

# Azure ML Tutorial

## Contents

Introduction .....	1
Create your workspace .....	2
Create your first Machine Learning Experiment.....	6
Importing data .....	7
Simple Pre-Processing.....	12
Actual Machine Learning .....	16
Results.....	20
Publishing your experiment.....	21
Creating a Windows 10 application (UWP).....	24
Wrap up .....	32

## Introduction

This tutorial aims to introduce you to AzureML and show you how you can easily create your own experiment and publish it into a service. We will do this through an example. We are going to use a dataset containing vote information from US elections at 1984 and make a simple Windows 10 Application “predicting” whether you would be a democrat or a republican at that time.

This tutorial is meant for people that now about Machine Learning and would like to see how this platform works and what it has to offer but it is also meant for beginners that would like to get started. However, the purpose of the tutorial is not to teach data analysis but to provide an overview of the tools and the platform offered by Microsoft Azure.

Links:

Github: <https://github.com/georschi/Azure-Machine-Learning-and-UWP-application>

Machine Learning Experiment:

<http://gallery.cortanaintelligence.com/Experiment/AzureMLTutorialExperiment-1>

## Create your workspace


The first thing you need to do is to create an account at the Azure Machine Learning Workspace.

You can visit <https://studio.azureml.net> and sign up for free with a Microsoft Account, without a need of having an Azure Subscription.

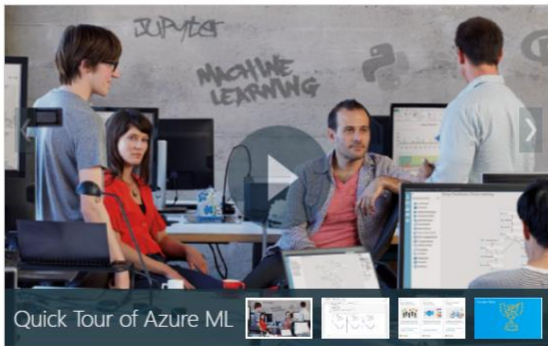
[studio.azureml.net/#](#)

chine Learning Studio

Introducing:  
Competitions



Learn More ➔



Quick Tour of Azure ML

Welcome to Azure Machine Learning

Try it for free

No Azure subscription? No credit card? No problem! Choose anonymous Guest Access, or sign in with your work or school account, or a Microsoft account.

Sign In ➔

Not an Azure ML user?  
[Sign up here](#)

[Pricing & FAQ](#)

By using this free version, you agree to be bound by the Microsoft Azure Website Terms of Use.

Webinars **NEW!**

**Decoding Brain Signals**  
Aired on August 02, 2016  
Patients who have injuries or tumors on the neuron connectivity have difficulties in connecting the visual stimulus and cognition. We will discuss general introduction of the ML

**Inside the Data Science VM**  
Aired on June 21, 2016  
DSVM is a custom Azure Virtual Machine image that is published on the Azure marketplace and available on both Windows and Linux. It contains several popular data science and

**Predictive Maintenance Modeling**  
Aired on July 05, 2016  
Predictive maintenance is one of the top demanded applications of predictive modelling and is seen as a life-saver in asset-heavy industries such as manufacturing and

Welcome to Azure

**Quick Evaluation**  
Guest Workspace  
8-hour trial  
No sign-in required.  
[Enter](#)  

- No hassle instant access
- Stock sample datasets
- ML models built in minutes
- Full range of ML algorithms

**Most Popular**  
Free Workspace  
\$0/month  
Don't already have a Microsoft account? Simply [sign up here](#).  
[Sign In](#)  

- Free access that never expires
- 10 GB storage on us
- R and Python scripts support
- Predictive web services

**Enterprise Grade**  
Standard Workspace  
\$9.99/month  
Azure subscription required  
Other charges may apply. [Read more](#).  
[Create Workspace](#)  

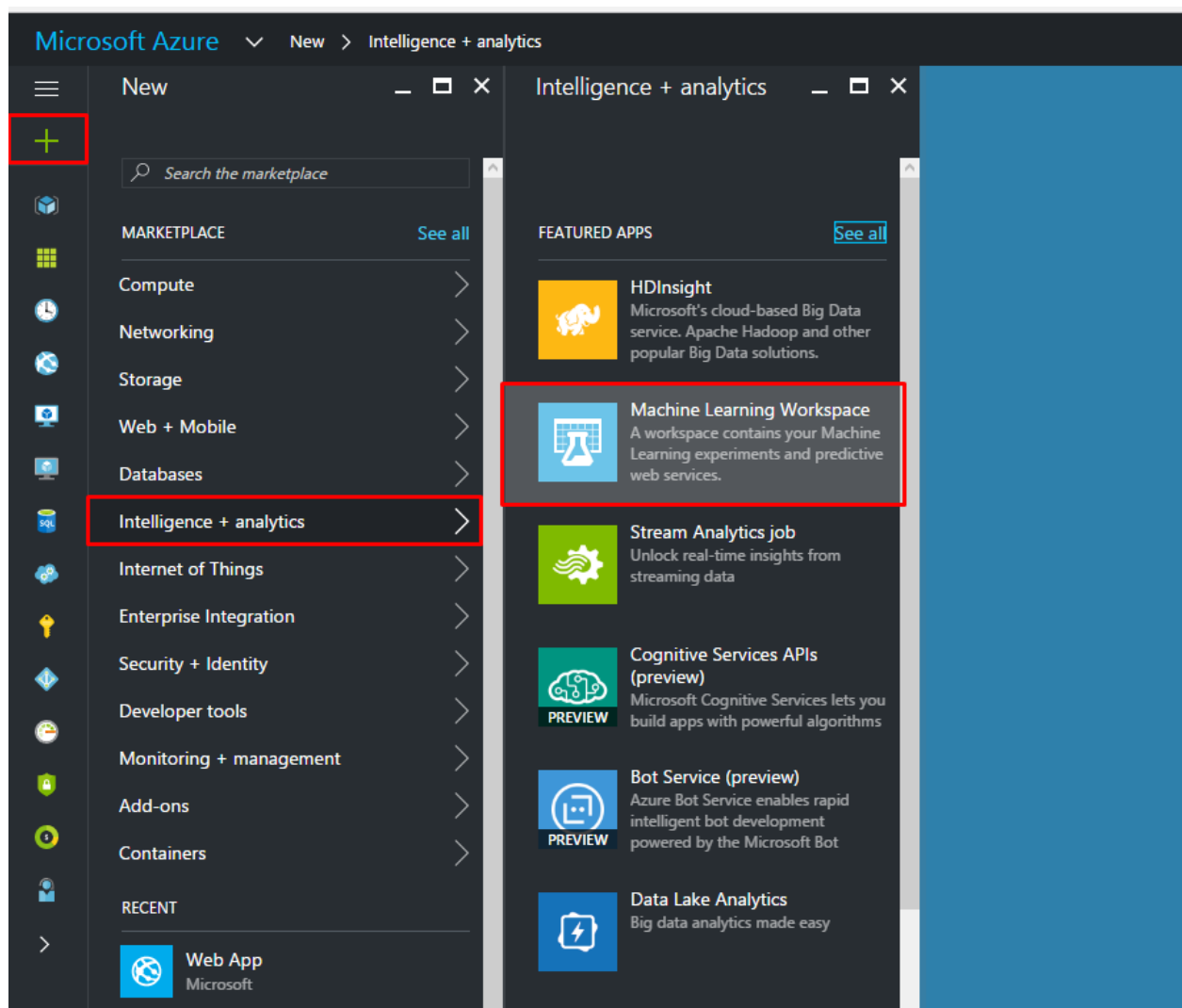
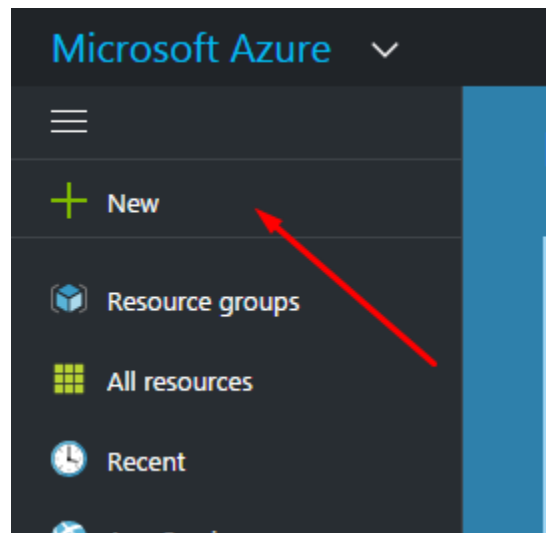
- Full SLA Support
- Bring your own Azure storage
- Parallel graph execution
- Elastic Web Service endpoints

We will discuss general introduction of the ML algorithm in decoding brain signal, and how to build the end-to-end analytics pipeline using

contains several popular data science and development tools both from Microsoft and from the open source community all pre-

industries such as manufacturing and aerospace due to its potential to provide significant cost savings by reducing downtime

In case you do have an active Azure Subscription you can create your workspace through the portal as is shown below



Machine Learning Work...  
Machine Learning Workspace

\* Workspace name

AzureMLTutorial

\* Subscription

Visual Studio Enterprise with MSDN

\* Resource group

Create new

Use existing

AzureMLTutorialRG

\* Location

West Europe

\* Storage account

Create new

Use existing

azuremltutorialstorage

Workspace pricing tier

Standard

\* Web service plan

Create new

Use existing

AzureMLTutorialPlan

\* Web service plan pricing tier

No pricing tier selected

Pin to dashboard

Create

Automation options

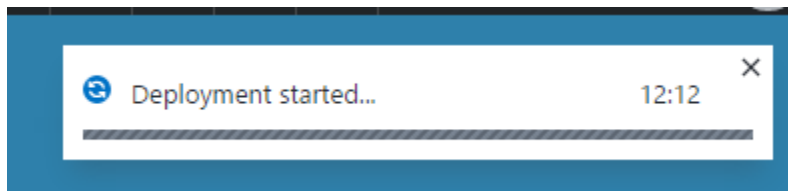
Choose your pricing tier

Browse the available plans

Choose the pricing tier for Machine Learning web services. [Learn more](#)

<div>DEVTEST Standard</div> <div>2 Compute Hours</div> <div>1,000 Transactions</div> <div>Manual Scaling</div> <div>0.00</div> <div>EUR/DAY (ESTIMATED)</div>	<div>S1 Standard</div> <div>25 Compute Hours</div> <div>100,000 Transactions</div> <div>Manual Scaling</div> <div>2.72</div> <div>EUR/DAY (ESTIMATED)</div>	<div>S2 Standard</div> <div>500 Compute Hours</div> <div>2,000,000 Transactions</div> <div>Manual Scaling</div> <div>27.20</div> <div>EUR/DAY (ESTIMATED)</div>
<div>S3 Standard</div> <div>12,500 Compute Hours</div> <div>50,000,000 Transactions</div> <div>Manual Scaling</div> <div>272.03</div> <div>EUR/DAY (ESTIMATED)</div>		

