Microsoft

Machine Learning

Azure Machine Learning Lab

Contents

Formatted: Font: (Default) Segoe UI Light, 14 pt, Complex Script Font: Segoe UI Light, 14 pt

Formatted: Complex Script Font: 14 pt

Formatted: Font: (Default) Segoe UI Light, 14 pt, Complex Script Font: Segoe UI Light, 14 pt

Formatted: Complex Script Font: 14 pt

Formatted: Font: (Default) Segoe UI Light, 14 pt, Complex Script Font: Segoe UI Light, 14 pt

Formatted: Complex Script Font: 14 pt

Formatted: Font: (Default) Segoe UI Light, 14 pt, Complex Script Font: Segoe UI Light, 14 pt

Formatted: Complex Script Font: 14 pt

Formatted: Font: (Default) Segoe UI Light, 14 pt, Complex Script Font: Segoe UI Light, 14 pt

Formatted: Complex Script Font: 14 pt

Formatted: Font: (Default) Segoe UI Light, 14 pt, Complex Script Font: Segoe UI Light, 14 pt

Formatted: Complex Script Font: 14 pt

Formatted: Font: (Default) Segoe UI Light, 14 pt, Complex Script Font: Segoe UI Light, 14 pt

Formatted: Complex Script Font: 14 pt

Formatted: Font: (Default) Segoe UI Light, 14 pt, Complex Script Font: Segoe UI Light, 14 pt

Formatted: Complex Script Font: 14 pt

Formatted: Font: (Default) Segoe UI Light, 14 pt, Complex Script Font: Segoe UI Light, 14 pt

Formatted: Font: (Default) Segoe UI Light, 14 pt, Complex Script Font: Segoe UI Light, 14 pt

Formatted: Complex Script Font: 14 pt

Formatted: Font: (Default) Segoe UI Light, 14 pt, Complex Script Font: Segoe UI Light, 14 pt

What is Azure Machine Learning?

This lab will introduce you to machine learning capabilities available in Microsoft Azure, specifically Microsoft Azure Machine Learning (Azure ML).

Azure ML is a fully managed machine learning platform that allows you to perform predictive analytics. Development of models (experiments) is achieved using the Azure ML Studio, a web based design environment that empowers both data scientists and domain specialists to build end-to-end solutions and significantly reduce the complexity of building and publishing predictive models. It has a simple drag-and-drop authoring and a catalogue of modules that provide functionality for an end-to-end workflow. More experienced users can also embed their own Python or R scripts in line in experiments and explore the data interactively with Jupyter Notebooks.

One of the most important features of Azure ML is its publishing service whereby a finished (trained) experiment can be exposed as a web API that can be consumed by any other application such as a website, mobile application etc. Each experiment can also be configured to be re-trained with new data using a separate API to maintain it.

Commented [EB1]: Should this be capitalised throughout?

In this short lab we'll explore how Azure ML works by looking at a simple scenario, involving the incidence of diabetes in the Pima Indian tribe from Arizona in the USA. We'll also see how Python scripts and Jupyter Notebooks can be used to aid in the exploration and evaluation of the data. We'll see how to publish the model by reviewing the sample code for a published experiment and testing the output from an Excel add-in.

The Problem Domain

During this lab, we will create and publish a model that predicts how likely a Pima Indian is to develop diabetes. Pima Indians have been found to have a high incidence of diabetes and this has been extensively researched. There is also a publicly available data set of 768 members of this tribe which has various features included and whether or not they are diabetic:

- Number of times pregnant
- Plasma glucose concentration - a 2 hours in an oral glucose tolerance test
- Diastolic blood pressure (mm Hg)
- Triceps skin fold thickness (mm)
- 2-Hour serum insulin (mu U/ml)
- Body mass index (weight in kg/(height in m)^2)
- Diabetes pedigree function
- Age (years)
- Class variable (0 or 1) to indicate whether they are diabetic or not

The model we'll create is a form of supervised learning, which means the data we'll use to build the model has the outcome or label included in the data. This model will use a binary (aka two class) classification technique by *classifying* the Pima Indians into two groups, those with diabetes and those who don't have the disease.

Formatted: Font: (Default) +Body (Calibri), 10 pt, Complex Script Font: +Body CS (Arial), 10 pt

Formatted: Font: (Default) Segoe UI Light, 11 pt, Font color: Auto, Complex Script Font: Segoe UI Light, 11 pt

Formatted: Font: (Default) Segoe UI Light, 11 pt, Font color: Auto, Complex Script Font: Segoe UI Light, 11 pt

Formatted: Font: (Default) Segoe UI Light, 11 pt, Font color: Auto, Complex Script Font: Segoe UI Light, 11 pt

Formatted: Font: (Default) Segoe UI Light, 11 pt, Font color: Auto, Complex Script Font: Segoe UI Light, 11 pt

Formatted: Font: (Default) Segoe UI Light, 11 pt, Font color: Auto, Complex Script Font: Segoe UI Light, 11 pt

Formatted: Font: (Default) Segoe UI Light, 11 pt, Font color: Auto, Complex Script Font: Segoe UI Light, 11 pt

Formatted: Font: (Default) Segoe UI Light, 11 pt, Font color: Auto, Complex Script Font: Segoe UI Light, 11 pt

Formatted: Font: (Default) Segoe UI Light, 11 pt, Font color: Auto, Complex Script Font: Segoe UI Light, 11 pt

Formatted: Font: (Default) Segoe UI Light, 11 pt, Font color: Auto, Complex Script Font: Segoe UI Light, 11 pt

Formatted: Font: (Default) Segoe UI Light, 11 pt, Font color: Auto, Complex Script Font: Segoe UI Light, 11 pt

Formatted: List Paragraph, Bulleted + Level: 1 + Aligned at: 0.25" + Indent at: 0.5"

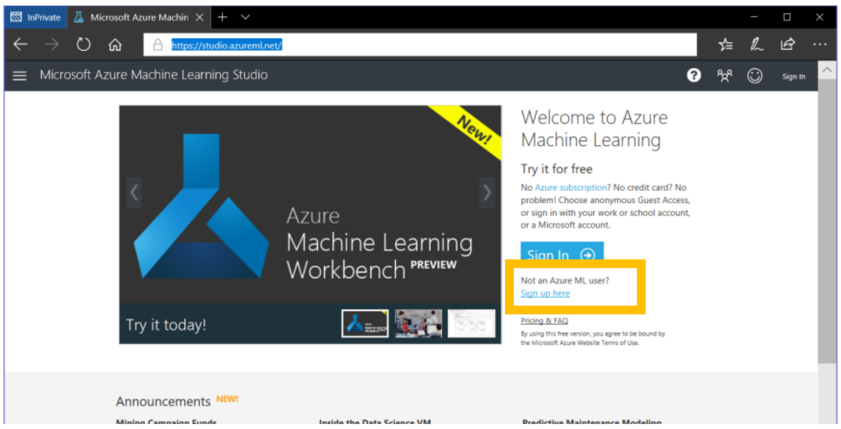
Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt

Formatted: Highlight

Formatted: Complex Script Font: Italic

First Time Setup

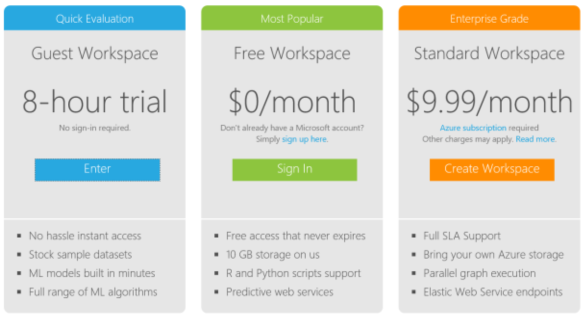
Microsoft Azure Machine Learning offers a free-tier service and a standard tier for which you need an Azure subscription. In order to access the free workspace, go to https://studio.azureml.net/, click on the sign up here link and sign in with a Microsoft account (@hotmail.com, @outlook.com etc.) and a free workspace is automatically created for you.



InPrivate
Microsoft Azure Machin
https://studio.azureml.net/
Microsoft Azure Machine Learning Studio
Sign In
New!
Azure Machine Learning Workbench PREVIEW
Try it today!
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.
Announcements NEW!

Microsoft Azure Machine Learning offers an 8 hour anonymous trial, a free-tier service and a standard tier (for which you need an Azure subscription). In order to access the free workspace, go to https://studio.azureml.net/

Formatted: Default Paragraph Font, Font: (Default) +Body (Calibri), 10 pt, Complex Script Font: +Body CS (Arial), 10 pt



Quick Evaluation	Most Popular	Enterprise Grade
Guest Workspace	Free Workspace	Standard Workspace
8-hour trial	$0/month	$9.99/month
No sign-in required.	Don't already have a Microsoft account? Simply sign up here.	Azure subscription required. Other charges may apply. Read more.
Enter	Sign In	Create Workspace
▪ No hassle instant access ▪ Stock sample datasets ▪ ML models built in minutes ▪ Full range of ML algorithms	▪ Free access that never expires ▪ 10 GB storage on us ▪ R and Python scripts support ▪ Predictive web services	▪ Full SLA Support ▪ Bring your own Azure storage ▪ Parallel graph execution ▪ Elastic Web Service endpoints

Select the Free workspace option and sign in with a Microsoft account (@hotmail.com, @outlook.com etc.) and a free workspace is automatically created for you.

If you don't have a Microsoft account, create one here: https://signup.live.com/signup.aspx

Formatted: Font: (Default) +Body (Calibri), 10 pt, Complex Script Font: +Body CS (Arial), 10 pt

Once on the Workspace homepage click on the link 'Sign-in to ML Studio' and you should land on an experiments page. From there find the 'New' button in the bottom left corner and choose 'Experiment' then 'Blank Experiment'. Notice there are sample experiments you can also use, edit and build on top of.

Getting Started

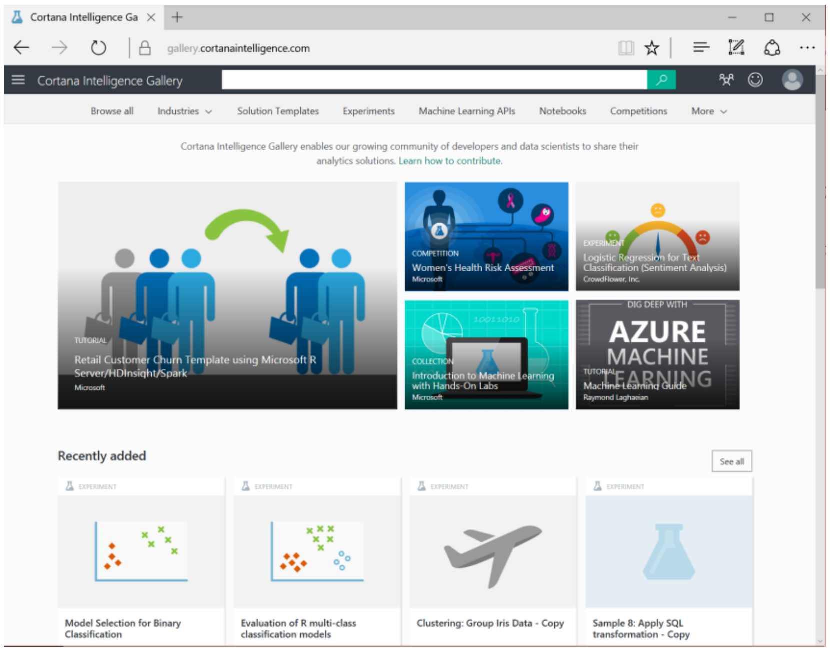
Formatted: Not Highlight

Formatted: Not Highlight

Now we can start building our experiment. Typically we would get hold of a data set and either connect to its source or upload a file:

- Azure ML accepts many formats including CSV, TSV, Plain Text and Zip files
- Alternatively, you can use the 'Import Data' Module to access your data from sources in Azure such as databases like SQL Azure , Blob storage , HD insight (Hadoop and Spark) and industry standard OData feeds.

It's also possible to adapt an existing experiment to our needs from one of the many samples in the Cortana Intelligence Gallery . If we open a new tab in our browser we can see the sort of samples that exist there which have either been created by Microsoft or the community of experts using the tool.

