1.Target

* Decision of the car price by car features.

2.Dataset

* Automobile data set has published by Jeffrey C. Schimer to UCI.

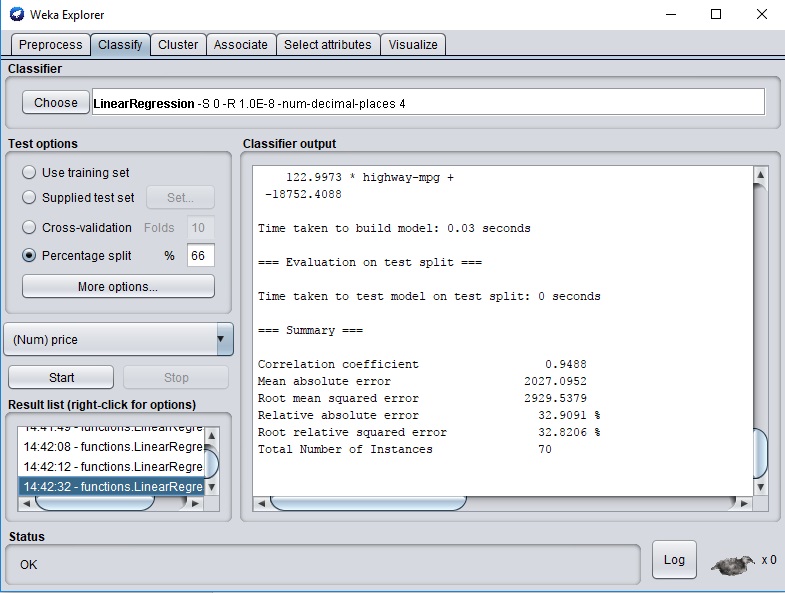
Dataset includes, 26 features and 205 instances

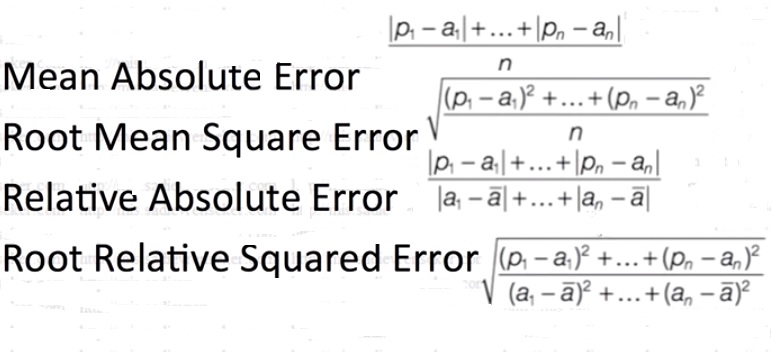
* Meaning of the features

|  |
| --- |
| Symbol:insurance +3risky -3safe |
| Normalized-loses: risk/car |
| Brand |
| Fuel |
| Aspiration:motor type |
| No-of-doors |
| Body-style |
| Engine-to-whell |
| Engine-location |
| No-of-cylinder |
| Engine-size |
| Fuel-system |
| Bore |
| Stroke |
| Compression ratio |
| Hp |
| Peak-rpm |
| City-mpg |
| Highway-mpg |
| Price |

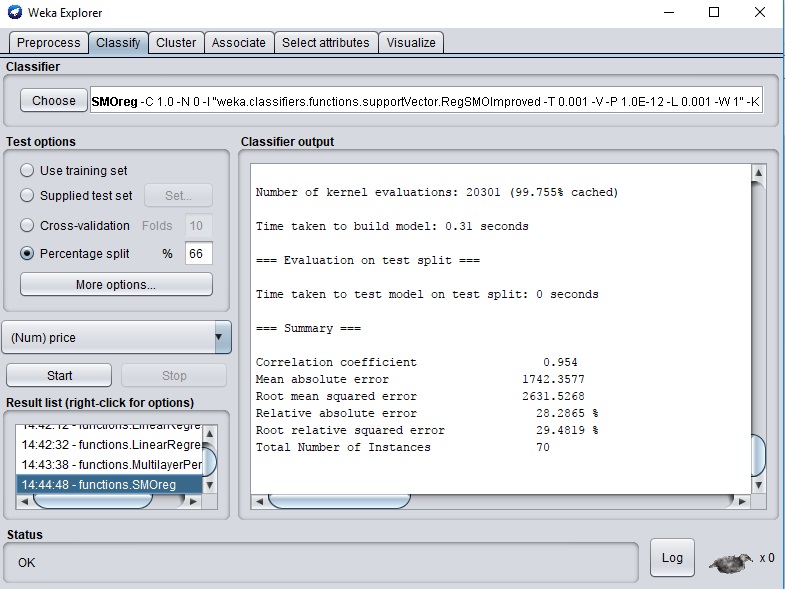
3.Classification via weka

* Classification models tried by.



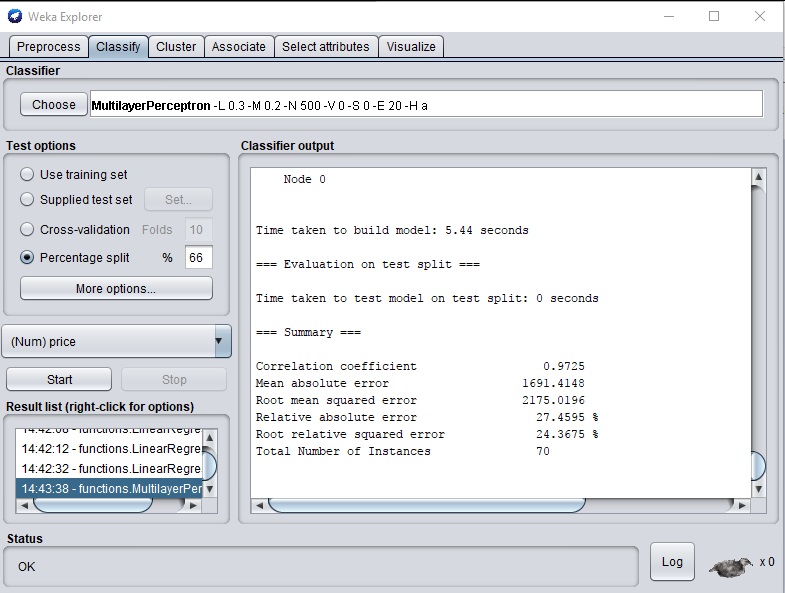


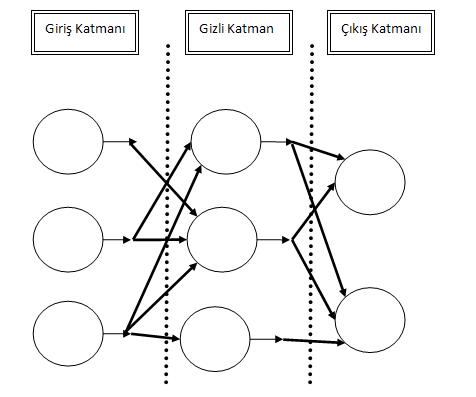
Mean Absolute Error: predicted-real/n  
Relative Absolute Error: Predicted deviation/ real deviation -avarage