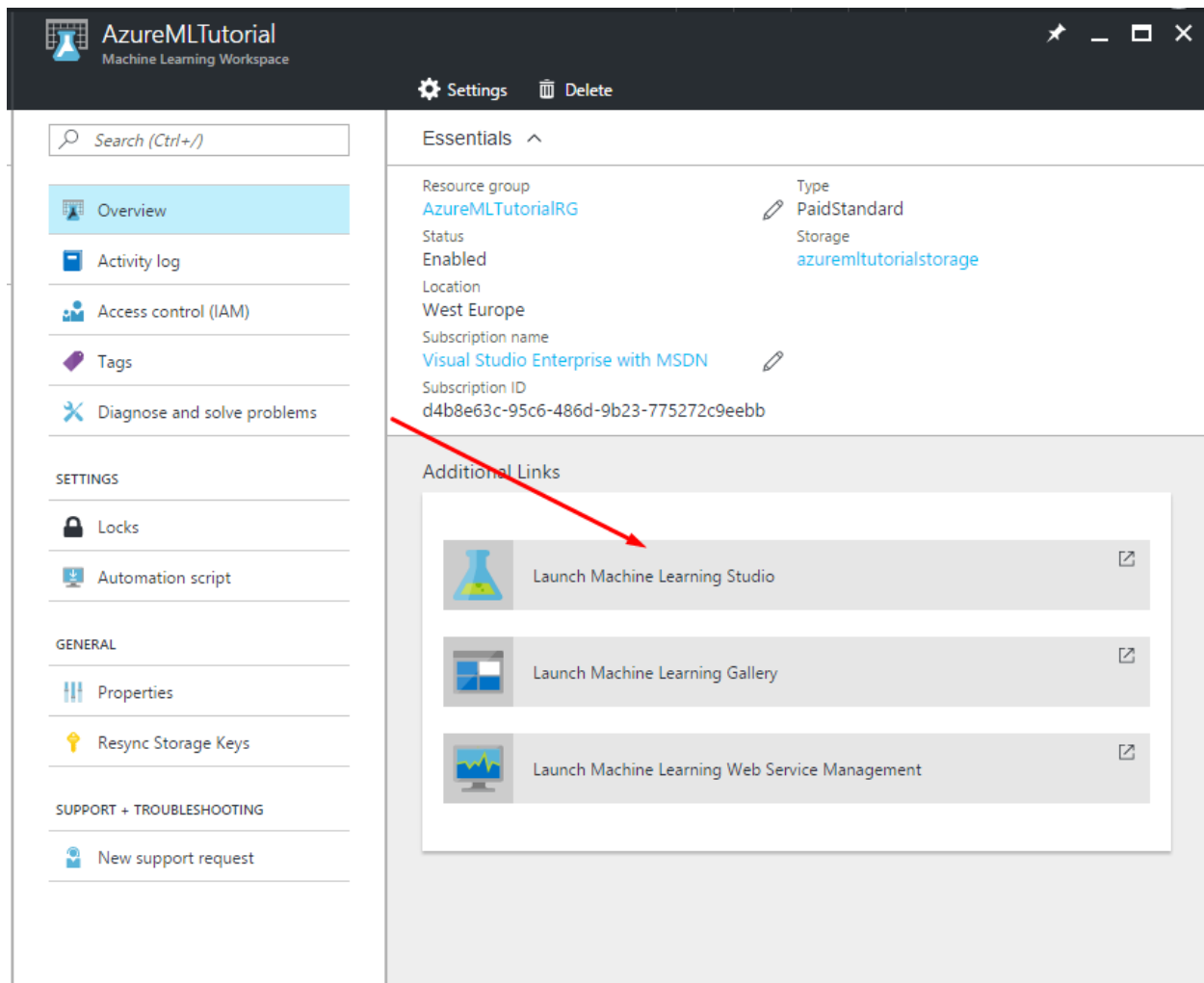
Select



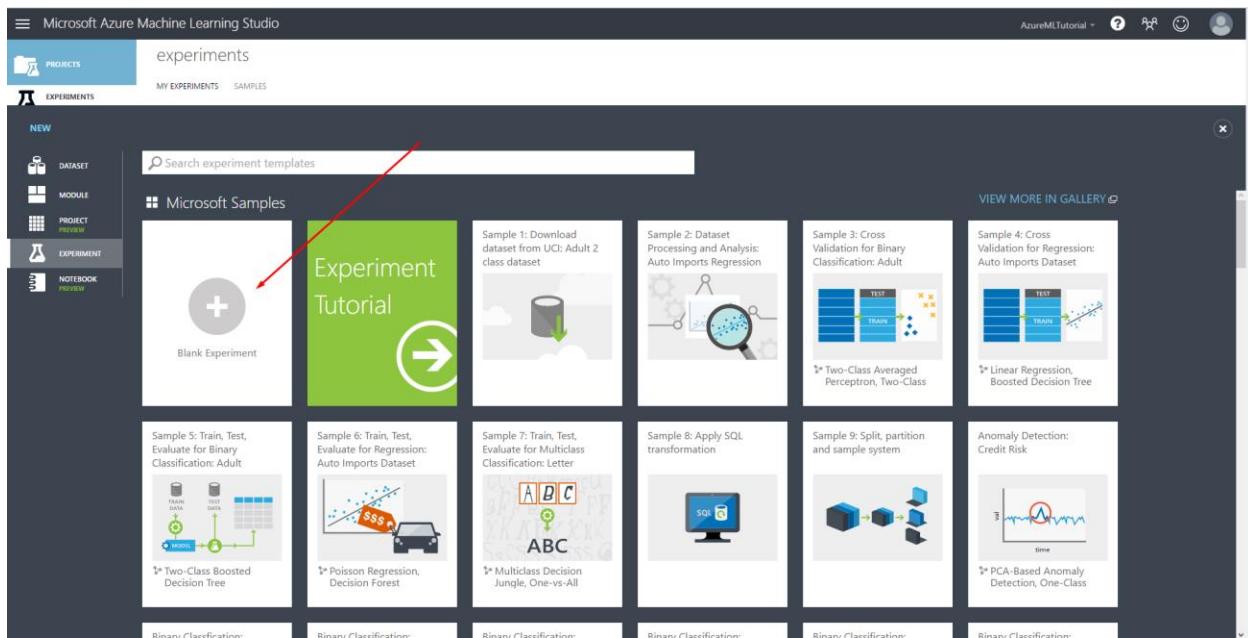
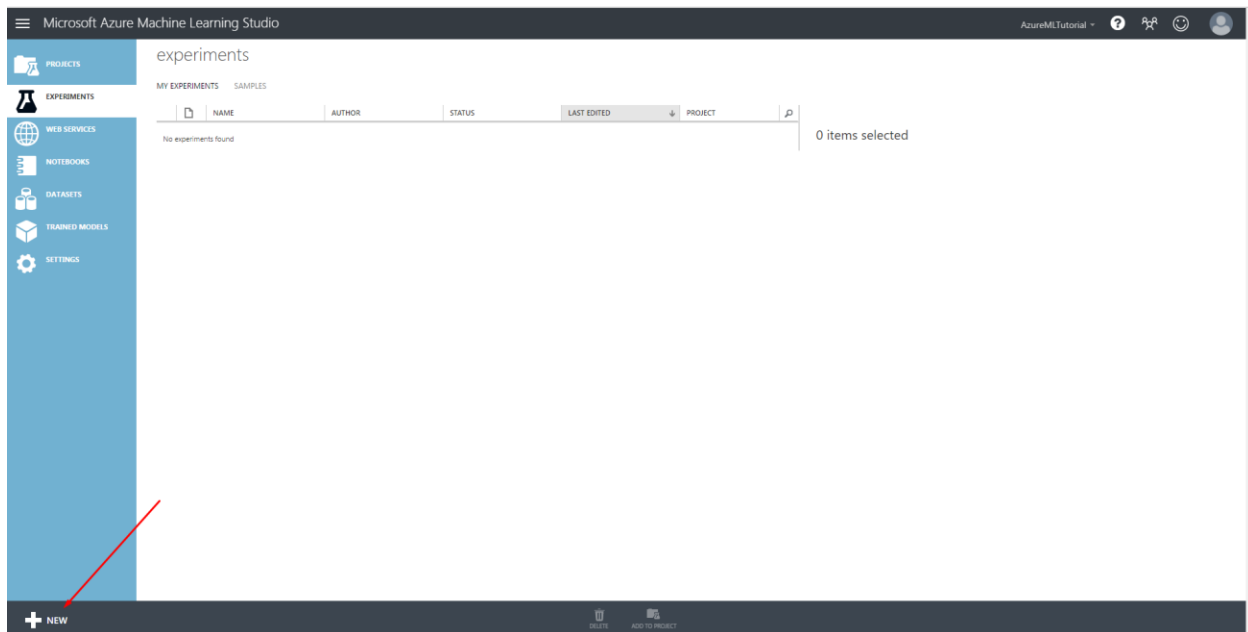
A blue notification bar with a circular arrow icon on the left, the text "Deployment started..." in the center, and a close button (X) on the right. Below the text is a progress bar. The time "12:12" is displayed on the right side of the bar.

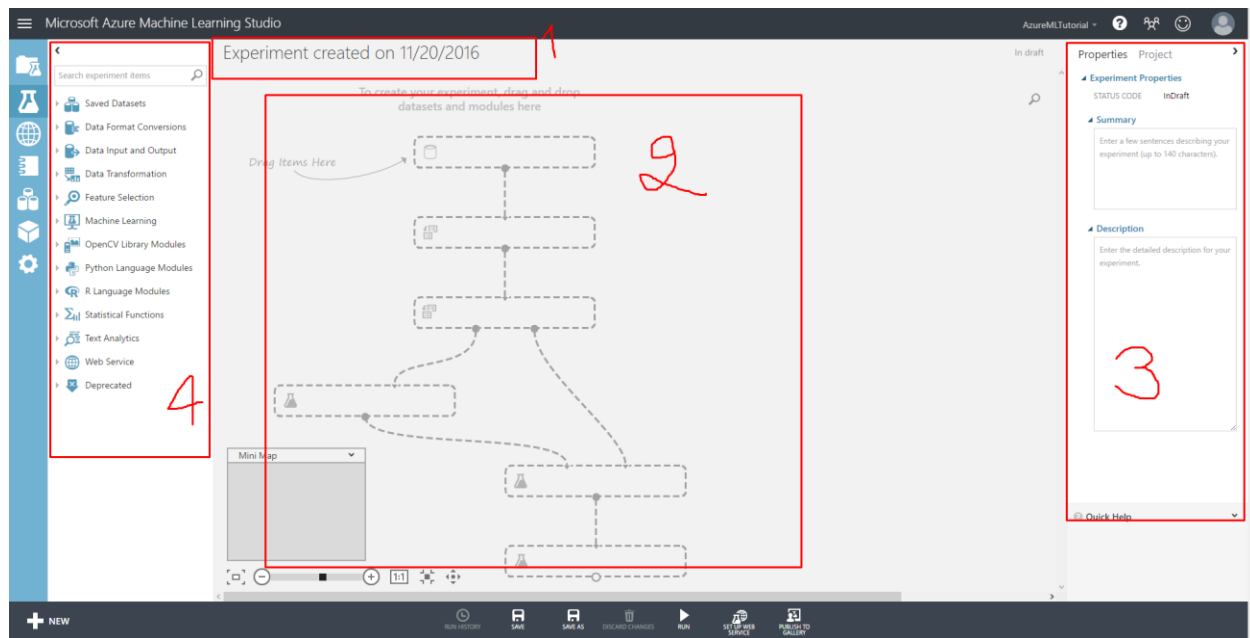
Georgios Schinas - November 2016

After a few second the deployment will have successfully finished and you can now head to you ML Workspace to start our project.



# Create your first Machine Learning Experiment

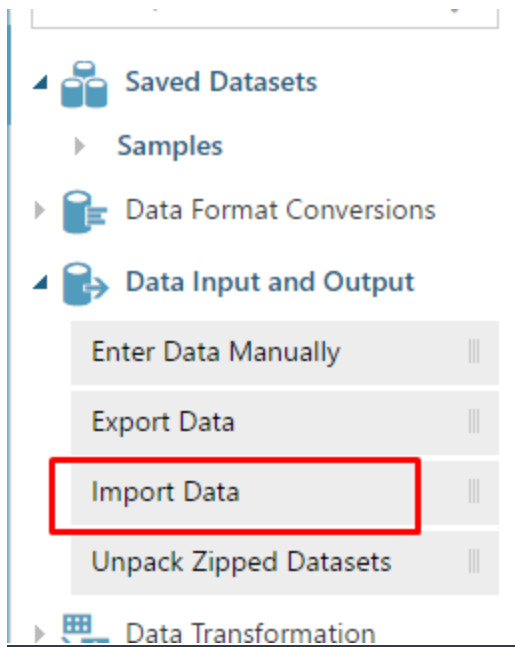




As soon as you create your new blank experiment you will come across a screen like the one above. At the top (1) you can see the name of your experiment. By clicking it we will change it into AzureMLTutorialExperiment. At the middle (2) you may see our canvas. The place where all your creations will come into life. At the far right (3) is the properties windows. From there we will be changing some settings to the modules we will include. Finally, at the left (4) there is list of all the things (datasets, models, tools etc.) that we can include in our experiments.

## Importing data

A great benefit of AzureML is the ease at which we can bring data into our experiment. You can either have your own datasets saved online or import them from anywhere in the internet (you can connect it to your SQL database if you would like to). We will import our data from a webpage.

