

AZURE MACHINE LEARNING

Predicting the Value of Hong Kong Properties

A Step-By-Step Tutorial Using Azure Machine Learning

Overview

Learn how companies like [Zillow](#) would predict the value of your property in Hong Kong. In this tutorial you will learn how to build a model to predict the real estate sales price of a property based upon various historical features about the house and the sales transaction.

About the Data

The Hong Kong Island areas of [Central, Sheung Wan and Sai Wan private housing dataset](#) is a Excel Comma Separated Value (CSV) text file, which include 29 features and 1139 observations. Each observation represents the sale of a home and each feature is an attribute describing the house or the circumstance of the sale. A data dictionary has been provided to explain the features, [click here for the full list of feature descriptions](#).

Objective of this tutorial

As a partner of a property investment company, your objective is to make a profit from investing in and the eventual sale of invested properties. To do this, you need a solid property prediction model based on historical property transactions. To enable the

prediction of future property prices from your prediction model compared against prevailing asking prices. So that the future sale of a property will bring in a nice profit.

Preprocessing & Data Exploration

Step 1 - Exploratory Data Analysis (EDA)

Open the file "[HKPropVisualizationinTableau.txt](#)" Hong Kong Property Map Visualization Using Tableau located in the "[HK Property Files](#)" folder. Follow the steps given in the document to explore the mapping functions of Tableau in relation to the "[HKProp Dataset.csv](#)" file.

Objective of EDA

By combining the powerful mapping functions of mapbox.com in Tableau Desktop, we can very quickly visualize our HK property dataset by locating properties of interest on top of a satellite image background and surrounding facilities.

Clean Missing Data

Most machine learning algorithms are unable to account for missing values and some treat it inconsistently from others. To address this, we must make sure our dataset contains no missing, "null," "NAN" Not A Number or "NA" values.

Replacement of missing values is the most versatile and preferred method because it allows us to keep our data. It also minimizes collateral damage to other columns because of one cell's bad behavior. In replacement, numerical values can easily be replaced with statistical values such as mean, median, or mode. While categories can be commonly dealt with by replacing with the mode or a separate categorical value for unknowns.

For simplicity, all categorical missing values were cleaned with the mode and all numeric features were cleaned using the median. To further improve a model's performance, custom cleaning functions should be tried and implemented on each individual feature rather than a blanket transformation of all columns.

Step 2 - Data Cleaning

Define Categorical Variables

We must now define which values are non-continuous or discrete by casting them as categorical. Mathematical approaches for continuous and non-continuous values differ greatly. [Nominal categorical features](#) were identified and cast as categorical data types using the [meta data editor](#) to ensure proper mathematical treatment by the machine learning algorithm. Another way to think of nominal categorical features are these are names or 'labels' without any quantitative values.

From the dataset, there are 4 features with missing values. Most machine learning algorithms do **NOT** accept missing values and will show an error when missing values are encountered in the dataset. So, we need to first "clean" missing values by treating them with substitute values.

Features with missing values are separately "Bed_Room", "Rehab_Year", "Tower" and "SaleableArea".

For "Bed_Room" feature, it should be a discrete number but from the dataset, it is a numeric feature with 677 missing values. We need to cast the feature as categorical and substitute a value of "0" for missing values.

1. Use "Clean Missing Data" module to replace the missing value with mode.

▲ Clean Missing Data

Columns to be cleaned

Selected columns:

Column names:

Bed_Room

Launch column selector

Minimum missing value ra... 

0


Maximum missing value ra... 

1

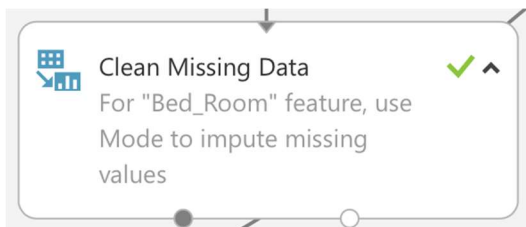
Cleaning mode

Replace with mode 

Cols with all missing values 

Remove 

☐ Generate missing valu... 