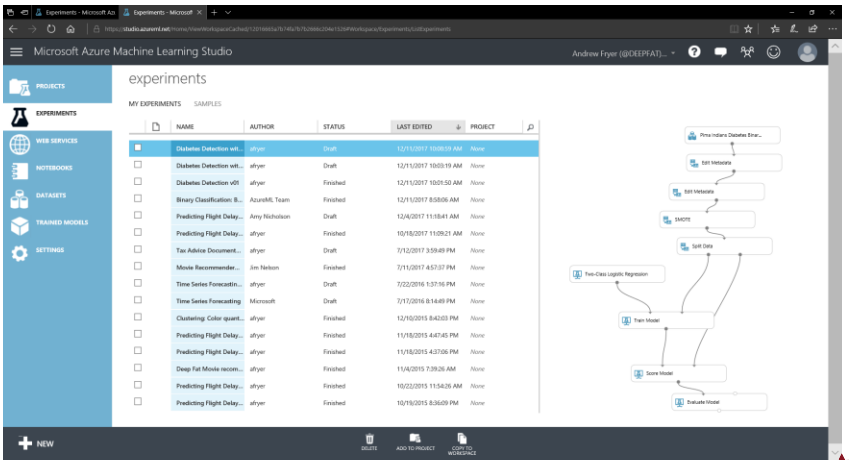

Cortana Intelligence Ga
gallery.cortanaintelligence.com
Cortana Intelligence Gallery
Browse all
Industries
Solution Templates
Experiments
Machine Learning APIs
Notebooks
Competitions
More
Cortana Intelligence Gallery enables our growing community of developers and data scientists to share their analytics solutions. Learn how to contribute.
TUTORIAL
Retail Customer Churn Template using Microsoft R Server/HDInsight/Spark
Microsoft
COMPETITION
Women's Health Risk Assessment
Microsoft
EXPERIMENT
Logistic Regression for Text Classification (Sentiment Analysis)
CrowdFlower, Inc.
COLLECTION
Introduction to Machine Learning with Hands-On Labs
Microsoft
DIG DEEP WITH
AZURE MACHINE LEARNING
TUTORIAL
Machine Learning Guide
Raymond Laghaeian
Recently added
See all
EXPERIMENT
Model Selection for Binary Classification
EXPERIMENT
Evaluation of R multi-class classification models
EXPERIMENT
Clustering: Group Iris Data - Copy
EXPERIMENT
Sample 8: Apply SQL transformation - Copy

For this lab we will start with a blank experiment. Go back to to https://studio.azureml.net/

 And sign in again if needed. We'll get a screen like this except that as a new user there won't be any experiments to display. Experiments are essentially models complete with the data associated with them

Formatted: Default Paragraph Font, Font: (Default) +Body (Calibri), 10 pt, Complex Script Font: +Body CS (Arial), 10 pt

Formatted: Highlight



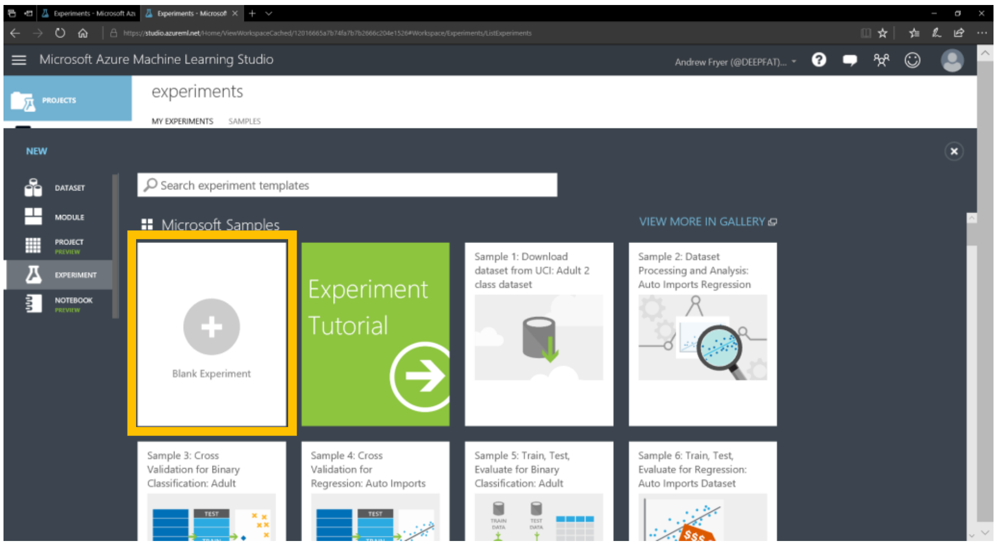
experiments

MY EXPERIMENTS   SAMPLES

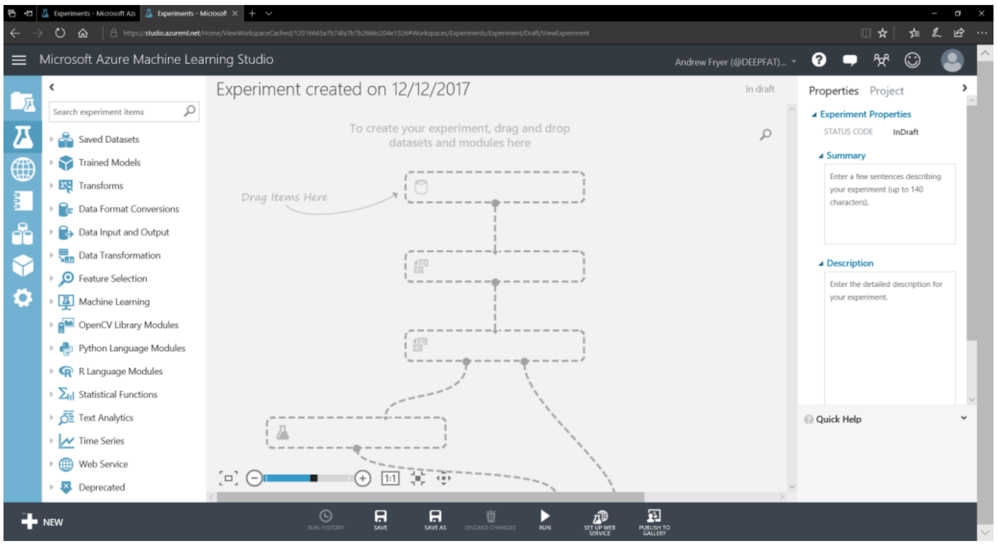
NAME	AUTHOR	STATUS	LAST EDITED	PROJECT
Diabetes Detection wit...	afryer	Draft	12/11/2017 10:08:59 AM	None
Diabetes Detection wit...	afryer	Draft	12/11/2017 10:03:19 AM	None
Diabetes Detection v01	afryer	Finished	12/11/2017 10:01:50 AM	None
Binary Classification: B...	AzureML Team	Finished	12/11/2017 8:58:06 AM	None
Predicting Flight Delay...	Amy Nicholson	Draft	12/4/2017 11:18:41 AM	None
Predicting Flight Delay...	afryer	Finished	10/18/2017 11:09:21 AM	None
Tax Advice Document...	afryer	Draft	7/12/2017 3:59:49 PM	None
Movie Recommender...	Jim Nelson	Finished	7/11/2017 4:57:37 PM	None
Time Series Forecastin...	afryer	Draft	7/22/2016 1:37:16 PM	None
Time Series Forecasting	Microsoft	Draft	7/17/2016 8:14:49 PM	None
Clustering: Color quant...	afryer	Finished	12/10/2015 8:42:03 PM	None
Predicting Flight Delay...	afryer	Finished	11/18/2015 4:47:45 PM	None
Predicting Flight Delay...	afryer	Finished	11/18/2015 4:37:06 PM	None
Deep Fat Movie recom...	afryer	Finished	11/4/2015 7:39:26 AM	None
Predicting Flight Delay...	afryer	Finished	10/22/2015 11:54:26 AM	None
Predicting Flight Delay...	afryer	Finished	10/19/2015 8:36:09 PM	None

On the left in blue with white icons are the different objects we use in ML studio such as Projects, Experiments, Web services, Jupyter Notebooks, Data Sets, Trained Models and Settings.

To start this tutorial we'll begin with a blank experiment. Click on the white plus sign on the bottom toolbar and select blank experiment:


Microsoft Azure Machine Learning Studio
Andrew Fryer (@DEEPFAT)...
PROJECTS
experiments
MY EXPERIMENTS
SAMPLES
NEW
DATASET
MODULE
PROJECT PREVIEW
EXPERIMENT
NOTEBOOK PREVIEW
Search experiment templates
Microsoft Samples
VIEW MORE IN GALLERY
Blank Experiment
Experiment Tutorial
Sample 1: Download dataset from UCI: Adult 2 class dataset
Sample 2: Dataset Processing and Analysis: Auto Imports Regression
Sample 3: Cross Validation for Binary Classification: Adult
Sample 4: Cross Validation for Regression: Auto Imports
Sample 5: Train, Test, Evaluate for Binary Classification: Adult
Sample 6: Train, Test, Evaluate for Regression: Auto Imports Dataset

We are now presented with the design surface where we can create our own experiments:

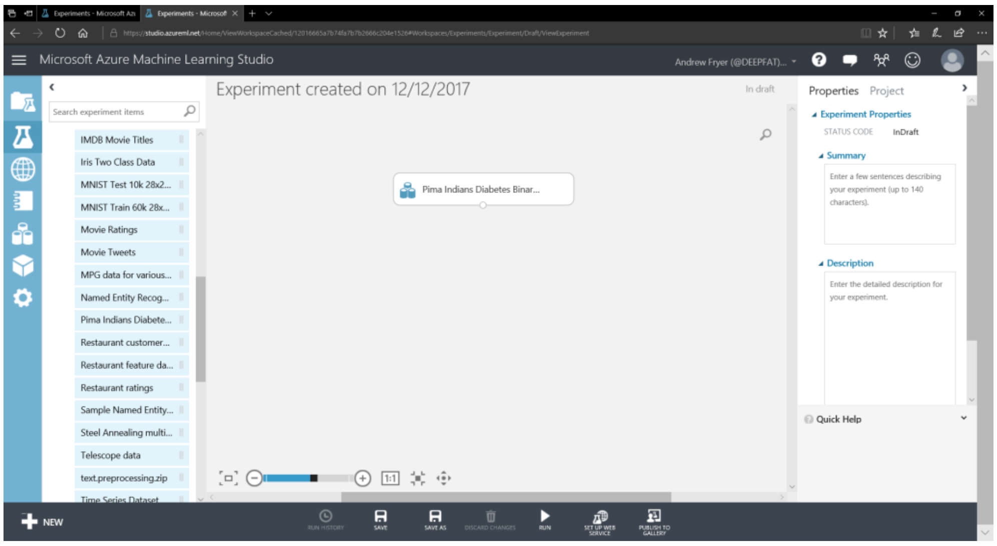


On the far right we have all the objects in ML studio as before, but now we have a list of modules next to that which we can drag on to the design surface in the middle of the screen. On the right are the properties of the module that has focused or the whole experiment if no module is selected.

Initially the design surface just has a dotted representation of an experiment until we start to create our own experiment.

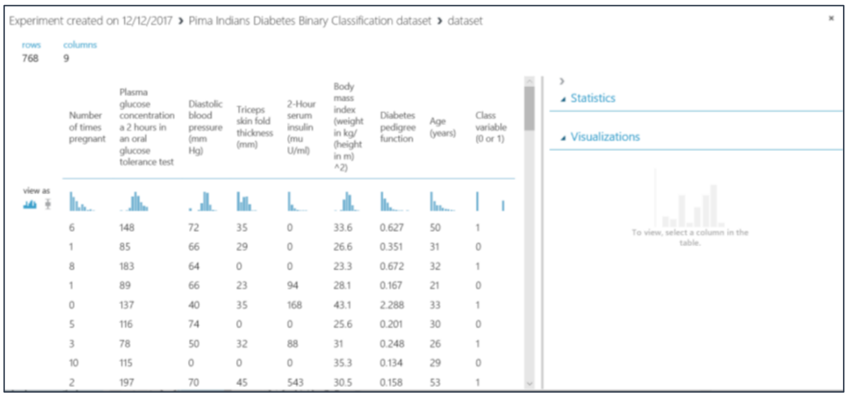
Typically we'd start work by connecting to a data source or just simply uploading a file to ML studio, but in this case we are going to use a built in sample dataset representing the Pima diabetes data mentioned earlier. Expand the Saved Datasets on the left, expand samples and scroll down to the Pima Indians Diabetes Binary Classification Dataset. Drag that onto the top centre of the design surface like so…

Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt



The first thing we should do is give our experiment a name – simply over type the "Experiment Created on …" and enter something like Pima Indians Diabetes Classification. Note there is no version control in ML studio; changes are saved for us but we can 'save as' and give our experiments meaningful names like V01 as we go along.