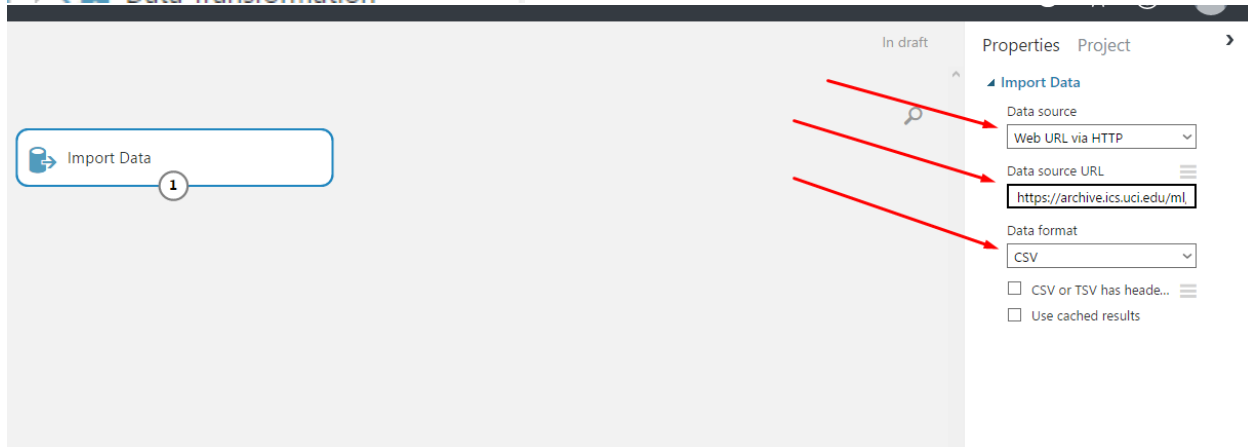


To use this module, we drag and drop it into the middle of our screen (or even double click it).

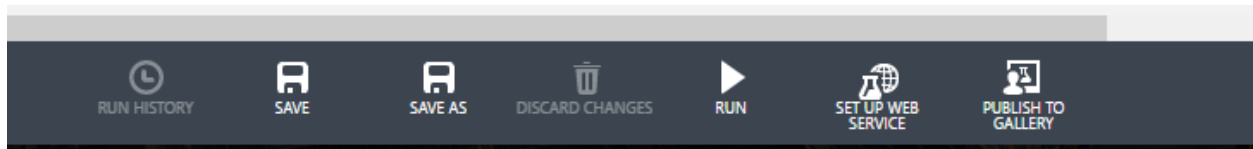
Now click it and notice the properties window. Change it so it seems like mine.

The URL we use is <https://archive.ics.uci.edu/ml/machine-learning-databases/voting-records/house-votes-84.data>

For your reference you can visit <http://archive.ics.uci.edu/ml/> to find a lot of databases on which you can test your skills

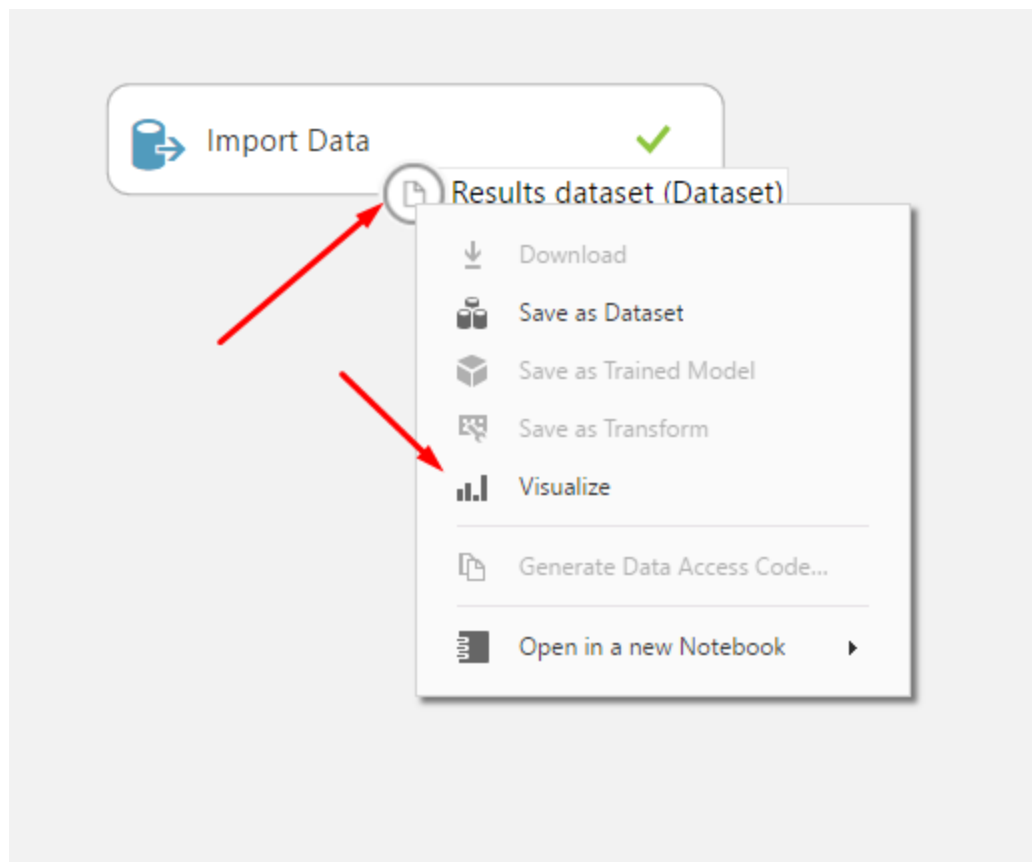


Now it's time for our first RUN! Notice the bar at the bottom of your screen and press RUN.



After a while the experiment will have run and now by clicking this little circle you can visualize the results





There are a few things to notice here.

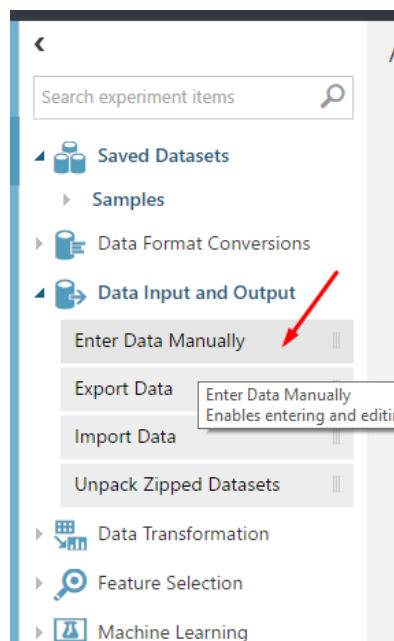
AzureMLTutorialExperiment > Import Data > Results dataset

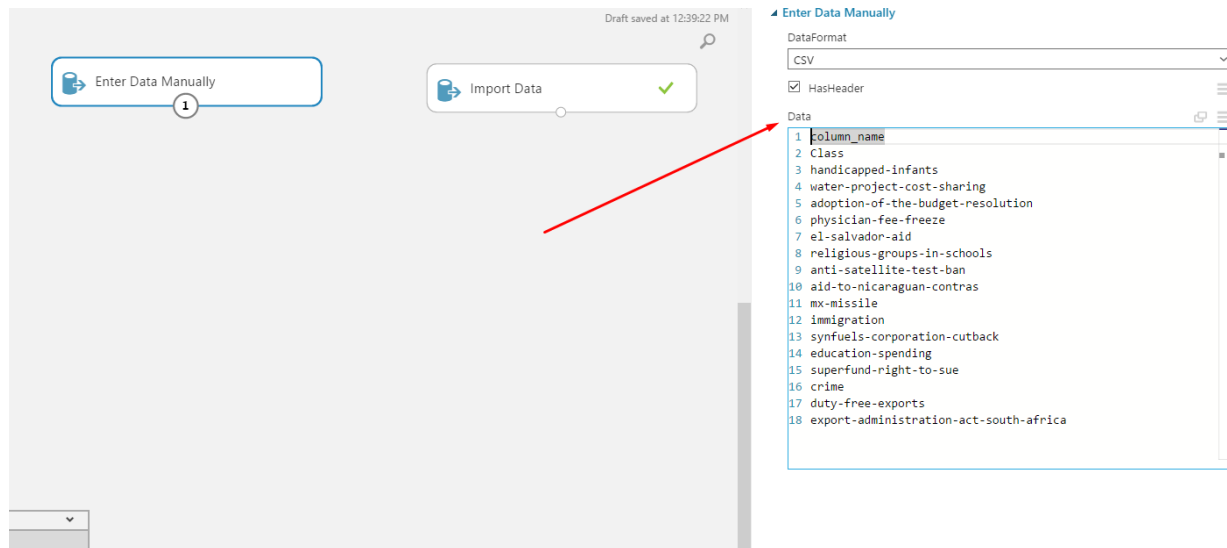
rows	columns
435	17

	Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15
view as															
	republican	n	y	n	y	y	y	n	n	n	y		y	y	y
	republican	n	y	n	y	y	y	n	n	n	n	n	y	y	y
	democrat		y	y		y	y	n	n	n	n	y	n	y	y
	democrat	n	y	y	n		y	n	n	n	n	y	n	y	n
	democrat	y	y	y	n	y	y	n	n	n	n	y		y	y
	democrat	n	y	y	n	y		n	n	n	n	n	n	y	y
	democrat	n	y	n	y	y	y	n	n	n	n	n	n		y
	republican	n	y	n	y	y	y	n	n	n	n	n	n	y	y
	republican	n	y	n	y	y	y	n	n	n	n	n	y	y	y
	democrat	y	y	y	n	n	n	y	y	y	n	n	n	n	n
	republican	n	y	n	y	y	n	n	n	n	n			y	y
	republican	n	y	n	y	y	y	n	n	n	n	y		y	y
	democrat	n	y	y	n	n	n	y	y	y	n	n	n	y	n
	democrat	y	y	y	n	n	y	y	y		y	y		n	n
	republican	n	y	n	y	y	y	n	n	n	n	n	y		
	republican	n	y	n	y	y	y	n	n	n	y	n	y	y	
	democrat	y	n	y	n	n	y	n	y		y	y	y		n
	democrat	y		y	n	n	n	y	y	y	n	n	n	y	n
	republican	n	y	n	y	y	y	n	n	n	n	n	y	y	

You can see that we have not names for our columns and there are some missing values. We will deal with this right now.

Let's enter manually the names of our columns.



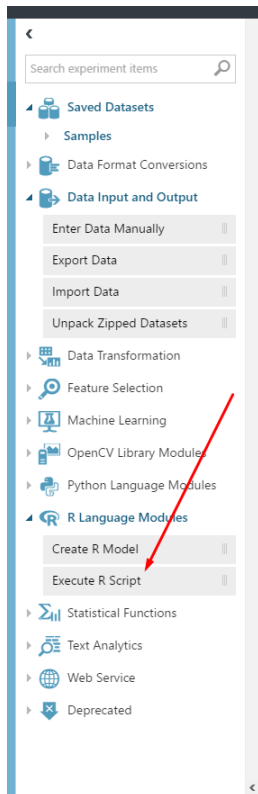


In there, we paste the names of the columns that we want to add.

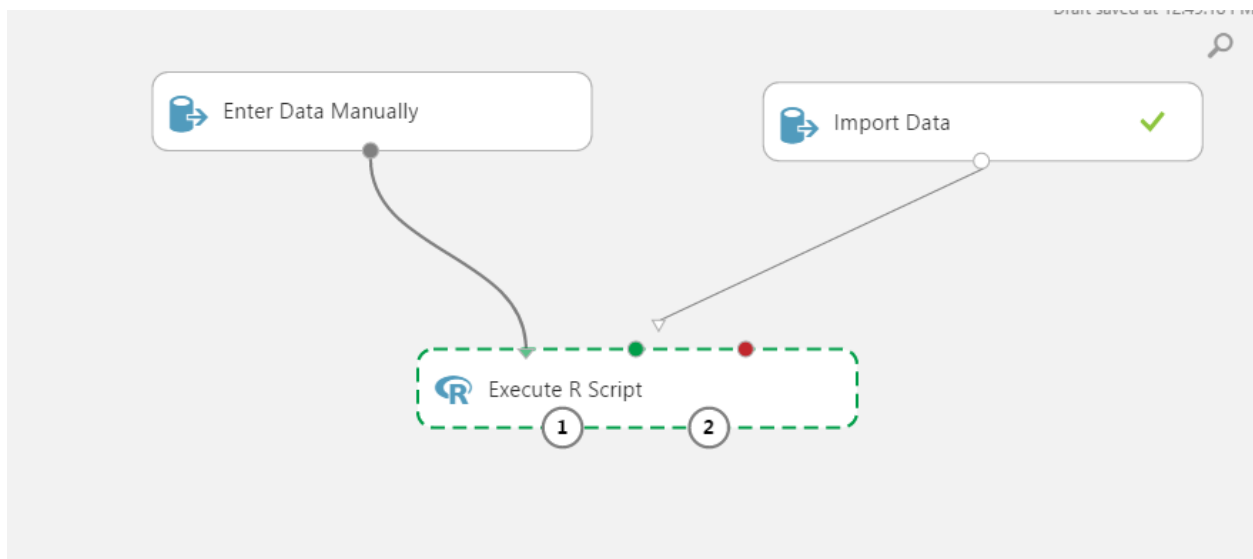
column\_name  
Class  
handicapped-infants  
water-project-cost-sharing  
adoption-of-the-budget-resolution  
physician-fee-freeze  
el-salvador-aid  
religious-groups-in-schools  
anti-satellite-test-ban  
aid-to-nicaraguan-contras  
mx-missile  
immigration  
synfuels-corporation-cutback  
education-spending  
superfund-right-to-sue  
crime  
duty-free-exports  
export-administration-act-south-africa

## Simple Pre-Processing

For now I have something special for you. You can execute your own R and Python scripts in your experiment! To do so, we import the module as shown in the picture



Drag and connect the modules as shown in the picture. This way you determine what the input to your script will be.



