# Chapter 11: Model deployment

## Introduction

This lesson includes an introduction to model deployment and highlights Altair SmartWorks Analytics Hub as a means to easily deploy program code for on-demand, scheduled and real time applications.

Figure 1: Contents

A screenshot of a computer

Description automatically generated

Model deployment is colloquially referred to as the last mile and sometimes this can seemingly be made longer when complexities such as data sources required, derivations applied, system requirements and model code format are taken into consideration.

Figure 2: Introduction

A screenshot of a computer

Description automatically generated

When developing models, these aspects should be taken into consideration to smooth the process and make model deployment straightforward.

Altair Analytics Workbench provides the facility to access model code for any created model. Code is output predominantly in either SAS or SQL format and can subsequently be applied as necessary.

Additionally, the score node can be used to score data and results can be exported using an appropriate Export group block.

ALTAIR provides an additional facility to assist in deployment, this is called Altair SmartWorks Analytics Hub.

Figure 3: Altair SmartWorks Analytics Hub

A screenshot of a computer

Description automatically generated

Altair Analytics Workbench Hub is accessed via browser and provides the facility to govern users, data sources, and deploy programs written in the languages of SAS, Python, R and SQL or a combination of these as APIs for use in real time, on-demand and scheduled processes.

Altair SmartWorks Analytics Hub will be used to illustrate the deployment of a SAS language program as an API for a real-time application.

## Demonstration

So, let’s demonstrate. This demonstration uses a project developed in a previous lesson. As can be seen, a decision tree, logistic regression and a decision forest model were developed.

Model code is available for each of these models from the Scoring code tab of the respective

Model reports.

Accessing the code for the decision tree it can be seen that the model code is available in either the language of SAS or SQL. The code creates two variables *P\_bad* and *P\_good* and assigns propensity scores to each observation based on the leaf node they are members of.

The code for any model code can be easily copied and used in a program file in the SAS Language perspective or in a code block in the Workflow perspective to score data.

The Score block can also be used to score data with any model. Here a new dataset, *Risk\_to\_score.csv*, is added to the Workflow. Opening this dataset shows that it contains 19024 observations and 12 variables, and some of these were used to build the decision tree model.

Connecting the decision tree and this data to the Score block automatically outputs a scored dataset.

Opening the dataset with the data profiler shows that two new variables have been added: *P\_bad* and *P\_good*.

The scored data can easily be exported to a desired destination format using an appropriate export block available from the Export group.

To illustrate, an Excel Export block is used to export the scored results to an Excel file. The scored file is output to the project folder for convenience and named *scored\_dt.*

Once complete the scored dataset is accessed and notice the new scored variables added. The scored dataset contains propensities scores and this may be all that is warranted.

To apply strategies based on the resulting propensities and include an outcome can also be accomplished.

## Adding Outcomes

To put this into perspective let’s say there is interest in assigning an outcome of either *accepted* or *rejected* based on the propensity contained in the variable *P\_bad* exceeding a threshold.

The threshold can be determined in a number of different ways but here let’s use an arbitrary cut-off of .5. An observation will be accepted if *P\_bad* is less than .5 and rejected otherwise.

This can easily be accomplished with a SAS Code block. Once dragged onto the Workflow canvas and the scored dataset connected the SAS Code block can be accessed.

A new dataset is added and its named changed to *scored\_strat*. Some code is added that assigns the value *accepted* to a variable named *Outcome* based on the variable *P\_bad* exhibiting a value less than 0.5, otherwise a value of *rejected* is assigned.

**data** &Output\_1;

set &Input\_1;

if P\_bad < 0.5 then outcome = 'accepted';

else outcome = 'rejected';

**run**;

Viewing the distribution of the resulting variable *outcome* from the Univariate Charts tab of the Data Profiler it can be seen that approximately 85% of the observations in this dataset have been assigned the value *accepted* and approximately 15% assigned the value *rejected.*

Even though a simple strategy was used here based on a simple cut-off, more complexity can be included in a similar way.

## Deploying programs as APIs using Altair SmartWorks Analytics Hub

Program code written in the languages of SAS, R, Python and SQL or a combination of these languages can be deployed as an API using Altair SmartWorks Analytics Hub.

To illustrate this deployment functionality, a scorecard model that assesses whether an applicant will be accepted or rejected will be used to demonstrate.

The scorecard includes four variables: *age*, *education\_num*, *relationship* and *sex* and as can be seen scores are assigned based on characteristics and summed to output a total score.

Based on the total score, the applicant is either accepted or rejected and there is interest in automating this process whereby applicants are directed to a webpage where they can supply their details and get a decision in real time.

The development, configuration, testing and uploading of programs to Altair SmartWorks Analytics Hub to deploy as APIs can be conducted solely from Altair Analytics Workbench.

The first step is to create a new Altair SmartWorks Analytics Hub project. Here a new project named scorecard\_ is created. This type of project is specific to the deployment of programs as APIs.

To deploy a program as an API requires two elements: the program code and the definition of the API endpoints. Adding any program to this type of project includes tabs to add code via the SAS Language Code tab and also configure the APIs endpoints via the Hub Configuration tab.