Now we have data we can immediately look at it, Right click on the circle at the bottom of the dataset and select visualise:

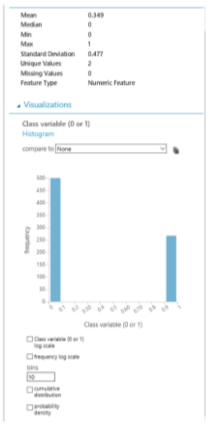


Experiment created on 12/12/2017 › Pima Indians Diabetes Binary Classification dataset › dataset

rows	columns
768	9

Number of times pregnant	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	Diastolic blood pressure (mm Hg)	Triceps skin fold thickness (mm)	2-Hour serum insulin (mu U/ml)	Body mass index (weight in kg/(height in m)^2)	Diabetes pedigree function	Age (years)	Class variable (0 or 1)
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1

We can quickly see key information about the data – there are only 768 rows in 9 columns. We can see stats about a numeric column by selecting it. Select the Class variable (0 or 1) and we'll see data about that column on the right hand side:



Mean	0.349
Median	0
Min	0
Max	1
Standard Deviation	0.477
Unique Values	2
Missing Values	0
Feature Type	Numeric Feature

We can immediately see that there are only two values (0 or 1☺) and that there are about 35% of these people with diabetes (in this case the mean of 0.349 gives us that empirically as well).

Ordinarily we'd want to clean and check this data before doing anything else with it, so what might that entail?

- Typically we'd want to check for null values and duplicates and have techniques to eliminate these.

Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt

Formatted: List Paragraph, Bulleted + Level: 1 + Aligned at: 0.25" + Indent at: 0.5"

Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt

- We may also want to adapt the data into new types of data - for example we might put numerical values into bins (aka quantisation) In this data set we might do that for any of the numeric data e.g. create five bins for different BMI values and call them very low, low, medium, high, very high.
- If we don't do that we could regularise the data so that all numeric values are scaled to be between 0 and 1 as this can help with certain types of algorithm and give better faster results.

Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt

What else can you think on based on what you have been taught so far?

Formatted: Font: Bold, Complex Script Font: Bold

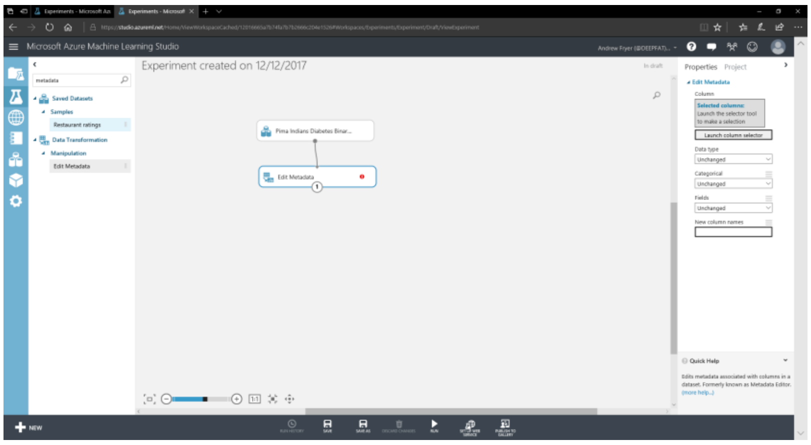
Formatted: Font: (Default) Segoe UI Light, 11 pt, Bold, Complex Script Font: Segoe UI Light, 11 pt, Bold

Formatted: Indent: Before: 0.25"

One thing we should do before we think about building a model is editing the metadata in two ways. We should declare what we are trying to predict, the label and data used to make that prediction, the features. Also the field names are very easy to understand but are not great to use in programming – they have special characters in them such as brackets and spaces which can make this difficult.

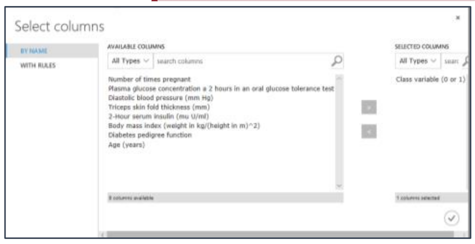
There is a module in ML studio called Edit Metadata which we can use to fix these issues.
Close the visualise window, and in the search window above the list of modules type metadata (this is the quickest way to locate a given module). Drag this module underneath the dataset module. Then connect the two modules by pointing to the small circle under the dataset module and dragging to the small circle above the edit metadata module:

Formatted: Indent: Before: 0.25"



Note the metadata module has a warning against it because its properties are not set. We can see those properties above because that module is selected. Click on the box on the right of the screen, Launch Column Selector:

Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt


Select columns

BY NAME
WITH RULES

AVAILABLE COLUMNS
All Types | search columns

Number of times pregnant
Plasma glucose concentration a 2 hours in an oral glucose tolerance test
Diastolic blood pressure (mm Hg)
Triceps skin fold thickness (mm)
2-Hour serum insulin (mu U/ml)
Body mass index (weight in kg/(height in m)^2)
Diabetes pedigree function
Age (years)

8 columns available

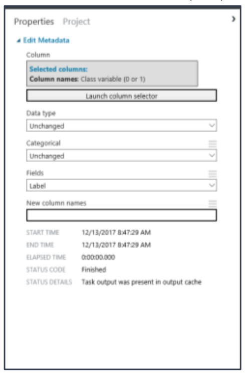
SELECTED COLUMNS
All Types | searc

Class variable (0 or 1)

1 columns selected

Select Class Variable and the > sign to move it to the SELECTED COLUMNS. Click on the tick box bottom right (if we just close the windows our selection is not saved).

In the edit metadata properties change the settings as shown by making the 'Fields' property to be 'Label':


Properties Project

Edit Metadata

Column

Selected columns:
Column names: Class variable (0 or 1)

Launch column selector

Data type: Unchanged

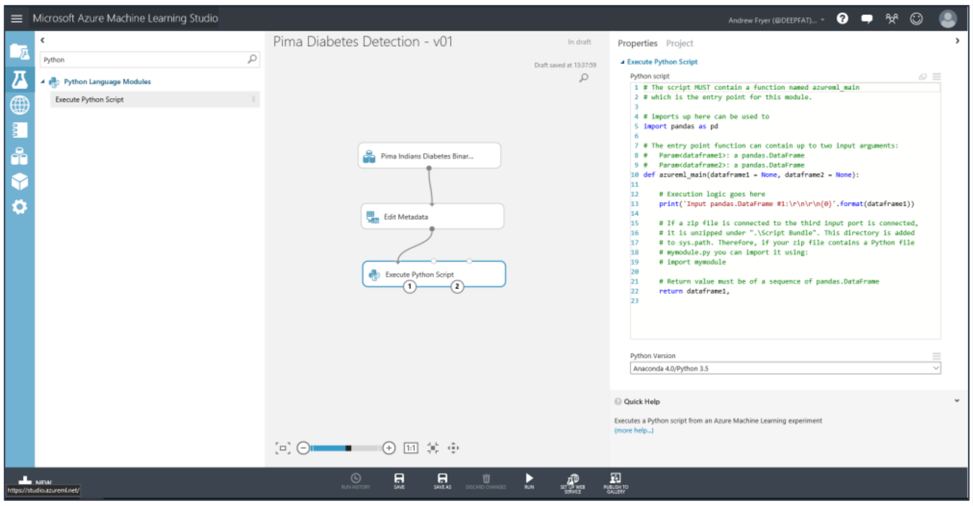
Categorical: Unchanged

Fields: Label

New column names:

START TIME	12/13/2017 8:47:29 AM
END TIME	12/13/2017 8:47:29 AM
ELAPSED TIME	0:00:00.000
STATUS CODE	Finished
STATUS DETAILS	Task output was present in output cache

All the other columns are still defined as features but if we want to change all of those feature names to computer friendly names then we'd need one Edit Metadata module for each column - so is there a better way? Yes - we can just write the code in R or Python. As we have already used Python on the course we'll use that. In the module search 'Python'. Drag the Python module onto the design surface and connect it as shown.



Execute Python Script

Python script

```
# The script MUST contain a function named azureml_main
# which is the entry point for this module.

# imports up here can be used to
import pandas as pd

# The entry point function can contain up to two input arguments:
#   Param<dataframe1>: a pandas.DataFrame
#   Param<dataframe2>: a pandas.DataFrame
def azureml_main(dataframe1 = None, dataframe2 = None):

    # Execution logic goes here
    print('Input pandas.DataFrame #1:\r\n\r\n{0}'.format(dataframe1))

    # If a zip file is connected to the third input port is connected,
    # it is unzipped under ".\Script Bundle". This directory is added
    # to sys.path. Therefore, if your zip file contains a Python file
    # mymodule.py you can import it using:
    # import mymodule

    # Return value must be of a sequence of pandas.DataFrame
    return dataframe1,

```

Python Version: Anaconda 4.0/Python 3.5

Quick Help

Executes a Python script from an Azure Machine Learning experiment
(more help...)

The Python module has three inputs. It can accept two datasets to enable us to code a join across them and we can add our own library on the right-hand input port by zipping it and uploading it. There are also two output ports, one for the data and one for the console output including any graphics. On the right there is a basic code editor which can be configured to work on various versions of Python. If we look at the stub code, dataframe1 is already setup with our data flowing in port 1. Clicking on help on any module will give us more information. The environment already has a lot of python libraries included and the script can import more as needed. The data comes in as Pandas dataframes.
In our case all we need to do is add this line of code

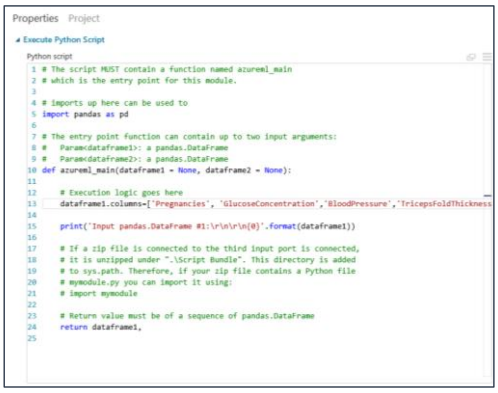
Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt

```
dataframe1.columns=['Pregnancies',
'GlucoseConcentration','BloodPressure','TricepsFoldThickness','SerumInsulin','BMI
','DiabetesPedigree','Age','HasDiabetes']
```

Formatted: Font: Lucida Console, 10 pt, Complex Script Font: 10 pt

Formatted: Font: Lucida Console, 10 pt, Complex Script Font: 10 pt

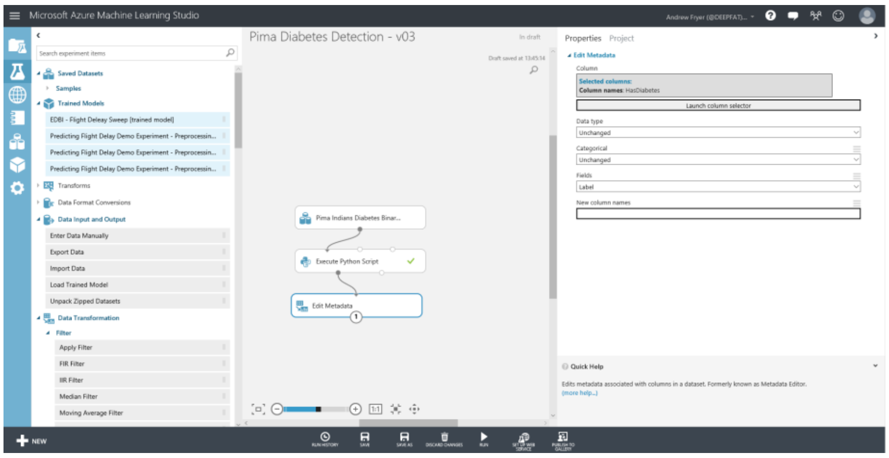
as line 13 just before the print statement



```
1  # The script MUST contain a function named azureml_main
2  # which is the entry point for this module.
3
4  # imports up here can be used to
5  import pandas as pd
6
7  # The entry point function can contain up to two input arguments:
8  #   Param<dataframe1>: a pandas.DataFrame
9  #   Param<dataframe2>: a pandas.DataFrame
10 def azureml_main(dataframe1 = None, dataframe2 = None):
11
12     # Execution logic goes here
13     dataframe1.columns=['Pregnancies', 'GlucoseConcentration','BloodPressure','TricepsFoldThickness
14
15     print('Input pandas.DataFrame #1:\r\n\r\n{0}'.format(dataframe1))
16
17     # If a zip file is connected to the third input port is connected,
18     # it is unzipped under ".\Script Bundle". This directory is added
19     # to sys.path. Therefore, if your zip file contains a Python file
20     # mymodule.py you can import it using:
21     # import mymodule
22
23     # Return value must be of a sequence of pandas.DataFrame
24     return dataframe1,
25
```

To check the code is working we can run the experiment. Click on the Run icon at the bottom of the screen. If all is well we'll see ticks for each step and if we right click on the left output of the Python module we'll see that our columns have been renamed.