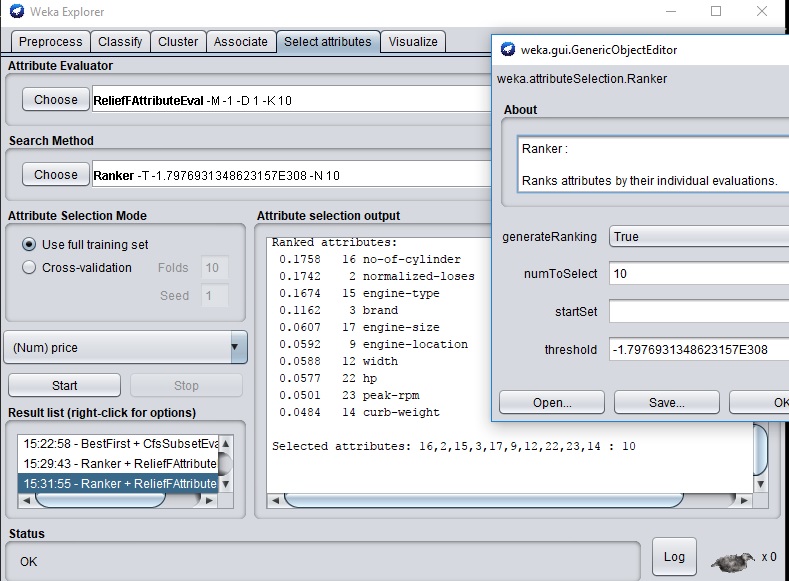
Support Vector Machine is SMOReg on Weka. For classification, this can make new line between two lines, so two lines can saparate. This main line, we can call border. Espacially for this border. 2 group of lines samples must far as far as possible. SVM can decide to make a border.

For using. for two group, we must make two parallel lines and then we can move this two lines for get close eachother we can make a border line.

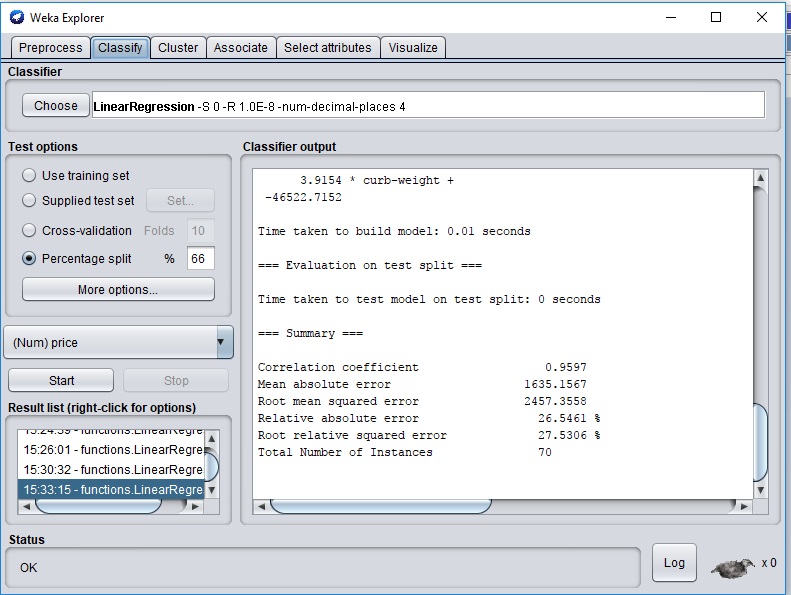


As we saw in the graph, there is input, secret and output secret neurons. Input and output main target is interaction of the system for outer inputs. Every neurons can make operation.

* According to decision of the price, we are looking for discrete variables thats why we are trying the variations of the regressions.
* Corelation coefficiencys are not so different from eachothers, that differences are about the using different kind of regressions equations.
* We have a lot of features so we must do feature selection. We did on ReliefAttributeEvaluation ranker 10



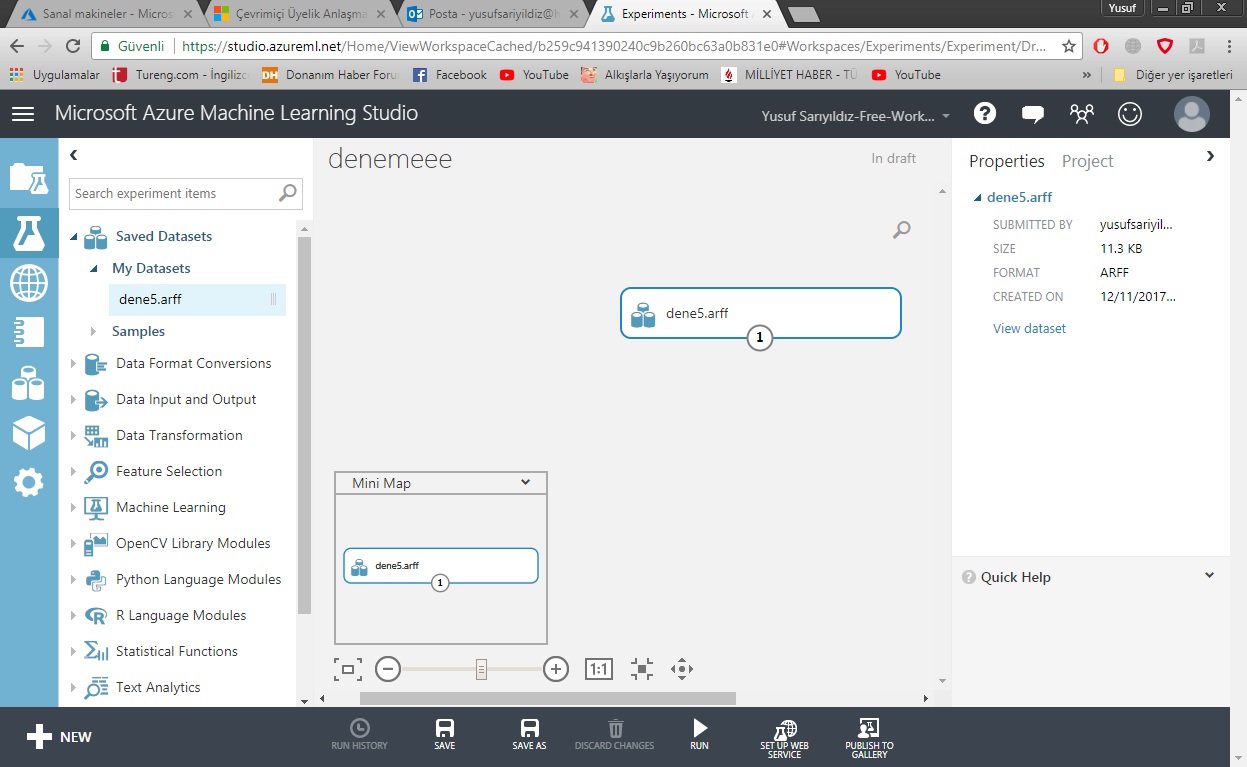
then we tred again our feature selectioned dataset, correlation coefficiency has increased



