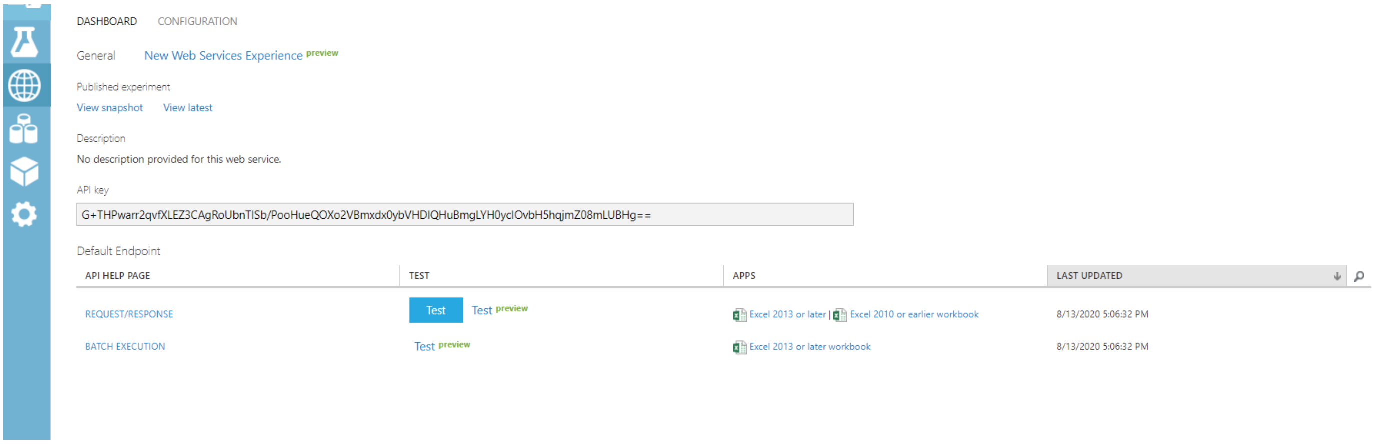
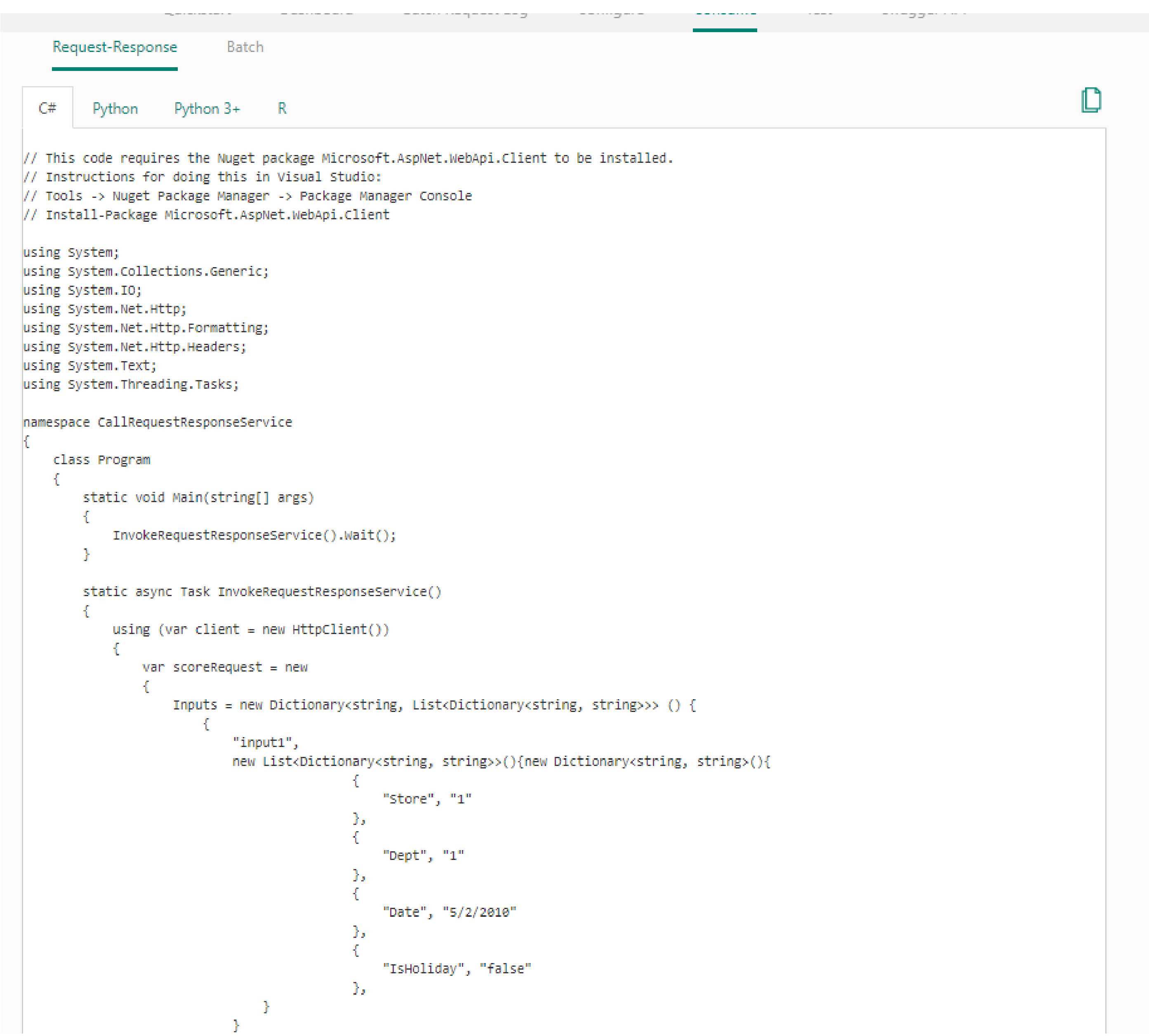


Figure 5 : Web Service , apiKey




```
// This code requires the Nuget package Microsoft.AspNet.WebApi.Client to be installed.
// Instructions for doing this in Visual Studio:
// Tools -> Nuget Package Manager -> Package Manager Console
// Install-Package Microsoft.AspNet.WebApi.Client

using System;
using System.Collections.Generic;
using System.IO;
using System.Net.Http;
using System.Net.Http.Formatting;
using System.Net.Http.Headers;
using System.Text;
using System.Threading.Tasks;

namespace CallRequestResponseService
{
    class Program
    {
        static void Main(string[] args)
        {
            InvokeRequestResponseService().Wait();
        }

        static async Task InvokeRequestResponseService()
        {
            using (var client = new HttpClient())
            {
                var scoreRequest = new
                {
                    Inputs = new Dictionary<string, List<Dictionary<string, string>>> () {
                        {
                            "input1",
                            new List<Dictionary<string, string>>(){new Dictionary<string, string>(){
                                            {
                                                "Store", "1"
                                            },
                                            {
                                                "Dept", "1"
                                            },
                                            {
                                                "Date", "5/2/2010"
                                            },
                                            {
                                                "IsHoliday", "false"
                                            },
                                }
                            }
```

