Data Science Essentials

Lab 6 – Introduction to Machine Learning

Overview

In this lab, you will use Azure Machine Learning to train, evaluate, and publish a classification model, a regression model, and a clustering model. The point of this lab is to introduce you to the basics of creating machine learning models in Azure ML, it is not intended to be a deep-dive into model design, validation and improvement.

Note: This lab builds on knowledge and skills developed in previous labs in this course.

What You'll Need

To complete this lab, you will need the following:

- An Azure ML account
- A web browser and Internet connection
- The files for this lab

Note: To set up the required environment for the lab, follow the instructions in the Setup Guide for this course.

Implementing a Classification Model

In this exercise, you will create a two-class classification model that predicts whether or not a person earns over $50,000 based on demographic features from an adult census dataset.

Demographic data provides information on a population of people. Classifying or segmenting demographic data is useful in several applications including marketing and political science. Segments in a population can be classified using various characteristics, or features, including income, education and location.

Prepare the Data

The source data for the classification model you will create is provided as a sample dataset in Azure ML. before you can use it to train a classification model you must prepare the data using some of the techniques you have learned previously in this course.

1. Open a browser and browse to https://studio.azureml.net. Then sign in using the Microsoft account associated with your Azure ML account.
2. Create a new blank experiment and name it **Adult Income Classification**.

3. In the **Adult Income Classification** experiment, drag the **Adult Census Income Binary Classification** sample dataset to the canvas.
4. Visualize the output of the dataset, and review the data it contains. Note that the dataset contains the following variables:
    - **age**: A numeric feature representing the age of the census respondent.
    - **workclass**: A string feature representing the type of employment of the census respondent.
    - **fnlwgt**: A numeric feature representing the weighting of this record from the census sample when applied to the total population.
    - **education**: A string feature representing the highest level of education attained by the census respondent.
    - **education-num**: A numeric feature representing the highest level of education attained by the census respondent.
    - **marital-status**: A string feature indicating the marital status of the census respondent.
    - **occupation**: A string feature representing the occupation of the census respondent.
    - **relationship**: A categorical feature indicating the family relationship role of the census respondent.
    - **race**: A string feature indicating the ethnicity of the census respondent.
    - **sex**: A categorical feature indicating the gender of the census respondent.
    - **capital-gain**: A numeric feature indicating the capital gains realized by the census respondent.
    - **capital-loss**: A numeric feature indicating the capital losses incurred by the census respondent.
    - **hours-per-week**: A numeric feature indicating the number of hours worked per week by the census respondent.
    - **native-country**: A string feature indicating the nationality of the census respondent.
    - **income**: A label indicating whether the census respondent earns $50,000 or less, or more than $50,000.

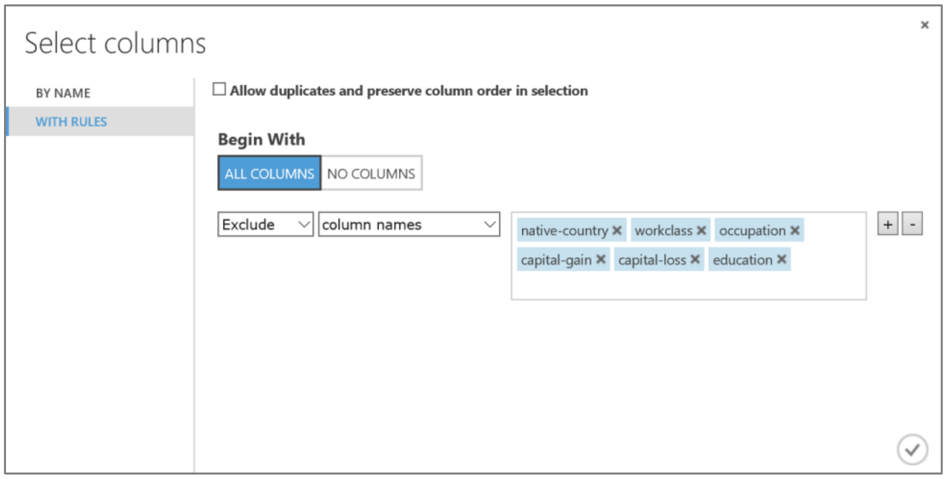
    Note: Before training a classification model on any dataset, it must be properly prepared so that the classification algorithm can work effectively with the data. In most real scenarios, you would need to explore the data and perform some data cleansing and transformation to prepare the data using the techniques described in the previous modules of this course; but for the purposes of this lab, the data exploration tasks have already been performed for you in order to determine the data transformations that are required. Specifically, some columns have been identified as redundant or not predictively useful, numeric values in the dataset must be scaled*, and string values must be converted to categorical features.

    *In this lab you will use the logistic regression model to train the classification model. Logistic regression is a linear computationally efficient method, widely employed by data scientists. The algorithm requires all numeric features to be on a similar scale. If features are not on a similar scale, those with a large numeric range will dominate the training of the model.

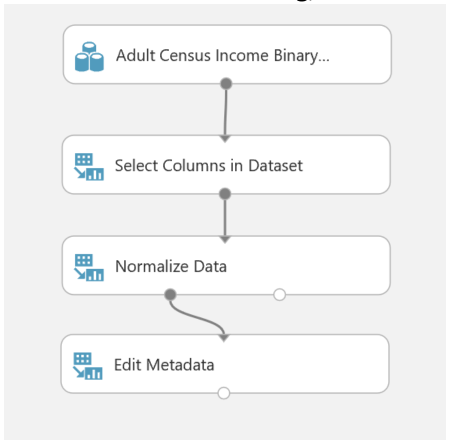
5. Add a **Select Columns in Dataset** module to the experiment, and connect the output of the dataset to its input.
6. Select the **Select Columns in Dataset** module, and in the **Properties** pane launch the column selector. Then use the column selector to exclude the following columns:
    - workclass

- education
- occupation
- capital-gain
- capital-loss
- native-country

You can use the **With Rules** page of the column selector to accomplish this as shown here:



7. Add a **Normalize Data** module to the experiment and connect the output of the **Select Columns in Dataset** module to its input.
8. Set the properties of the **Normalize Data** module as follows:
   - **Transformation method**: MinMax
   - **Use 0 for constant columns**: Unselected
   - **Columns to transform**: All numeric columns
9. Add an **Edit Metadata** module to the experiment, and connect the **Transformed dataset** (left) output of the **Normalize Data** module to its input.
10. Set the properties of the **Edit Metadata** module as follows:
    - **Column**: All string columns
    - **Data type**: Unchanged
    - **Categorical**: Make categorical
    - **Fields**: Unchanged
    - **New column names**: *Leave blank*
11. Verify that your experiment looks like the following, and then save and run the experiment:

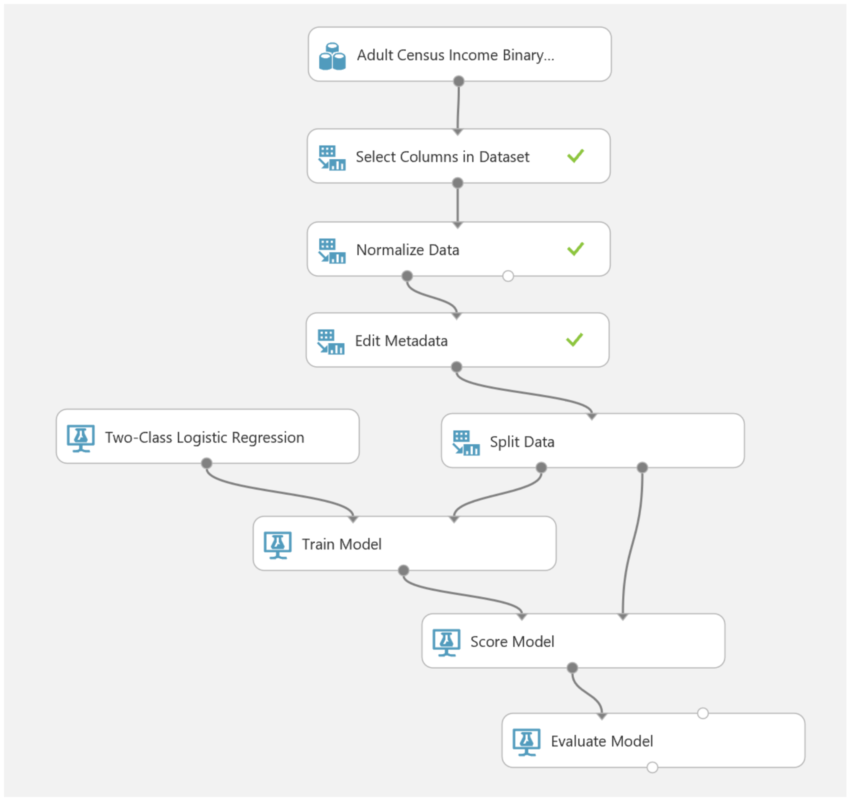


12. When the experiment has finished running, visualize the output of the **Edit Metadata** module and verify that:
    - The columns you specified have been removed.
    - All numeric columns now contain a scaled value between 0 and 1.
    - All string columns now have a **Feature Type** of *Categorical Feature*.

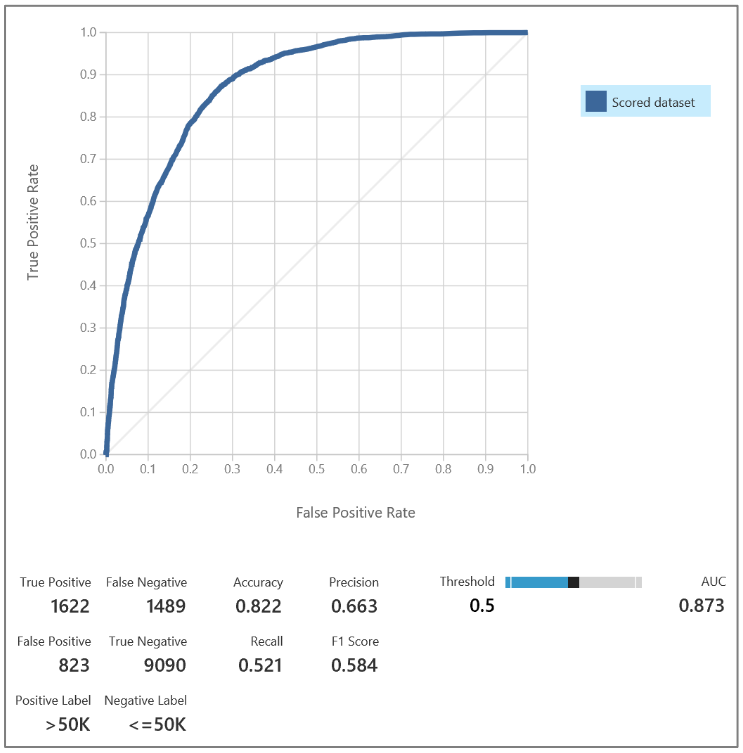
Create and Evaluate a Classification Model

Now that you have prepared the data, you will construct and evaluate a classification model. The goal of this model is to classify people by income level: low (**<=50K**) or high (**>50K**).

1. Add a **Split Data** module to the **Adult Income Classification** experiment, and connect the output of the **Edit Metadata** module to the input of the **Split Data** module. You will use this module to split the data into separate training and test datasets.
2. Set the properties of the **Split Data** module as follows:
    - **Splitting mode**: Split Rows
    - **Fraction of rows in the first output dataset**: 0.6
    - **Randomized split**: Checked
    - **Random seed**: 123
    - **Stratified split**: False
3. Add a **Train Model** module to the experiment, and connect the **Results dataset1** (left) output of the **Split Data** module to the **Dataset** (right) input of the **Train Model** module.
4. In the **Properties** pane for the **Train Model** module, use the column selector to select the **income** column. This sets the label column that the classification model will be trained to predict.
5. Add a **Two Class Logistic Regression** module to the experiment, and connect the output of the **Two Class Logistic Regression** module to the **Untrained model** (left) input of the **Train Model** module. This specifies that the classification model will be trained using the two-class logistic regression algorithm.
6. Set the properties of the **Two Class Logistic Regression** module as follows:
    - **Create trainer mode**: Single Parameter
    - **Optimization tolerance**: 1E-07
    - **L1 regularization weight**: 0.001
    - **L2 regularization weight**: 0.001
    - **Memory size for L-BFGS**: 20
    - **Random number seed**: 123
    - **Allow unknown categorical levels**: Checked
7. Add a **Score Model** module to the experiment. Then connect the output of the **Train Model** module to the **Trained model** (left) input of the **Score Model** module, and connect the **Results dataset2** (right) output of the **Split Data** module to the **Dataset** (right) input of the **Score Model** module.
8. On the **Properties** pane for the **Score Model** module, ensure that the **Append score columns to output** checkbox is selected.
9. Add an **Evaluate Model** module to the experiment, and connect the output of the **Score model** module to the **Scored dataset** (left) input of the **Evaluate Model** module.
10. Verify that your experiment resembles the figure below, then save and run the experiment.


Adult Census Income Binary...
Select Columns in Dataset
Normalize Data
Edit Metadata
Two-Class Logistic Regression
Split Data
Train Model
Score Model
Evaluate Model

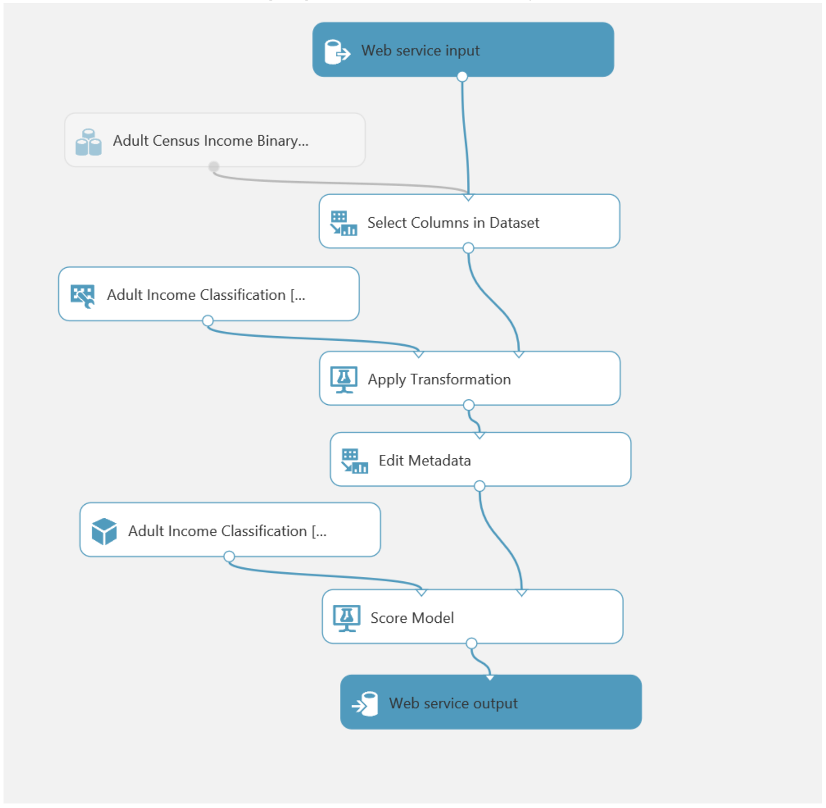
11. When the experiment has finished running, visualize the output of the **Score Model** module, and compare the predicted values in the **Scored Labels** column with the actual values from the test data set in the **income** column.
12. Visualize the output of the **Evaluate Model** module, and review the ROC curve (shown below). The larger the area under this curve (as indicated by the **AUC** figure), the better a classification model predicts when compared to a random guess. Then review the **Accuracy** figure for the model, which should be around **0.82**. This indicates that the classifier model is correct 82% of the time, which is a good figure for an initial model.