Unfortunately, one of the outputs from the Python module loses the fact that we declared the Field as 'Label' earlier, as the data is passed into and out of a pandas dataframe. To fix that we need to re order the modules as shown. You can right-click the arrow connections to delete them, move the modules, and then re-add the connections:



Column
Selected columns:
Column names: HasDiabetes

Data type: Unchanged

Categorical: Unchanged

Fields: Label

New column names:

Edits metadata associated with columns in a dataset. Formerly known as Metadata Editor.

Right click on the Execute Python module and select Run Selected to rerun just this module. This is necessary to get ML studio to recognise the new column names.

Now we'll need to change the Edit Metadata module to recognise the new column name for the label (HasDiabetes). Select the Edit Metadata module and launch the column selector, select 'By Name' on the left-hand side and select HasDiabetes.

Now we are ready to start to build a model.

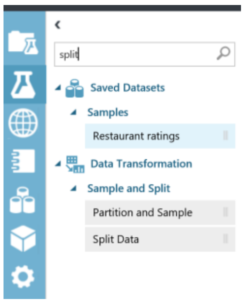
Training the Model

At this point in the experimentation process we are assuming we have a clean set of data that can be used to train and evaluate a model. We are moving from the business knowledge domain to the machine learning domain. We have identified that we are trying to predict the label (now renamed to be HasDiabetes) which can be 1 or 0 (two class classification) against a set of data about Pima Indians.

We will train a model against our data using one of the built-in algorithms in Azure ML. Once the model has been trained it can be used to make predictions about whether a given person is likely to have diabetes that the model has not seen. In order to check how good the model is, we must evaluate the predicted answer against the original value (in the HasDiabetes label column). This is just like any good science experiment - we want a control group to check the accuracy of the model.

To do this we split the data into two random sets and we'll want to use most of the data (typically 80%) for training and reserve 20% for scoring. We'll also want to stratify the split over the label to ensure that each of these separate groups of data have the same ration of values in the label i.e. the train score and evaluate sets of data will have the same ratio of people with and without diabetes.

In Azure ML we do this by using the built-in split module. The easiest way to find this is to type 'split' into the search box at the top of the list modules:



Now drag and drop the 'Split Data' module onto the workspace. Connect this to the output of the 'Edit metadata' module and set the 'fraction of rows in the first output dataset' field to 0.8 (80%).

Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt

Formatted: List Paragraph, Indent: Before: 0.25"

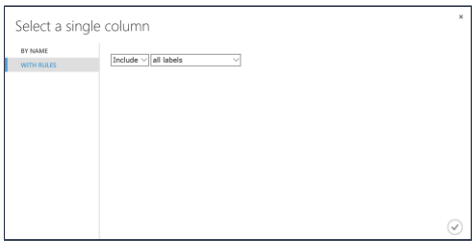
Commented [EB3]: This is a very complex sentence I had to re read several times, I think I know what it says but it could do with a bit of love..

Commented [AF(4R3]: Is that clearer for you?

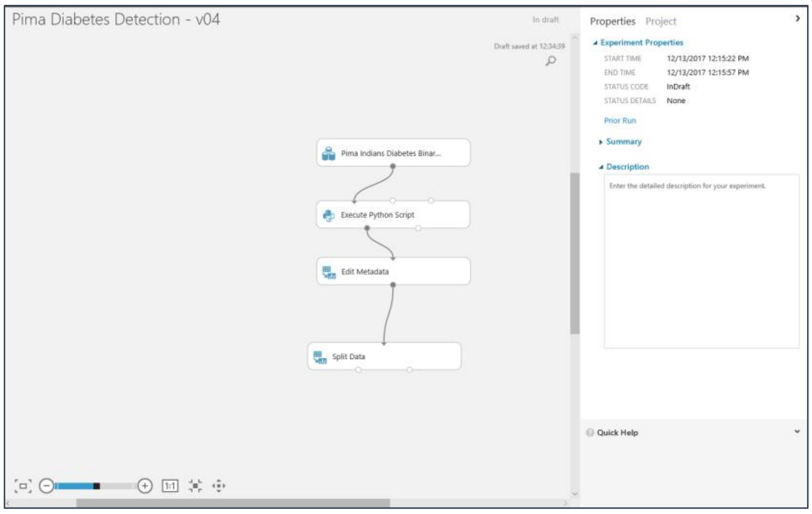
Commented [EB5]: Need to close the dataset first. Also need to either show or tell where to get the split data module from Data trans Sample and Split etc.

Set the 'Stratified Split' option to True and launch the column selector. Select with rules, and then set the dropdown boxes to 'Include' and 'all labels'. Click the checkbox to close the dialog to save changes. This step should appear as shown below:

Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt



The properties of the stratified split should now look like this:



Experiment Properties

START TIME	12/13/2017 12:15:22 PM
END TIME	12/13/2017 12:15:57 PM
STATUS CODE	InDraft
STATUS DETAILS	None

Prior Run

Summary

Description

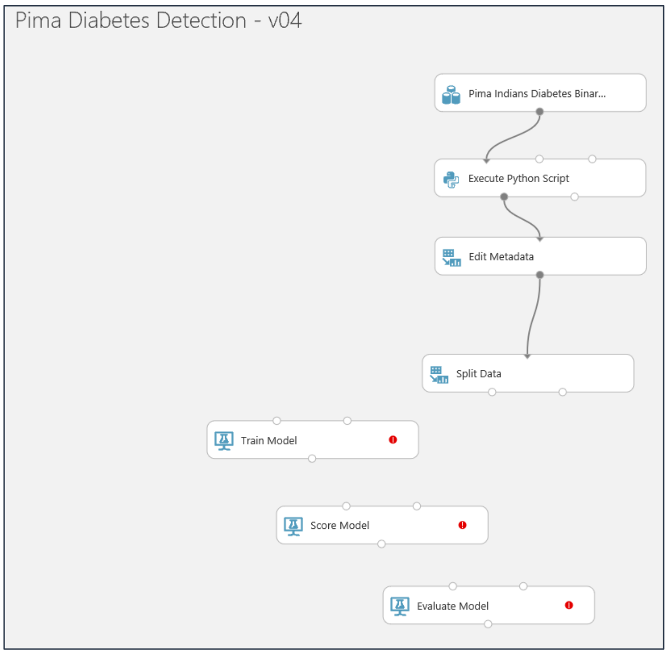
Enter the detailed description for your experiment.

We can check the split is working by rerunning the experiment (run is at the bottom of the screen) and then visualising the output of each of the Split Data ports. Right-click the first output port and select 'Visualize' - observe the number of rows in the data, and so the same for the second output port. We can now check that the mean of each set of data is roughly the same – remember we just need to highlight the HasDiabetes column to see basic stats about it.

Now we can setup the process of training scoring and evaluating a model based on the Cross Industry Standard Process for Data Mining (CRIP-DM).

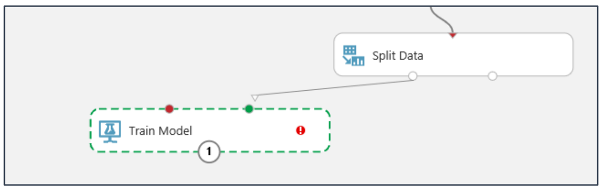
Formatted: Font: (Default) +Body (Calibri), 10 pt, Complex Script Font: +Body CS (Arial), 10 pt

We'll need three modules for this Train Model, Score Model, and Evaluate Model. Find each of these and drag them onto the design surface like this..



Now we can wire these together as follows:

- We'll use 80% of the data for training the model so the left output of Split Data goes to the Train Model. Notice it can only be connected to one of the input ports marked in green



- We'll want to score the output of the trained model so connect the output of Trained Model to the input of Score model – again we can only select one port of Score Model for this
- We'll use the other 20% of the split data to score the model by comparing it with the output of the trained model so connect the other output of split to the remaining port of Score Model module.
- Finally we can use the built in Evaluate Model to show the output of Score Model visually and as a set of statistics. Connect the Output of Score Model to the Left input of Evaluate Model

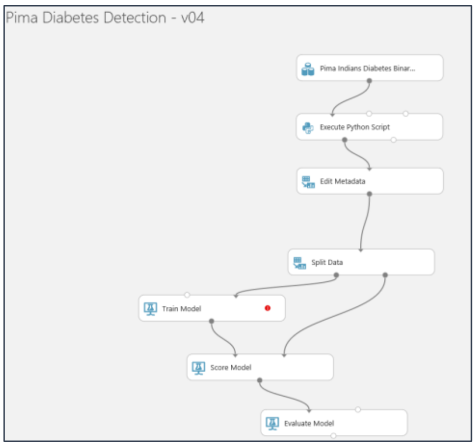
Your diagram should now look like this and there will be a warning on the Train Model module.

Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt

Formatted: List Paragraph, Bulleted + Level: 1 + Aligned at: 0.25" + Indent at: 0.5"

Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt

Formatted: Font: Bold, Complex Script Font: Bold



There are two reasons for this warning:

- We haven't specified what we are trying to predict – unfortunately ML Studio doesn't automatically select the label. So on the properties for Train Model launch the column select, go to 'With Rules' and select 'Include all labels'. Don't worry that there is still a warning...
- Because we haven't specified how to train the model!

Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt

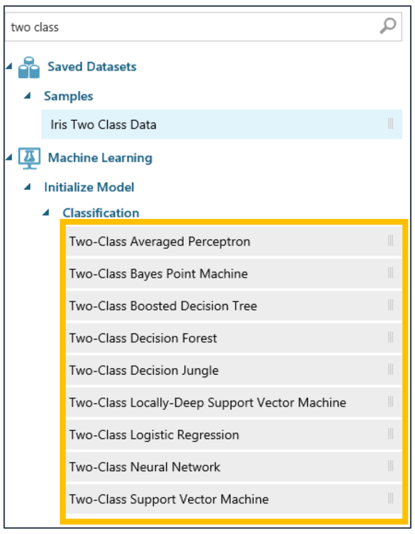
Formatted: List Paragraph, Bulleted + Level: 1 + Aligned at: 0.25" + Indent at: 0.5"

Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt

Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt

Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt

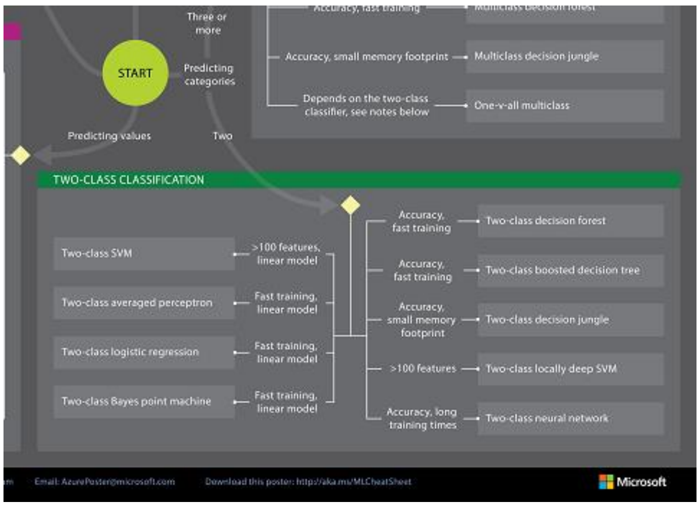
This model is a two class classification problem and if we type 'two class' into the module search we can see that there are several algorithms we can use:



two class
Saved Datasets
Samples
Iris Two Class Data
Machine Learning
Initialize Model
Classification
Two-Class Averaged Perceptron
Two-Class Bayes Point Machine
Two-Class Boosted Decision Tree
Two-Class Decision Forest
Two-Class Decision Jungle
Two-Class Locally-Deep Support Vector Machine
Two-Class Logistic Regression
Two-Class Neural Network
Two-Class Support Vector Machine