Figure 6 : Consume Web Service Part 1

```
                    },
                },
                GlobalParameters = new Dictionary<string, string>() {
                }
            };

            const string apiKey = "abc123"; // Replace this with the API key for the web service
            client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue( "Bearer", apiKey);
            client.BaseAddress = new Uri("https://ussouthcentral.services.azureml.net/workspaces/9c7eacb777ca45079d3a311454a48511/services/c
284dfa711c8462488effaea2d008d1f/execute?api-version=2.0&format=swagger");

            // WARNING: The 'await' statement below can result in a deadlock
            // if you are calling this code from the UI thread of an ASP.Net application.
            // One way to address this would be to call ConfigureAwait(false)
            // so that the execution does not attempt to resume on the original context.
            // For instance, replace code such as:
            //      result = await DoSomeTask()
            // with the following:
            //      result = await DoSomeTask().ConfigureAwait(false)

            HttpResponseMessage response = await client.PostAsJsonAsync("", scoreRequest);

            if (response.IsSuccessStatusCode)
            {
                string result = await response.Content.ReadAsStringAsync();
                Console.WriteLine("Result: {0}", result);
            }
            else
            {
                Console.WriteLine(string.Format("The request failed with status code: {0}", response.StatusCode));

                // Print the headers - they include the requert ID and the timestamp,
                // which are useful for debugging the failure
                Console.WriteLine(response.Headers.ToString());

                string responseContent = await response.Content.ReadAsStringAsync();
                Console.WriteLine(responseContent);
            }
        }
    }
}
}
```

Figure 7 : Consume Web Service Part 2 , apiKey

Consume Azure Machine Learning in ASP.NET CORE MVC

With Azure Machine Learning Web Services, an external application (ASP.NET) can communicate with the Machine Learning scoring model in real time with API call. The API call will return the prediction results from the model to the external ASP.NET application.

Azure Machine Learning has two types of services:

Request-Response Service(RRS) and Batch Execution Service(BES). RRS will be used in the ASP.NET Project while BES will be used and tested for the model's accuracy via Excel.

Request-Response Service (RRS)

RRS is a low latency, highly scalable service that provides an interface to the stateless models created and deployed from the Machine Learning Studio. To generate the result, in ASP.NET, a .cshtml razor page is created for the UI and to get the user inputs to pass the parameters from the inputs into the Store Controller.

Cleaner Coding and calling Azure Machine Learning API- QtyPredictionServices.cs

To facilitate cleaner and readable codes, in **Figure 8**, a Services folder is created and the codes from **Figure 6** and **7** are added into QtyPredictionServices.cs in **Figure 9** so the Store controller or any other controllers/ functions can call QtyPredictionServices and reduce duplicate codes. In QtyPredictionServices.cs , to make the API service call, we pass the unique API key in **Figure 10**, that was created when the web service was deployed.