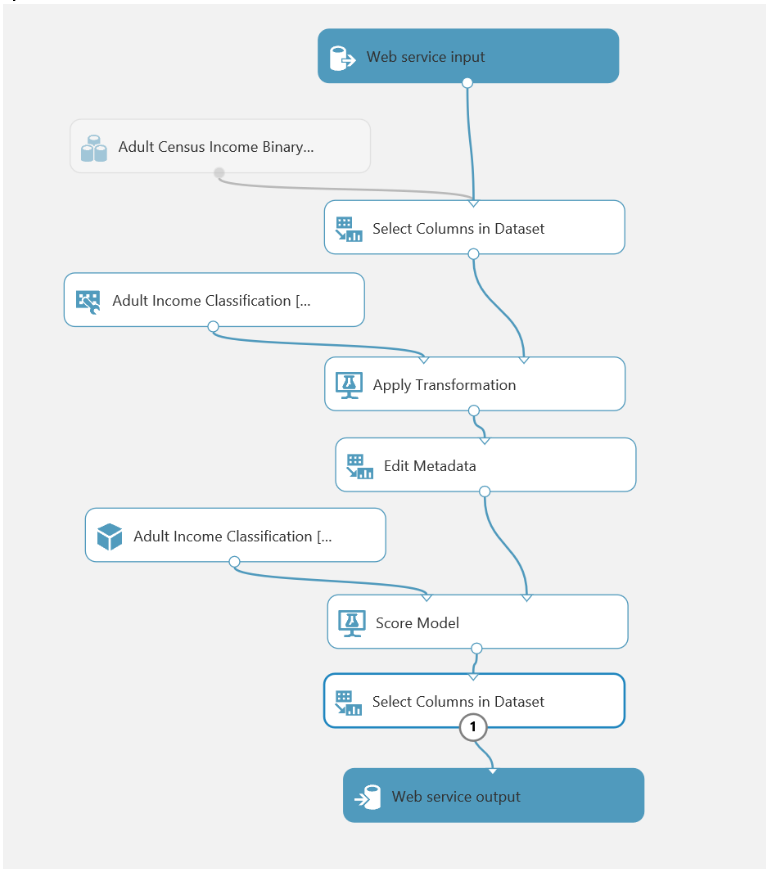
True Positive	False Negative	Accuracy	Precision	Threshold	AUC
1622	1489	0.822	0.663	0.5	0.873
False Positive	**True Negative**	**Recall**	**F1 Score**		
823	9090	0.521	0.584		
Positive Label	**Negative Label**				
>50K	<=50K				

Publish the Model as a Web Service

1. With the **Adult Income Classification** experiment open, click the **SET UP WEB SERVICE** icon at the bottom of the Azure ML Studio page and click **Predictive Web Service [Recommended]**. A new **Predictive Experiment** tab will be automatically created.
2. Verify that, with a bit of rearranging, the Predictive Experiment resembles this figure:


Web service input
Adult Census Income Binary...
Select Columns in Dataset
Adult Income Classification [...
Apply Transformation
Edit Metadata
Adult Income Classification [...
Score Model
Web service output

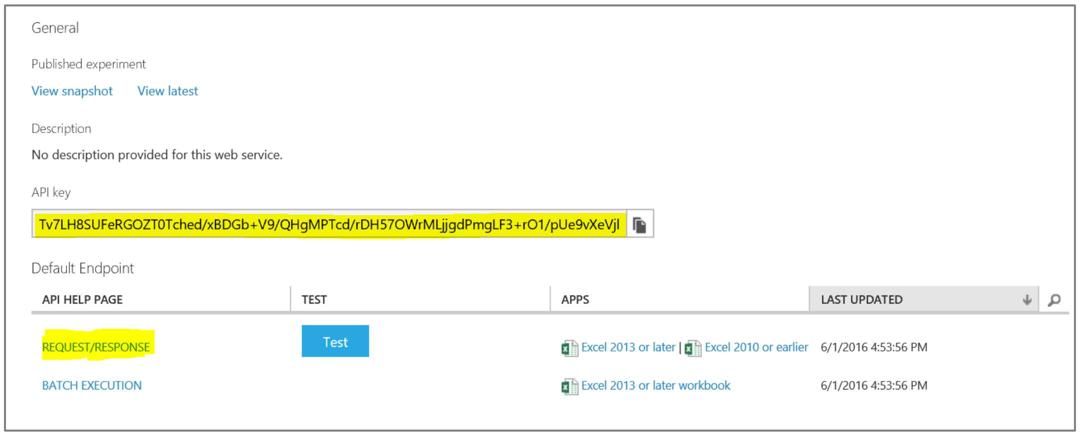
3. Delete the connection between the **Score Model** module and the **Web service output** module.
4. Add a **Select Columns in Dataset** module to the experiment, and connect the output of the **Score Model** module to its input. Then connect the output of the **Select Columns in Dataset** module to the input of the **Web service output** module.
5. Select the **Select Columns in Dataset** module, and use the column selector to select only the **Scored Labels** column. This ensures that when the web service is called, only the predicted value is returned.
6. Ensure that the predictive experiment now looks like the following, and then save and run the predictive experiment:


Web service input
Adult Census Income Binary...
Select Columns in Dataset
Adult Income Classification [...
Apply Transformation
Edit Metadata
Adult Income Classification [...
Score Model
Select Columns in Dataset
1
Web service output

7. When the experiment has finished running, visualize the output of the last **Select Columns in Dataset** module and verify that only the **Scored Labels** column is returned.

Deploy and Use the Web Service

1. In the **Adult Income Classification [Predictive Exp.]** experiment, click the **Deploy Web Service** icon at the bottom of the Azure ML Studio window.
2. Wait a few seconds for the dashboard page to appear, and note the **API key** and **Request/Response** link. You will use these to connect to the web service from a client application.


General

Published experiment

View snapshot View latest

Description

No description provided for this web service.

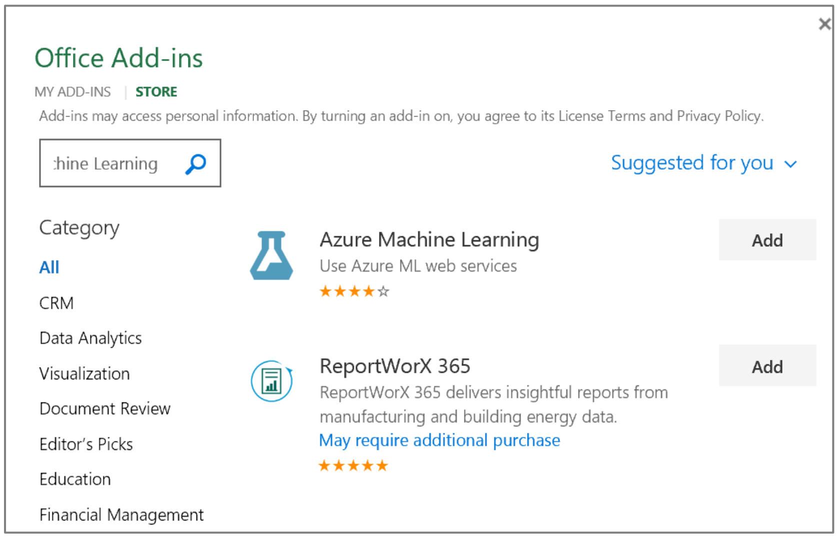
API key

Tv7LH8SUFeRGOZT0Tched/xBDGb+V9/QHgMPTcd/rDH57OWrMLjjgdPmgLF3+rO1/pUe9vXeVjl

Default Endpoint

API HELP PAGE	TEST	APPS	LAST UPDATED
REQUEST/RESPONSE	Test	Excel 2013 or later | Excel 2010 or earlier	6/1/2016 4:53:56 PM
BATCH EXECUTION		Excel 2013 or later workbook	6/1/2016 4:53:56 PM

3. Leave the dashboard page open in your web browser, and open a new browser tab.
4. In the new browser tab, navigate to https://office.live.com/start/Excel.aspx. If prompted, sign in with your Microsoft account (use the same credentials you use to access Azure ML Studio.)
5. In Excel Online, create a new blank workbook.
6. On the **Insert** tab, click **Office Add-ins**. Then in the **Office Add-ins** dialog box, select **Store**, search for *Azure Machine Learning*, and add the **Azure Machine Learning** add-in as shown below:


Office Add-ins

MY ADD-INS | STORE

Add-ins may access personal information. By turning an add-in on, you agree to its License Terms and Privacy Policy.