On the properties window you can write your own code.

Properties Project

#### ▲ Execute R Script

R Script

```
1 # Map 1-based optional input ports to variables
2 dataset1 <- mam1.mapInputPort(1) # class: data.frame
3 dataset2 <- mam1.mapInputPort(2) # class: data.frame
4
5 colnames(dataset2) <- c(dataset1['column_name'])$column_name
6 dataset2 = na.omit(dataset2)
7
8 # Select data.frame to be sent to the output Dataset port
9 mam1.mapOutputPort("dataset2");
```

Random Seed

The variables dataset1 and dataset2 contain the datasets we gave as input.

The lines 5&6 change the name of the columns and then we delete all those lines that contain a missing character. NOTE that you could do something else with the missing values, but it's not the purpose of this tutorial to deal with these issues.

```
colnames(dataset2) <- c(dataset1['column_name'])$column_name;
dataset2 = na.omit(dataset2)
```

By running again the experiment and visualizing the result you can see that our dataset is now ready for machine learning.



## Select a single column

BY NAME

WITH RULES

AVAILABLE COLUMNS

All Types search columns

Class

handicapped-infants

water-project-cost-sharing

adoption-of-the-budget-resolution

physician-fee-freeze

el-salvador-aid

religious-groups-in-schools

anti-satellite-test-ban

aid-to-nicaraguan-contras

mx-missile

immigration

synfuels-corporation-cutback

education-spending

superfund-right-to-sue

crime

17 columns available

SELECTED COLUMNS

All Types search columns

0 columns selected

port Data

Saving...

Filter Based Feature Selection

Feature scoring method

Pearson Correlation

☒ Operate on feature columns only

Target column

Selected columns:

Column names: Class

Launch column selector

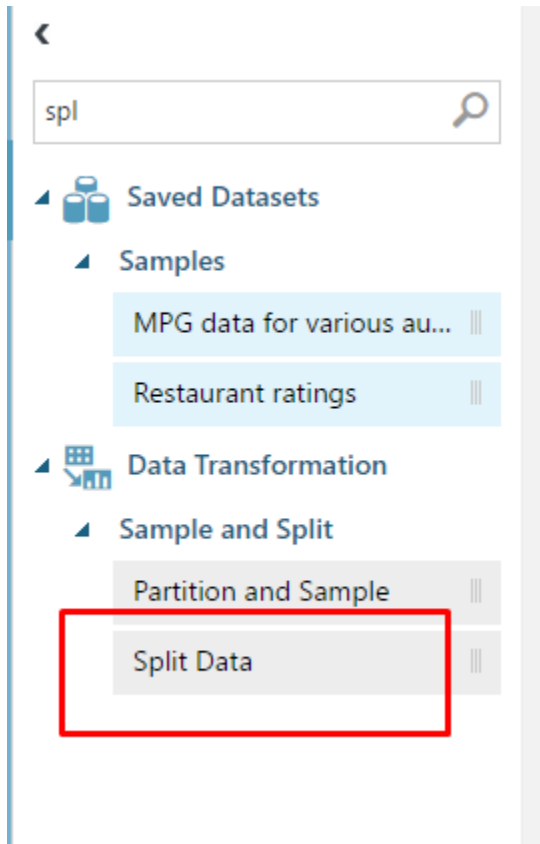
Number of desired features

6

## Actual Machine Learning

It's time for real ML now. We need to split our data into training and test sets, train our model and compare it with our test set.

Let's do it!





In draft  
Draft saved at 1:30:29 PM

Properties Project

**Split Data**

Splitting mode  
Split Rows

Fraction of rows in the first output...  
0.75

☒ Randomized split

Random seed  
1234

☐ Stratified split  
False

Quick Help

The screenshot shows the Azure ML Studio interface. On the left, a workflow diagram consists of five modules: 'Enter Data Manually', 'Import Data', 'Execute R Script', 'Filter Based Feature Selection', and 'Split Data'. The 'Split Data' module is highlighted with a red circle and has a red oval around its input port. On the right, the 'Properties' pane for the 'Split Data' module is open, showing settings for 'Splitting mode' (Split Rows), 'Fraction of rows in the first output...' (0.75), 'Randomized split' (checked), 'Random seed' (1234), and 'Stratified split' (False). The 'Split Data' module is also labeled with '1' and '2' at its output ports.

Microsoft Azure Machine Learning Studio

AzureMLTutorialExperiment

In draft  
Draft saved at 1:30:41 PM

Properties Project

**Two-Class Bayes Point Machine**

Number of training iterations  
30

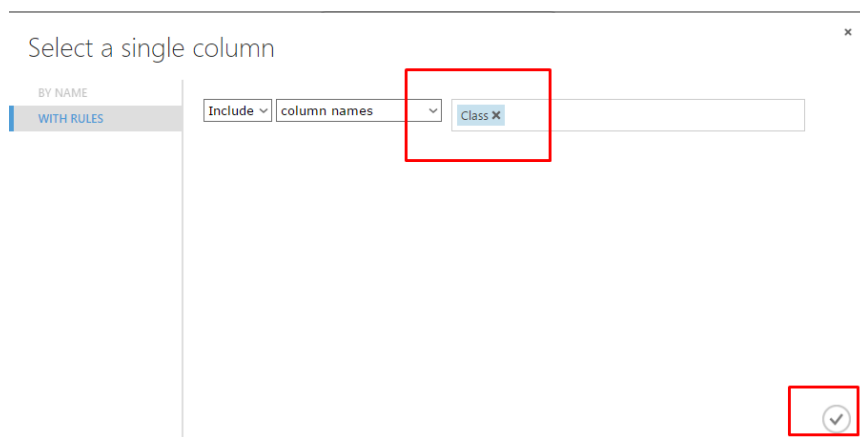
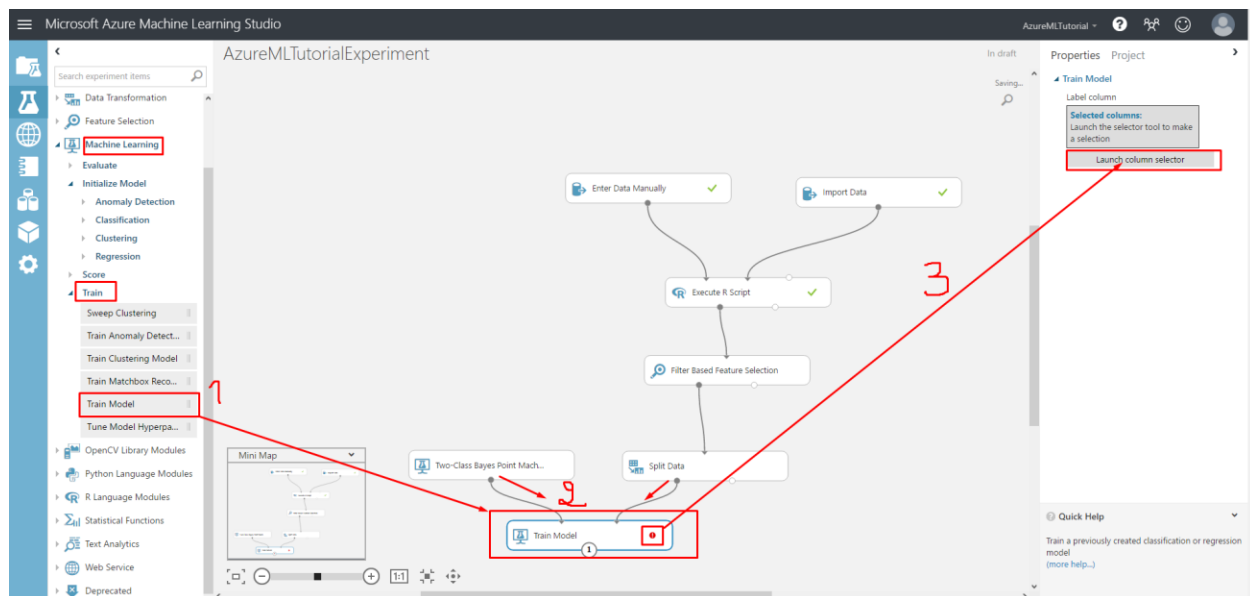
☒ Include bias

☒ Allow unknown values in cate...

Quick Help

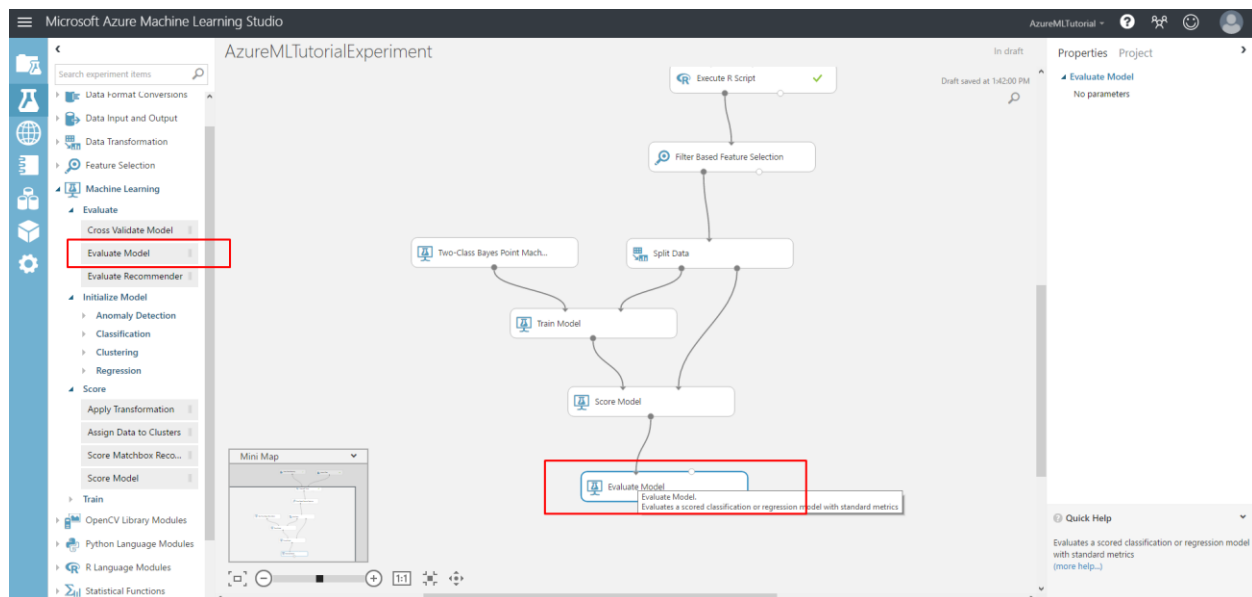
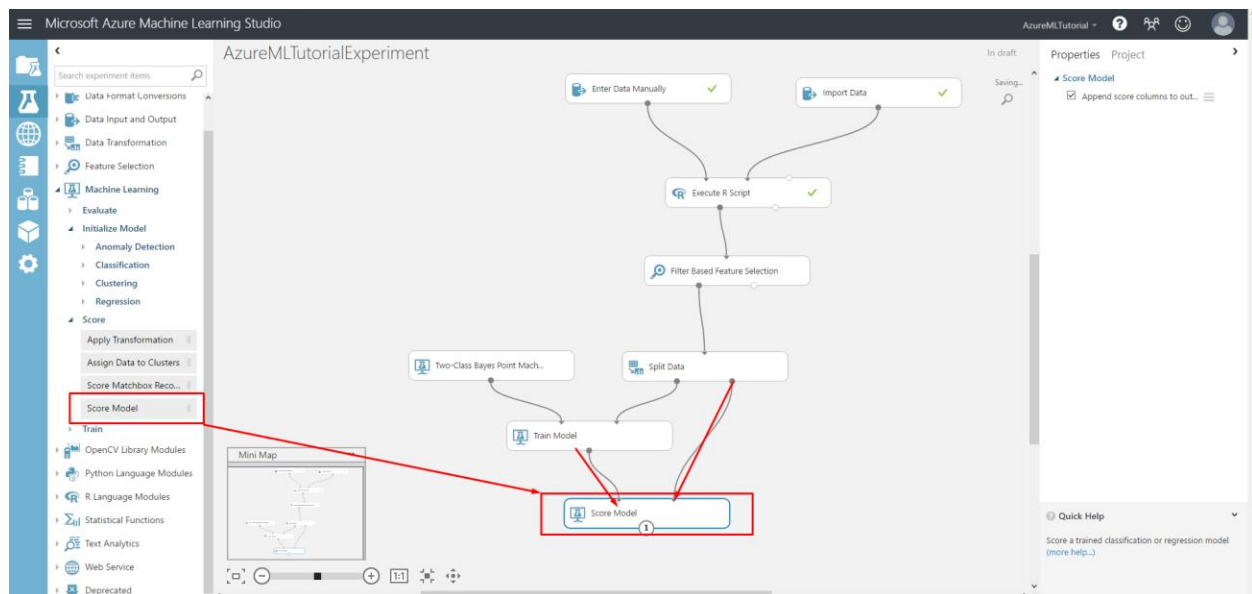
Create a Bayes Point Machine binary classification model  
(more help...)