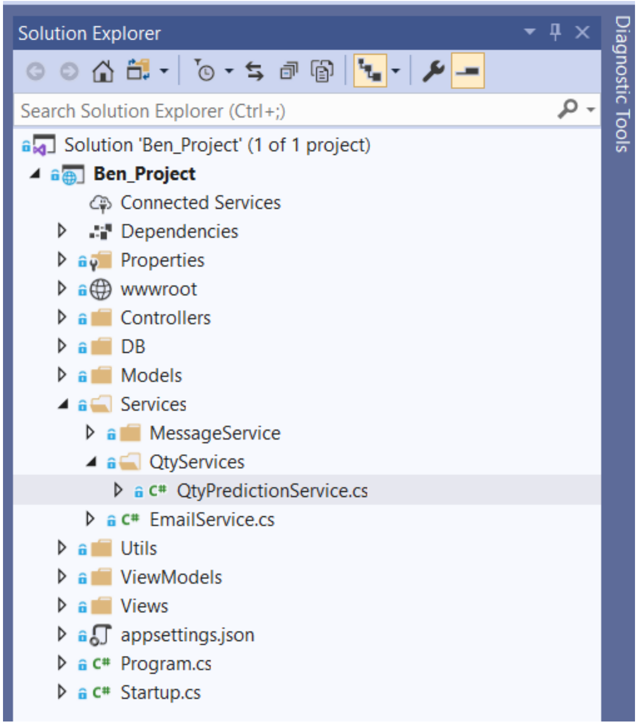


Figure 8: Services, QtyServices

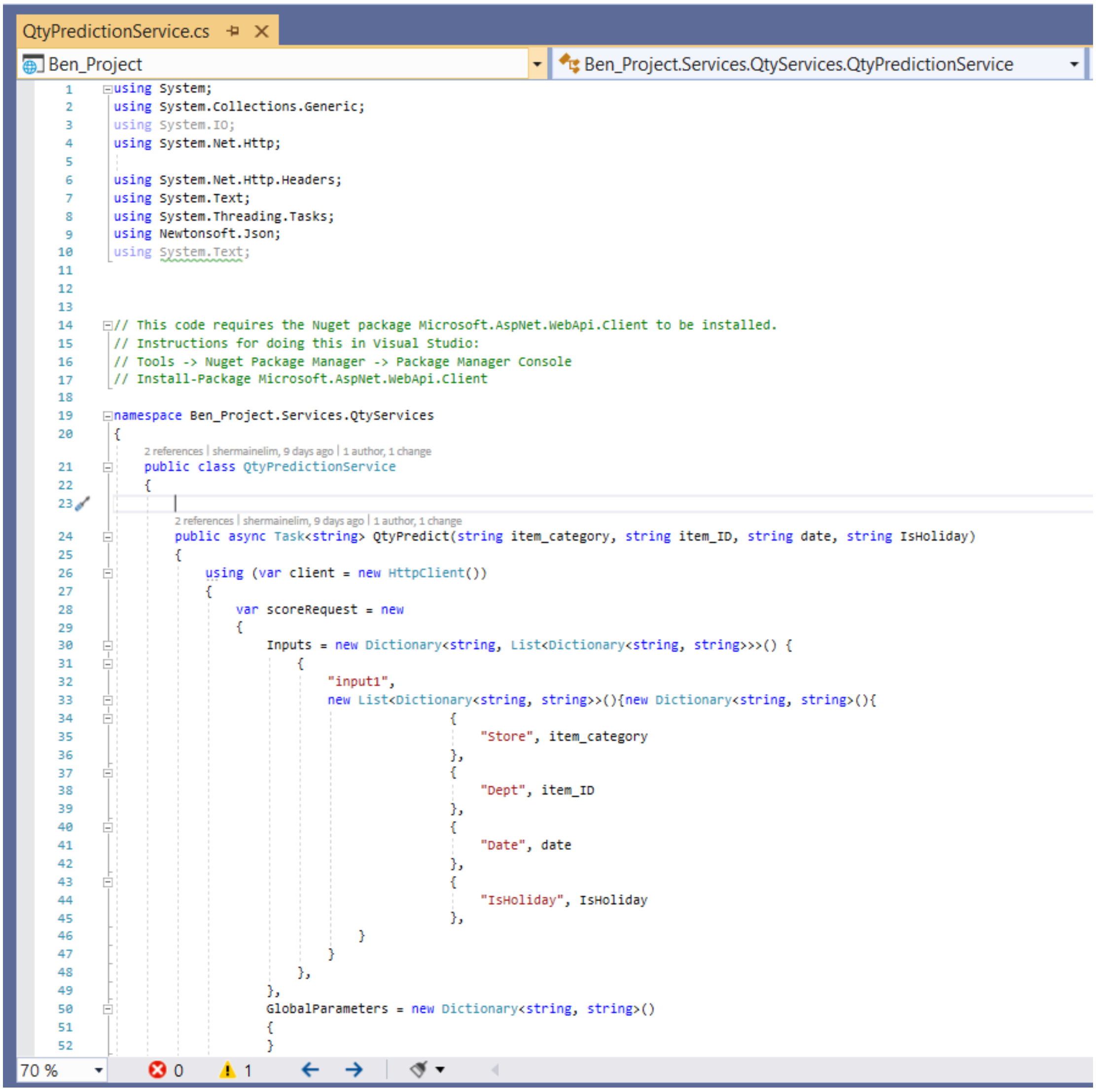
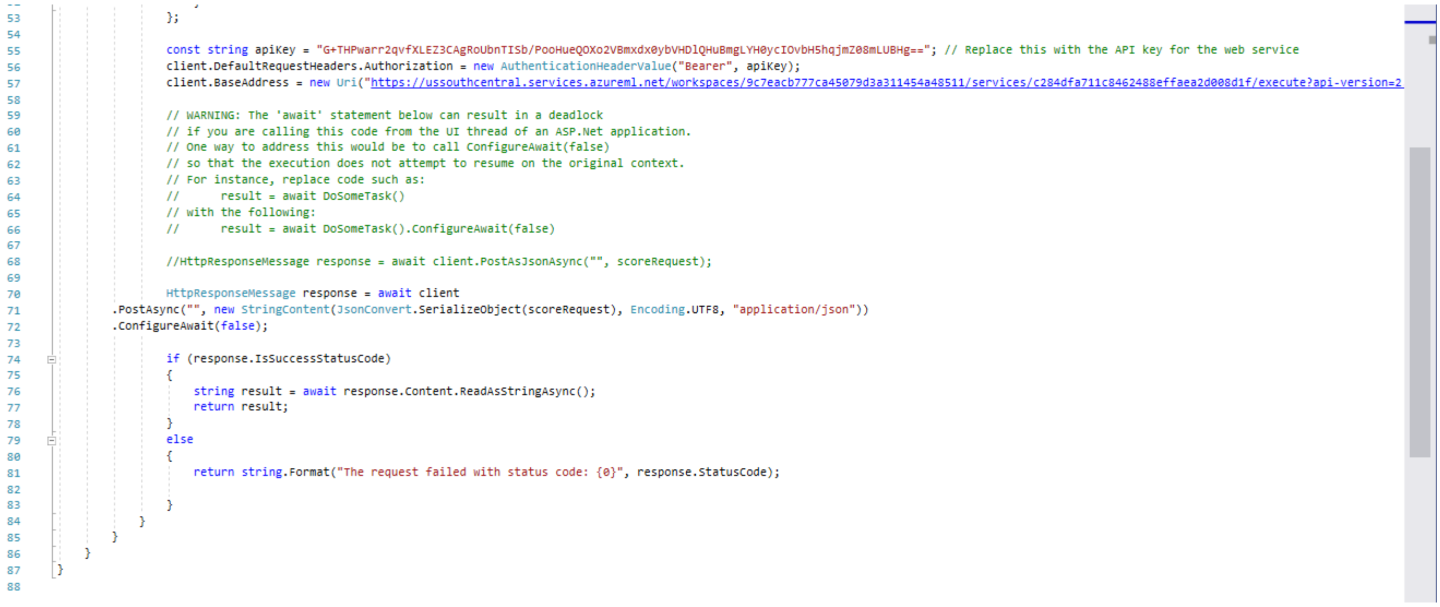


Figure 9: Part 1_QtyPredictionService.cs with QtyPredict and Parameters (string item_category, string item_ID, string data, string isHoliday)



```
                };

                const string apiKey = "G+THPwarr2qvfXLEZ3CAgRoUbnTISb/PooHueQOXo2VBmxdx0ybVHDlQHuBmgLYH0ycIOvbH5hqjmZ08mLUBHg=="; // Replace this with the API key for the web service
                client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", apiKey);
                client.BaseAddress = new Uri("https://ussouthcentral.services.azureml.net/workspaces/9c7eacb777ca45079d3a311454a48511/services/c284dfa711c8462488effaea2d008d1f/execute?api-version=2

                // WARNING: The 'await' statement below can result in a deadlock
                // if you are calling this code from the UI thread of an ASP.Net application.
                // One way to address this would be to call ConfigureAwait(false)
                // so that the execution does not attempt to resume on the original context.
                // For instance, replace code such as:
                //      result = await DoSomeTask()
                // with the following:
                //      result = await DoSomeTask().ConfigureAwait(false)

                //HttpResponseMessage response = await client.PostAsJsonAsync("", scoreRequest);

                HttpResponseMessage response = await client
            .PostAsync("", new StringContent(JsonConvert.SerializeObject(scoreRequest), Encoding.UTF8, "application/json"))
            .ConfigureAwait(false);

                if (response.IsSuccessStatusCode)
                {
                    string result = await response.Content.ReadAsStringAsync();
                    return result;
                }
                else
                {
                    return string.Format("The request failed with status code: {0}", response.StatusCode);

                }
            }
        }
    }
}
```

Figure 10: Part 2_QtyPredictionService.cs with apiKey

Cleaner Coding and calling Azure Machine Learning API- Razor View (.cshtml)

On Azure Machine Learning, the Testing User Interface as shown in Figure 11. As reference to **Figure 11**, we create a similar User Interface with Razor View in ASP.NET in **Figure 12**. **Figure 12** contains a form that when PredictionBtn is submitted, the data - item_category, item_ID, date, isHoliday goes to the logic, IActionResult Prediction(string item_category, string item_ID, string date, string IsHoliday) in the Store Controller of **Figure 13**.

The logic will process, and check whether there is enough stock or need to order. Each input also contains validation to ensure no null empty fields, inputs requiring string can only accept string and inputs requiring int can only accept int value.