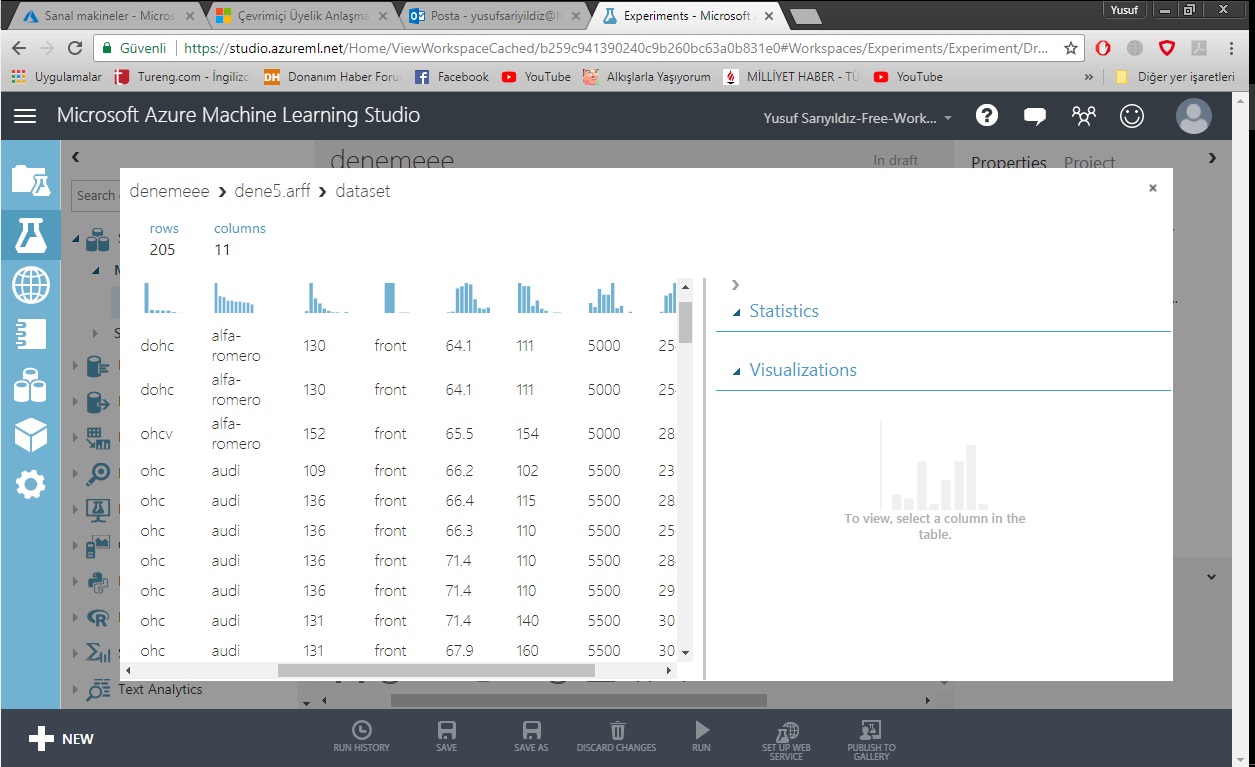
4. Project on Azure Machine Learning.

Make an account on azureml.net. Then, you can make a Project step by step.

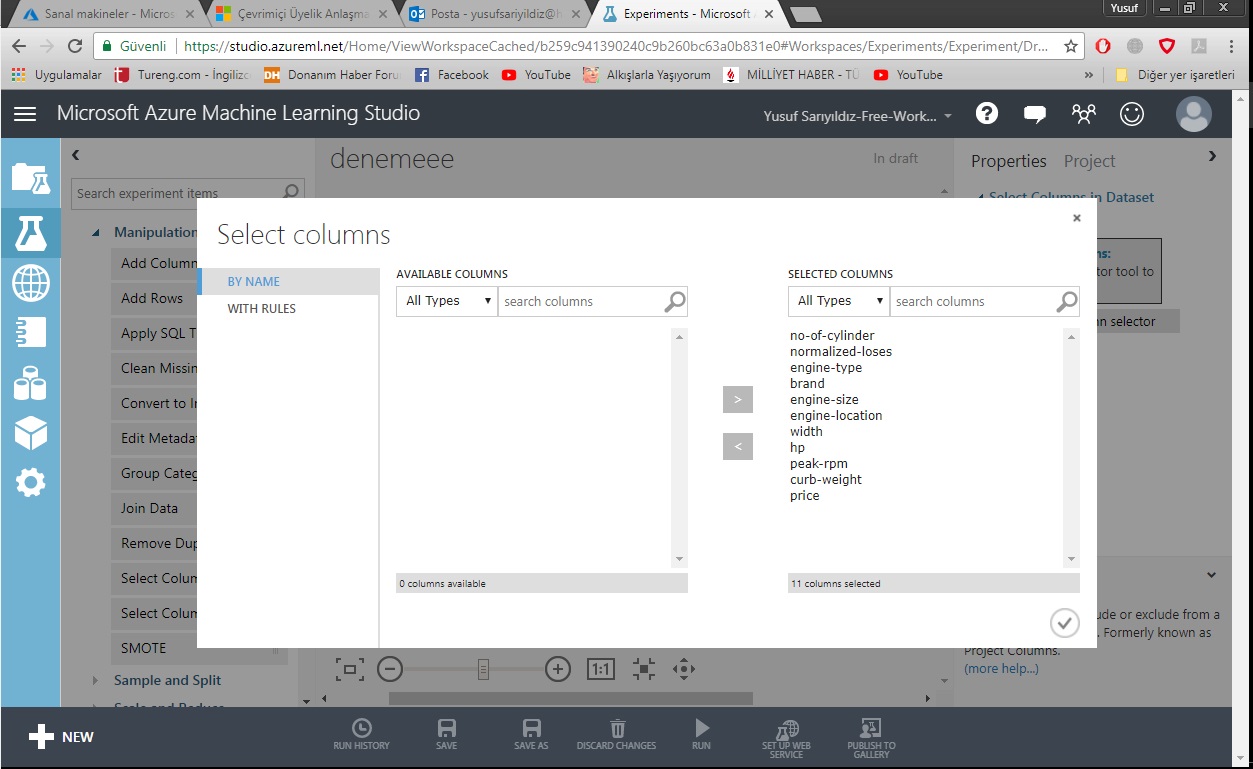
1. Upload your arff file to system.



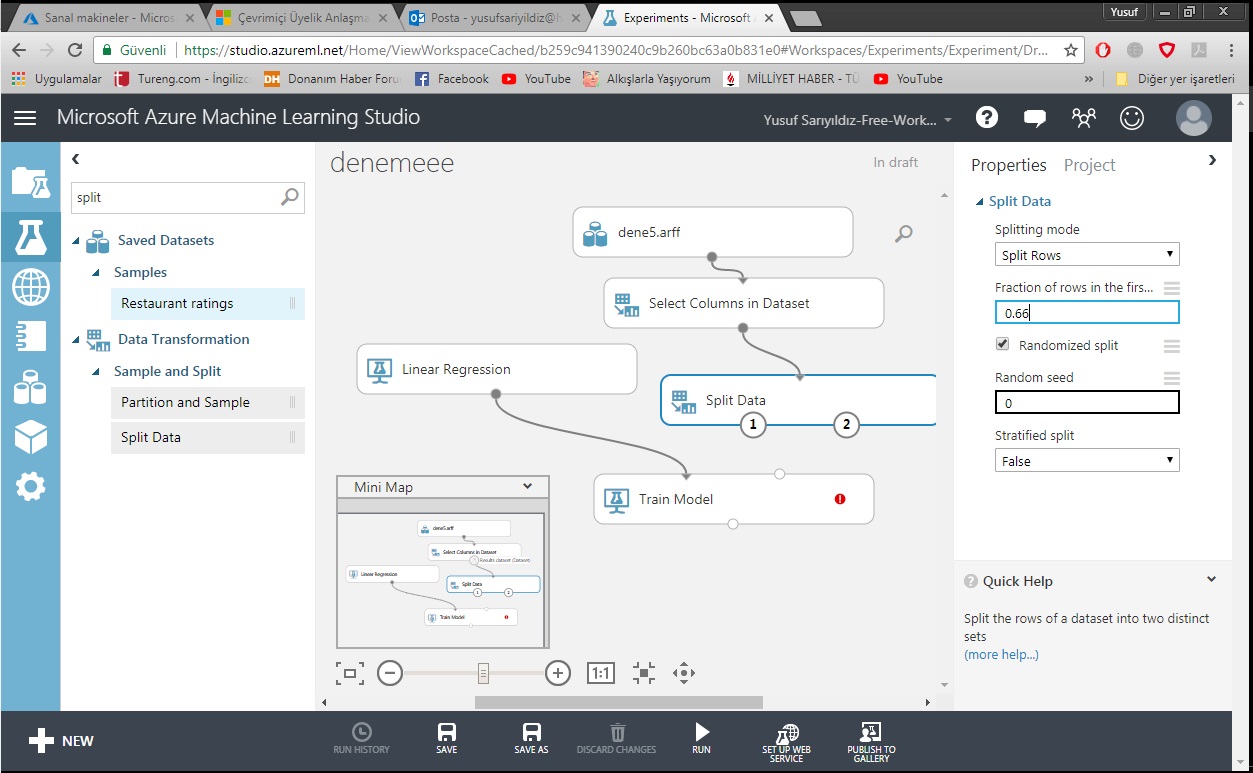
1. We can see attributes and samples via right click.