Any of these will work but some are better than others in certain circumstances, so which one should we use in this case? There is a cheat sheet in the Azure ML documentation and below is the part of that for two class classification:

Formatted: Font: (Default) +Body (Calibri), 10 pt, Complex Script Font: +Body CS (Arial), 10 pt

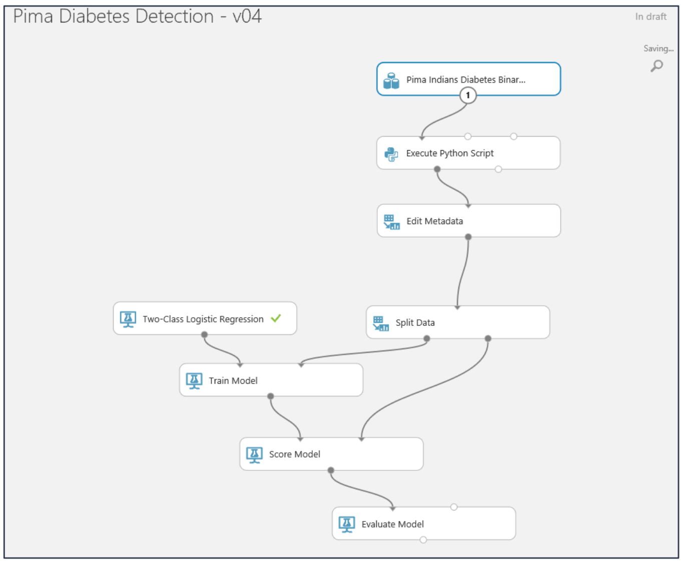


Three or more
START
Predicting categories
Predicting values
Two
Accuracy, fast training
Multiclass decision forest
Accuracy, small memory footprint
Multiclass decision jungle
Depends on the two-class classifier, see notes below
One-v-all multiclass
TWO-CLASS CLASSIFICATION
Two-class SVM
>100 features, linear model
Two-class averaged perceptron
Fast training, linear model
Two-class logistic regression
Fast training, linear model
Two-class Bayes point machine
Fast training, linear model
Accuracy, fast training
Two-class decision forest
Accuracy, fast training
Two-class boosted decision tree
Accuracy, small memory footprint
Two-class decision jungle
>100 features
Two-class locally deep SVM
Accuracy, long training times
Two-class neural network
Email: AzurePoster@microsoft.com
Download this poster: http://aka.ms/MLCheatSheet
Microsoft

As we can see some models are fast, some are designed for accuracy, and others can handle lots of features such as pixels in an image.

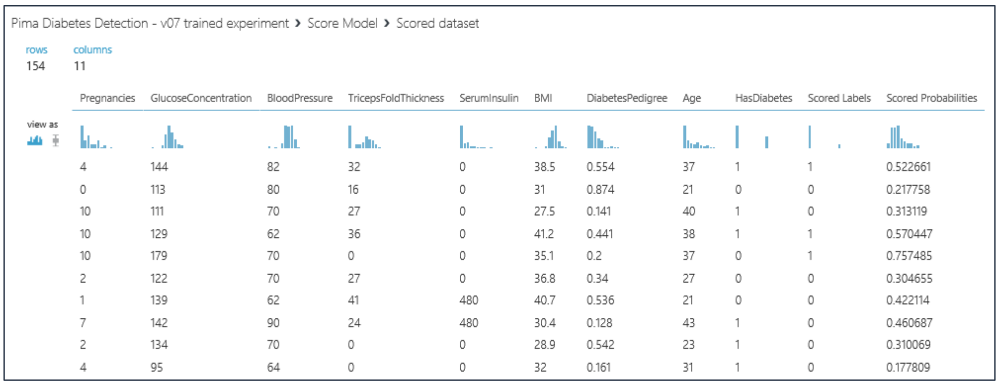
For speed we'll start with Two-Class Logistic Regression. Drag this module onto the design surface above Train Model and connect its output to Train Model..

Formatted: Font: (Default) +Body (Calibri), 10 pt, Complex Script Font: +Body CS (Arial), 10 pt



As there are no longer any red warnings we can run the experiment – this should take a few minutes. (Ask a lab supervisor if this isn't the case!)

Once it has completed we can quickly see how well the model performed. Firstly if we look at the output from the Score Model (right click on the small circle and select Visualise) we can see we have two new columns Scored Label and Scored Probability. We can then see how the Scored Label compares to the HasDiabetes column:

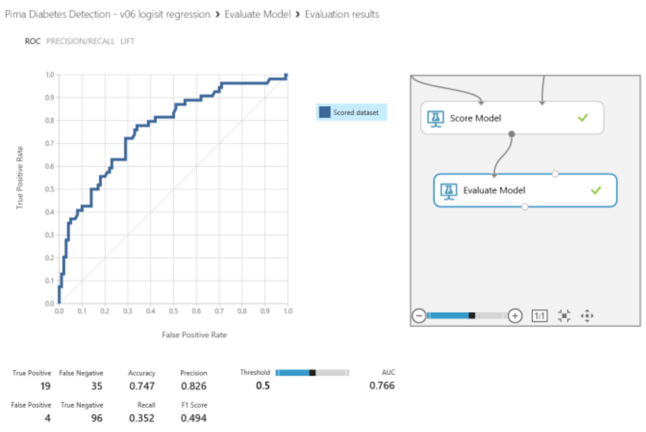


Pima Diabetes Detection - v07 trained experiment › Score Model › Scored dataset

rows 154 columns 11

Pregnancies	GlucoseConcentration	BloodPressure	TricepsFoldThickness	SerumInsulin	BMI	DiabetesPedigree	Age	HasDiabetes	Scored Labels	Scored Probabilities
4	144	82	32	0	38.5	0.554	37	1	1	0.522661
0	113	80	16	0	31	0.874	21	0	0	0.217758
10	111	70	27	0	27.5	0.141	40	1	0	0.313119
10	129	62	36	0	41.2	0.441	38	1	1	0.570447
10	179	70	0	0	35.1	0.2	37	0	1	0.757485
2	122	70	27	0	36.8	0.34	27	0	0	0.304655
1	139	62	41	480	40.7	0.536	21	0	0	0.422114
7	142	90	24	480	30.4	0.128	43	1	0	0.460687
2	134	70	0	0	28.9	0.542	23	1	0	0.310069
4	95	64	0	0	32	0.161	31	1	0	0.177809

However if we want understand the trend across all of the data then we need to visualise the output from the Evaluate Model module:



True Positive	False Negative	Accuracy	Precision	Threshold	AUC
19	35	0.747	0.826	0.5	0.766
False Positive	**True Negative**	**Recall**	**F1 Score**		
4	96	0.352	0.494		

The Receiver Operator Curve above shows us how well the model performs across the data and the greater area under the curve the better the model is – up to a point that is. If the curve is very close to being an upside down 'L' then the model is overfitted or there may be a feature in there that is actually a function of the label. Underneath the curve is the confusion matrix. This grid of numbers is controlled by the threshold setting on the right - if the scored probability of a record is over the threshold value (in this case 0.5) then it is scored as 1.

The calculations for accuracy etc. are as follows:

Accuracy = True Positives + True negatives over the overall total = 0.747

Recall = true positives divided by true positives + false negatives = 0.352

Precision = true positives divided by true positives + false positives = 0.826

F1 score = .2 * (precision* recall)/ (precision + recall) = 0.494

For extra credit you can do the evaluation in Python. Connect another Python module and set it to the SAME VERSION as the earlier Python module (only one version can be used in any one experiment). The code you'll need is this:

Formatted: Complex Script Font: 8 pt, Bold

Formatted: Complex Script Font: 8 pt, Bold

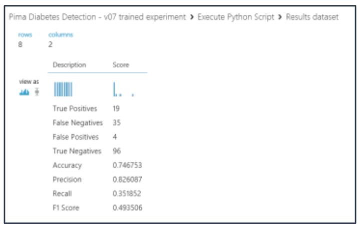
Formatted: Complex Script Font: 8 pt, Bold

Formatted: Complex Script Font: 8 pt, Bold

Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt

```python
import matplotlib
matplotlib.use(‘TkAgg’)
import sklearn.metrics as m
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import confusion_matrix as cm

def azureml_main(dataframe1 = None):
    dataframe1 = dataframe1.dropna()
    # pick the label out of the dataframe and the positive label and plot the RoC curve
    r1 = m.roc_curve(dataframe1[‘HasDiabetes’], dataframe1[‘Scored Probabilities’], pos_label= 1)
    plt.plot(r1[0], r1[1], 'r-', label=‘Boosted Trees’);
    plt.grid(‘on’);
    plt.legend(loc=‘best’)
    plt.savefig(‘roc.png’)
    #Derive test statistics to show the accuracy of the model and output on POrt1
    cmarray =  cm(dataframe1['HasDiabetes'], dataframe1['Scored Labels'])
    TrueNeg, FalsePos = cmarray[0]
    FalseNeg, TruePos = cmarray[1  TruePos = float(TruePos)
    FalseNeg = float(FalseNeg)
    FalsePos= float(FalsePos)
    TrueNeg= float(TrueNeg)

    Accuracy = (TruePos + TrueNeg) /(TruePos+ TrueNeg + FalsePos + FalseNeg)
    Recall = TruePos /(TruePos + FalseNeg)
    Precision = TruePos /(TruePos+ FalsePos)
    F1Score = 2 * (Precision * Recall)/(Precision + Recall)

    data = {'Description': ['True Positives','False Negatives','False Positives','True 
Negatives','Accuracy','Precision','Recall','F1 Score'],'Score': 
[TruePos,FalseNeg,FalsePos,TrueNeg,Accuracy,Precision,Recall,F1Score]}
    dataframe1 =pd.DataFrame(data,columns=['Description','Score'])
    return dataframe1,
```

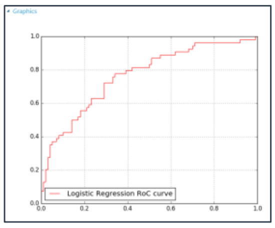
The confusion matrix will show up as the output data of the module and the plot will appear on port 2:



Pima Diabetes Detection - v07 trained experiment › Execute Python Script › Results dataset

rows 8 columns 2

Description	Score
True Positives	19
False Negatives	35
False Positives	4
True Negatives	96
Accuracy	0.746753
Precision	0.826087
Recall	0.351852
F1 Score	0.493506



Publishing a Web Service

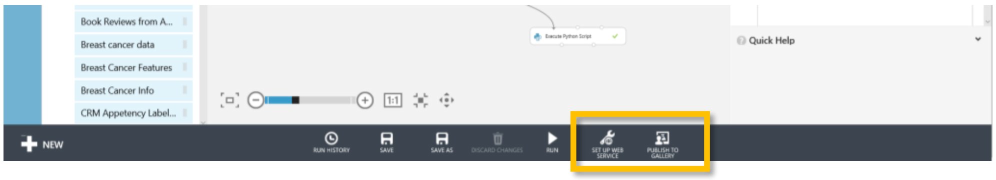
Formatted: Space After: 4 pt

Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt

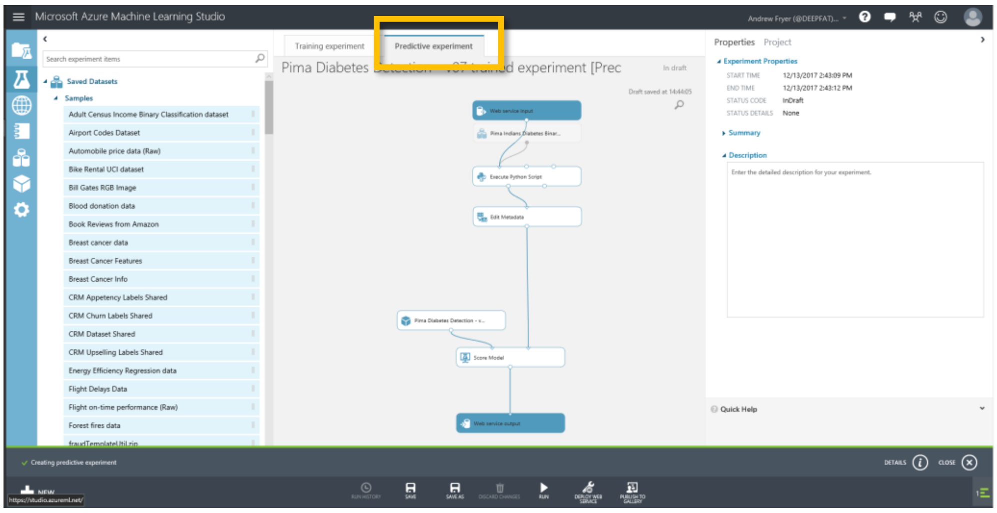
Formatted: All caps

We have trained a model to identify diabetes in Pima Indians and checked to see how it is working. Now we want to use one of Azure ML's key features – operationalising the classifier, by publishing an API to expose the model we have created. In order to do this, we first need to create a Training Experiment. This can then be published to the Azure ML web API service to make it available for other users or applications to use as a web service or a REST endpoint.