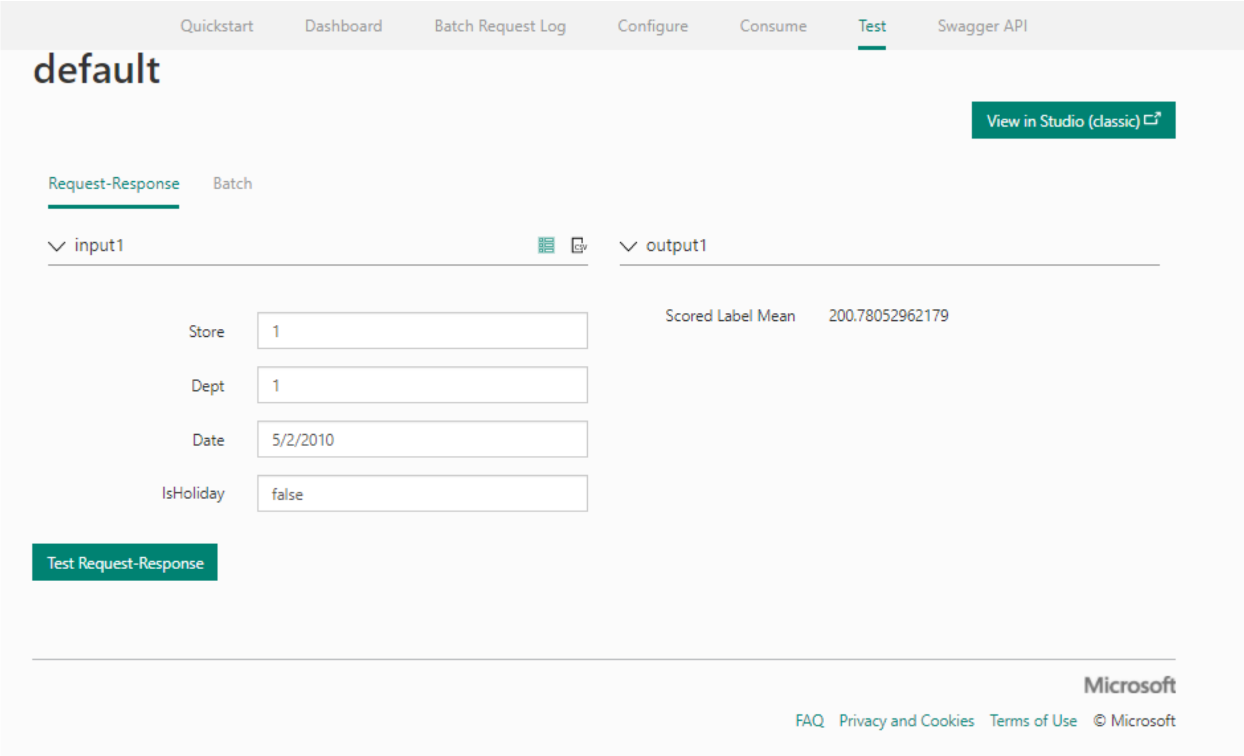


Figure 11: Azure Machine Learning Testing User Interface

```
<p style="color:red">@TempData["Error"]</p>


<h2>@TempData["Message"]</h2>
<h2>@TempData["result"]</h2>

<form asp-controller="Store" asp-action="Prediction" method="post">
    <table align="center">
        <tr>
            <td>Item Category:</td>
            <td>
                <input id="item_category" name="item_category" value="@ViewData["item_category"]" />

            </td>
        </tr>
        <tr>
            <td>Item ID:</td>
            <td><input id="item_ID" name="item_ID" value="@ViewData["item_ID"]" /></td>
        </tr>

        <tr>
            <td>Date:</td>
            <td>
                <input id="date" name="date" type="date" value="@ViewData["date"]" />
            </td>
        </tr>

        <tr>
            <td>IsHoliday?( write true or false only):</td>
            <td>
                <input id="IsHoliday" name="IsHoliday" type="radio" value="true" />
                "true    "
                <input id="IsHoliday" name="IsHoliday" type="radio" value="false" />
                "false"
            </td>
        </tr>

        <tr>
            <td colspan="2" align="right">
                <input id="PredictBtn" type="submit" value="Prediction" />
            </td>
        </tr>
    </table>
</form>
```

Figure 12: Create Razor Form View

StoreController.cs* StoreController.cs QtyPredictionService.cs

Ben_Project.Controllers.StoreController

```
        //TempData["result"] = "";

        return View();
    }

    0 references | shermainelim, 6 days ago | 1 author, 4 changes
    public IActionResult Prediction(string item_category, string item_ID, string date, string IsHoliday)
    {
        int number;
        var result5 = int.TryParse(item_category, out number);
        var result6 = int.TryParse(item_ID, out number);

        if (item_category == null || item_ID == null || date == null || IsHoliday == null)
        {
            TempData["Error"] = "Enter the empty fields";
            return RedirectToAction("Index");
        }
        else if (result5 == false || result6 == false)
        {
            TempData["Error"] = "Enter only int fields";
            return RedirectToAction("Index");
        }

        int itemid = Int32.Parse(item_ID);
        Stock stock = _dbContext.Stocks.SingleOrDefault(x => x.Stationery.Id == itemid);
        int safetyStock = stock.Stationery.ReorderLevel;
        int currentStock = stock.Qty;
        var result = new QtyPredictionService().QtyPredict(item_category, item_ID, date, IsHoliday).Result;

        var result2 = result.Replace("Results", "")
            .Replace("output1", "")
            .Replace("Scored Label Mean", "")
            .Replace("{", "")
            .Replace("}", "")
            .Replace(":", "")
            .Replace("[", "")
            .Replace("]", "")
            .Replace('"', 'o')
            .Replace("o", "");

        TempData["Message"] = result2;

        double final = Math.Round(Double.Parse(result2));
        if (((final + safetyStock) > currentStock))
        {
            TempData["result"] = "You should order : " + ((final + safetyStock) - currentStock);
        }
        else if ((final + safetyStock) < currentStock)
        {
            TempData["result"] = "You have enough stock";
        }

        return RedirectToAction("Index");
    }
```

No issues found

Figure 13: Logic in Store Controller

UI to use Demand Forecasting

Figure 14 shows the user interface created from the MVC codes and the inputs need. **Figure 15** shows the user interface for the demand forecasting results. **Figure 16** and **17** shows the user input validation.