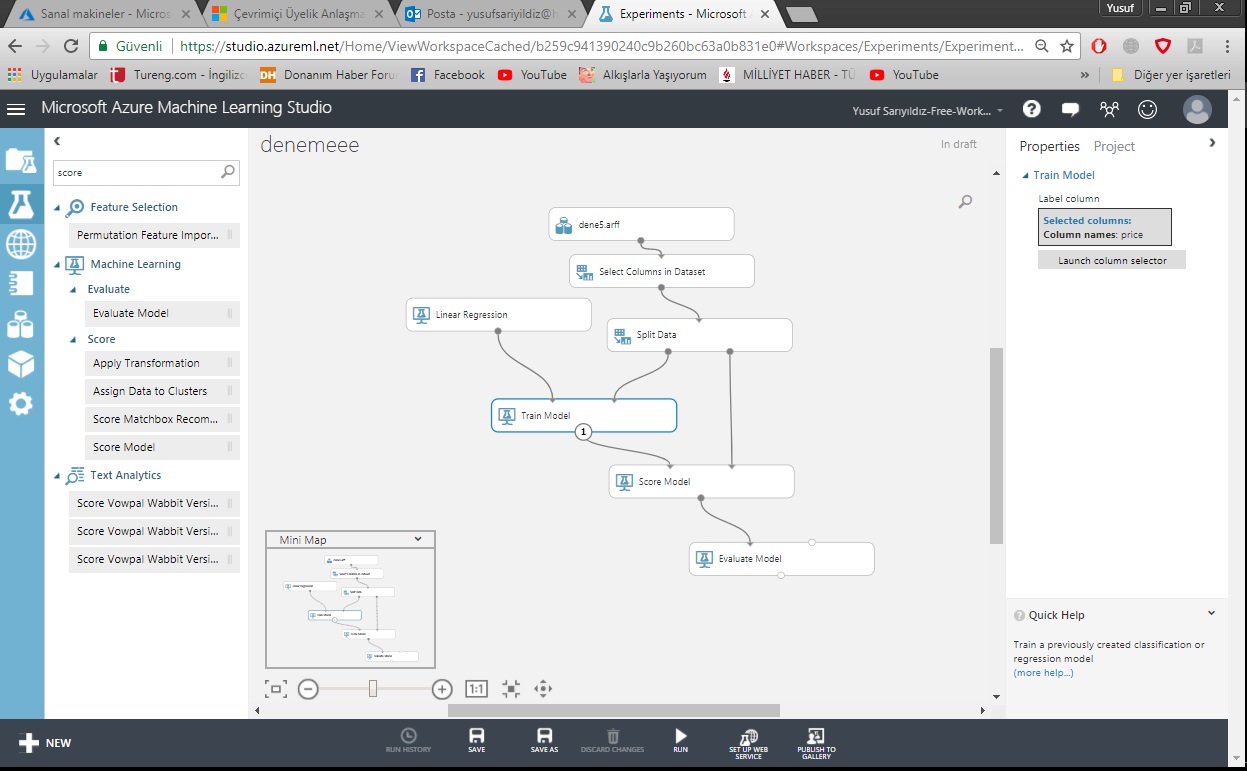
1. For using columns, we click select coumn in dataset. On the right side we press launch column selector.



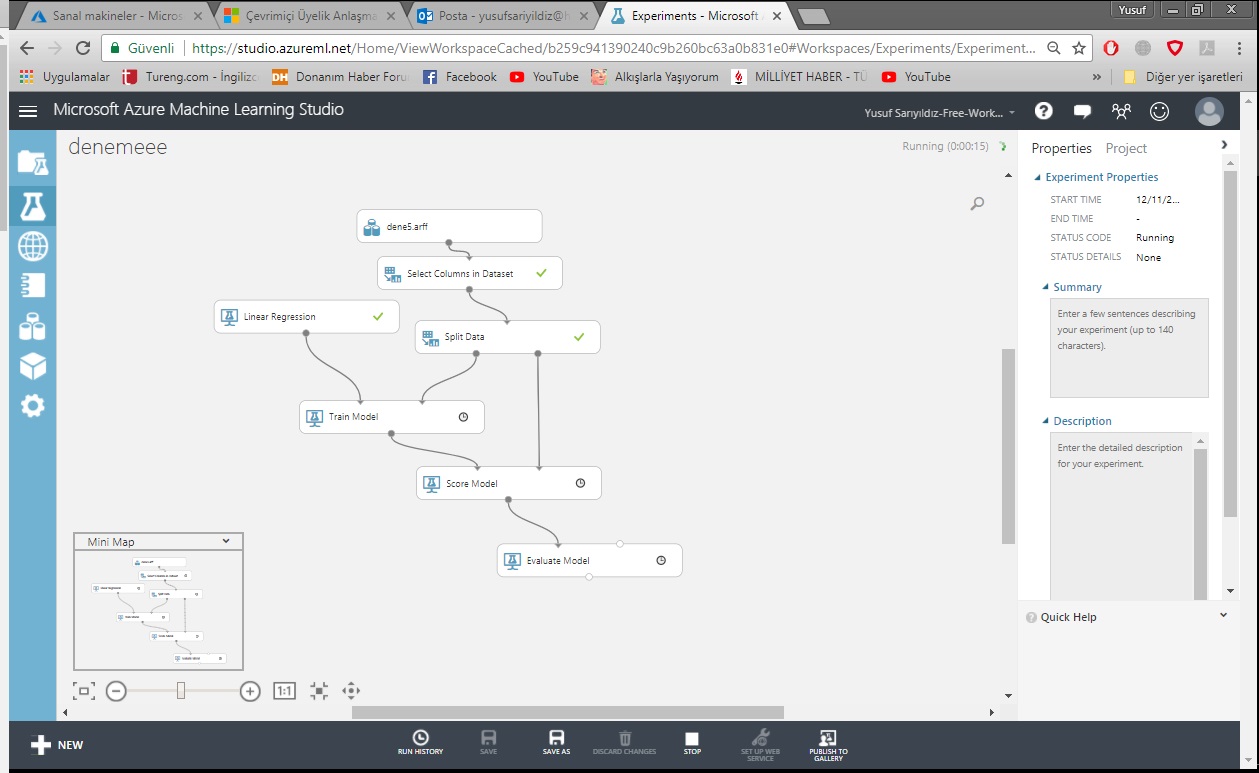
1. Saparate data via split. We write 0.66 for fraction