The feature [Rehab_Year] represents the year of building rehabilitation scheme – which is part of efforts by the [Urban Renewal Authority of Hong Kong](#) in rehabilitating aging buildings. We need to reflect an adjusted age of the properties after it has been rehabilitated in this way, to distinguish them from properties that were never rehabilitated.

The [Rehab_Year] feature is a discrete value. But from the dataset, it has 1049 missing values. We need to substitute the missing values with “0”.

1. Use “Clean Missing Data” module to replace the missing value with “0”

Clean Missing Data

Columns to be cleaned

Selected columns:

Column names:

Rehab_Year

Launch column selector

Minimum missing value ra... 


0

Maximum missing value ra... 

1

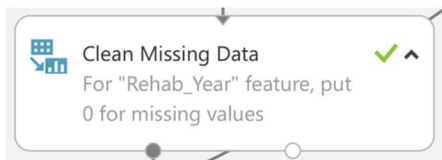
Cleaning mode

Custom substitution value 

Replacement value 

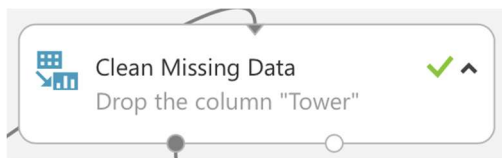
0

☐ Generate missing valu... 



The [Tower] feature represents the name of the tower of an estate and is not a significant feature for machine learning and the whole feature (i.e. column) is removed.

Use "Clean Missing Data" module to remove entire column.



Properties Project

Clean Missing Data

Columns to be cleaned

Selected columns:
Column names: Tower

Launch column selector

Minimum missing value ra...

0

Maximum missing value ra...

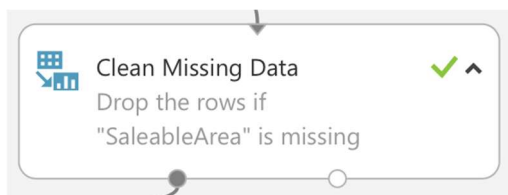
1

Cleaning mode

Remove entire column

The “[SaleableArea] feature is significant but from the dataset, there were 61 missing values and these observations (i.e. row) should be dropped, resulting in 1078 rows (observation points) from the original 1139 rows.

Use “Clean Missing Data” module to remove an entire row.



Properties Project

Clean Missing Data

Columns to be cleaned

Selected columns:
Column names:
SaleableArea

Launch column selector

Minimum missing value ra...

0

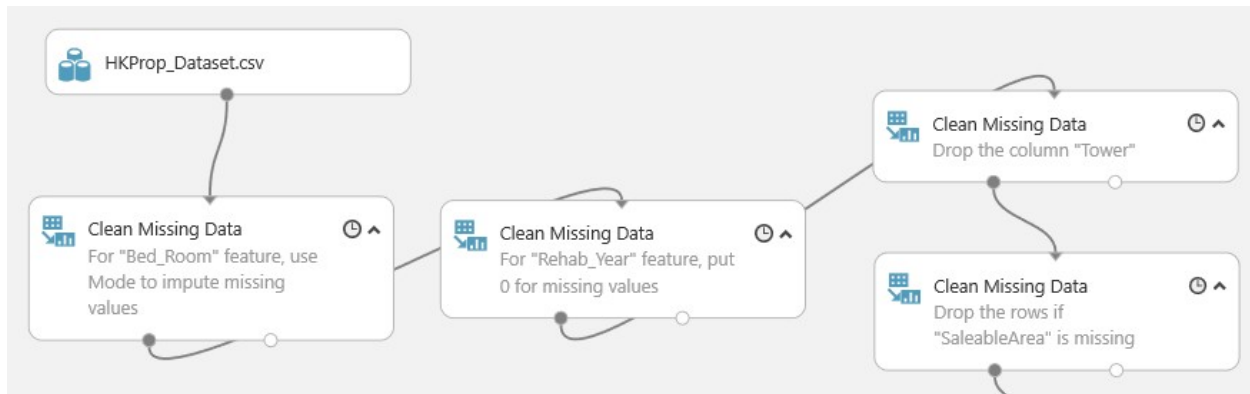
Maximum missing value ra...