The screenshot shows the Azure ML Studio interface. On the left, a workflow diagram consists of five modules: 'Enter Data Manually', 'Import Data', 'Execute R Script', 'Filter Based Feature Selection', and 'Split Data'. The 'Split Data' module is highlighted with a red circle and has a red oval around its input port. On the right, the 'Properties' pane for the 'Two-Class Bayes Point Machine' module is open, showing settings for 'Number of training iterations' (30), 'Include bias' (checked), and 'Allow unknown values in cate...' (checked). The 'Two-Class Bayes Point Machine' module is also labeled with '1' and '2' at its output ports.



We need to select a column to tell the model what feature is it trying to predict.

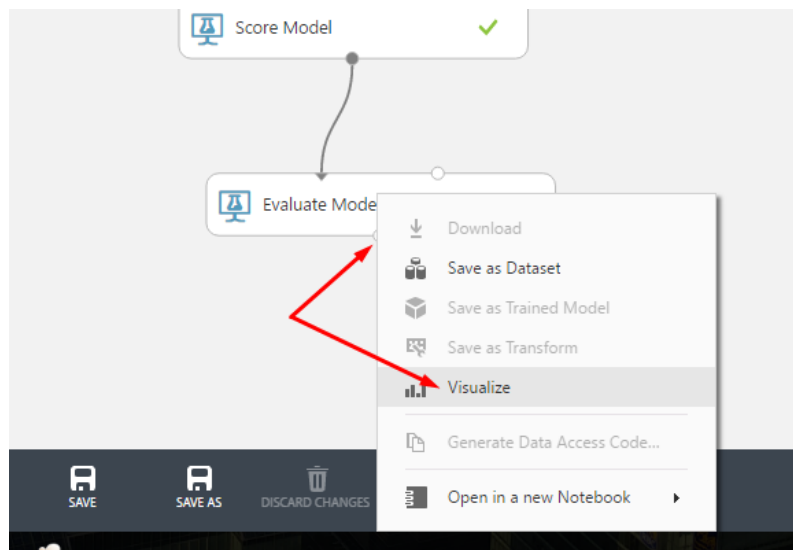
Up to this point we have trained our model. But we need to see how good it is. So we have 2 more modules to add.



Let's now run our experiment!

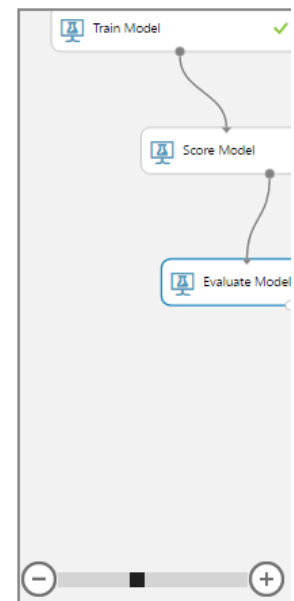
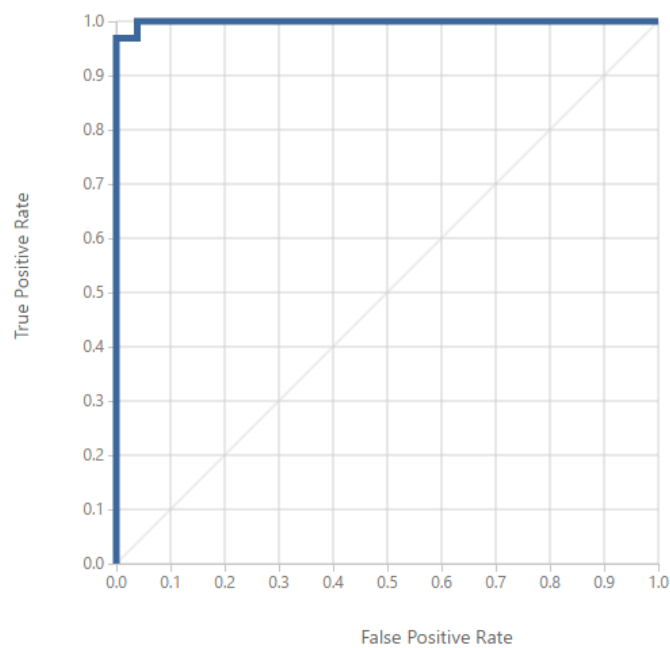
## Results

The experiment should have run in about one minute. Let's see the results.



AzureMLTutorialExperiment > Evaluate Model > Evaluation results

ROC PRECISION/RECALL LIFT



True Positive	False Negative	Accuracy	Precision	Threshold	AUC
31	1	0.983	1.000	0.5	0.999
False Positive	True Negative	Recall	F1 Score		
0	26	0.969	0.984		

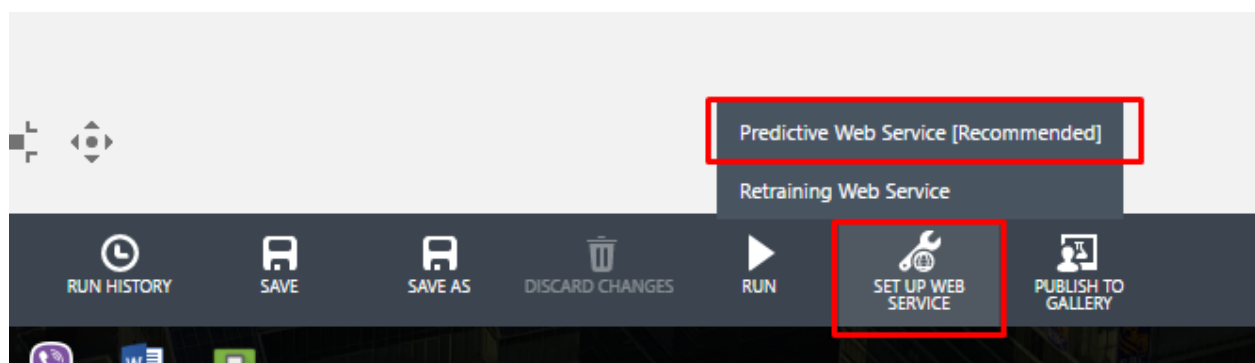
This beautifully organized page contains a lot of useful information for a data analyst. It provides the ROC curve, the confusion matrix and on top of that it provides the analyst with a threshold slider where you can change the threshold of your model and see in real time how the other metrics change!

For our example, we see some great results. One could argue that these results are not representative since we omitted all those rows with missing values that could substantially contribute to the performance of our model. This might be the case but again it is out of the scope of this tutorial.

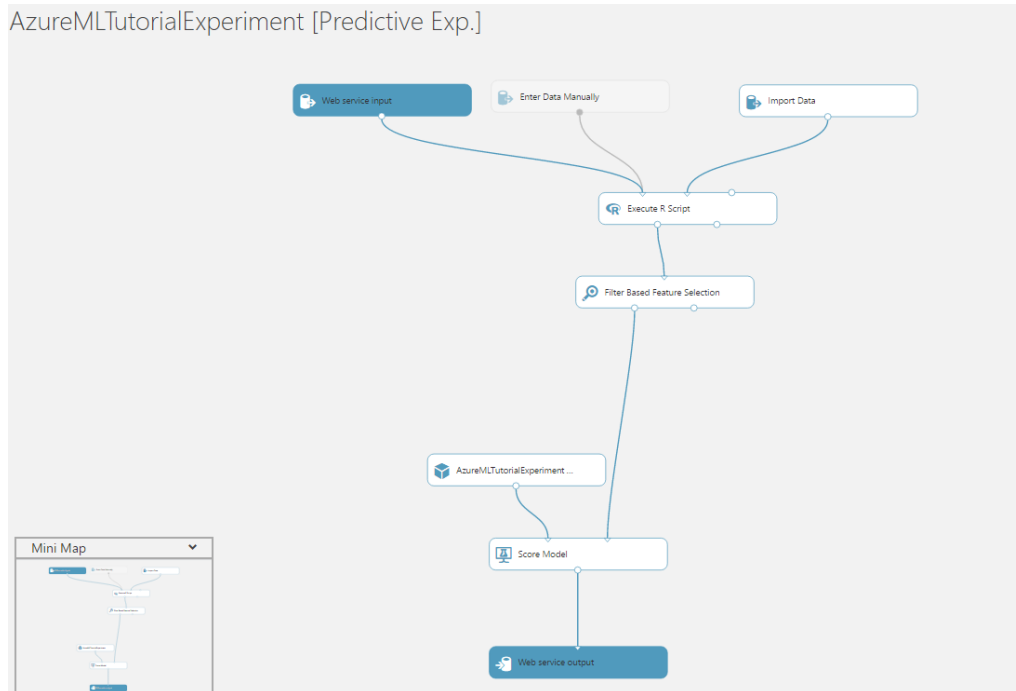
Our model has been trained and therefor now it's time to publish it and see how to integrate it within an app.

## Publishing your experiment

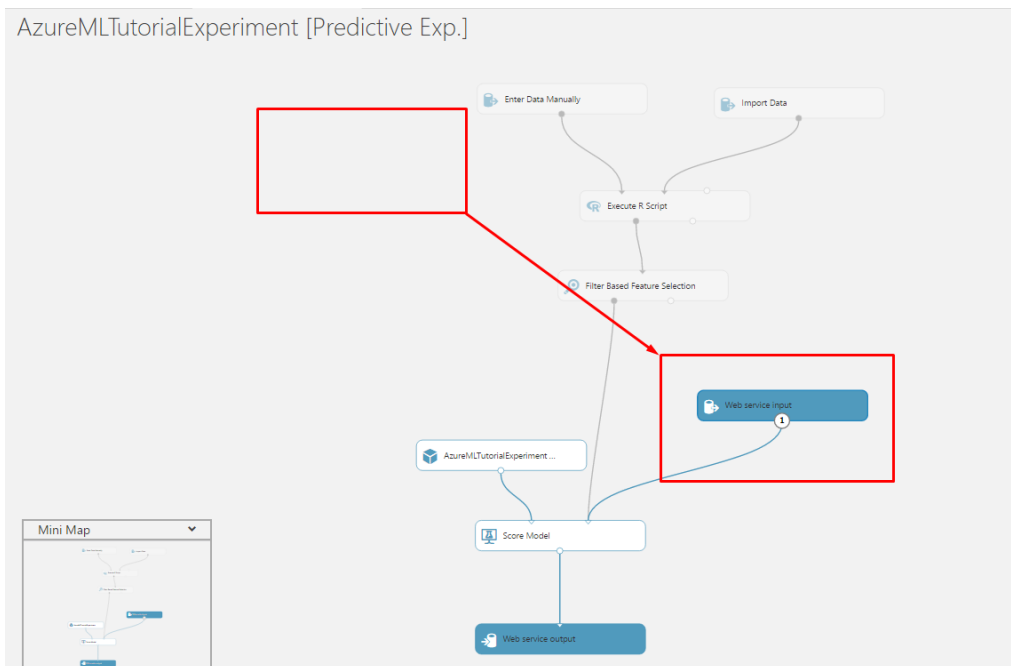
What we want to do is to setup a web service. Just hover over the SET UP WEB SERVICE and click the first option. In case you are not able to do that, you might need to re-run your experiment and then you will be good to go. We click and wait for a bit.