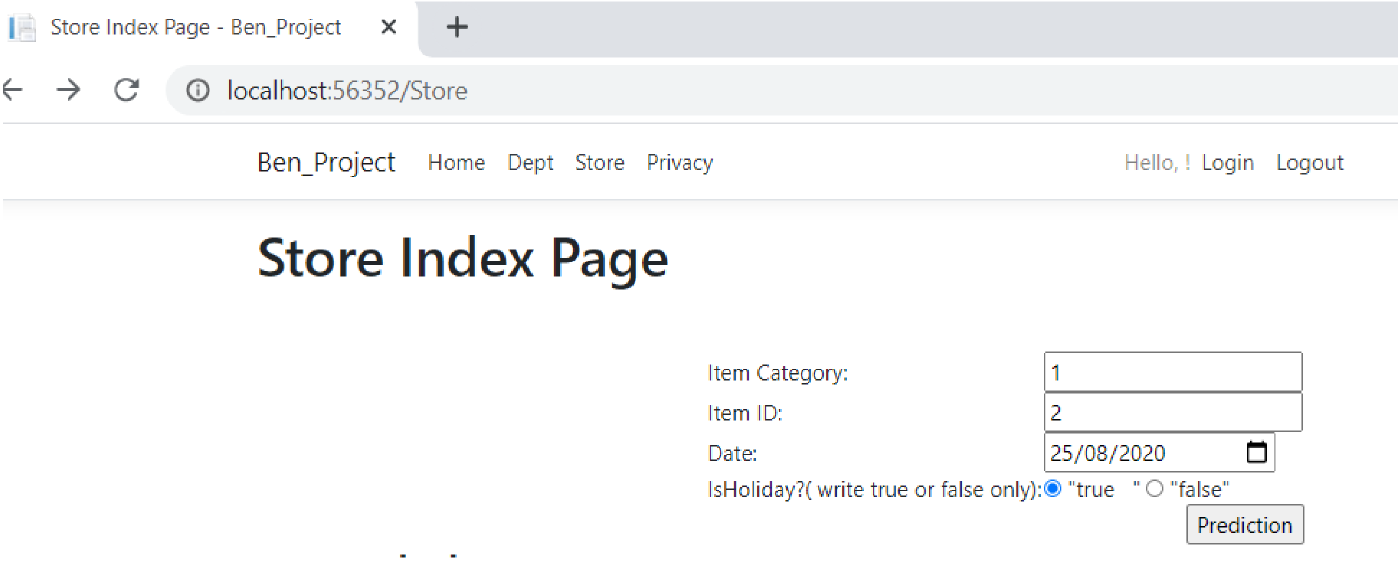


Figure 14: User Interface: Demand Forecasting Inputs

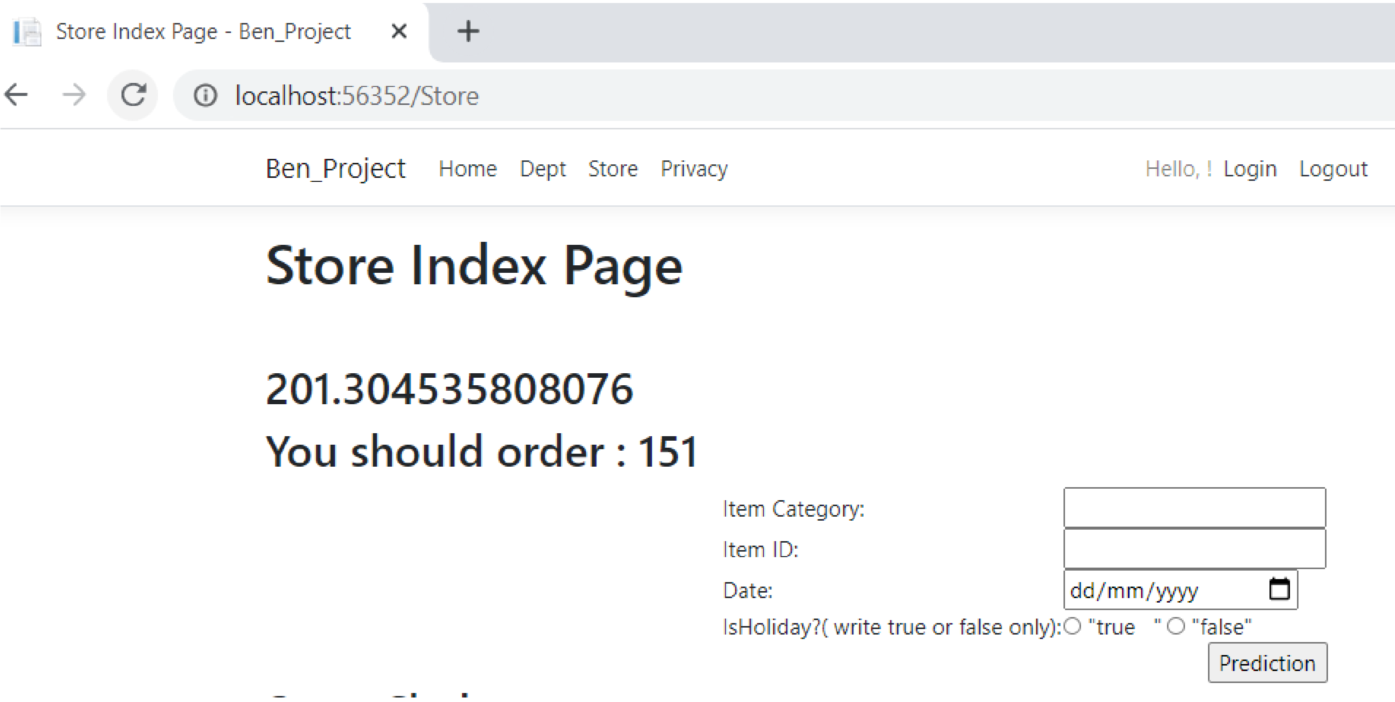


Figure 15: User Interface : Demand Forecasting Results

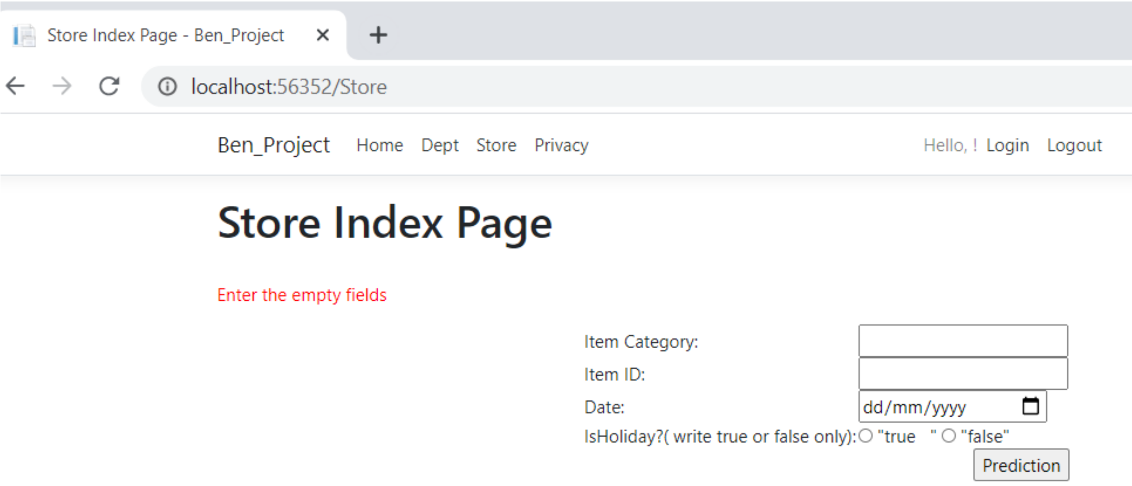


Figure 16: User Interface : Demand Forecasting Validation-1

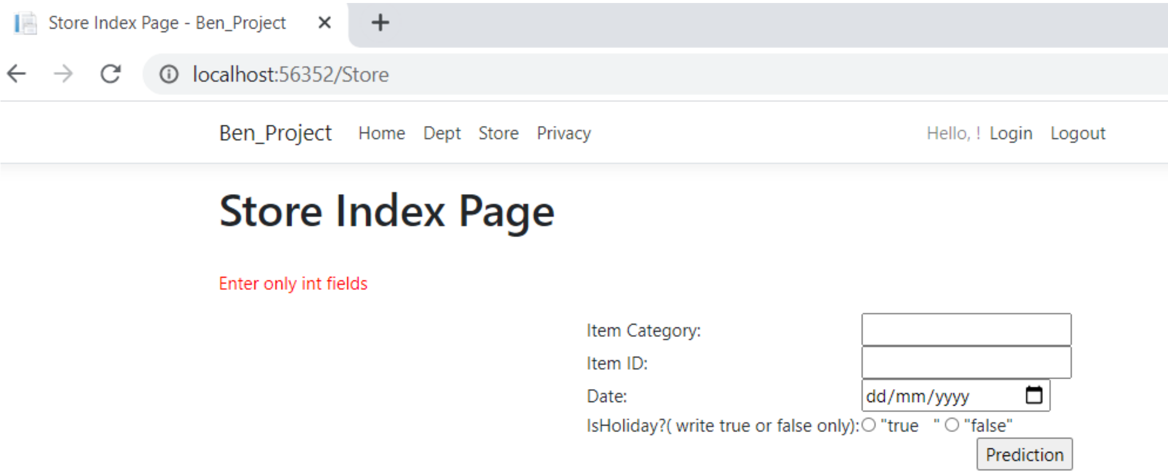


Figure 17: User Interface : Demand Forecasting Validation-2

Purchase Order (PO) Form with Demand Forecasting

The demand forecasting is integrated to the Purchase Order Form and displays as under the Column "Predicted Quantity" so the store clerk knows the Quantity to order from supplier as shown in **Figure 18.**