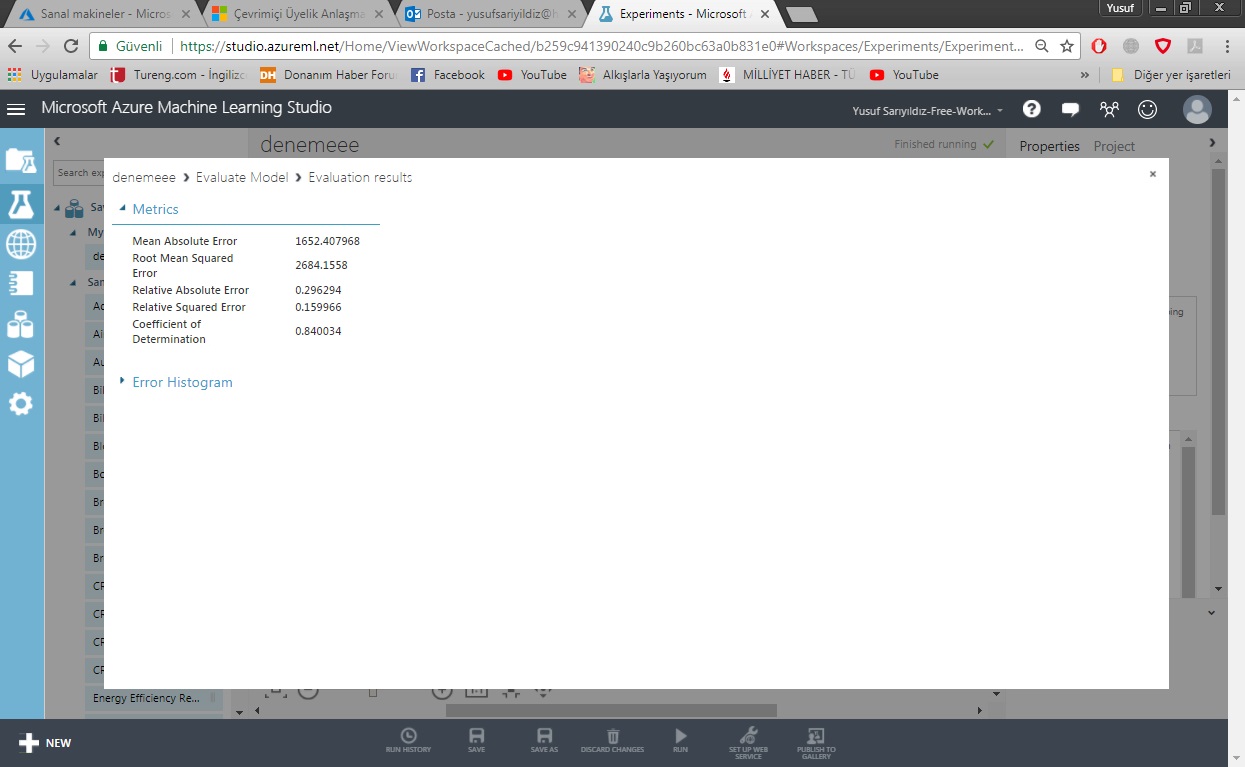
1. Match score model and evaluate model. Then, match score’s second part and split data.



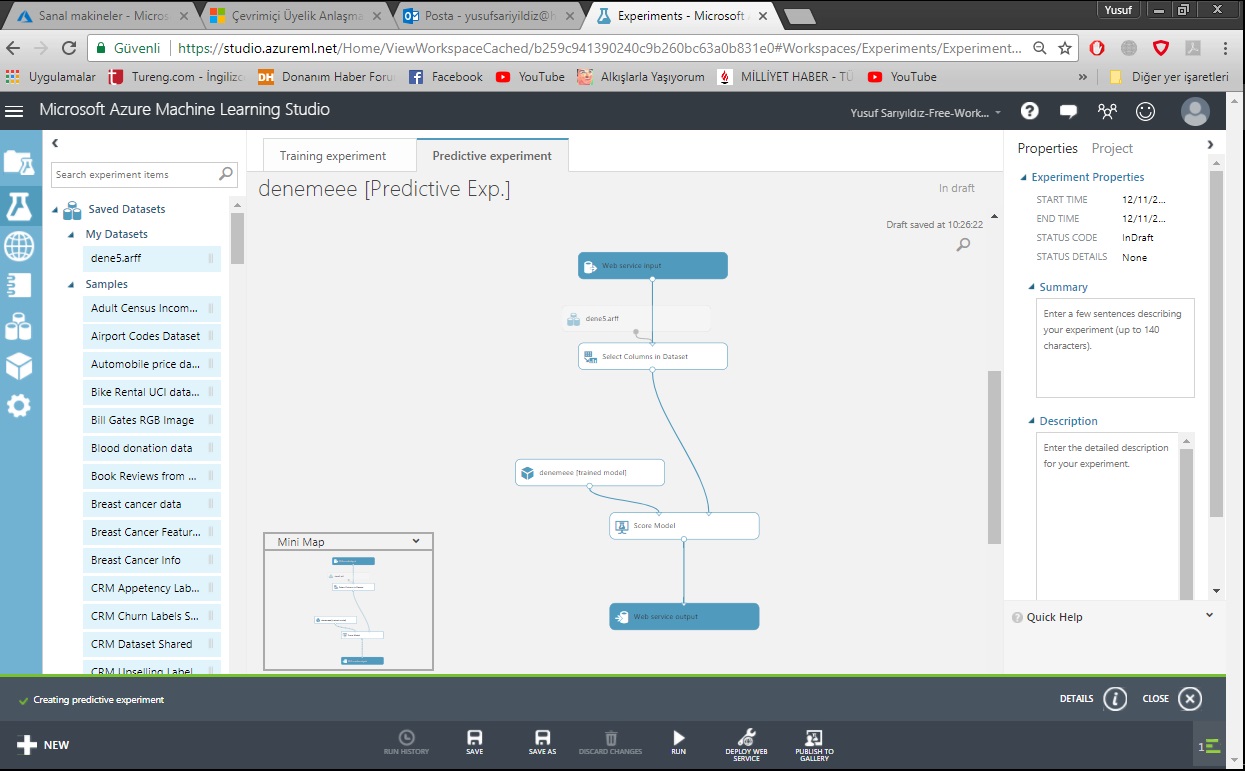
1. Press save and run botton



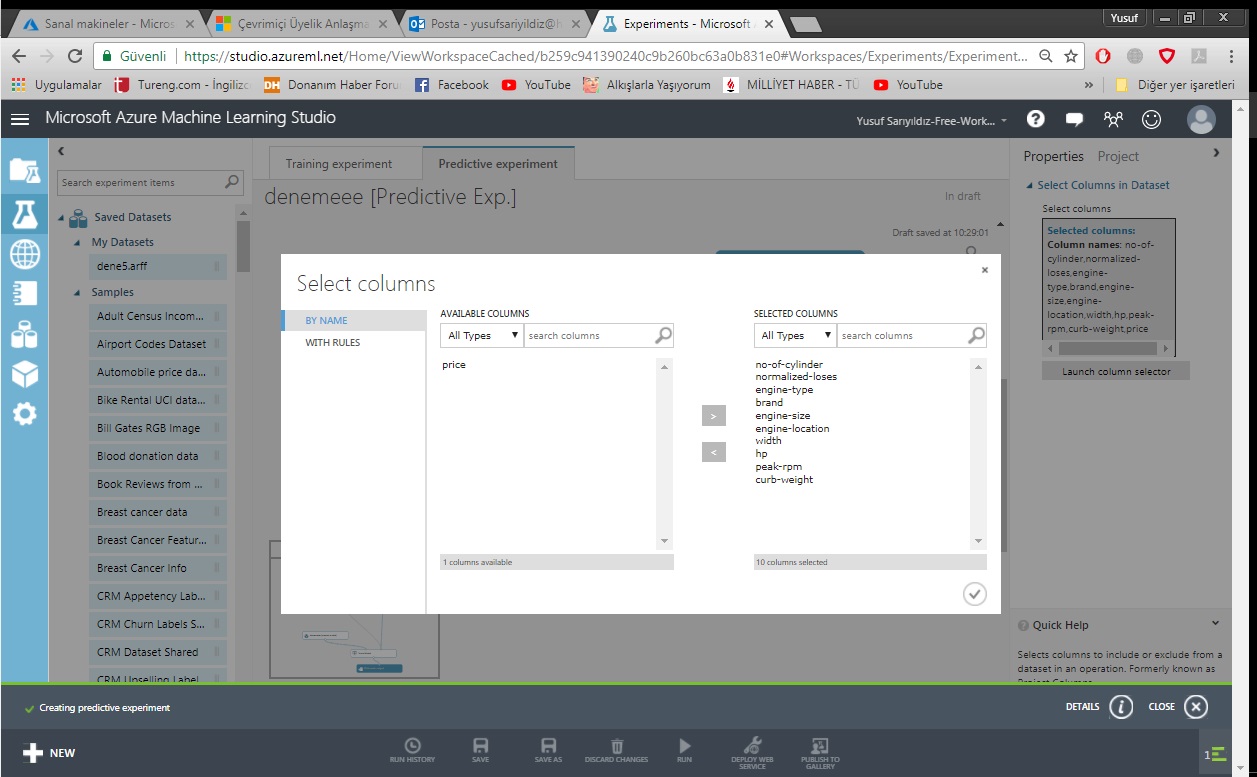
1. Right click to evaluate then click to visualize can Show corelation coefficiancies.