This may sound like a hard task, however once your experiment has been run successfully - there are green ticks by each module you will see a button on the toolbar at the bottom of the screen become active:



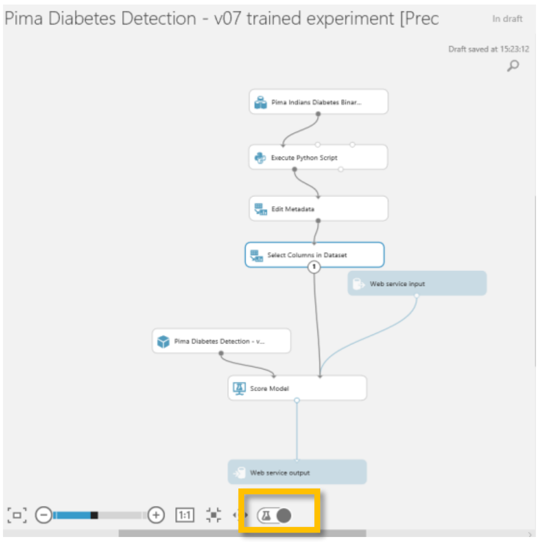
After clicking the 'Predictive Web Service (Recommended)' option our experiment appears to get redrawn and consolidated. What has really happened, is that ML has created a new Predictive experiment tab at the top of the screen.



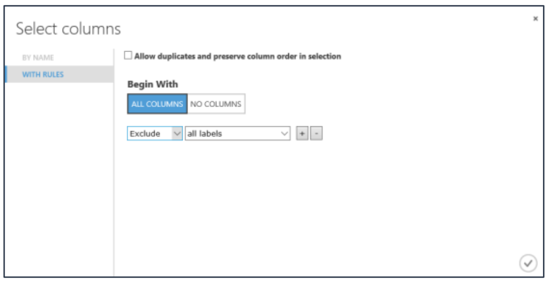
and this is what we are looking at. Our original experiment is still there as it was on the Training experiment tab. Note there are two new modules in blue, "Web service input" and "Web service output" which represent the data format that will flow into and out of the web service we are creating. The Web service input will use the same fields as the input to the module it is connected to, in this case what flows into the top metadata module.

While the wizard has done a basic job of placing where the input and outputs on our predictive experiment are, it is not perfect. The data flowing through 'feed 1' in the above diagram contains our label (HasDiabetes) which is what we are trying to predict. While this is valid for training we shouldn't have it here.

First we flip the toggle switch to experiment mode at the bottom of the experiment. Then we can more clearly see what is going on. Now we can reconnect the Web service input to connect with the score model module and adding a 'Select columns in a Dataset' module as shown:


Pima Diabetes Detection - v07 trained experiment [Prec
In draft
Draft saved at 15:23:12
Pima Indians Diabetes Binar...
Execute Python Script
Edit Metadata
Select Columns in Dataset
Web service input
Pima Diabetes Detection - v...
Score Model
Web service output

set this module to exclude all labels:


Select columns

BY NAME
WITH RULES

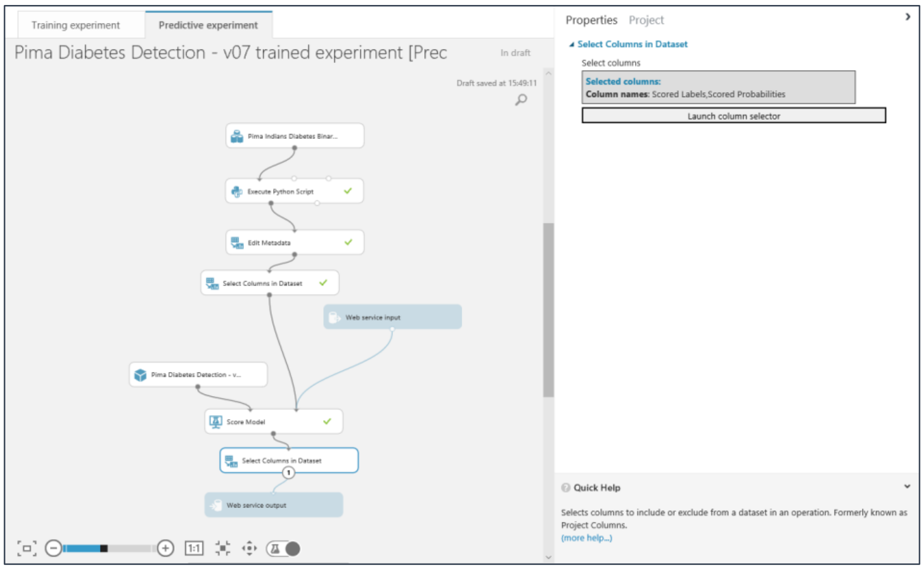
☐ Allow duplicates and preserve column order in selection

Begin With

[✓ ALL COLUMNS] NO COLUMNS

Exclude | all labels

If we now think about what fields we want to return to the application or web site that will call our web service we are creating, then all we need are the scored labels (the prediction) and scored probabilities (the probability of the prediction being true). So we should add another 'Select columns in a Dataset' module as shown to just return those fields (in the columns selector) :



Select Columns in Dataset

Select columns

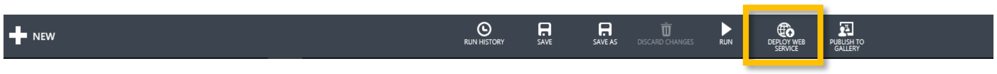
Selected columns:
Column names: Scored Labels,Scored Probabilities

Launch column selector

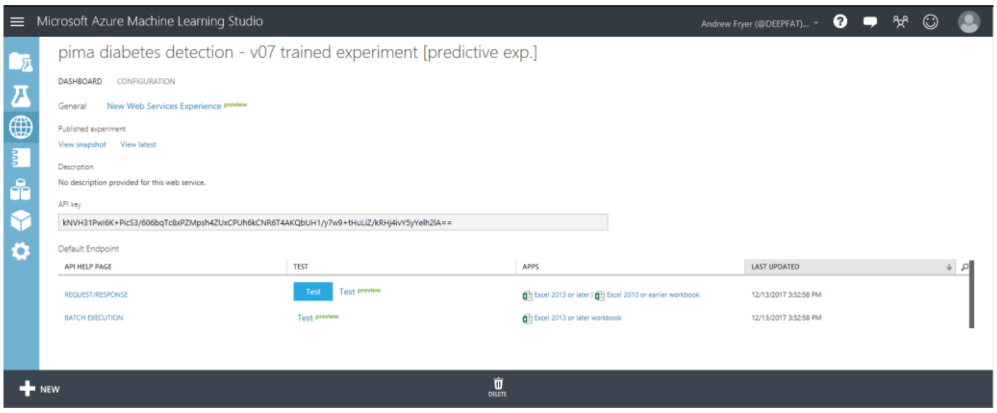
Quick Help

Selects columns to include or exclude from a dataset in an operation. Formerly known as Project Columns.
(more help...)

We must now run this predictive experiment as this allows ML to validate the predictive experiment before we can publish it as web service. After a successful run we'll see that the deploy web service icon is available on the bottom toolbar



All we need to do is click on it and select deploy web service (classic). After a few seconds we'll see we are taken to the web service section of ML studio and our new web service is displayed:


Microsoft Azure Machine Learning Studio
Andrew Fryer (@DEEPFAT)...
DASHBOARD
CONFIGURATION
NEW
DELETE

pima diabetes detection - v07 trained experiment [predictive exp.]

General   New Web Services Experience preview

Published experiment
View snapshot   View latest

Description
No description provided for this web service.

API key
kNVH31Pwl6K+PicS3/606bqTc8xPZMpsh4ZUxCPUh6kCNR6T4AKQbUH1/y7w9+tHuLiZ/kRHj4ivY5yYelh2lA==

Default Endpoint

API HELP PAGE	TEST	APPS	LAST UPDATED
REQUEST/RESPONSE	Test Test preview	Excel 2013 or later | Excel 2010 or earlier workbook	12/13/2017 3:32:58 PM
BATCH EXECUTION	Test preview	Excel 2013 or later workbook	12/13/2017 3:32:58 PM

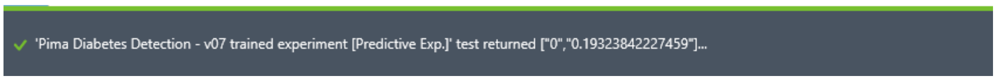
We can see some general information about the WEB API like the API key which is will be used for authorization purposes. There are also hyperlinks to help pages for the two endpoints of the service, a Request/Response endpoint and a Batch Execution endpoint. Notice we are also given some Excel spreadsheets where we can test the service as well.

On the Dashboard tab click on the Test button alongside the REQUEST/RESPONSE API as a quick sense check to see if our web service is working as expected.

Enter the following values in the Enter data to predict dialog:

- Pregnancies = 4
- Glucose Concentration = 95
- BloodPressure = 64
- TricepFoldThickness = 0
- SerumInsulin = 0
- BMI = 32
- DiabetesPedigree = 0.2
- Age = 40

The result you receive back after processing returns a JSON like output.



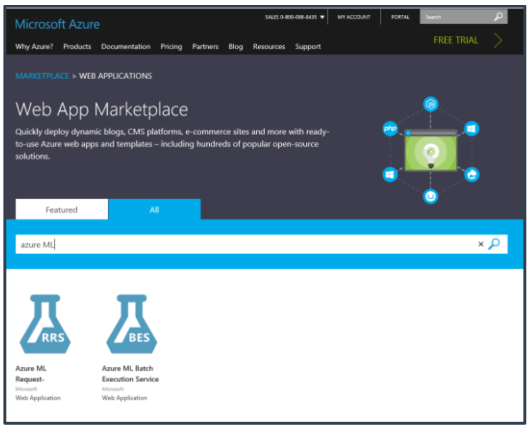
✓ 'Pima Diabetes Detection - v07 trained experiment [Predictive Exp.]' test returned ["0","0.19323842227459"]...

Notice we just get back the Scored Label and the Scored Probabilities. So in the case below we can say that given the parameters above the classifier has predicted the flight will be more than 15 mins delayed (1) and it is very confident in its decision (0.997861325740814).

Optional Exercise – create a Web Site using the api

We can also test our new API using a partially configured web site in the Azure Web App Marketplace. Simply go to the site and look for Azure ML.

This will create a web app in our Azure subscription. You will need an Azure subscription for this.