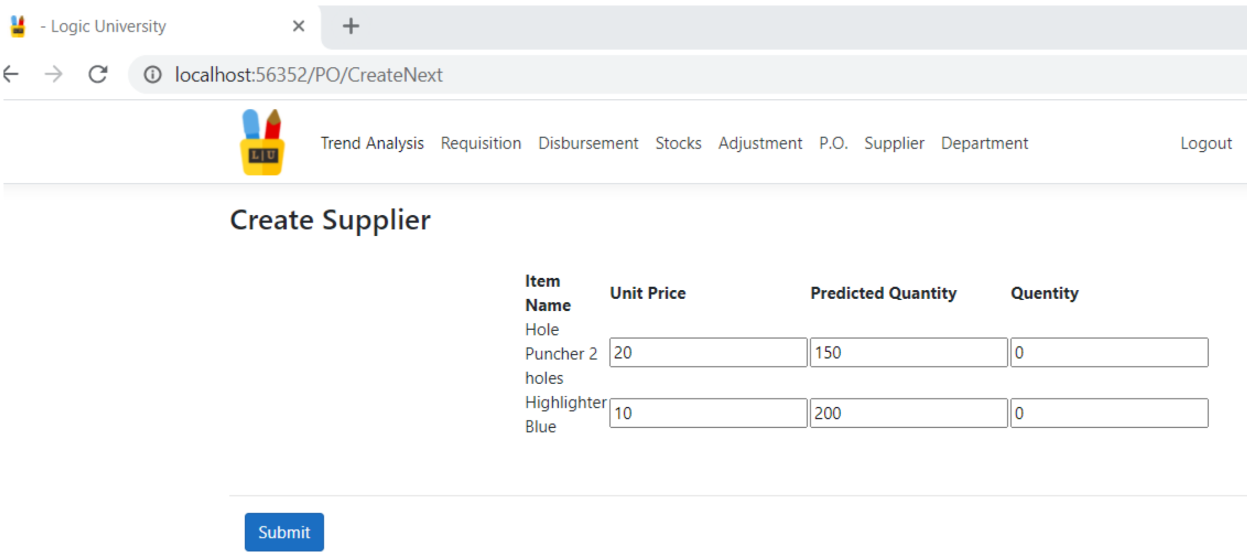


Figure 18: PO Form with Demand Forecasting

Batch Execution Service(BES) and Testing Accuracy of Model in Excel -Part 1

An asynchronous service that scores a batch for data records. A BES can be executed with Excel. The Excel .xlsb is downloaded from Azure Machine Learning. When opening this Excel file, Azure Machine Learning will load (**Figure 19**) to do Batch prediction for many rows. In **Figure 20**, we open another csv file, the csv file that was used to train the model and we copy the columns Store, Dept, Date, Weekly_Sales, Holiday until row 600 to the .xlsb Machine Learning Excel in **Figure 19.**