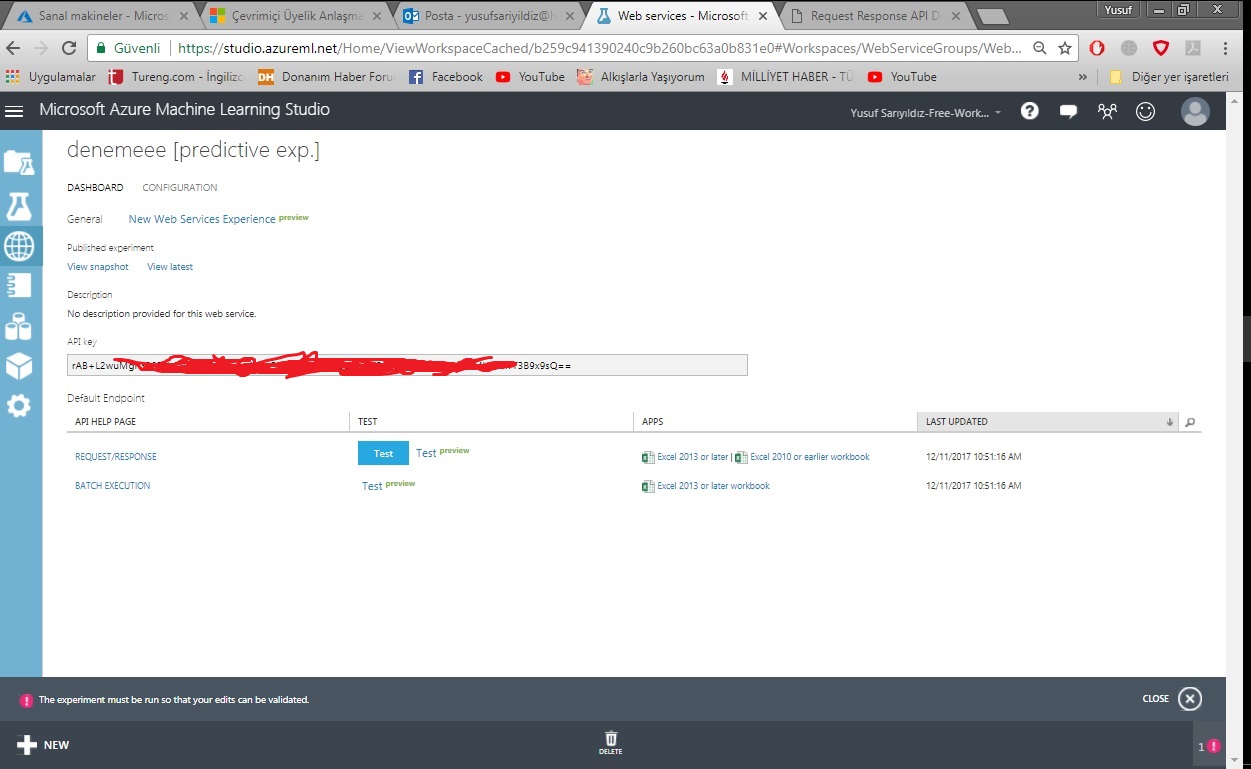
8.Press set up web service then, trained model made by Azure.



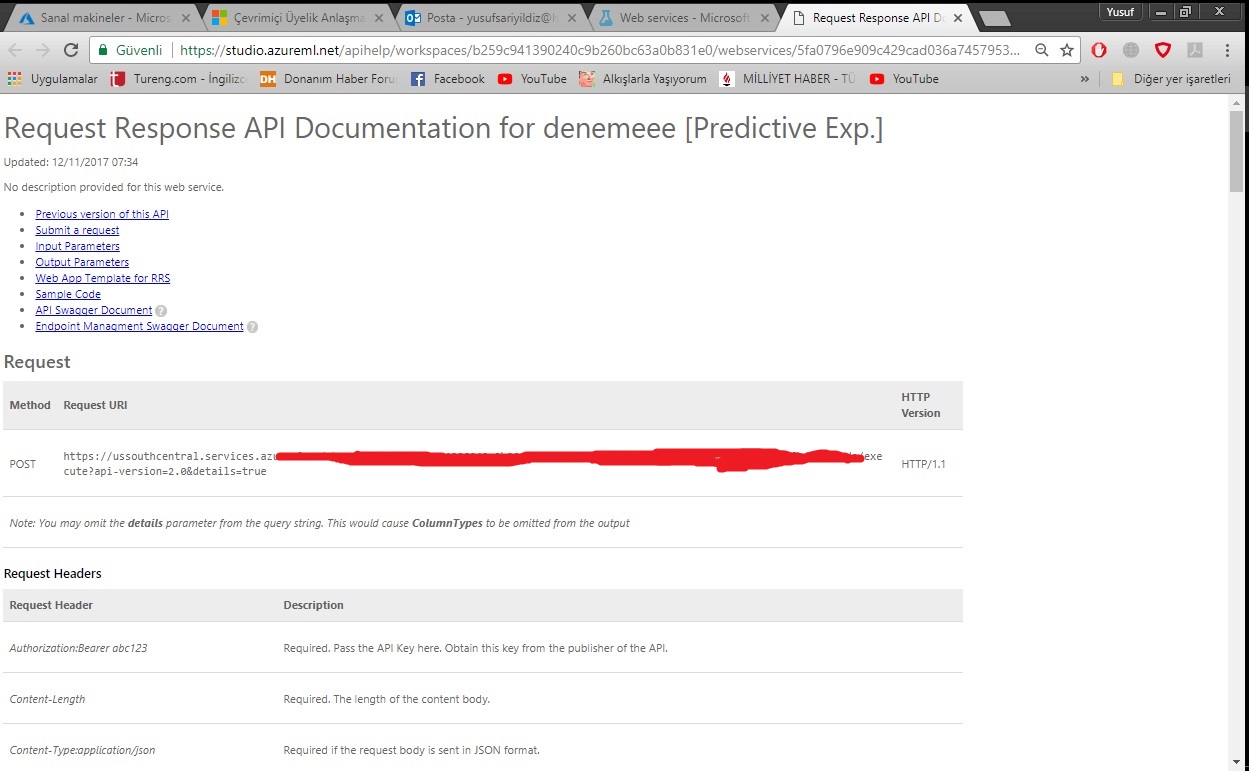
9.Our model can give price now. So we can drop off price from select column.