1

Cleaning mode

Remove entire row

Our Azure Machine Learning experiment demonstrates how the 4 types of data cleaning modules may be connected.



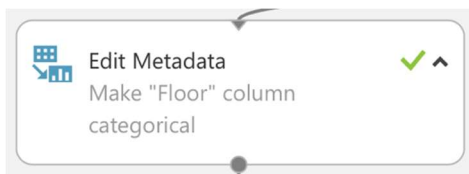
End of Step 2 – Data Cleaning

Step 3 - Feature Engineering

In this tutorial, we will perform feature engineering on the columns of [Floor], [Kindergarten], [Primary_Schools], [Secondary_Schools], [Build_Ages] and [Rehab_Year].

The feature [Floor] has a range of value from 0 to 53. We want to group it into 3 floor categories (i.e. High, Medium and Low).

1. Use “Edit Metadata” module to make the [Floor] to categorical.



Edit Metadata

Column

Selected columns:
Column names: Floor

Launch column selector

Data type

String

Categorical

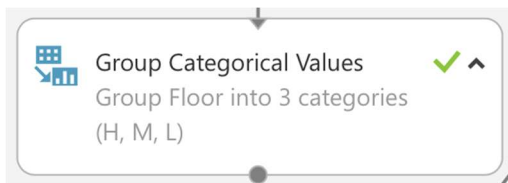
Make categorical

Fields

Features

New column names

2. Use “Group Categorical Values” module to group [Floor] into 3 categories. Follow the screen below to set 3 levels with the numbers as given below
 - H equals to (31-53)
 - M equals to (16-30)
 - L are remaining floors



Group Categorical Values

Selected columns

Selected columns:
Column names: Floor

Launch column selector

Output mode

Append

Default level name

L

New number of levels

3

Name of new level 1

H

Comma-separated list of o...

1,32,33,34,35,36,37,38,39,40

Name of new level 2

M

Comma-separated list of o...

16,17,18,19,20,21,22,23,24,25

3. Use "Edit Metadata" module to rename the column to [Floor_(H/M/L)]

Edit Metadata

Column

Selected columns:
Column names: Floor (2)

Launch column selector

Data type

Unchanged

Categorical

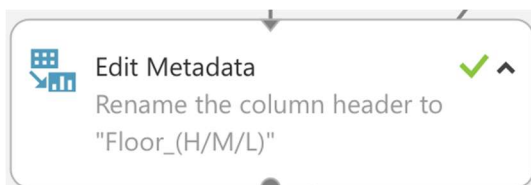
Unchanged

Fields

Unchanged

New column names

Floor_(H/M/L)



The features [Kindergarten], [Primary_Schools] and [Secondary_Schools] are similar and we combine them to form a new feature column called [Edu_Inst]. Follow the steps below to achieve this.

1. Use “Apply Math Operation” module to add [Kindergarten] and [Primary_Schools]. The “append output mode” appends the result column in the last column of the dataset and the column name is [Add(Primary_Schools_Kindergrarten)].

Apply Math Operation

Category
Operations

Basic operation
Add

Operation argument type
ColumnSet

Operation argument

Selected columns:

Column names:

Kindergarten

Launch column selector

Column set

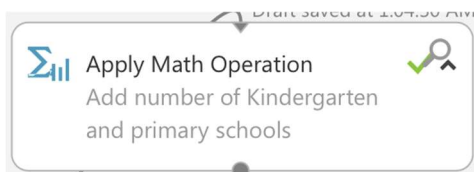
Selected columns:

Column names:

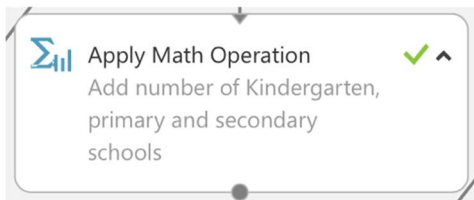
Primary_Schools

Launch column selector

Output mode
Append



2. Use “Apply Math Operation” module to combine [Secondary_Schools] and [Add(Primary_Schools_Kindergrarten)]. The “Inplace” output mode overwrites the result column in step 1 and the column name remains unchanged.