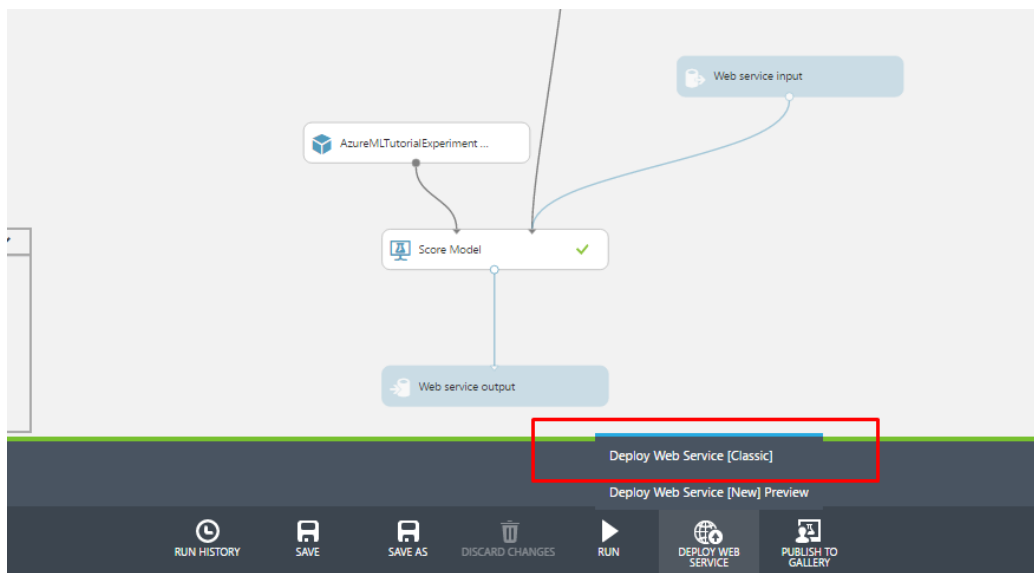
Your main screen should be something like this



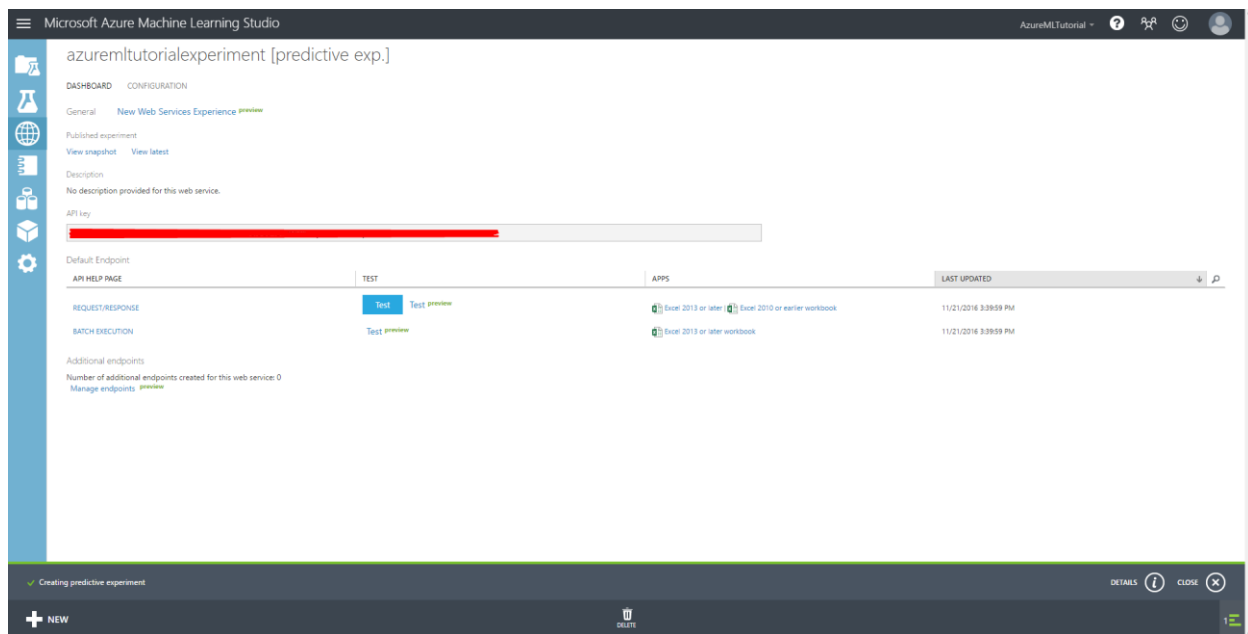
We need to change where the input of the web service is and place it above the Score Model. (in case you needed to do some preprocessing with your input you would need to place it in an appropriate place)



Since we made this change we need to press RUN again. Then just press Deploy Web Service [Classic]



In a moment, you will arrive in a page like the following



Note your API Key since we are going to need it. By pressing Test a window pops up and asks us to enter values for our input and by clicking the ok button we get a result.

Test AzureMLTutorialExperiment [Predictive Exp.] Service

### Enter data to predict

CLASS

HANDICAPPED-INFANTS

WATER-PROJECT-COST-SHARING

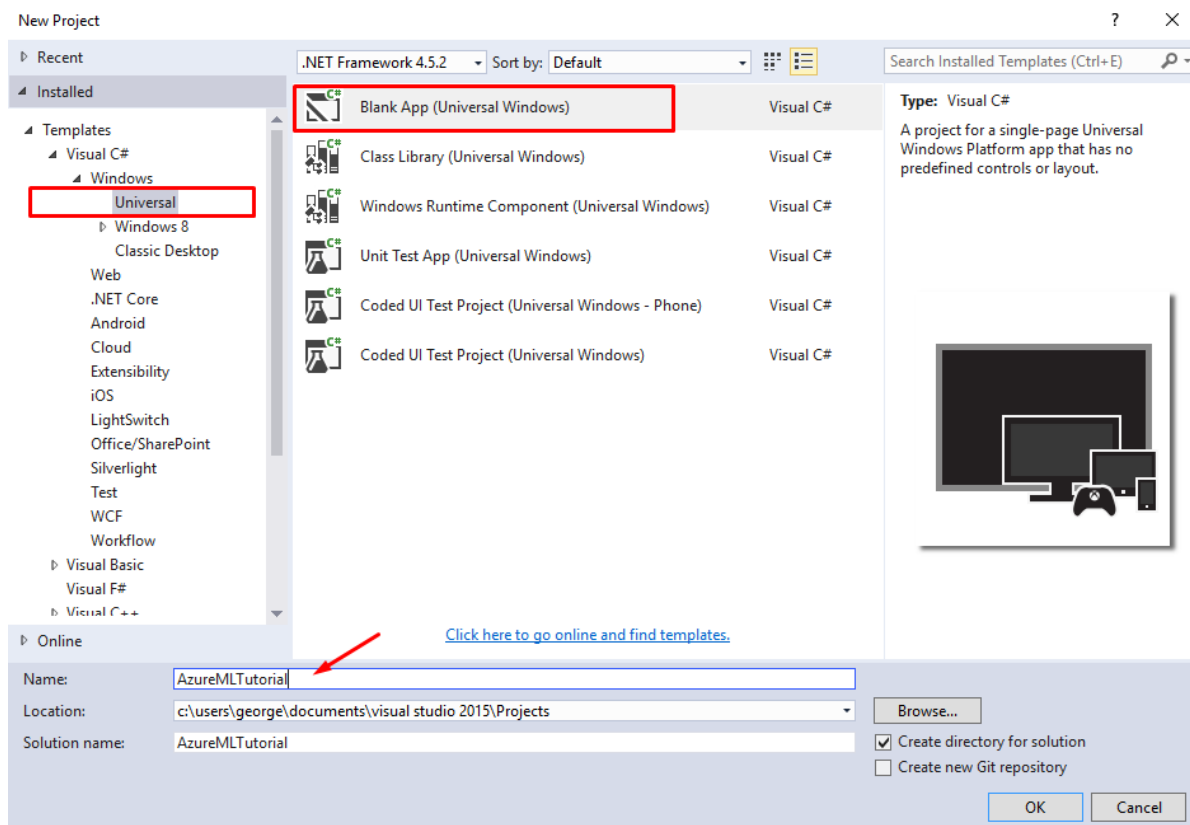
ADOPTION-OF-THE-BUDGET-RESOLUTION

PHYSICIAN-FEE-FREEZE

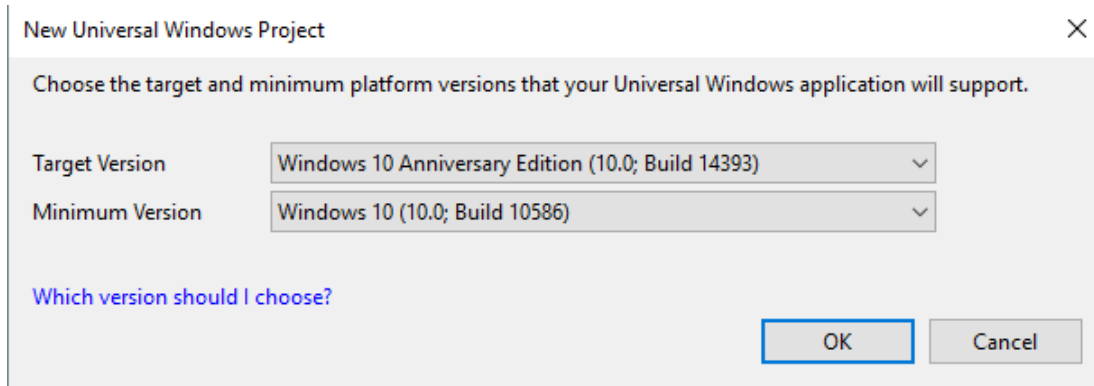
At this point if you know how to make http calls from your code you are good to go. You've learnt all you need it to get started with Azure Machine Learning.

## Creating a Windows 10 application (UWP)

We start by opening Visual Studio Community and start a New Project.