Once the program has been added the code for the scorecard is copied and pasted into the SAS Language Code page.

The scorecard assigns points based on variable characteristics and sums results. Rather than an applicant being faced with a score, some additional code is included to output a message based on the score. If the score exceeds the threshold of 550 a message contained in a variable called Decision is returned congratulating the application on being approved and if not, a message requesting the applicant call is returned.

Finally, the code is wrapped in a datastep. Here, the dataset will be called output, and the only detail that is retained is the response, contained in the variable 'decision', a *keep* statement is used to retain this variable only.

The API endpoints which are simply the inputs the program accepts, in this case values for the variables *age*, *education*, *relationship* and *sex* and the output returned are easily configured from the Hub Configuration tab.

First, the program path is supplied. It is advised to assign a value here that corresponds with the name of the program file and here scorecard\_ is assigned.

The input parameters to this API will be supplied as Macro variables and this is selected from the Parameter style drop-down from the Parameters pane.

Clicking the Add button enables the definition of each input parameter with the ability to assign a default value among other characteristics and here defaults are supplied for all parameters.

From the Results pane, output is added by clicking Add. The output type is selected as a dataset and its name corresponds with that used in the code, here, output.

Some of the results details are populated from the results parameter definition. Additional details include providing a dataset name and the results format. The dataset name used here corresponds with the name supplied previously, and the results format is selected as JSON, but note other options are available.

Once complete clicking CTRL+S saves the configuration. The API can be selected and tested from the Workflow perspective by right-clicking the API and selecting either Run Program or Run Program With Parameters.

Here, Run Program With Parameters is selected and a dialog is displayed that provides the ability to supply parameters, notice that the defaults supplied earlier are used and values are automatically populated.

Clicking Run executes the program and a message displays stating:

The result from running a Hub API Program are shown in the ‘Hub API Executions’ view. Would you like to open this view now?

Clicking Yes opens this view with details in relation to execution accessible. Clicking the output shows that a congratulatory message has been returned and this reflects that the program is successful and configured correctly.

Note also that a log is generated and this can be referenced in the event the program does not run successfully.

The next steps is to relay the API to Altair SmartWorks Analytics Hub. This requires that the Hub is logged into and this can be accomplished by selecting ALTAIR > Hub > Log In.

Once logged, the project can be uploaded to Hub by right-clicking the project and selecting the option: Upload to Hub… and providing details such as the Repository and Group the API should be saved to and its Name and Version.

Once OK is clicked, a message displays relaying that the upload has been successful. Once this is complete, the program can be deployed, and its API URL accessed and used as necessary.

This is accomplished by logging in to Altair SmartWorks Analytics Hub via browser and selecting the tile: Deployment Services.

From Deployment, New is clicked and the Deployment type selected as Artefact the uploaded program can be easily selected and using simple click steps the program can be deployed to any configured environment – note that environments are installed at the point of initially configuring Altair SmartWorks Analytics Hub, these are machines that house the APIs and from which they can be called.

Once these steps are complete the environment is published and the API is now available from the directory, where its URL is accessible.

Note that the directory provides the facility to test the API via the Execute tab and clicking Run. Here, as is already understood the API is functioning correctly.

The crudest method of calling the API is to pass the variable values along with the call to the API. Using this in a browser, it can be seen that a response is successfully received.

A much more elegant way to deploy this is to create a web page to accept the model input values and return a response once submitted and this is the method employed here.

The web page is programmed to take the input values, pass these along with a call to the API and display a response.

Note that the scorecard can be easily updated by changing the code, here, a shorter message is used. Deploying an updated version is a matter of click steps to save results, upload to Altair SmartWorks Analytics Hub and publish to the environment used.

Returning to the web page and entering the same values used previously and clicking Submit returned the updated response.

Altair SmartWorks Analytics Hub not only provides the facility to call deployed programs via web service but also from Microsoft Excel using the Altair SmartWorks Analytics Hub Excel plugin.

The program deployed previously was modified to accept an input dataset to batch score observations with the scorecard and outputs a score and a decision.

This program can be called via Microsoft Excel from the Altair SmartWorks Analytics Hub tab. Once logged to Altair SmartWorks Analytics Hub, all programs contained there are accessible. Here the scorecard is called and the data to score supplied via familiar Microsoft Excel dialog.

The existing worksheet is selected as The Results Location and an appropriate cell selected to display results.

Clicking Run calls the program and returns, in this instance a score and a decision to the Microsoft Excel spreadsheet.

Note that these are two examples of the ability to deploy programs as APIs using Altair Analytics Workbench and a myriad of other possibilities can be addressed.

It should suffice to say that any process, whether that be for etl, real-time scoring, on-demand, batch scoring or scheduling processing, as long as the process can be coded in the language of SAS, R, Python, SQL or a combination of these, it can be deployed as an API using Altair Analytics Workbench.

## Summary

This lesson included an introduction to model deployment and highlighted Altair SmartWorks Analytics Hub as a means to easily deploy programs as APIs.