Apply Math Operation

Category

Operations

Basic operation

Add

Operation argument type

ColumnSet

Operation argument

Selected columns:

Column names:

Secondary_Schools

Launch column selector

Column set

Selected columns:

Column names:

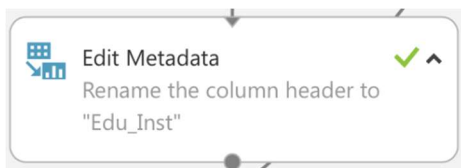
Add(Primary_Schools_Kinde

Launch column selector

Output mode

Inplace

3. Use “Edit Metadata” module to rename the column to “Edu_Inst”. This new column is the combination of Kindergarten, Primary and Secondary schools of the area of interest.



▲ Edit Metadata

Column

Selected columns:

Column names:

Add(Primary_Schools_Kinde

Launch column selector

Data type

Unchanged

Categorical

Unchanged

Fields

Unchanged

New column names

Edu_Inst

The feature [Rehab_Year] and [Build_Age] are used to calculate the effective building age of a building. As some buildings are rehabilitated, their effective building ages should be discounted.

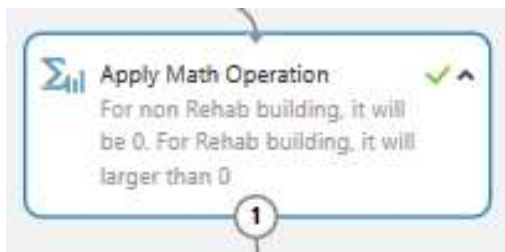
The formula for calculating the effective building ages has 2 parts and is given below:

1. For buildings without rehabilitation, their building ages are their effective building ages.
e.g. if an observation's [Rehab_Year] is equal to 0, that means the building is not rehabilitated. Therefore, the effective building age should equal to [Build_Age].
2. For building with rehabilitation, their building ages should be a fraction of discount, in our initial assessment, we use 2/3 or 0.66 to discount their building ages.
e.g. if an observation's [Rehab_Year] is 2009, that means the building has rehabilitated on year 2009. Hence the effective building age should be equal to $0.66 * [\text{Build_Age}]$.

The calculation of the effective building age of a property is given below.

1. Use "Apply Math Operation" module to Multiply [Rehab_Year] by [Build_Ages]. In this step, for the building without rehabilitation, the new value becomes 0. For rehabilitated buildings, the new value will be greater than 0.

The “Append” output mode appends the result column in the last column of the dataset with column name [Multiply(Build_Ages_Rehab_Year)].



▲ Apply Math Operation

Category

Operations ▼

Basic operation

Multiply ▼

Operation argument type

ColumnSet ▼

Operation argument

Selected columns:
Column names:
Rehab_Year

Launch column selector

Column set

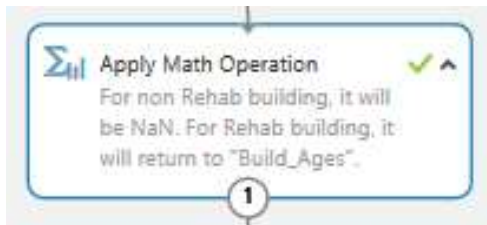
Selected columns:
Column names:
Build_Ages

Launch column selector

Output mode

Append ▼

2. Use “Apply Math Operation” module to Divide [Multiply(Build_Ages_Rehab_Year)] by [Rehab_Year]. In this step, for the building without rehabilitation, the math operation is 0/0 resulting in “NaN”. For rehabilitated buildings, the quotient should return to [Build_Ages]. The “Inplace” output mode overwrite the result column in step 1 and the column name has no change (i.e. Multiply(Build_Ages_Rehab_Year))