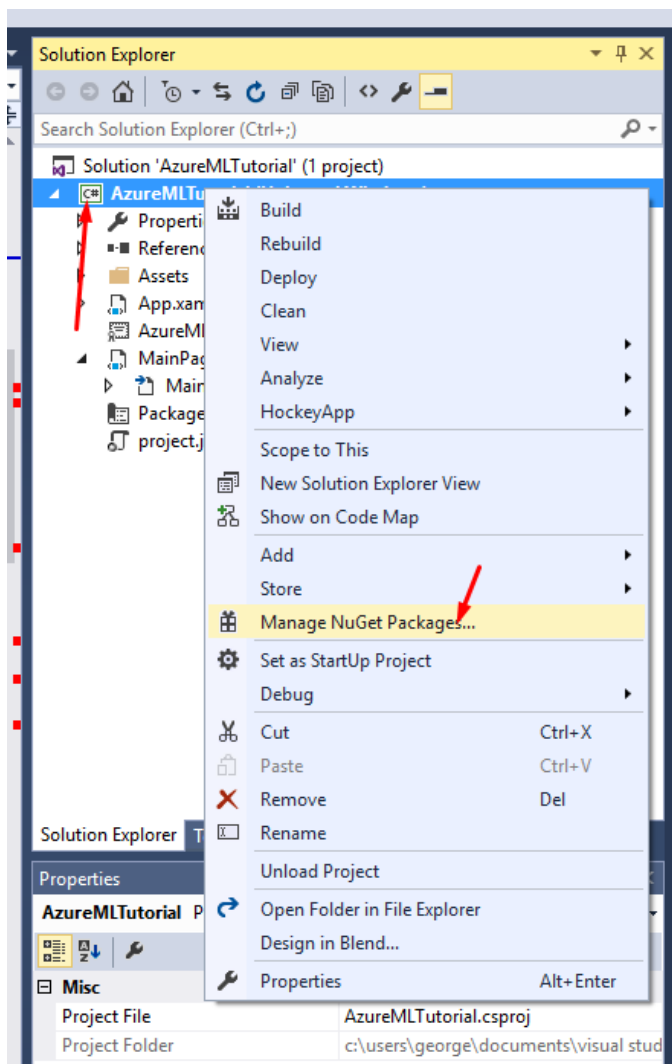


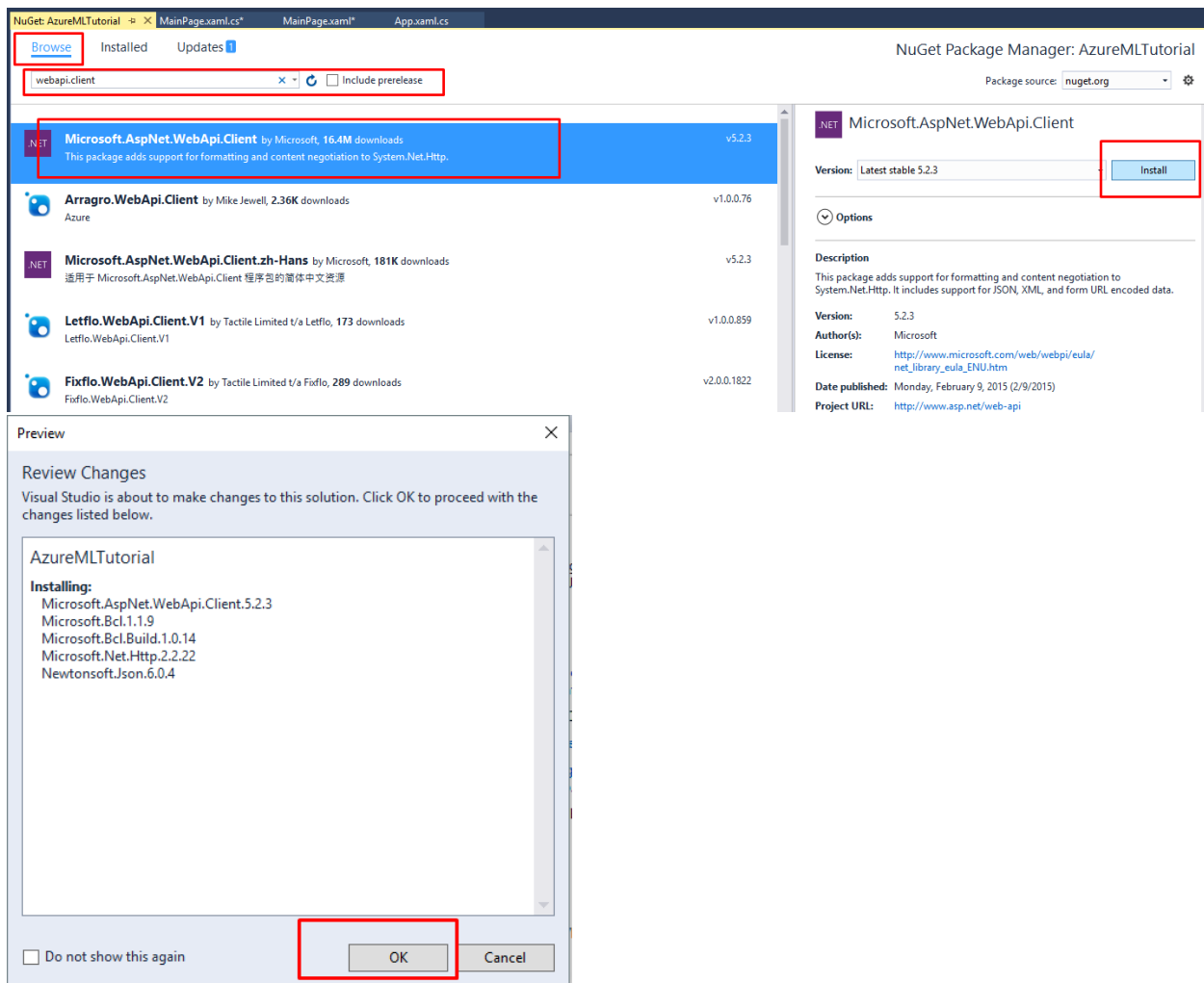




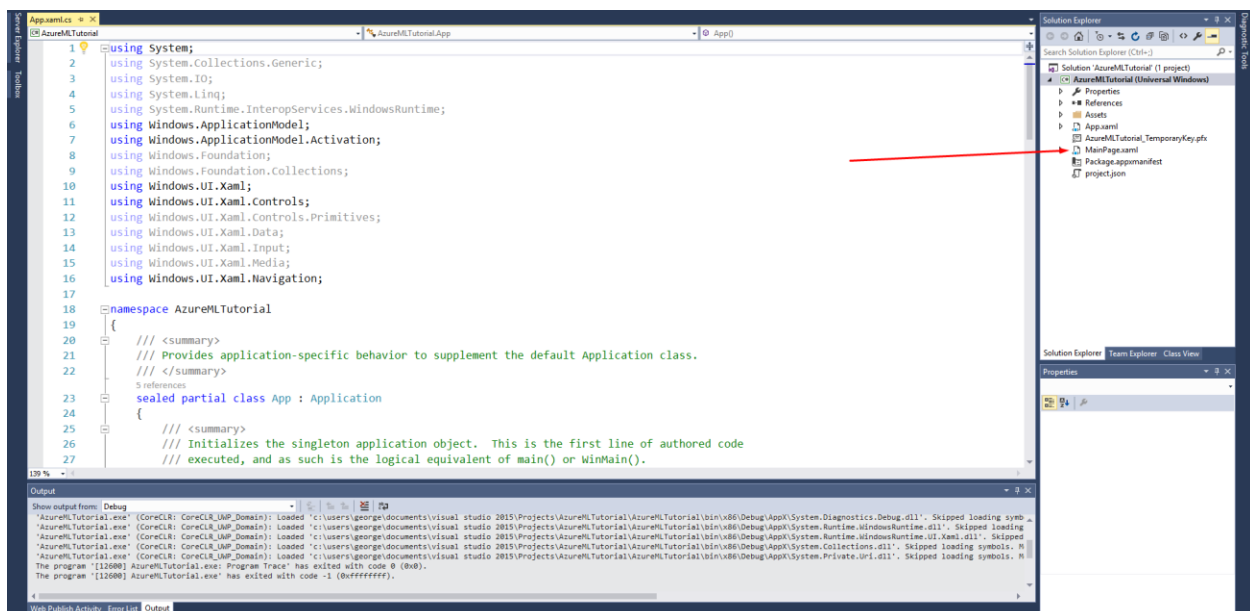
The first thing we should do is to add the nuget package that will help us with the web service calls.

Follow my lead!





Let's go now to the MainPage.xaml



What we want to do is to create a layout as simple as possible. So we add to our Grid a few grid rows and columns, by adding the following lines of code under the Grid. For consistency with my code, give a name to the grid myGrid. Please go on ahead and copy paste the code either from below or from github.

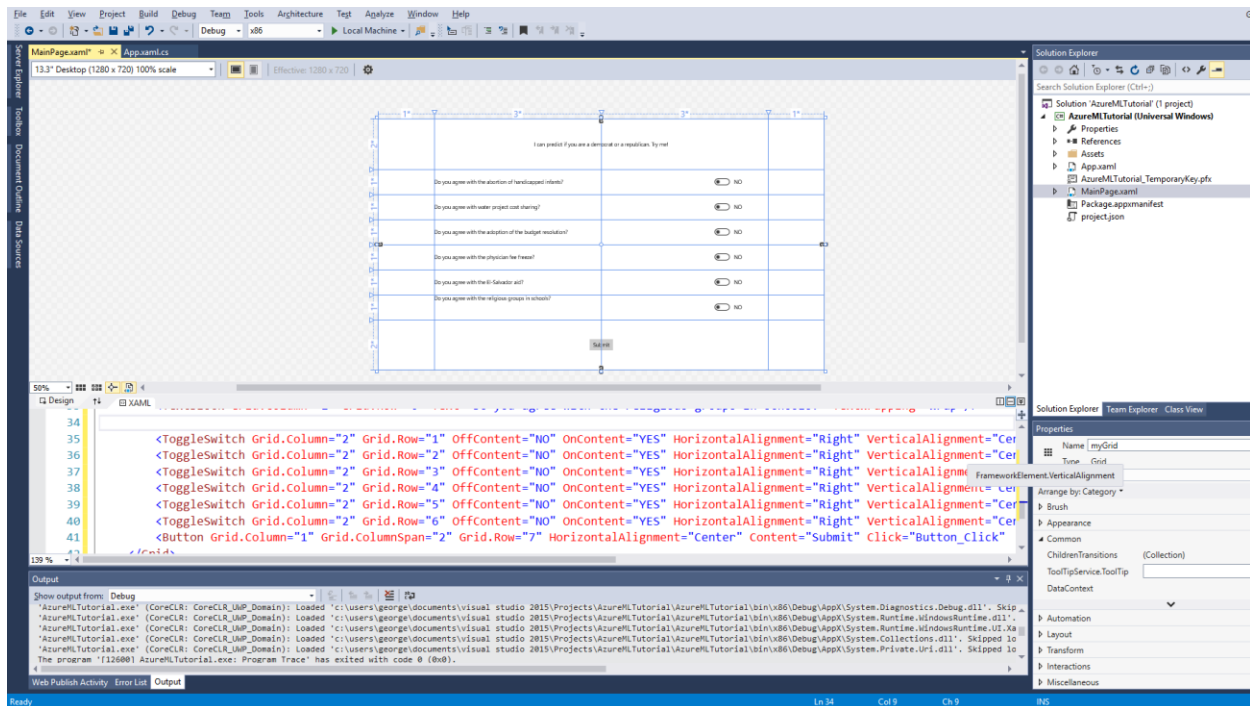
```
<Grid.RowDefinitions>
    <RowDefinition Height="2*" />
    <RowDefinition />
    <RowDefinition />
    <RowDefinition />
    <RowDefinition />
    <RowDefinition />
    <RowDefinition />
    <RowDefinition Height="2*" />
</Grid.RowDefinitions>
<Grid.ColumnDefinitions>
    <ColumnDefinition Width="*" />
    <ColumnDefinition Width="3*" />
    <ColumnDefinition Width="3*" />
    <ColumnDefinition Width="*" />
</Grid.ColumnDefinitions>
```

Below these definitions we will add the following code that in the one row puts the questions and at the other it give the user the freedom to choose between yes and no.

```
<TextBlock Grid.ColumnSpan="2" Grid.Column="1" Text="I can predict if you are a
democrat or a republican. Try me!" HorizontalAlignment="Center"
VerticalAlignment="Center" TextWrapping="Wrap" />
<TextBlock Grid.Column="1" Grid.Row="1" Text="Do you agree with the abortion of
handicapped infants?" TextWrapping="Wrap" VerticalAlignment="Center" />
<TextBlock Grid.Column="1" Grid.Row="2" Text="Do you agree with water project cost
sharing?" TextWrapping="Wrap" VerticalAlignment="Center" />
<TextBlock Grid.Column="1" Grid.Row="3" Text="Do you agree with the adoption of the
budget resolution?" TextWrapping="Wrap" VerticalAlignment="Center" />
<TextBlock Grid.Column="1" Grid.Row="4" Text="Do you agree with the physician fee
freeze?" TextWrapping="Wrap" VerticalAlignment="Center" />
<TextBlock Grid.Column="1" Grid.Row="5" Text="Do you agree with the El-Salvador aid?"
TextWrapping="Wrap" VerticalAlignment="Center" />
<TextBlock Grid.Column="1" Grid.Row="6" Text="Do you agree with the religious groups in
schools?" TextWrapping="Wrap" />

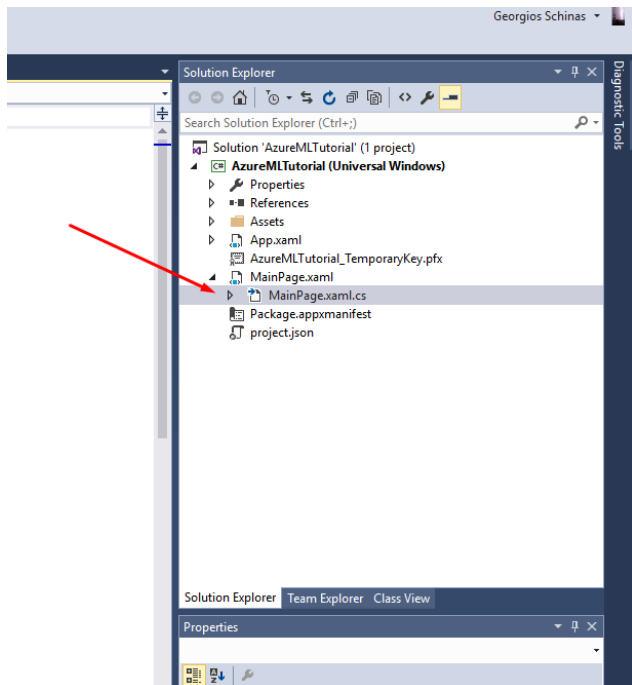
<ToggleSwitch Grid.Column="2" Grid.Row="1" OffContent="NO" OnContent="YES"
HorizontalAlignment="Right" VerticalAlignment="Center" />
<ToggleSwitch Grid.Column="2" Grid.Row="2" OffContent="NO" OnContent="YES"
HorizontalAlignment="Right" VerticalAlignment="Center" />
<ToggleSwitch Grid.Column="2" Grid.Row="3" OffContent="NO" OnContent="YES"
HorizontalAlignment="Right" VerticalAlignment="Center" />
<ToggleSwitch Grid.Column="2" Grid.Row="4" OffContent="NO" OnContent="YES"
HorizontalAlignment="Right" VerticalAlignment="Center" />
<ToggleSwitch Grid.Column="2" Grid.Row="5" OffContent="NO" OnContent="YES"
HorizontalAlignment="Right" VerticalAlignment="Center" />
<ToggleSwitch Grid.Column="2" Grid.Row="6" OffContent="NO" OnContent="YES"
HorizontalAlignment="Right" VerticalAlignment="Center" />
<Button Grid.Column="1" Grid.ColumnSpan="2" Grid.Row="7" HorizontalAlignment="Center"
Content="Submit" Click="Button_Click" />
```