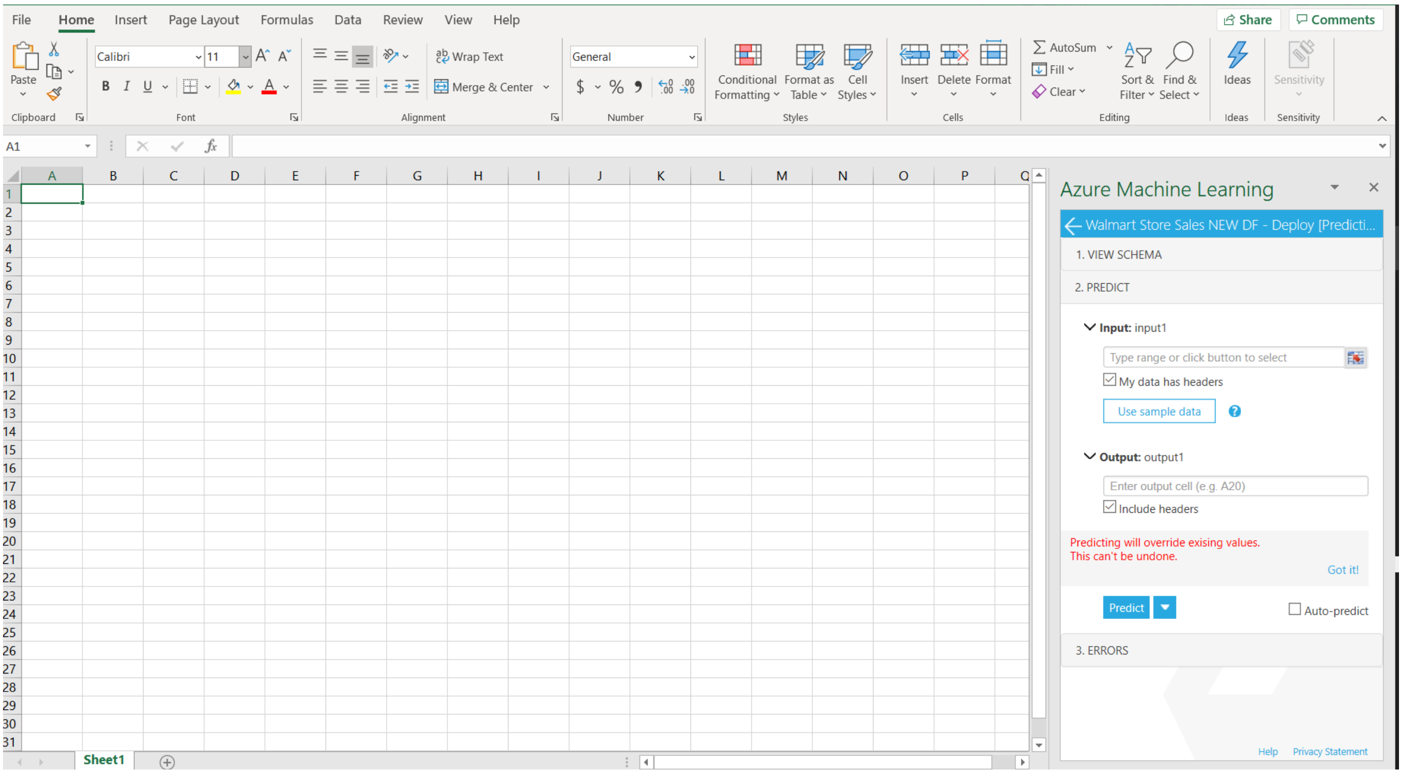
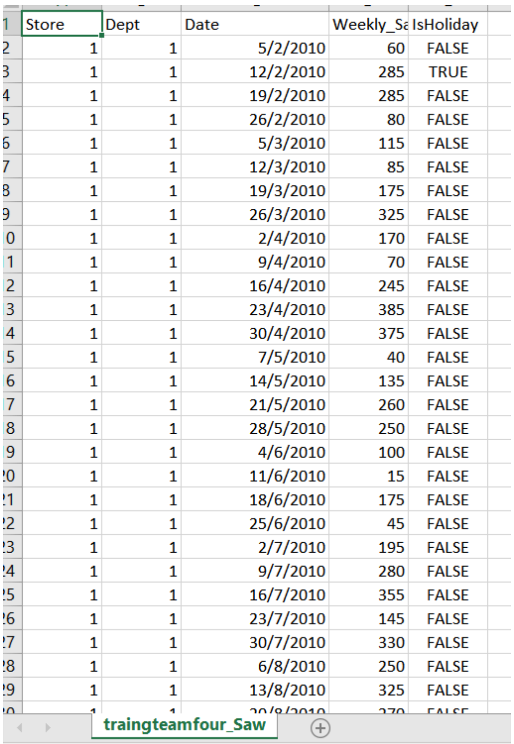


Figure 19: Batch Execution Service with Azure Machine Learning in Excel



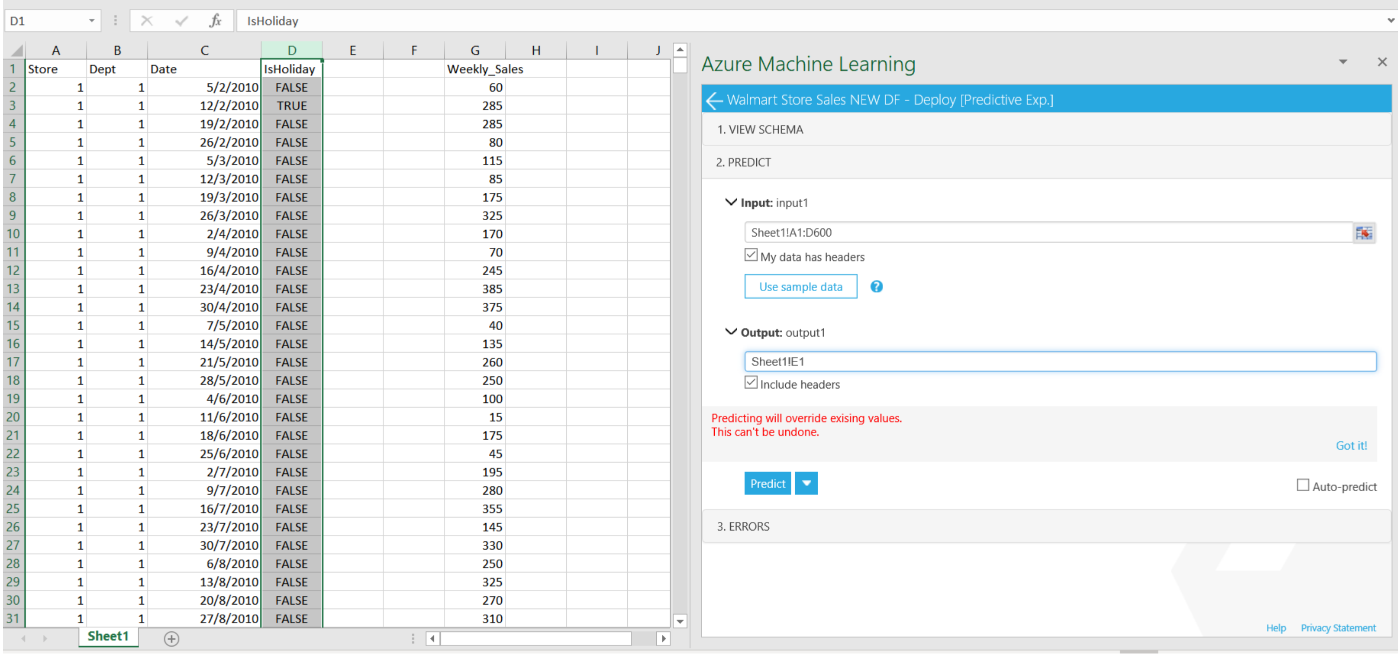
Store	Dept	Date	Weekly_Sa	IsHoliday
1	1	5/2/2010	60	FALSE
1	1	12/2/2010	285	TRUE
1	1	19/2/2010	285	FALSE
1	1	26/2/2010	80	FALSE
1	1	5/3/2010	115	FALSE
1	1	12/3/2010	85	FALSE
1	1	19/3/2010	175	FALSE
1	1	26/3/2010	325	FALSE
1	1	2/4/2010	170	FALSE
1	1	9/4/2010	70	FALSE
1	1	16/4/2010	245	FALSE
1	1	23/4/2010	385	FALSE
1	1	30/4/2010	375	FALSE
1	1	7/5/2010	40	FALSE
1	1	14/5/2010	135	FALSE
1	1	21/5/2010	260	FALSE
1	1	28/5/2010	250	FALSE
1	1	4/6/2010	100	FALSE
1	1	11/6/2010	15	FALSE
1	1	18/6/2010	175	FALSE
1	1	25/6/2010	45	FALSE
1	1	2/7/2010	195	FALSE
1	1	9/7/2010	280	FALSE
1	1	16/7/2010	355	FALSE
1	1	23/7/2010	145	FALSE
1	1	30/7/2010	330	FALSE
1	1	6/8/2010	250	FALSE
1	1	13/8/2010	325	FALSE

Figure 20: CSV Dataset used to train model

Batch Execution Service(BES) and Testing Accuracy of Model in Excel -Part 2

After pasting the no of rows from the csv sheet that was used to train the model to the .xlsb Excel with BES Azure Machine Learning, we only want the inputs from Column A-D (Store, Dept, Date, IsHoliday).