Apply Math Operation

Category
Operations

Basic operation
Divide

Operation argument type
ColumnSet

Operation argument
Selected columns:
Column names:
Rehab_Year

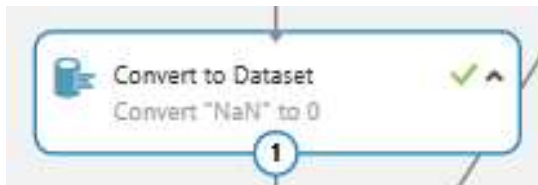
Launch column selector

Column set
Selected columns:
Column names:
Multiply(Build_Ages_Ref
< >

Launch column selector

Output mode
Inplace

3. Use “Convert to Dataset” module to convert “NaN” to a value of 0. In this step, building ages of the buildings without rehabilitation will change to 0.



Convert to Dataset

Action
ReplaceValues

Replace
Custom

Custom value
NaN

New value
0

4. Use “Edit Metadata” to rename the column to “Build_Age_with_URA”

Properties Project

▲ Edit Metadata

Column

Selected columns:

Column names:

Multiply(Build_Ages_Rehab_

Launch column selector

Data type

Unchanged

Categorical


Unchanged

Fields

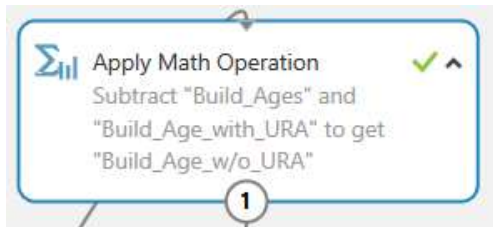
Unchanged

New column names

Build_Age_with_URA



5. Use “Apply Math Operation” to subtract [Build_Ages] and [Build_Age_with_URA].
 - a. In this step, for rehabilitated buildings, their building ages between [Build_Ages] and [Build_Age_with_URA] are the same and the difference should be 0. For the buildings without rehabilitation, the different of the subtraction should be equal to the [build_Ages].
 - b. The “Append” output mode appends a column at the end of the dataset with column name [Subtract(Build_Ages_Build_Age_with_URA)].



Apply Math Operation

Category
Operations

Basic operation
Subtract

Operation argument type
ColumnSet

Operation argument
Selected columns:
Column names:
Build_Age_with_URA

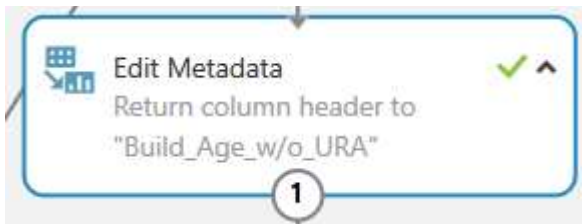
Launch column selector

Column set
Selected columns:
Column names:
Build_Ages

Launch column selector

Output mode
Append

6. Use "Edit Metadata" to rename the column to "Build_Ages_w/o_URA"



Edit Metadata

Column
Selected columns:
Column names:
Subtract(Build_Ages_Bui
< >

Launch column selector

Data type
Unchanged

Categorical
Unchanged

Fields
Unchanged

New column names
Build_Age_w/o_URA

7. Now, we have separated a column [Build_Ages] into 2 columns [Build_Age_with_URA] and [Build_Age_w/o_URA]. For buildings with rehabilitation, their building age should be discounted to reflect the effect of rehabilitation.
8. Use “Apply Math Operation” module to multiply [Build_Age_URA] by 2/3 or 0.66. The product of the math operation will create a decimal. The “Append” output mode appends a column to the end of the dataset with column name (Multiply(Build_Age_with_URA_\$0.66)).