Your screen should look like this



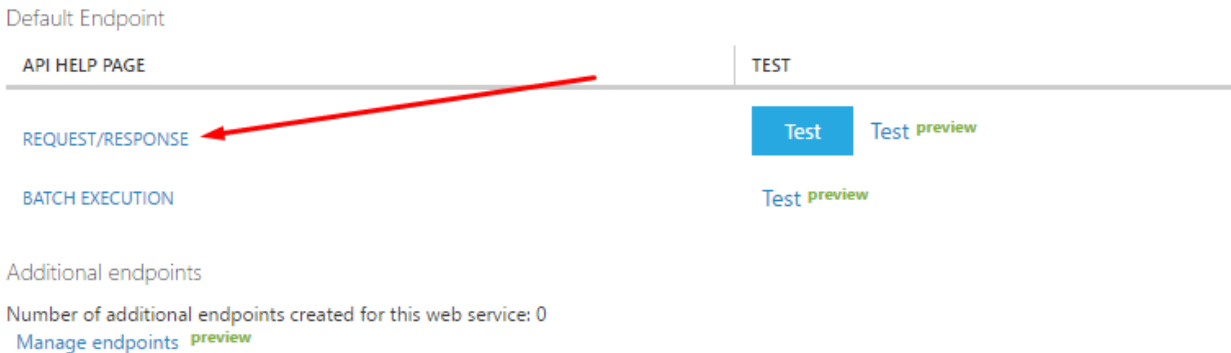
Now, open the MainPage.xaml.cs and right before the constructor of the MainPage declare a variable

`private string[] answers = new string[6];`



6 references  
`public sealed partial class MainPage : Page`  
`{`  
`private string[] answers = new string[6];`  
1 reference  
`public MainPage()`  
`{`  
`this.InitializeComponent();`  
`}`

Now return to your browser and click REQUEST/RESPONSE as shown below, which will take to a very helpful page.



That page explains exactly what happens with the http calls and gives you hints and help to implement them in any platform, so if interested, give it some time. For our project we will take advantage of some C# code that has been regenerated for us. Simply scroll down a bit and you'll find it.

Sample Code

C# Python R Select sample code

```
// 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
{
    public class StringTable
    {
        public string[] ColumnNames { get; set; }
        public string[,] Values { get; set; }
    }

    static async Task InvokeRequestResponseService()
```

From that code, copy the function `static async Task InvokeRequestResponseService()` which contains the basic structure to call the service. After you copy-paste it there will be a few error indications but don't worry! That's only because this C# code was generated for a console application and not a UWP. Make a few changes to your code so that it looks like the one below (or you directly that instead)

```
private async Task InvokeRequestResponseService()
{
    using (var client = new HttpClient())
    {
        var scoreRequest = new
        {
            Inputs = new Dictionary<string, StringTable>() {
```

```

        {
            "input1",
            new StringTable()
            {
                ColumnNames = new string[] { "Class", "handicapped-
infants", "water-project-cost-sharing", "adoption-of-the-budget-resolution",
"physician-fee-freeze", "el-salvador-aid", "religious-groups-in-schools"},
                Values = new string[,] { { "null", answers[0],
answers[1], answers[2], answers[3], answers[4], answers[5] }, }
            }
        },
        GlobalParameters = new Dictionary<string, string>()
        {
        }
    };
    const string apiKey = ""; // Replace this with your API key
    client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", apiKey);

    client.BaseAddress = new Uri("enter your own uri here as in the sample
code");

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

    if (response.IsSuccessStatusCode)
    {
        string json = await response.Content.ReadAsStringAsync();
        dynamic result = JsonConvert.DeserializeObject<dynamic>(json);
        var whatAreYou = result.Results.output1.value.Values[0][7];
        var thatMuch = result.Results.output1.value.Values[0][8];
        float percent = (float)thatMuch;
        percent = (percent < 0.5) ? (1 - percent) * 100 : percent * 100;
        percent = (int)percent;
        var dialog = new MessageDialog(String.Format("You are a {0} and I'm
{1}% sure about it!", whatAreYou, percent));
        await dialog.ShowAsync();
    }
    else
    {
        // custom code to deal with a problem at the call
    }
}
}

```

In that code what you need to notice to put your own API Key and Uri. The errors that remain will go away once you add this class in your code

```

public class StringTable
{
    public string[] ColumnNames { get; set; }
    public string[,] Values { get; set; }
}

```

```

90     }
91 }
92 public class StringTable
93 {
94     public string[] ColumnNames { get; set; }
95     public string[,] Values { get; set; }
96 }
97 }
98

```

and after you add these lines at the top of the page

```

using System.Threading.Tasks;
using System.Net.Http;
using System.Net.Http.Headers;
using Newtonsoft.Json;
using Windows.UI.Popups;

```

There is one final thing to add here and we are done! This is nothing other than calling that function and filling up the variable *answers[]* with the appropriate content, when the button submit is clicked.

To do so, add the following code and you're good to go (when writing the xaml part you may have noticed that at the button we already added the click event `Click="Button_Click"`).

```

1 reference
private async void Button_Click(object sender, RoutedEventArgs e)
{
    foreach (FrameworkElement item in myGrid.Children)
    {
        if (item is ToggleSwitch) answers[Grid.GetRow(item) - 1] = (((ToggleSwitch)item).IsOn == true) ? "y" : "n";
    }
    await InvokeRequestResponseService();
}
}

```

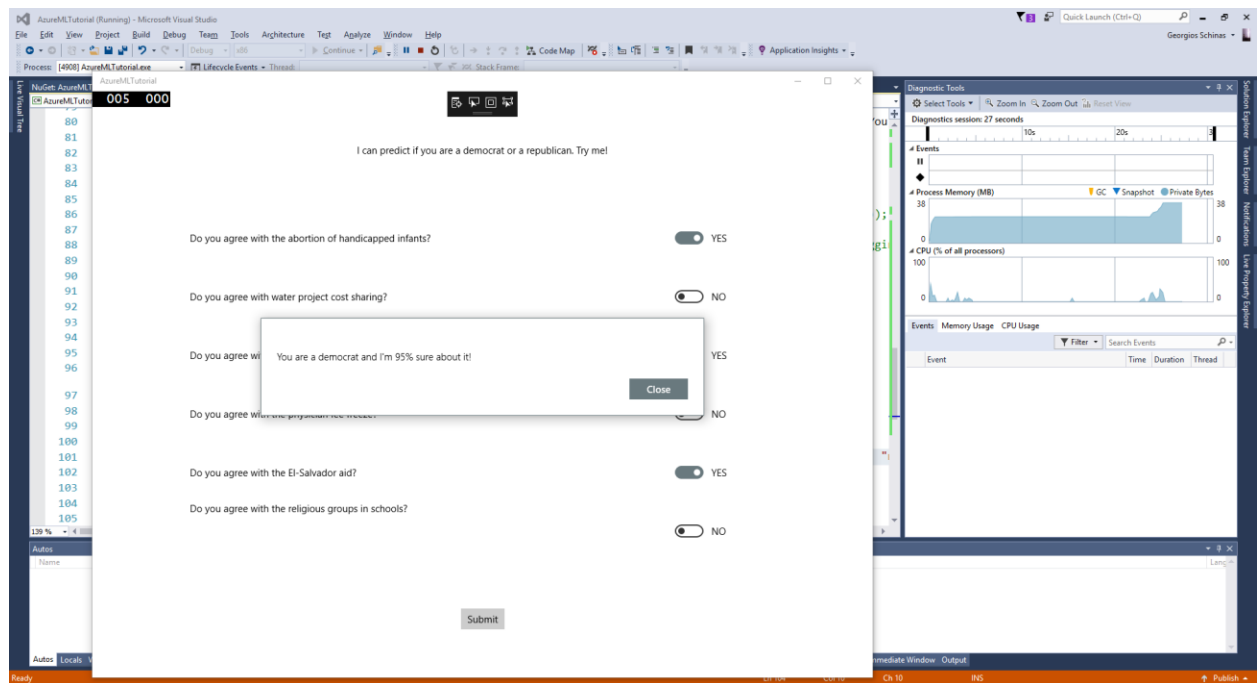
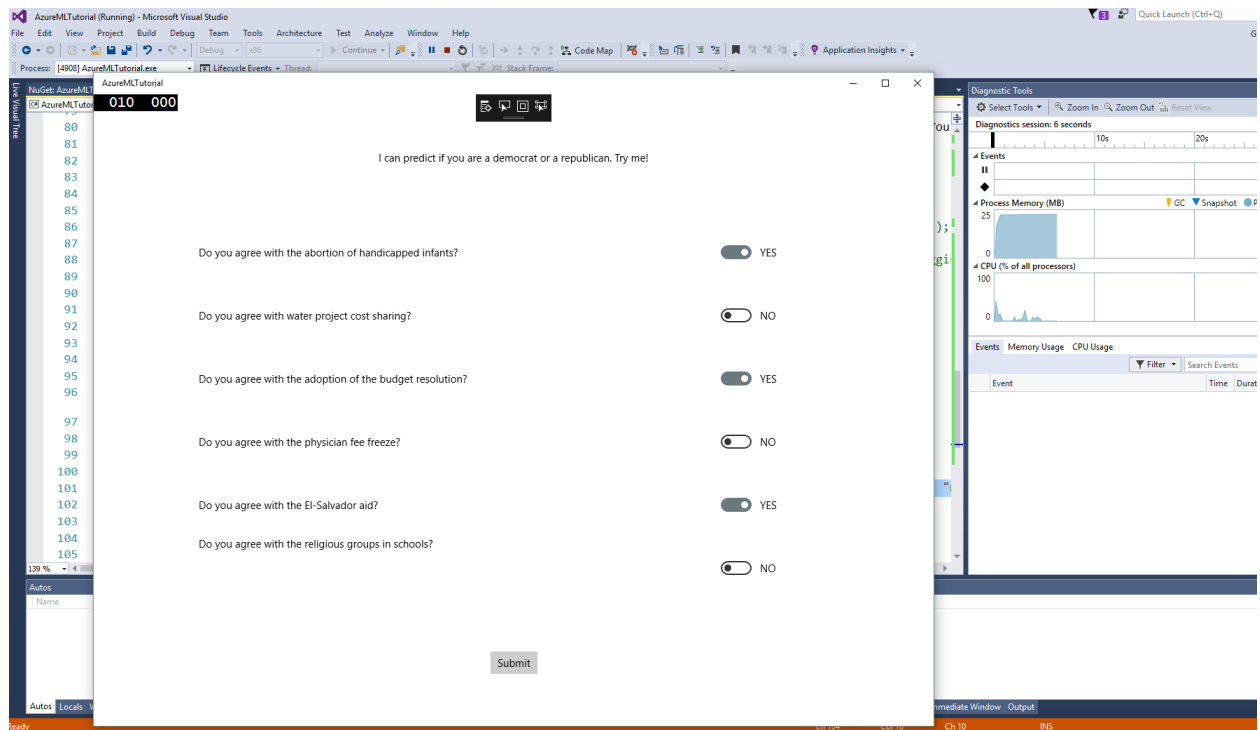
```

private async void Button_Click(object sender, RoutedEventArgs e)
{
    foreach (FrameworkElement item in myGrid.Children)
    {
        if (item is ToggleSwitch) answers[Grid.GetRow(item) - 1] =
        (((ToggleSwitch)item).IsOn == true) ? "y" : "n";
    }
    await InvokeRequestResponseService();
}

```

What this function does is to search all the ToggleSwitch items we have and put their respective value in the variable *answers[]* at the correct position. You may need to take some time to agree with that line of code, don't worry!

Run the application and have fun, it should look like this:



Note: my answers here were chosen at random, nothing personal :P

## Wrap up

If you've reached this point you have successfully finished this tutorial. Congratulations!

Thanks for staying with me throughout my first tutorial. Please feel free to leave any comments or questions, See you soon!