:hine Learning

Suggested for you

Category

All

CRM

Data Analytics

Visualization

Document Review

Editor's Picks

Education

Financial Management

Azure Machine Learning

Use Azure ML web services

★★★★☆

Add

ReportWorX 365

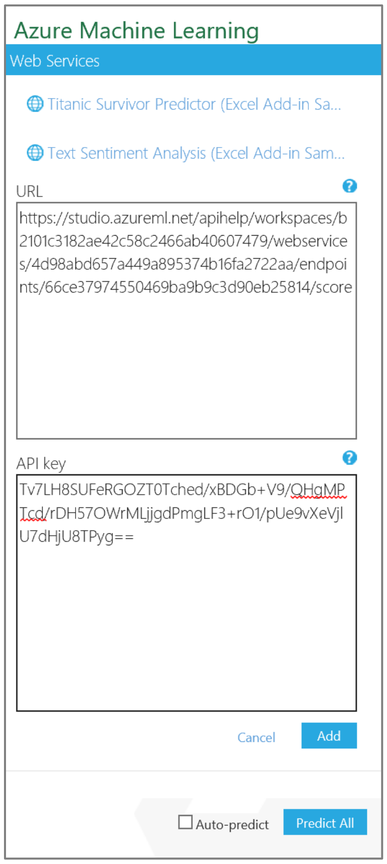
ReportWorX 365 delivers insightful reports from manufacturing and building energy data.

May require additional purchase

★★★★★

Add

7. After the add-in is installed, in the **Azure Machine Learning** pane on the right of the Excel workbook, click **Add Web Service**. Boxes for the URL and API key of the web service will appear.
8. On the browser tab containing the dashboard page for your Azure ML web service, right-click the **Request/Response** link you noted earlier and copy the web service URL to the clipboard. Then return to the browser tab containing the Excel Online workbook and paste the URL into the URL box.
9. On the browser tab containing the dashboard page for your Azure ML web service, click the **Copy** button for the **API key** you noted earlier to copy the key to the clipboard. Then return to the browser tab containing the Excel Online workbook and paste it into the **API key** box.
10. Verify that the **Azure Machine Learning** pane in your workbook now resembles this, and click **Add**:


Azure Machine Learning
Web Services
Titanic Survivor Predictor (Excel Add-in Sa...
Text Sentiment Analysis (Excel Add-in Sam...
Cancel
Add
Auto-predict
Predict All

URL

https://studio.azureml.net/apihelp/workspaces/b2101c3182ae42c58c2466ab40607479/webservices/4d98abd657a449a895374b16fa2722aa/endpoints/66ce37974550469ba9b9c3d90eb25814/score

API key

Tv7LH8SUFeRGOZT0Tched/xBDGb+V9/QHgMPTcd/rDH57OWrMLjjgdPmgLF3+rO1/pUe9vXeVjlU7dHjU8TPyg==

11. After the web service has been added, in the **Azure Machine Learning** pane, click **1. View Schema** and note the *inputs* expected by the web service (which consist of the fields in the original Adult Census dataset) and the *outputs* returned by the web service (the **Scored Labels** field).
12. In the Excel worksheet select cell A1. Then in the **Azure Machine Learning** pane, collapse the **1. View Schema** section and in the **2. Predict** section, click **Use sample data**. this enters some sample input values in the worksheet.
13. Modify the sample data in row 2 as follows:
    - **age**: 39
    - **workclass**: Private
    - **fnlwgt**: 77500
    - **education**: Bachelors
    - **education-num**: 13
    - **marital-status**: Never-married
    - **occupation**: Adm-clerical
    - **relationship**: Not-in-family
    - **race**: White
    - **sex**: Male
    - **capital-gain**: 2200
    - **capital-loss**: 0
    - **hours-per-week**: 40
    - **native-country**: United-States
    - **income**: Unknown
14. Select the cells containing the input data (cells A1 to O2), and in the **Azure Machine Learning** pane, click the button to select the input range and confirm that it is **'Sheet1'!A1:O2**.

15. Ensure that the **My data has headers** box is checked.
16. In the **Output** box type **P1**, and ensure the **Include headers** box is checked.
17. Click the **Predict** button, and after a few seconds, view the predicted label in cell P2.
18. Change the marital-status value in cell F2 to **Married** and click **Predict** again. Then view the updated label that is predicted by the web service.
19. Try changing a few of the input variables and predicting the income classification. You can add multiple rows to the input range and try various combinations at once.

Implementing a Regression Model

In this exercise you will perform regression on the automobiles dataset. This dataset contains a number of characteristics for each automobile. These characteristics, or features, are used to predict the price of the automobile.

Prepare the Data

Note: If you completed Lab 5: *Transforming Data*, then you can skip this procedure and open the **Autos** experiment you created previously.

1. In Azure ML Studio, create a new experiment called **Autos**.
2. Create a new dataset named **autos.csv** by uploading the **autos.csv** file in the lab files folder for this module to Azure ML.
3. Create a second new dataset named **makes.csv** by uploading the **makes.csv** file in the lab files folder for this module to Azure ML.
4. Add the **autos.csv** and **makes.csv** datasets to the **Autos** experiment.
5. Use the following instructions to add a script module to prepare the data:

If you prefer to work with R:

- Add an **Execute R Script** module to the experiment and connect the output of the **autos.csv** dataset to its **Dataset1** (left-most) input, and the output of the **makes.csv** dataset to its **Dataset2** (middle) input.
- Replace the default R script in the **Execute R Script** module with the following code (which you can copy and paste from **PrepAutos.R** in the lab files folder for this module):

```
## Define a function to join the data frames
join.auto <- function(autos, makes){
  require(dplyr) ## Make sure dplyr is loaded
  left_join(autos, makes, by = 'make-id')
}

## Function to clean and prepare the auto data
prep.auto <- function(df, col.names){
  require(dplyr) ## Make sure dplyr is loaded

  ## set the column names.
  names(df) <- col.names

  ## Eliminate unneeded columns
  df <- df[,!(names(df) %in%
              c('symboling', 'normalizedlosses', 'makeid'))]

  ## Add a log transformed column for price using dplyr mutate
  df <- df %>% mutate(lnprice = log(price))
  ## Remove rows with NAs
  df <- df[complete.cases(df), ]

  ## Remove duplicate rows
  df <- df %>% filter (! duplicated(df,))

  ## Consolidate the number of cylinders
  df <- df %>%
    mutate(cylinders = ifelse(cylinders %in% c("four", "three",
"two"), "three-four",
           ifelse(cylinders %in% c("five", "six"), "five-six",
"eight-twelve")))

  df
}

## Function to find outliers
id.outlier <- function(df){
  ## Use ifelse to filter for outliers
  df[, "outlier"] <- ifelse(df[,"enginesize"] > 190 |
                             df[, "weight"] > 3500|
                             df[, "citympg"] > 40, '1', '0')
  df
}

## Define column names (change - to . for R)
col.names <- c('symboling', 'normalizedlosses', 'makeid',
'fueltype', 'aspiration', 'doors',
               'body', 'drive', 'engineloc', 'wheelbase',
               'length', 'width', 'height', 'weight',
'enginetype',
               'cylinders', 'enginesize', 'fuelsystem', 'bore',
'stroke',
               'compression', 'horsepower', 'rpm', 'citympg',
               'highwaympg', 'price', 'make')

## R code to prep the auto data
autos <- maml.mapInputPort(1) # read autos data frame from port 1
makes <- maml.mapInputPort(2) # read makes data frame from port 2
require(dplyr) # make sure dplyr is loaded
joined <- join.auto(autos, makes) ## Join the data frames
prepped <- prep.auto(joined, col.names) # Transform the data
outliers <- id.outlier(prepped) # mark the outliers
## remove rows where outlier is 1, then remove outlier column
out <- outliers %>% filter(outlier == 0) %>% subset(select=-
outlier)
maml.mapOutputPort("out") ## Output the prepared data frame
```