Weekly_Sales from the csv training dataset will not be included in the Prediction as we want the Machine Learning to predict the Weekly_Sales then cross check with the csv training dataset Weekly Sales to obtain the accuracy of the prediction. Weekly_Sales refers to the Ordered Quantity from Departments. The output for the Prediction of Weekly_Sales will be in Column E. We then press the blue button "Predict". **Figure 21** as shown.



Store	Dept	Date	IsHoliday	Weekly_Sales
1	1	5/2/2010	FALSE	60
1	1	12/2/2010	TRUE	285
1	1	19/2/2010	FALSE	285
1	1	26/2/2010	FALSE	80
1	1	5/3/2010	FALSE	115
1	1	12/3/2010	FALSE	85
1	1	19/3/2010	FALSE	175
1	1	26/3/2010	FALSE	325
1	1	2/4/2010	FALSE	170
1	1	9/4/2010	FALSE	70
1	1	16/4/2010	FALSE	245
1	1	23/4/2010	FALSE	385
1	1	30/4/2010	FALSE	375
1	1	7/5/2010	FALSE	40
1	1	14/5/2010	FALSE	135
1	1	21/5/2010	FALSE	260
1	1	28/5/2010	FALSE	250
1	1	4/6/2010	FALSE	100
1	1	11/6/2010	FALSE	15
1	1	18/6/2010	FALSE	175
1	1	25/6/2010	FALSE	45
1	1	2/7/2010	FALSE	195
1	1	9/7/2010	FALSE	280
1	1	16/7/2010	FALSE	355
1	1	23/7/2010	FALSE	145
1	1	30/7/2010	FALSE	330
1	1	6/8/2010	FALSE	250
1	1	13/8/2010	FALSE	325
1	1	20/8/2010	FALSE	270
1	1	27/8/2010	FALSE	310

Figure 21: Columns to Predict Azure Machine Learning in Excel

Batch Execution Service(BES) and Testing Accuracy of Model in Excel -Part 3

Azure Machine Learning will generate the Scored Label Mean. Scored Label Mean is the Predicted Weekly_Sales by the Machine Learning. This is shown in **Figure 22**.

Next, we sum up the entire Scored Label Mean from Row 2 to 600 in cell F2. We also sum up the entire Weekly_Sales from Row 2 to 600 in F3. We get the Prediction Accuracy of the model by dividing the sum of the entire Weekly_Sales with the sum of the entire Scored Label Mean, in cell F4 (=F2/F3) and the accuracy results is 96% meaning the prediction accuracy is quite good. This is shown in **Figure 23**.

Store	Dept	Date	IsHoliday	Scored Label Mean		Weekly_Sales
1	1	5/2/2010	FALSE	199.8458714		60
1	1	12/2/2010	TRUE	199.8458714		285
1	1	19/2/2010	FALSE	199.8458714		285

Figure 22: Scored Label Mean

F4 =F2/F3

	A	B	C	D	E	F	G	H	I
1	Store	Dept	Date	IsHoliday	Scored Label Mean	Prediction Accuracy	Weekly_Sales		
2	1	1	5/2/2010	FALSE	199.8458714	120175.1218	60		
3	1	1	12/2/2010	TRUE	199.8458714	124585	285		
4	1	1	19/2/2010	FALSE	199.8458714	0.964603458	285		

Figure 23: Azure Machine Learning Model Accuracy

Improvements

AutoML- Automated Machine Learning

Automated Machine Learning can automatically select and train Machine Learning models.

Microsoft's AutoML can build a set of Machine Learning models automatically, intelligently select models for training and recommend the best ones based on the Machine Learning problem and data type. Normally, models are manually produced and compared against dozens of models. With AutoML, it can select the right algorithm and help tune hyperparameters(Sue, 2016). AutoML currently supports classification, forecasting and regression problems. This is useful for the forecasting used in Logic University. With AutoML, it saves time, effort while producing excellent results.

AutoML Integration with Azure Machine Learning

AutoML can be used with Azure Machine Learning to optimize model scoring. An automated ML training experiment is designed and run by first identifying the ML problem, specifying the source, and formatting of the labelled training dataset. The compute target for model training and automated ML parameters are then configured (Sue, 2016) .

References:

Arvan et al., 2018 Arvan, M., Fahimnia, B., Reisi, M., Siemsen, E. (2018). Integrating Human Judgement into Quantitative Forecasting Methods: A Review. *Omega*. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0305048317311155 (Accessed: 25 August 2020).

Azure Machine Learning documentation (2020). Available at: https://docs.microsoft.com/en-us/azure/machine-learning/ (Accessed: 25 August 2020).

Budek, K. (2018) *Powering up demand forecasting with machine learning - deepsense.ai*, *deepsense.ai*. Available at: https://deepsense.ai/powering-up-demand-forecasting-with-machine-learning/ (Accessed: 25 August 2020).

ML Studio (classic): Deploy a web service - Azure (2017). Available at: https://docs.microsoft.com/en-us/azure/machine-learning/studio/deploy-a-machine-learning-web-service (Accessed: 25 August 2020).

Puchalsky et al., 2018

W. Puchalsky, Gl T. Ribeiro, C.P. Da Veiga, R.Z. Freire, Santos Coelho, L Dos**Agribusiness time series forecasting using Wavelet neural networks and metaheuristic optimization: An analysis of the soybean sack price and perishable products demand**

International Journal of Production Economics, 203 (2018), pp. 174-189. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0925527318302548 (Accessed: 25 August 2020).

Sue, A. (2019) 6 reasons Microsoft has become the go-to for machine learning. Available at: https://medium.com/datadriveninvestor/6-reasons-microsoft-has-become-the-go-to-for-machine-learning-e642864ef5f5 (Accessed: 25 August 2020).

Tarallo, E. et al. (2019) "Machine Learning in Predicting Demand for Fast-Moving Consumer Goods: An Exploratory Research", *IFAC-PapersOnLine*, 52(13), pp. 737-742. doi: 10.1016/j.ifacol.2019.11.203. Available at: https://www.sciencedirect.com/science/article/pii/S240589631931153X (Accessed: 25 August 2020).

Taranenko, L. (2020) *How to Apply Machine Learning in Demand Forecasting for Retail? - MobiDev*, *MobiDev*. Available at: https://mobidev.biz/blog/machine-learning-methods-demand-forecasting-retail#:~:text=Machine%20learning%20techniques%20allow%20predicting,Accelerates%20data%20processing%20speed (Accessed: 25 August 2020).