Formatted: Complex Script Font: Bold

Formatted: Font: (Default) Segoe UI Light, 11 pt, Bold, Complex Script Font: Segoe UI Light, 11 pt, Bold

Formatted: Complex Script Font: Bold

Formatted: Font: (Default) Segoe UI Light, 11 pt, Bold, Complex Script Font: Segoe UI Light, 11 pt, Bold

Formatted: Complex Script Font: Bold

Formatted: Complex Script Font: Bold

Formatted: Font: (Default) Segoe UI Light, 11 pt, Bold, Complex Script Font: Segoe UI Light, 11 pt, Bold

Formatted: Complex Script Font: Bold

Formatted: Complex Script Font: Bold

Formatted: Font: (Default) Segoe UI Light, 11 pt, Bold, Complex Script Font: Segoe UI Light, 11 pt, Bold

Formatted: Complex Script Font: Bold

Formatted: Complex Script Font: Bold

Formatted: Font: (Default) Segoe UI Light, 11 pt, Bold, Complex Script Font: Segoe UI Light, 11 pt, Bold

Formatted: Complex Script Font: Bold

Formatted: Font: (Default) Segoe UI Light, 11 pt, Bold, Complex Script Font: Segoe UI Light, 11 pt, Bold

Formatted: Complex Script Font: Bold

Formatted: Font: (Default) Segoe UI Light, 11 pt, Bold, Complex Script Font: Segoe UI Light, 11 pt, Bold

Formatted: Complex Script Font: Bold

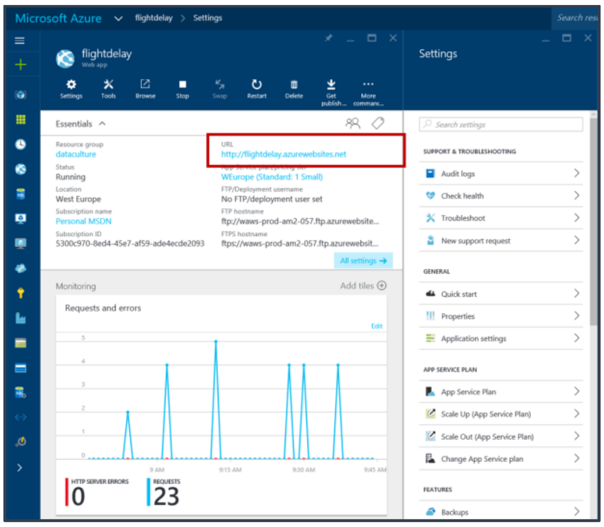
Formatted: Font: (Default) Segoe UI Light, 11 pt, Bold, Complex Script Font: Segoe UI Light, 11 pt, Bold

Formatted: Normal, Indent: Before: 0.04"

Formatted: Font: (Default) Segoe UI Light, 11 pt, Bold, Complex Script Font: Segoe UI Light, 11 pt, Bold

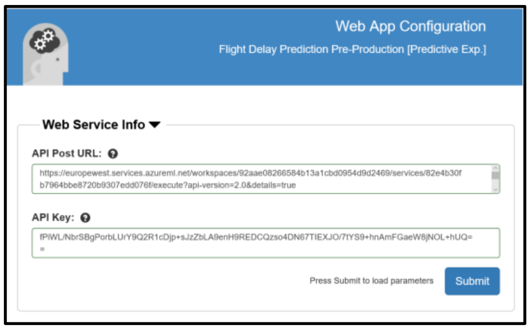
Formatted: Font: 20 pt, Complex Script Font: 20 pt

Formatted: Font: 20 pt, Complex Script Font: 20 pt



Resource group: dataculture
Status: Running
Location: West Europe
Subscription name: Personal MSDN
Subscription ID: 5300c970-8ed4-45e7-af59-ade4ecde2093
URL: http://flightdelay.azurewebsites.net
WEurope (Standard: 1 Small)
FTP/Deployment username: No FTP/deployment user set
FTP hostname: ftp://waws-prod-am2-057.ftp.azurewebsite...
FTPS hostname: ftps://waws-prod-am2-057.ftp.azurewebsit...
HTTP SERVER ERRORS 0
REQUESTS 23

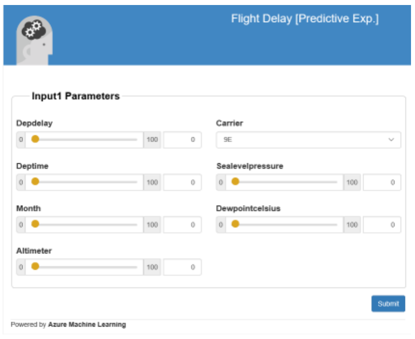
If we then click on the URL of the new site we'll be presented with a simple page where we can enter our API URL and API key.


Web App Configuration
Flight Delay Prediction Pre-Production [Predictive Exp.]
Web Service Info
Press Submit to load parameters
Submit

API Post URL: https://europewest.services.azureml.net/workspaces/92aae08266584b13a1cbd0954d9d2469/services/82e4b30fb7964bbe8720b9307edd076f/execute?api-version=2.0&details=true

API Key: fPiWL/NbrSBgPorbLUrY9Q2R1cDjp+sJzZbLA9enH9REDCQzso4DN67TIEXJO/7tYS9+hnAmFGaeW8jNOL+hUQ==

The key is on the API dashboard and the URL is at the top of the Request response page. Click Submit and close the page. Go back to the Azure Portal and open the site again and enter some trial values..

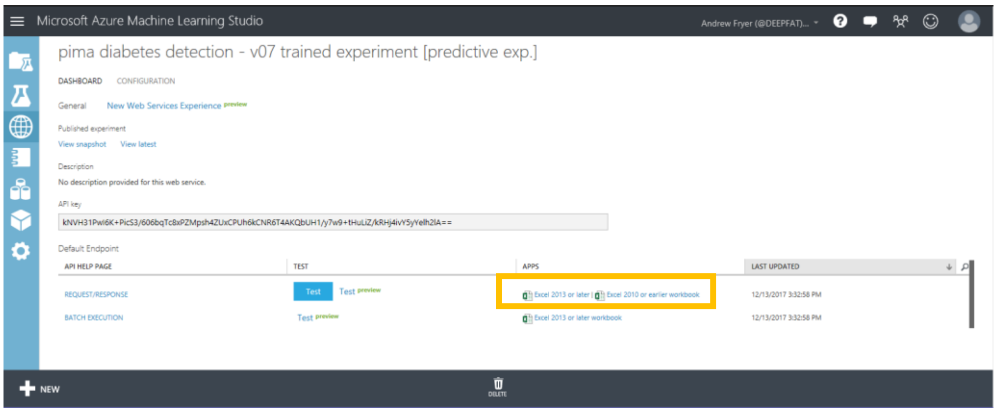

Flight Delay [Predictive Exp.]
Input1 Parameters
Submit
Powered by Azure Machine Learning

Depdelay: 0 – 100, 0
Carrier: 9E
Deptime: 0 – 100, 0
Sealevelpressure: 0 – 100, 0
Month: 0 – 100, 0
Dewpointcelsius: 0 – 100, 0
Altimeter: 0 – 100, 0

Optional Exercise – using Excel to call the API

We can also see how we can interact with the new API from Excel, if you have Excel on your machine. Below is what the Excel 2013 version looks like which uses the new Excel add-ins to automatically setup a connection to the API we have and also allows us to use sample data to test.

From the web services page click on the web service and select the right Excel version for your laptop



Microsoft Azure Machine Learning Studio — Andrew Fryer (@DEEPFAT)...

pima diabetes detection - v07 trained experiment [predictive exp.]

DASHBOARD CONFIGURATION

General New Web Services Experience preview

Published experiment
View snapshot View latest

Description
No description provided for this web service.

API key
kNVH31Pwl6K+PicS3/606bqTc8xPZMpsh4ZUxCPUh6kCNR6T4AKQbUH1/y7w9+tHuLiZ/kRHj4ivY5yYelh2IA==

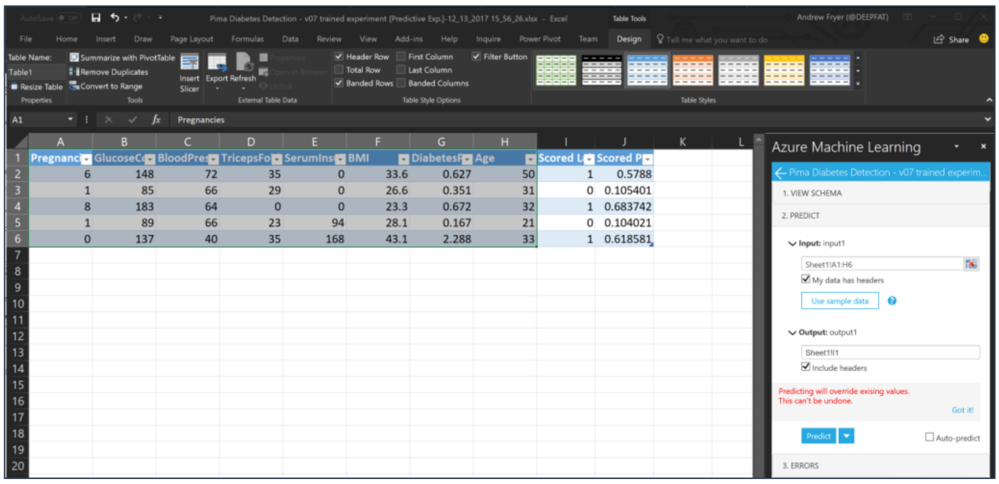
Default Endpoint

API HELP PAGE	TEST	APPS	LAST UPDATED
REQUEST/RESPONSE	Test Test preview	Excel 2013 or later | Excel 2010 or earlier workbook	12/13/2017 3:32:58 PM
BATCH EXECUTION	Test preview	Excel 2013 or later workbook	12/13/2017 3:32:58 PM

Accept the warning and open the spreadsheet once it's downloaded. We need to enable content to load the Azure ML add. Once that appears we can click on the Use sample data box in the Azure Learning pane on the right. Select the sample data as Input rows and I1 as the Output and click Predict. You should see the new columns for Scored Labels and Scored Probabilities like this:

Commented [EB10]: Columns al wrong the data uses columns a-o

And add a new screen shot for the new data



A1 — Pregnancies

	A	B	C	D	E	F	G	H	I	J
1	Pregnanci	GlucoseCo	BloodPres	TricepsFo	SerumIns	BMI	DiabetesP	Age	Scored L	Scored P
2	6	148	72	35	0	33.6	0.627	50	1	0.5788
3	1	85	66	29	0	26.6	0.351	31	0	0.105401
4	8	183	64	0	0	23.3	0.672	32	1	0.683742
5	1	89	66	23	94	28.1	0.167	21	0	0.104021
6	0	137	40	35	168	43.1	2.288	33	1	0.618581

Azure Machine Learning

Pima Diabetes Detection - v07 trained experim...

1. VIEW SCHEMA

2. PREDICT

Input: input1
Sheet1!A1:H6
☑ My data has headers
Use sample data

Output: output1
Sheet1!I1
☑ Include headers

Predicting will override exising values.
This can't be undone.
Got it!

Predict ☐ Auto-predict

3. ERRORS

Now our sample data has the scored labels and scored probabilities against it and we could now enter our own totally unseen data to test the model.

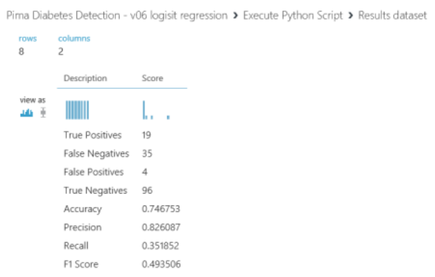
Formatted: Indent: Before: 0"

Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt

Formatted: Justified, Indent: Before: 0"

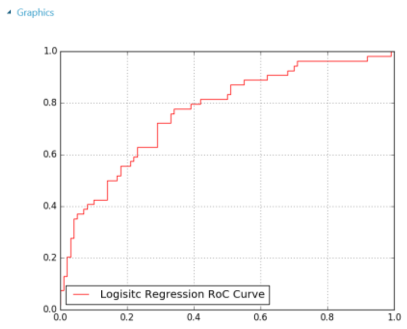
```
    TruePos = float(TruePos)
    FalseNeg = float(FalseNeg)
    FalsePos= float(FalsePos)
    TrueNeg= float(TrueNeg)

    Accuracy = (TruePos + TrueNeg) /(TruePos+ TrueNeg + FalsePos + FalseNeg)
    Recall =   TruePos  /(TruePos + FalseNeg)
    Precision =  TruePos /(TruePos+ FalsePos)
    F1Score = 2 * (Precision * Recall)/(Precision + Recall)

    data = {'Description': ['True Positives','False Negatives','False Positives','True 
Negatives','Accuracy','Precision','Recall','F1 Score'],'Score': 
[TruePos,FalseNeg,FalsePos,TrueNeg,Accuracy,Precision,Recall,F1Score]}
    dataframe1 =pd.DataFrame(data,columns=['Description','Score'])
    return dataframe1,
```

where the confusion matrix will come out of port 1 of the Python module and the plot will be in port 2:


Pima Diabetes Detection - v06 logisit regression › Execute Python Script › Results dataset

rows 8 columns 2

Description	Score
True Positives	19
False Negatives	35
False Positives	4
True Negatives	96
Accuracy	0.746753
Precision	0.826087
Recall	0.351852
F1 Score	0.493506



This is not a very accurate model in that the true positives are very low compared to the errors – where the model predicts a Pima Indian has diabetes where they don't (false positives) and has wrongly identified some Indians as not having diabetes when they do.