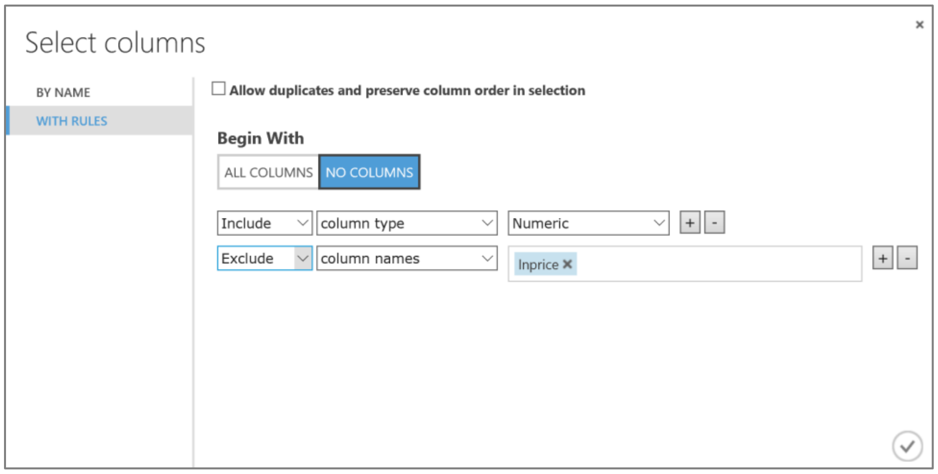
If you prefer to work with Python:

1. Add an **Execute Python Script** module to the experiment and connect the output of the **autos.csv** dataset to its **Dataset1** (left-most) input, and the output of the **makes.csv** dataset to its **Dataset2** (middle) input.

2. Replace the default Python script in the **Execute Python Script** module with the following code (which you can copy and paste from **PrepAutos.py** in the lab files folder for this module):

```
## Function to join automotive data
def auto_join(autos, makes):
    import pandas as pd
    key = ['make-id'] ## Define key column
    ## Return the joined dataframe
    return pd.merge(autos, makes, on = key, how = 'left')

## Function to prepare the automotive data
def prep_auto(df, col_names):
    import pandas as pd
    import numpy as np

    ## Assign names to columns
    df.columns = col_names

    ## Drop unneeded columns
    drop_list = ['symboling', 'normalizedlosses', 'makeid']
    df.drop(drop_list, axis = 1, inplace = True)

    ## Remove rows with missing values
    df = df[~pd.isnull(df).any(axis=1)]

    ## Add a log transformed column for price
    df['lnprice'] = np.log(df['price'].as_matrix())

    ## Remove duplicate rows
    df.drop_duplicates(inplace = True)

    ## Create a column with new levels for the number of
cylinders
    df['cylinders'] = ['four-or-less' if x in ['two', 'three',
'four'] else
                                 ('five-six' if x in ['five',
'six'] else
                                 'eight-twelve') for x in
df['cylinders']]

    df = id_outlier(df)  # mark outliers
    df = df[df.outlier == 0] # filter for outliers
    df.drop('outlier', axis = 1, inplace = True)

    return df

def id_outlier(df):
    ## Create a vector of 0 of length equal to the number of rows
    temp = [0] * df.shape[0]
    ## test each outlier condition and mark with a 1 as required
    for i, x in enumerate(df['enginesize']):
        if (x > 190): temp[i] = 1
    for i, x in enumerate(df['weight']):
        if (x > 3500): temp[i] = 1
        for i, x in enumerate(df['citympg']):
            if (x > 40): temp[i] = 1
        df['outlier'] = temp # append a column to the data frame
        return df

    def azureml_main(autos, makes):
        ## join datasets
        df = auto_join(autos, makes)

        ## Define column names
        col_names = ['symboling', 'normalizedlosses', 'makeid',
    'fueltype', 'aspiration', 'doors',
           'body', 'drive', 'engineloc', 'wheelbase',
           'length', 'width', 'height', 'weight', 'enginetype',
           'cylinders', 'enginesize', 'fuelsystem', 'bore', 'stroke',
           'compression', 'horsepower', 'rpm', 'citympg',
           'highwaympg', 'price', 'make']

        ## Call function to prep auto data and return
        return prep_auto(df, col_names)
```

6. After you've created your choice of custom script module, drag a **Normalize Data** module onto the canvas and connect the output from the script module to its input.
7. On the properties pane of the **Normalize Data** module select the following settings:
    - **Transformation method**: ZScore
    - **Use 0 for constant columns when checked**: Unchecked
    - **Columns to transform**: Include all numeric columns except for **lnprice**, as shown here:


Select columns

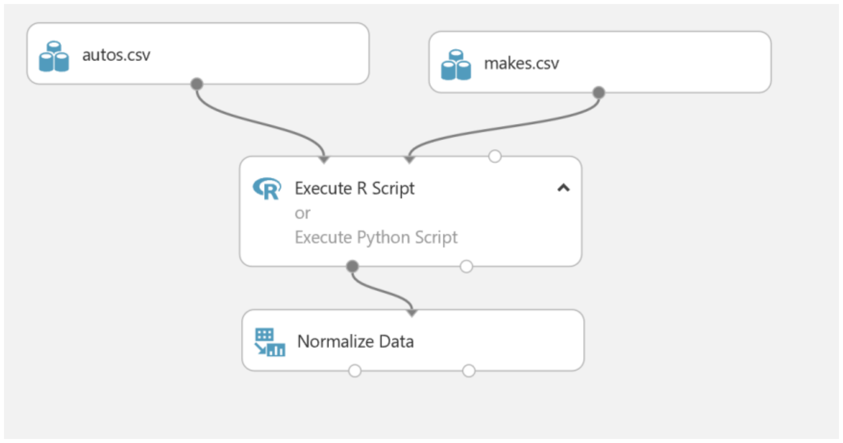
BY NAME
WITH RULES

☐ Allow duplicates and preserve column order in selection

Begin With
ALL COLUMNS | NO COLUMNS

Include | column type | Numeric
Exclude | column names | lnprice

8. verify that your experiment looks similar to this:



9. Save and run the experiment, and then visualize the output of the **Normalize Data** module to see the transformed data.

Remove Redundant Columns and Split the Data

You have cleaned and scaled the automobile data, and generated a label named **lnprice** that is the log of the original **price** column. Your model will predict the **lnprice** for automobiles based on their features. You must now remove the redundant price column, and split the data into a training set and a test set so you can build your model.