Apply Math Operation

Category: Operations

Basic operation: Multiply

Operation argument type: Constant

Constant operation ar...: 0.66

Column set:

Selected columns:

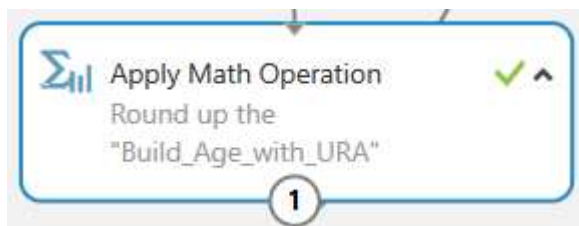
Column names:

Build_Age_with_URA

Launch column selector

Output mode: Append

9. Use “Appy Math Operation” module to round up the discounted building ages. “Constant Precision” is set to 0 to round up to an integer value with no decimal places. The “Inplace” output mode overwrites the result column in step 8 and the column name has no change (i.e. Multiply(Build_Age_with_URA_\$0.66)).



Apply Math Operation

Category
Rounding

Rounding operation
RoundUp

Precision Type
Constant

Constant Precision
0

Column set
Selected columns:
Column names:
Multiply(Build_Age_with
< >
Launch column selector

Output mode
Inplace

10. Use "Edit Metadata" module to rename the column header of "Multiply(Build_Age_with_URA_\$0.66)" to "Discount_Build_Age".



Edit Metadata

Column
Selected columns:
Column names:
Multiply(Build_Age_with
< >
Launch column selector

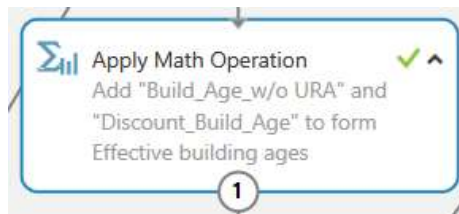
Data type
Unchanged

Categorical
Unchanged

Fields
Unchanged

New column names
Discount_Build_Age

11. Use “Apply Math Operation” module to add [Build_Age_w/o URA] and [Discount_Build_Age] to form Effective building ages. The “Append” output mode appends a column at the end of the dataset with column name (Add(Discount_Build_Age_Build_Age_w/o_URA)).



Apply Math Operation

Category
Operations

Basic operation
Add

Operation argument type
ColumnSet

Operation argument
Selected columns:
Column names:
Build_Age_w/o_URA
Launch column selector

Column set
Selected columns:
Column names:
Discount_Build_Age
Launch column selector

Output mode
Append

12. Use “Edit Metadata” module to rename column header to "Eff_Build_Ages".



Edit Metadata

Column
Selected columns:
Column names:
Add(Discount_Build_Age_Build_Age_w/o_URA)
< >
Launch column selector

Data type
Unchanged

Categorical
Unchanged

Fields
Unchanged

New column names
Eff_Build_Ages

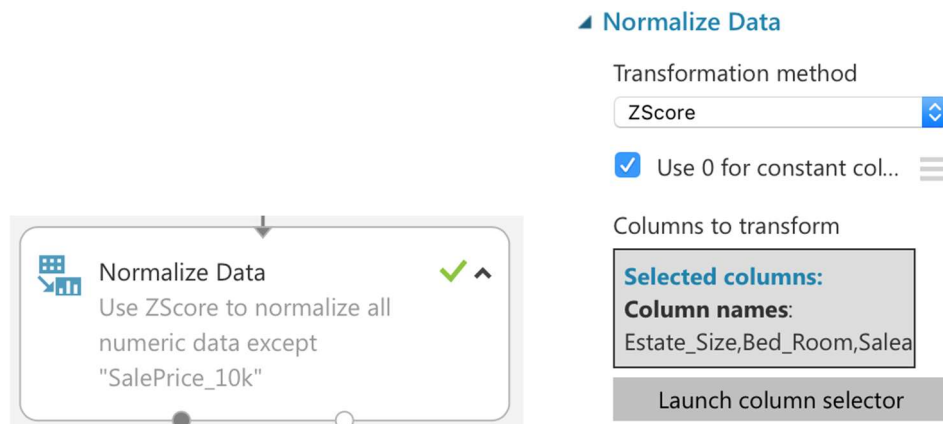
After the feature engineering of the above 3 features, now we start prepare the features for machine learning algorithms and model creation.