Why is the model not doing well and what could we do to improve it.

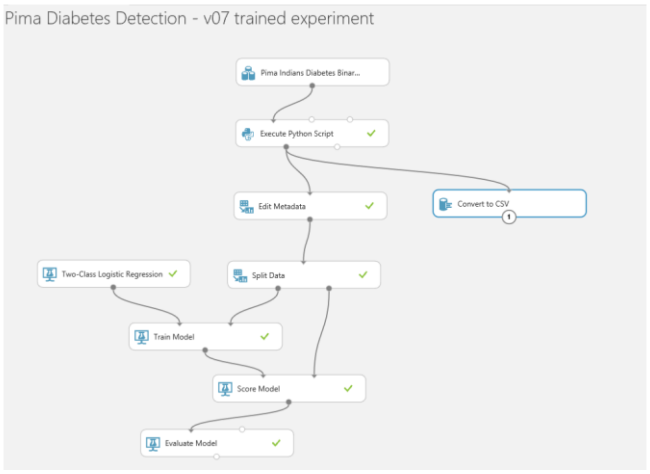
Before we review that let's see how to publish the model assuming it is fit for production.

|

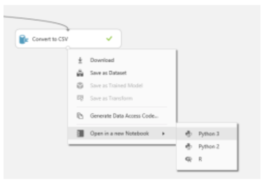
Jupyter Notebooks

Our experiment seems to be quite inaccurate, so should start to investigate the data more closely (in fact we should really have started by doing this!). We saw at the start of this lab that we can get basic statistics just by visualising the data and there are also statistical modules in ML Studio to do this. However, we might wish to bring our own tools to bear and a great way for those with Python or R skills to do this is with Jupyter Notebooks. These are ad hoc scripts which we can attach to our experiments or use on their own.

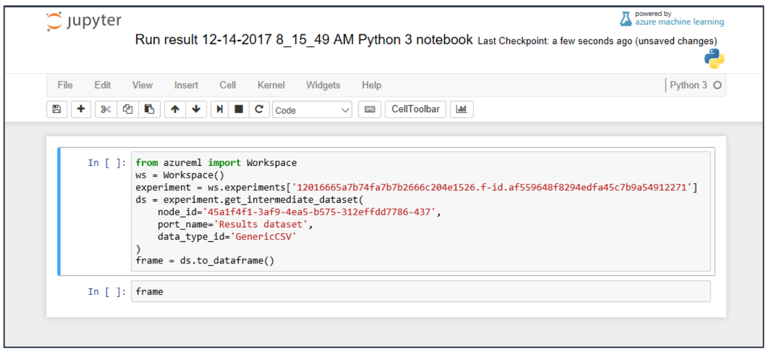
To use these notebooks, we need to extract the data to a format (.csv) that the notebooks can use and this can quickly be done with the Convert to CSV module which we'll connect to the output of the Execute Python Script module as shown:


Pima Diabetes Detection - v07 trained experiment
Pima Indians Diabetes Binar...
Execute Python Script
Edit Metadata
Convert to CSV
Two-Class Logistic Regression
Split Data
Train Model
Score Model
Evaluate Model

Right click on the Export to CSV module and click Run Selected to just run that module. Once it has finished right click again and select Open in a New Notebook -> Python 3



We'll now be taken to a new browser window with our Notebook in:

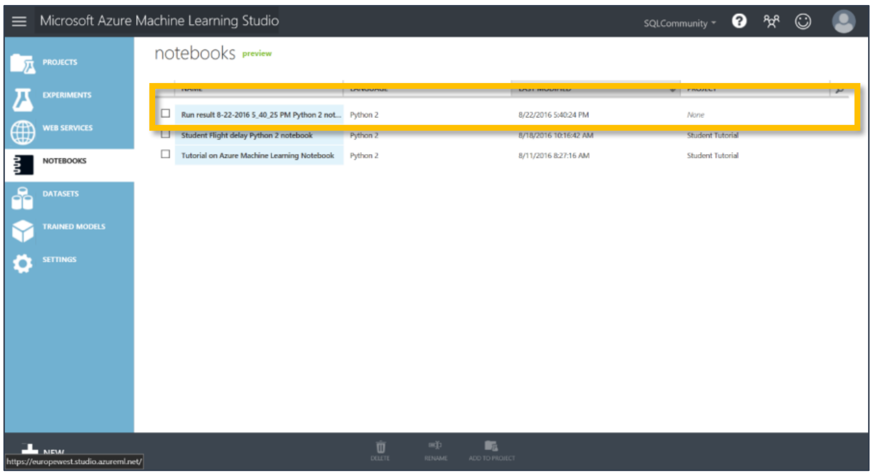


Run result 12-14-2017 8_15_49 AM Python 3 notebook Last Checkpoint: a few seconds ago (unsaved changes)

```
In [ ]: from azureml import Workspace
        ws = Workspace()
        experiment = ws.experiments['12016665a7b74fa7b7b2666c204e1526.f-id.af559648f8294edfa45c7b9a54912271']
        ds = experiment.get_intermediate_dataset(
            node_id='45a1f4f1-3af9-4ea5-b575-312effdd7786-437',
            port_name='Results dataset',
            data_type_id='GenericCSV'
        )
        frame = ds.to_dataframe()

In [ ]: frame
```

Notice we already have a stub script in it which takes our data and loads it into a dataframe called frame. The first thing we should do is rename our notebook to something meaningful as this notebook will persist in our ML workspace. So rename it click on save and confirm this by going back to the browser tab for ML studio and clicking on the Notebook icon on the left (it's not yet renamed in the screenshot below):

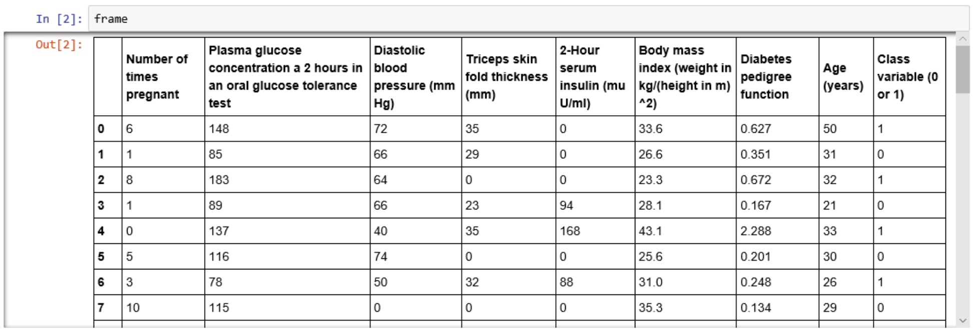
Commented [EB11]: How?) I know you click the title and change it but it took me a while…) o rclick file --- rename



NAME	LANGUAGE	LAST MODIFIED	PROJECT
Run result 8-22-2016 5_40_25 PM Python 2 not...	Python 2	8/22/2016 5:40:24 PM	None
Student Flight delay Python 2 notebook	Python 2	8/18/2016 10:16:42 AM	Student Tutorial
Tutorial on Azure Machine Learning Notebook	Python 2	8/11/2016 8:27:16 AM	Student Tutorial

We can now go back to the Notebook tab and add some of our own code. Firstly, we need to run the cell with the existing code in to load the dataframe with data from our experiment which we do by setting focus on the cell and clicking on the run ▶ icon in the top toolbar.

In the second cell there is just the command `frame` and if we run this cell we'll see a sample of our data:



```
In [2]: frame
Out[2]:
```

	Number of times pregnant	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	Diastolic blood pressure (mm Hg)	Triceps skin fold thickness (mm)	2-Hour serum insulin (mu U/ml)	Body mass index (weight in kg/(height in m)^2)	Diabetes pedigree function	Age (years)	Class variable (0 or 1)
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0

What we can do now is use any of the many libraries and function in Python to analyse this data without any further effort. In this case let's see how the various features correlate with each other. This plot only works against numerical data and fortunately all our data is numeric. Add a new cell to the notebook (with the **+** sign) and enter this:

```
%matplotlib inline
import pip
import pandas
import numpy as np
import pandas as pd
import matplotlib
matplotlib.use("agg")
import matplotlib.pyplot as plt
```

#Now we have these libraries we can create the plot itself:

```
cm=correlationFrame.corr()
fig=plt.figure()
plt.imshow(cm,interpolation='nearest')
plt.xticks(list(range(0,len(cm.columns))),list(cm.columns.values), rotation=90)
plt.yticks(list(range(0,len(cm.columns))),list(cm.columns.values))
plt.colorbar()
```

If we run this cell we'll get a plot like this:

Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt

Formatted: Font: Bold, Complex Script Font: Bold

Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt

Formatted: Font: (Default) Lucida Console, Font color: Purple, Complex Script Font: Segoe UI Light

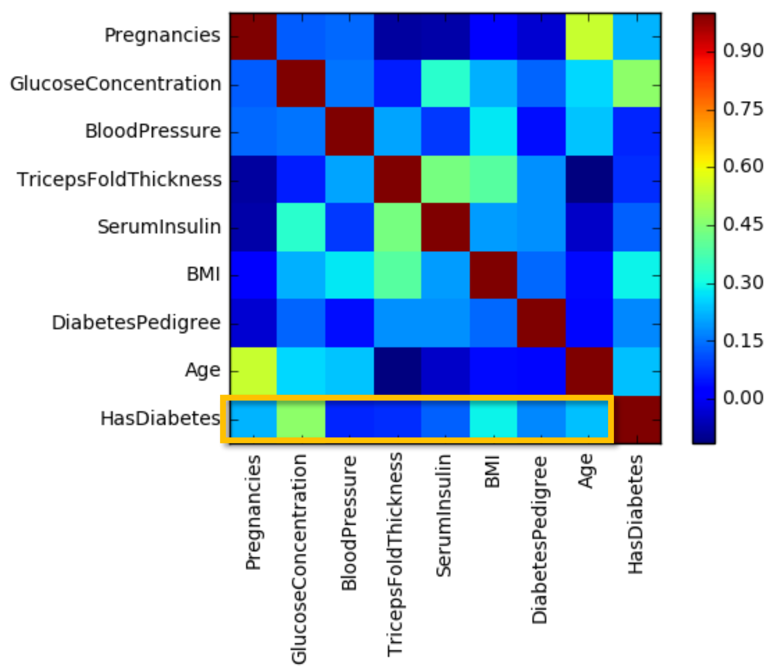
Formatted: Font: (Default) Lucida Console, Font color: Accent 6, Complex Script Font: Segoe UI Light

Commented [EB12]: Warning received

```
/home/nbcommon/env/lib/python2.7/site-packages/matplotlib/__init__.py:1350: UserWarning:  This call to matplotlib.use() has no effect
because the backend has already been chosen;
matplotlib.use() must be called *before* pylab, matplotlib.pyplot,
or matplotlib.backends is imported for the first time.

  warnings.warn(_use_error_msg)
```

Commented [EB13]: Typo removed extra a

```
Out[5]: <matplotlib.colorbar.Colorbar at 0x7f5eaeeb70f0>
```



This shows us is how the various columns in our dataset are correlated. For example the correlation on the leading diagonal is always one because this simply a comparison of a column with itself.

What we are interested is how each of these columns correlates with HasDiabetes (the yellow box) and here we can see there isn't a strong correlation between any of the features and the label, the strongest being GlucoseConcentration and the weakest BloodPressure and TricepFoldThickness.

What this does show is that is a simply matter to use Jupyter notebooks against any step in our experiments to gain a deeper insight into our data

Formatted: Normal, No bullets or numbering

Formatted: Font: (Default) Segoe UI Light, 11 pt, Complex Script Font: Segoe UI Light, 11 pt

Conclusion

This lab was intended to introduce you to the basic concepts of Machine Learning such as binary classification, feature selection, training and testing a model and using Azure Machine Learning. A web service was created to operationalize and deploy the model for production.

Next Steps:

- Check out the UK Data Developer page on MSDN here:
- Check out the Azure ML Gallery and download and edit/run other types of machine learning experiments: https://gallery.azureml.net/
- Also check out other Azure Services that can be used with Machine Learning such as HDInsight, Data Factory and Stream Analytics: http://azure.microsoft.com/en-us/services/

Formatted: Complex Script Font: Bold

Formatted: Font: (Default) Segoe UI Light, 11 pt, Bold, Complex Script Font: Segoe UI Light, 11 pt, Bold