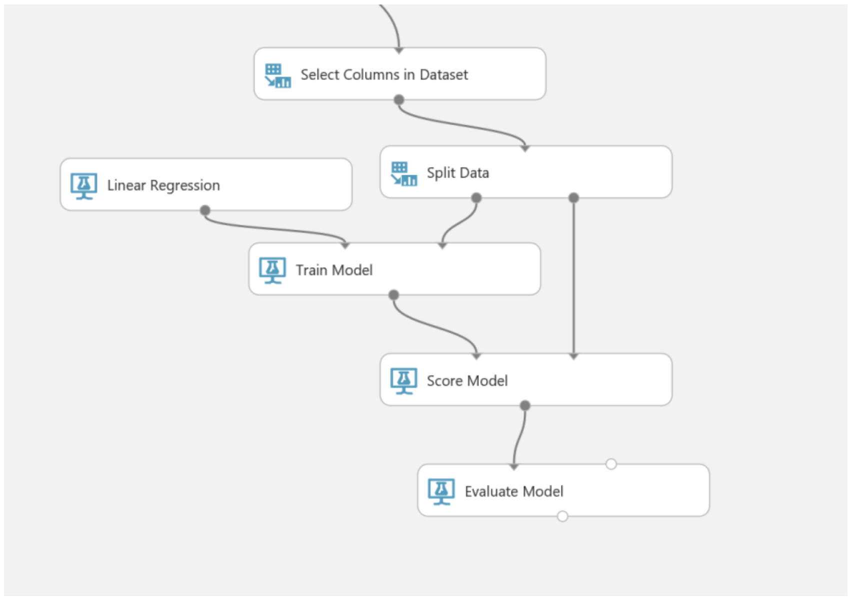
1. Add a **Select Columns in Dataset** module to the **Autos** experiment, and connect the output from the **Normalize Data** module to its input.
2. Use the columns elector to configure the **Select Columns in Dataset** module to select all columns <u>excluding</u> the **price** column.
3. Add a **Split Data** module to the experiment, and connect the output of the **Select Columns in Dataset** module to the input of the **Split Data** module. You will use this module to split the data into separate training and test datasets.
4. Set the properties of the **Split Data** module as follows:
   - **Splitting mode**: Split Rows
   - **Fraction of rows in the first output dataset**: 0.7
   - **Randomized split**: Checked
   - **Random seed**: 123
   - **Stratified split**: False

Train and Test a Linear Regression Model

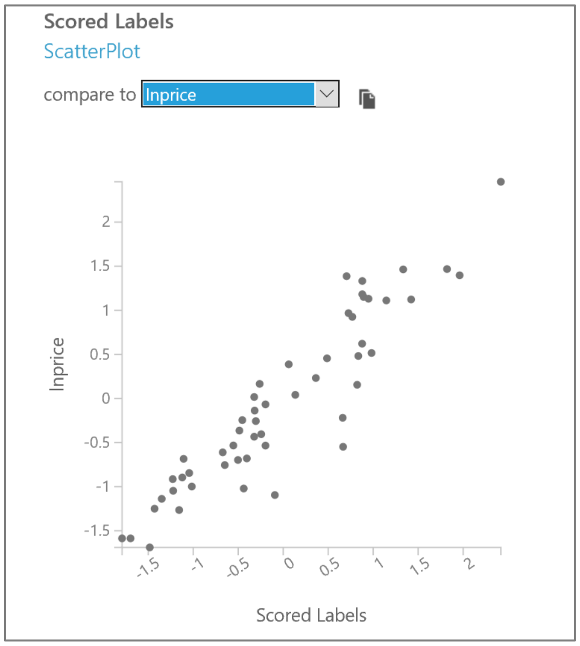
1. Add a **Train Model** module to the experiment, and connect the **Results dataset1** (left) output of the **Split Data** module to the **Dataset** (right) input of the **Train Model** module.
2. In the **Properties** pane for the **Train Model** module, use the column selector to select the **lnprice** column. This sets the label column that the regression model will be trained to predict.
3. Add a **Linear Regression** module to the experiment, and connect the output of the **Linear Regression** module to the **Untrained model** (left) input of the **Train Model** module. This specifies that the regression model will be trained using the linear regression algorithm.
4. Set the properties of the **Linear Regression** module as follows:
   - **Solution method: Ordinary Least Squares**
   - **L2 regularization weight**: 0.001
   - **Include intercept term**: Unchecked
   - **Random number seed**: 123

- **Allow unknown categorical levels**: Checked

5. Add a **Score Model** module to the experiment. Then connect the output of the **Train Model** module to the **Trained model** (left) input of the **Score Model** module, and connect the **Results dataset2** (right) output of the **Split Data** module to the **Dataset** (right) input of the **Score Model** module.
6. On the **Properties** pane for the **Score Model** module, ensure that the **Append score columns to output** checkbox is selected.
7. Add an **Evaluate Model** module to the experiment, and connect the output of the **Score model** module to the **Scored dataset** (left) input of the **Evaluate Model** module.
8. Verify that the lower part of your experiment resembles the figure below, then save and run the experiment.



9. When the experiment has finished, visualize the output of the **Score Model** module and select the **Scored Labels** column. Then, in the **compare to** drop-down, select **lnprice** and verify that the resulting scatter-plot chart resembles this:



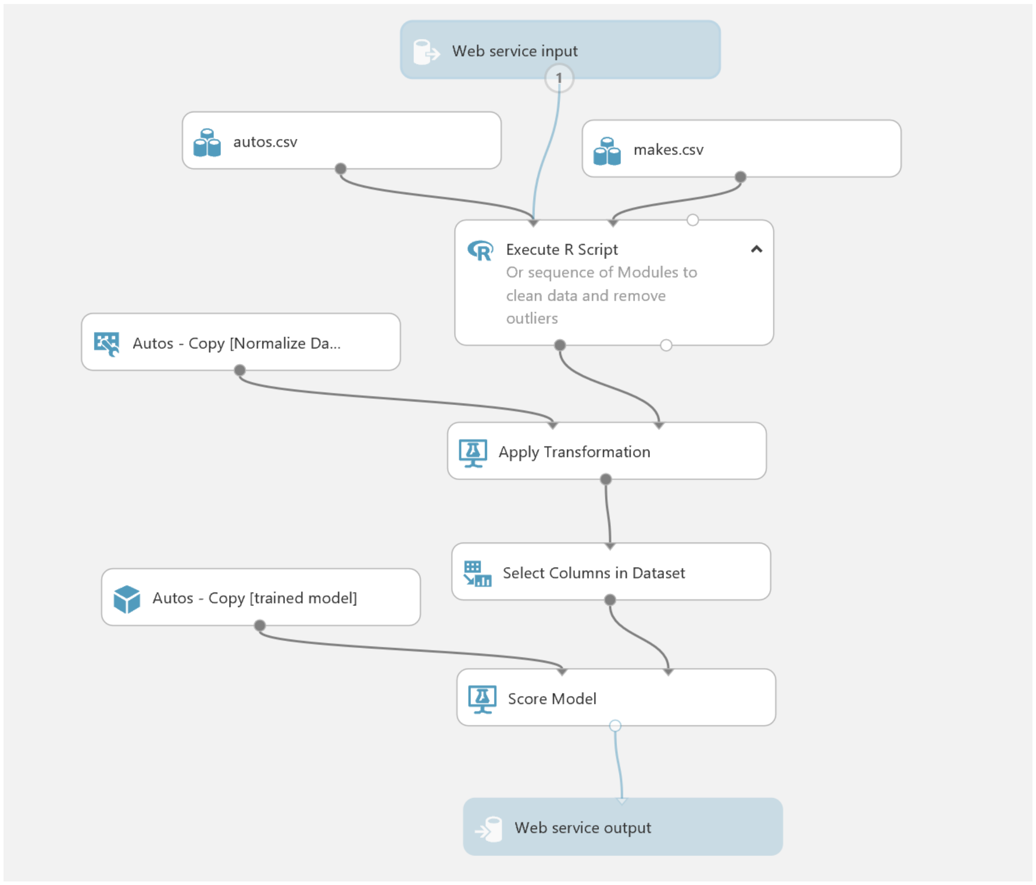
Note that the values of the **Scored Label** and **lnprice**, mostly fall on an imaginary straight diagonal line with little dispersion, which indicates the model is a reasonably good fit.

10. Visualize the output of the **Evaluate Model** module and note the metrics for the model. The root mean squared error (RMSE) should be around 0.154, which is less than half of the standard deviation of the **lnprice** column in the source data after the numeric values have been normalized (which is around 0.4), indicating that the model seems to be a good fit.