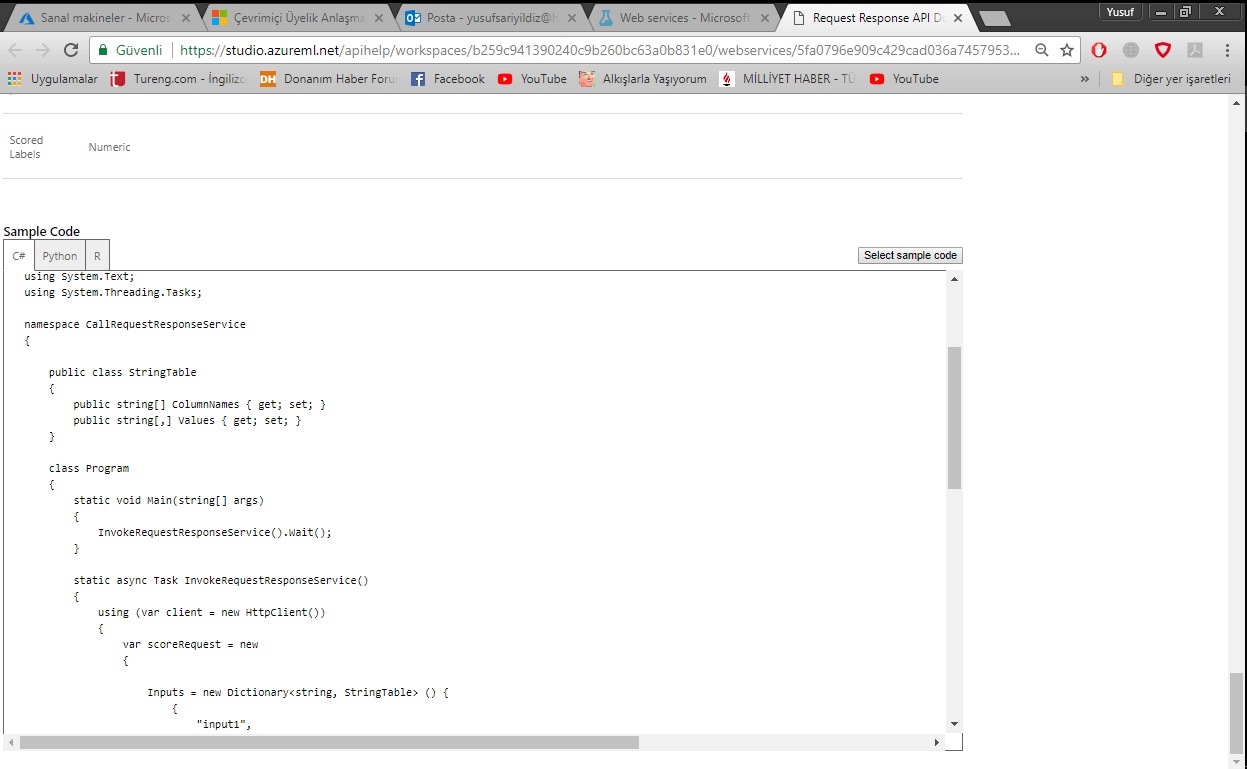
10. Press deploy. The new page has opened. We can see apikey on the page.



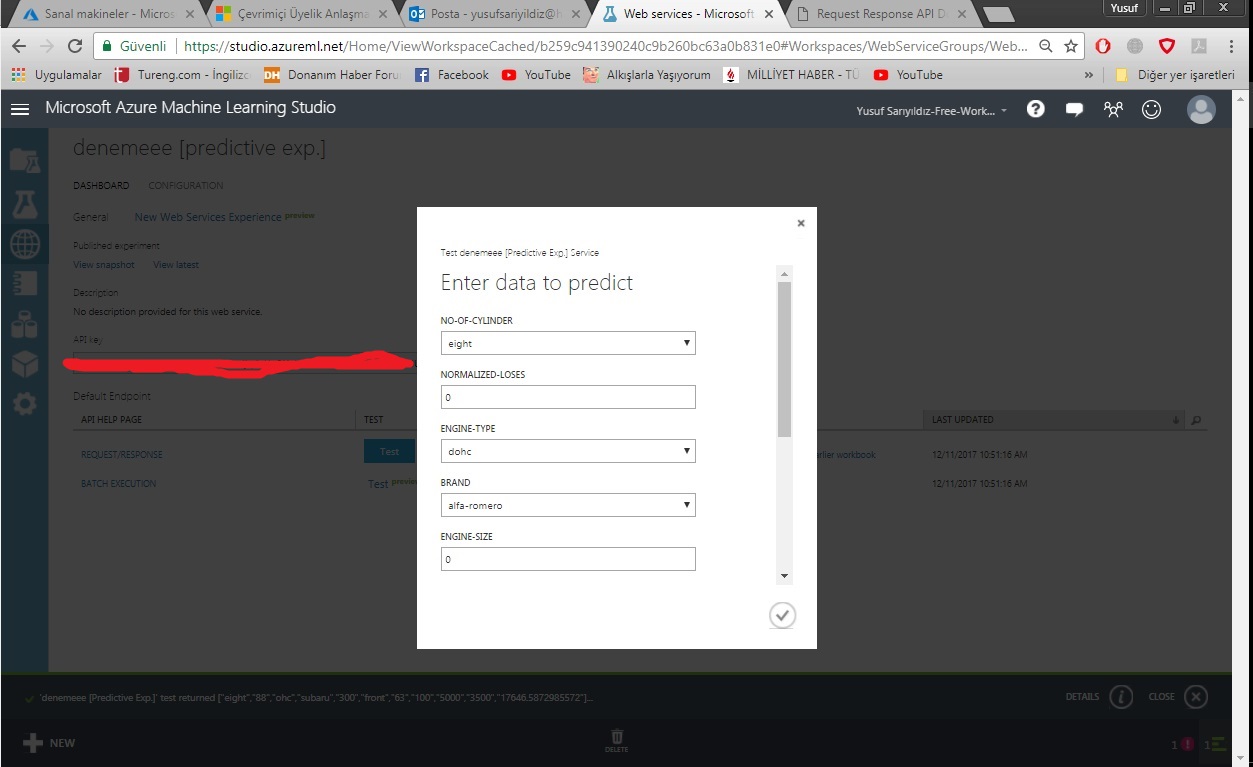
11. On clicking request respond. How to use page is avaible.