Publish a Web Service for the Regression Model

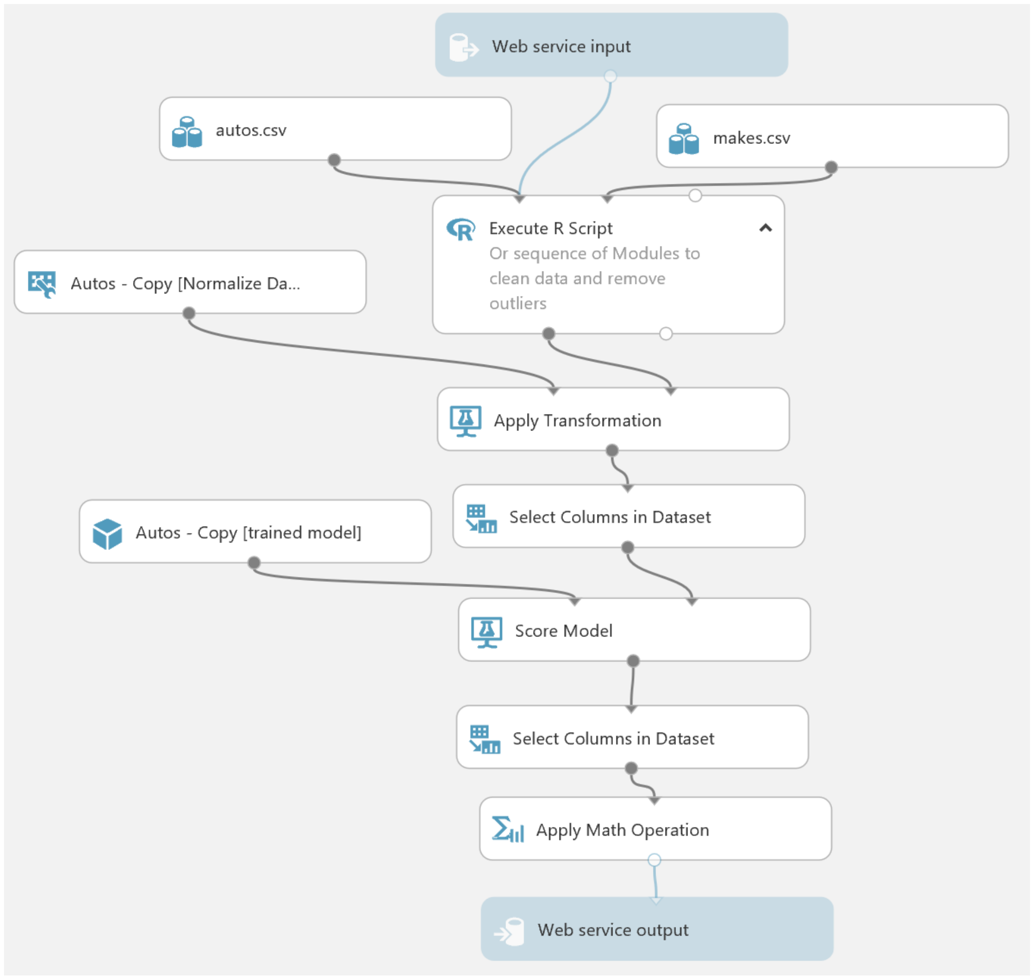
Now that you have created the regression model, you can publish the experiment as a web service and use it from a client application; just as you did previously for the classification model.

1. With the **Autos** experiment open, click the **SET UP WEB SERVICE** icon at the bottom of the Azure ML Studio page and click **Predictive Web Service [Recommended]**. A new **Predictive Experiment** tab will be automatically created.
2. Verify that, with a bit of rearranging, the **Predictive Experiment** resembles this figure:


Web service input
1
autos.csv
makes.csv
Execute R Script
Or sequence of Modules to clean data and remove outliers
Autos - Copy [Normalize Da...
Apply Transformation
Select Columns in Dataset
Autos - Copy [trained model]
Score Model
Web service output

The web service currently returns the scored labels and all of the other fields. You must modify it so return only the predicted price. However, the scored label in this case is actually the natural log of the predicted automobile price, so you must convert this back to an absolute value.

3. Delete the connection between the **Score Model** module and the **Web service output** module.
4. Add a **Select Columns in Dataset** module to the experiment, and connect the output of the **Score Model** module to its input. Then in the Properties pane for the **Select Columns in Dataset** module, use the column selector to select only the **Scored Labels** column.
5. Add an **Apply Math Operation** module to the experiment and connect the output from the **Select Columns in Dataset** module to its input. Then set the properties of the **Apply Math Operation** module as follows:
   - **Category**: Basic
   - **Basic math function**: Exp
   - **Column set**: use the column selector to select the **Scored Labels** column.
   - **Output mode**: ResultOnly
6. Ensure that the predictive experiment now looks like the following, and then save and run it:


Web service input
autos.csv
makes.csv
Execute R Script
Or sequence of Modules to clean data and remove outliers
Autos - Copy [Normalize Da...
Apply Transformation
Autos - Copy [trained model]
Select Columns in Dataset
Score Model
Select Columns in Dataset
Apply Math Operation
Web service output

7. When the experiment has finished running, visualize the output of the **Apply Math Operation** module and verify that the predicted price is returned.

Deploy and Use the Web Service

1. In the **Autos [Predictive Exp.]** experiment, click the **Deploy Web Service** icon at the bottom of the Azure ML Studio window.
2. Wait a few seconds for the dashboard page to appear, and note the **API key** and **Request/Response** link.
3. Create a new blank Excel Online workbook at https://office.live.com/start/Excel.aspx and insert the Azure Machine Learning add-on. Then add the **Autos [Predictive Exp.]** web service, pasting the **Request/Response** URL and **API key** into the corresponding text boxes.
4. Use the web service to predict an automobile price based on the following values:
    - **symboling**: 0
    - **normalized-losses**: 0
    - **make-id**: 1
    - **fuel-type**: gas
    - **aspiration**: turbo
    - **num-of-doors**: four
    - **body-style**: sedan
    - **drive-wheels**: fwd
    - **engine-location**: front

- **wheel-base**: 106
- **length**: 192.5
- **width**: 71.5
- **height**: 56
- **curb-weight**: 3085
- **engine-type**: ohc
- **num-of-cylinders**: five
- **engine-size**: 130
- **fuel-system**: mpfi
- **bore**: 3.15
- **stroke**: 3.4
- **compression-ratio**: 8.5
- **horsepower**: 140
- **peak-rpm**: 5500
- **city-mpg**: 17
- **highway-mpg**: 22
- **price**: 0

5. Note the predicted price, and then change the **fuel-type** to *diesel* and predict the price again.

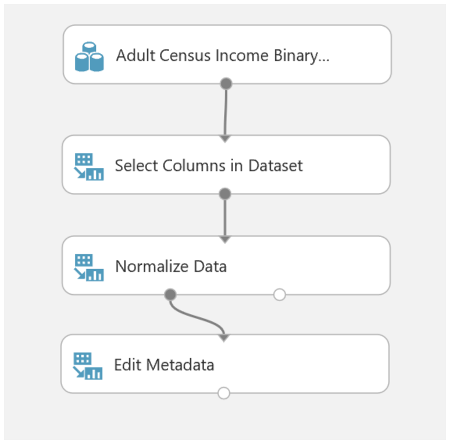
Creating a Clustering Model

In this exercise you will perform k-means cluster analysis on the Adult Census Income Binary Classification Dataset. You will determine how many natural clusters these data contain and evaluate which features define this structure.

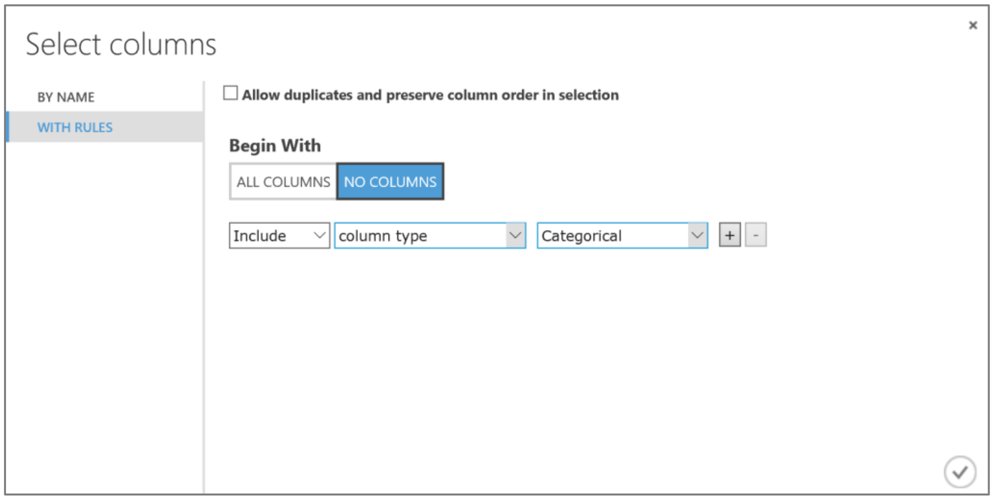
Prepare the Data

Note: If you did not complete the first exercise in this lab (*Implementing a Classification Model*), go back and perform the first procedure in that exercise (*Prepare the Data*) before performing this procedure).

1. In Azure ML Studio, open the **Adult Income Classification** experiment you created in the first exercise of this lab.
2. At the bottom of the experiment page, click **Save As** and create a copy of the experiment with the name **Adult Income Clustering**.
3. Delete all modules in the experiment other than the following:



4. Add a **Convert to Indicator Values** module to the experiment and connect the output of the **Edit Metadata** module to its input.
5. Configure the properties of the **Convert to Indicator Values** module to select all *categorical* columns, as shown here:


Select columns

BY NAME
WITH RULES

☐ Allow duplicates and preserve column order in selection

Begin With
ALL COLUMNS [✓ NO COLUMNS]

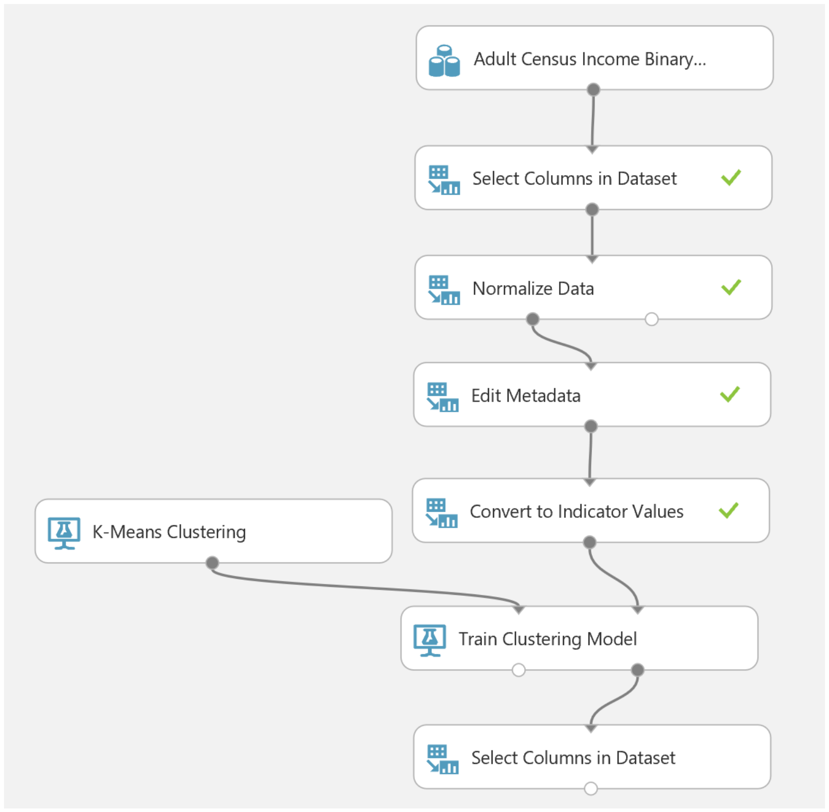
Include | column type | Categorical

6. Select the **Overwrite Categorical Columns** property for the **Convert to Indicator Values** module.
7. Save and run the experiment, and when the experiment has finished running, visualize the output of the **Convert to Indicator Values** module to verify that the categorical features in the dataset are now represented as numeric indicator columns for each category value, with a 0 or 1 to indicate which categories apply to each row. This structure will make the clustering algorithm more effective.

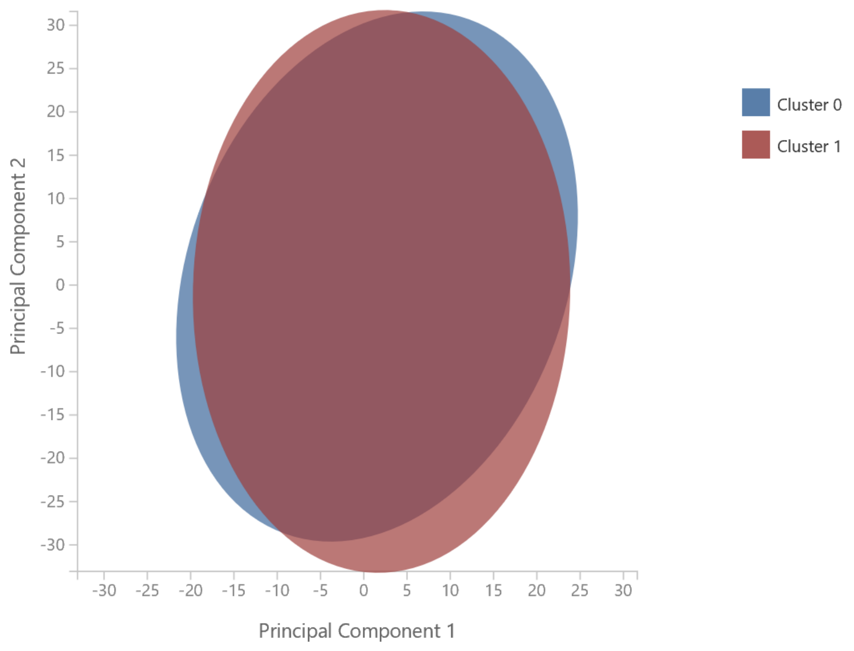
Create a K-Means Clustering Model

Now that the data is prepared, you are ready to create a clustering model.

1. Add a **K-Means Clustering** module to the experiment, and set its properties as follows:
   - **Create trainer mode**: Single Parameter
   - **Number of Centroids**: 2
   - **Initialization**: Random
   - **Random number seed**: 4567
   - **Metric**: Euclidian
   - **Iterations**: 100
   - **Assign Label Model**: Ignore label column
2. Add a **Train Clustering Model** module to the experiment, and connect the output of the **K-Means Clustering** module to its **Untrained model** (left) input and the output of the **Convert to Indicator Values** module to its **Dataset** (right) input.
3. Configure the properties of the **Train Clustering Model** module to select all columns and enable the **Check for Append or Uncheck for Result Only** option.
4. Add a **Select Columns in Dataset** module to the experiment, and connect the **Results** (right) output of the **Train Clustering Model** module to its input.
5. Configure the properties of the **Select Columns in Dataset** module to select all features.
6. Verify that the experiment resembles this, and then save and run the experiment.


Adult Census Income Binary...
Select Columns in Dataset
Normalize Data
Edit Metadata
Convert to Indicator Values
K-Means Clustering
Train Clustering Model
Select Columns in Dataset

7. When the experiment has finished running, visualize the **Results** (right) output of the **Train Clustering Model** module and note the visualization that shows the two clusters that have been generated.



8. Visualize the output of the **Select Columns in Dataset** module and view the **Assignments** column at the right end of the table, which shows the two clusters {0,1}. The visualization shows the proportions of people assigned to each cluster. Interestingly, the cluster assignments do not

correspond exactly with the high and low income classifications, indicating that there may be some more complex combination of factors that differentiates the people in this dataset.

Summary

In this lab you:

- Create a classification model.
- Created a Regression model.
- Created a clustering model.