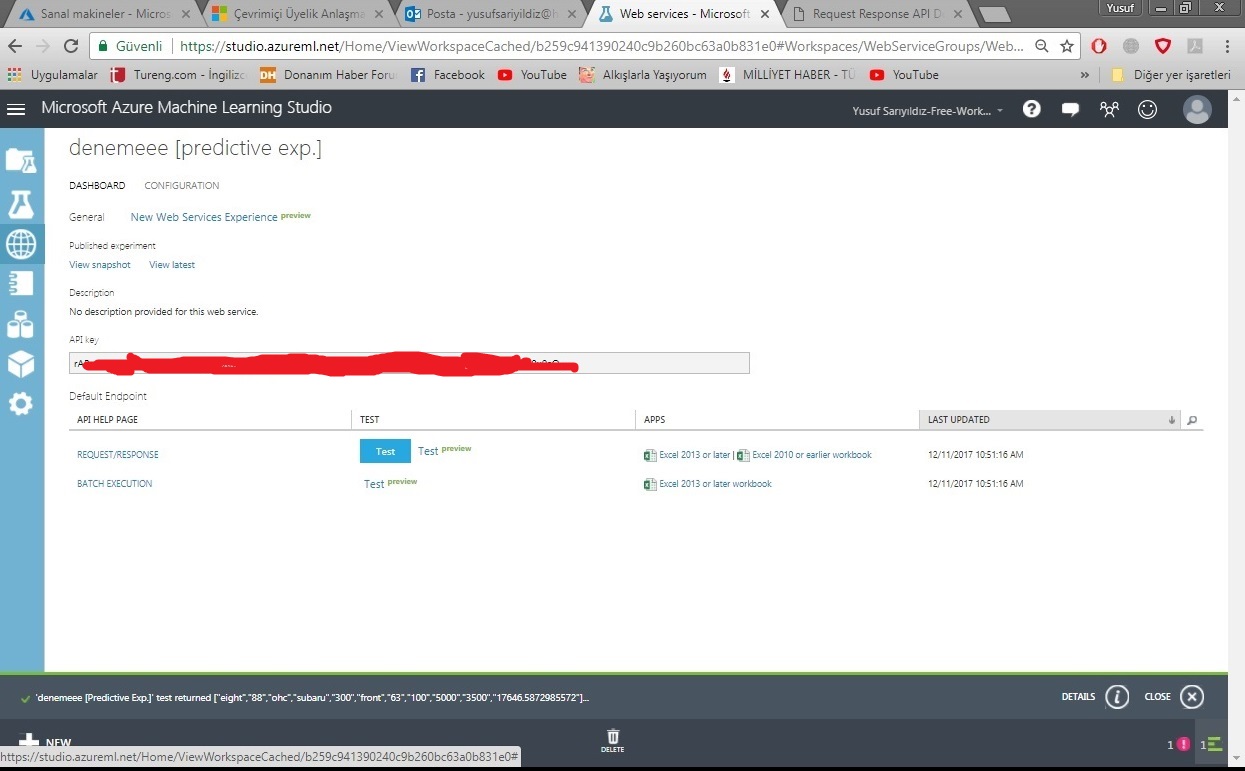
12. Sample usage of c# is avaible



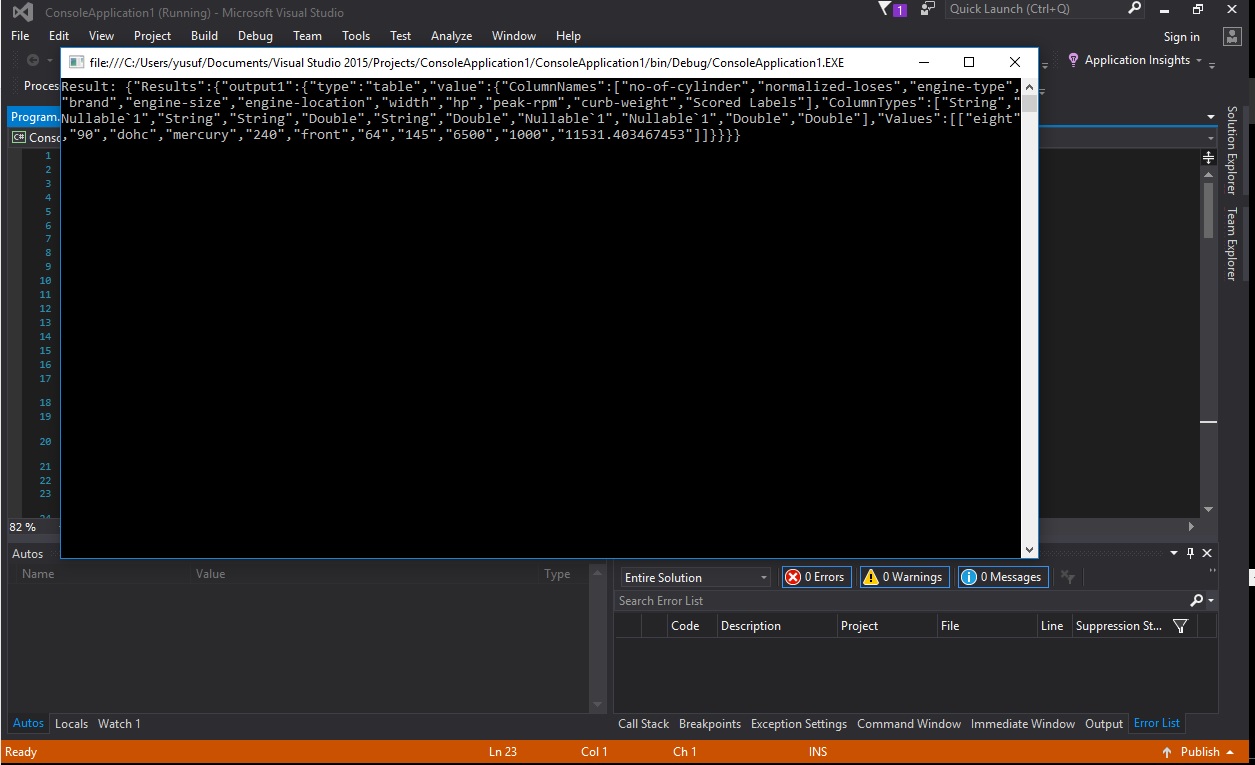
13. We can test via test button



14. We can see results on grey area



15. Result of console application.



Codes

// 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; }

}

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, StringTable>() {

{

"input1",

new StringTable()

{

ColumnNames = new string[] {"no-of-cylinder", "normalized-loses", "engine-type", "brand", "engine-size", "engine-location", "width", "hp", "peak-rpm", "curb-weight"},

// Values = new string[,] { { "eight", "0", "dohc", "alfa-romero", "0", "front", "0", "0", "0", "0" }, { "eight", "0", "dohc", "alfa-romero", "0", "rear", "0", "0", "0", "0" }, }

// Values = new string[,] { { "six", "200", "ohc", "subaru", "200", "rear", "64", "145", "6500", "1000" } }

Values = new string[,] { { "eight", "90", "dohc", "mercury", "240", "front", "64", "145", "6500", "1000" } }

//4073

//11531

/\* ----num-of-cylinders: eight, five, four, six, three, twelve, two.----

---normalized-losses: continuous from 65 to 256.------

----engine-type: dohc, dohcv, l, ohc, ohcf, ohcv, rotor.----

----brand: alfa-romero, audi, bmw, chevrolet, dodge, honda,

isuzu, jaguar, mazda, mercedes-benz, mercury,

mitsubishi, nissan, peugot, plymouth, porsche,

renault, saab, subaru, toyota, volkswagen, volvo----

------ engine-size: continuous from 61 to 326.---------

-------engine-location: front, rear.------

------- hp: continuous from 48 to 288.------

--------width: continuous from 60.3 to 72.3.------------

------horsepower: continuous from 48 to 288.-------

----------peak-rpm: continuous from 4150 to 6600.-----------

-------curb-weight: continuous from 1488 to 4066.---------------- \*/

}

},

},

GlobalParameters = new Dictionary<string, string>()

{

}

};

const string apiKey = "rAB+L2aaaaaaaaaaaaaaaaaaaaaBDDb7l/sQlda170ebBoIUf/cK9Uu+cSJMUxbAvT3B9x9sQ=="; // 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/workaaaaaaaaspaces/b25aaaaaaaaaaaaaaaaaae0/services/0daaaaaaaaaaaaaaaaac1/execute?api-version=2.0&details=true");

// 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);

}

System.Threading.Thread.Sleep(30008800);//görmek için beklettik

}

}

}

}