1. Use “Select Columns in Dataset” module to select [bed_room], [SalePrice_10K], [SaleableArea], [Eff_Build_Ages],[Bus_Route] and [Parks].
 - a. These features are selected because these were flagged as being significant to the analysis results in the Permutation Feature Importance (PFI) module. PFI will be described further in Week 4.

2. For selected numeric variable features, we want to make sure they are in numeric. Use “Edit Metadata” module to make numeric variable features numeric.

3. For all numeric features other than [SalePrice_10k], we apply a transformation method of “ZScore” to normalize the data, to bring about numeric features in a similar range. If this is not done, outliers or features with large values will result in the machine learning algorithm ‘overfitting’.

Use “Normalize Data” module with transformation method of “ZScore” for all numeric features except [SalePrice_10k].



Normalize Data

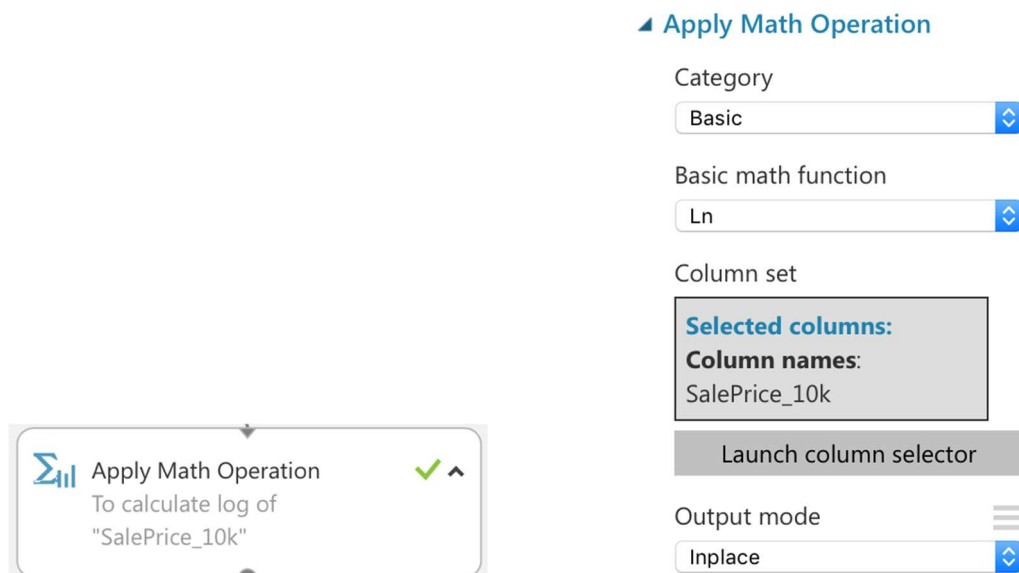
Transformation method
ZScore

☒ Use 0 for constant col...

Columns to transform
Selected columns:
Column names:
Estate_Size, Bed_Room, Salea

Launch column selector

4. The predictor or target variable [SalePrice_10k], is a numeric value with ranging from 200 to 18800. We apply a log (Ln) function on the [SalePrice_10k] to transform it to a similar dimension with other features given in step 3.
 - a. In the future tutorial, we will use exponential function to reserve the predicted value in ln function to predicted [SalePrice_10k].
 - b. Use “Apply Math Operation” module to select “Basic” in Category and “Ln” in Basic math function as show in below screens. The “Inplace” output mode make the Ln results overwrite in the same column.



Apply Math Operation

Category
Basic

Basic math function
Ln

Column set
Selected columns:
Column names:
SalePrice_10k

Launch column selector

Output mode
Inplace

End of Step 3 – Feature Engineering

Step 4 – Choosing a Machine Learning Algorithm

Recall that the objective of this tutorial is to make a profit from investing in and the eventual sale of invested properties, by creating a model that accurately predicts the future sales price of a private property based on historical property transactions.

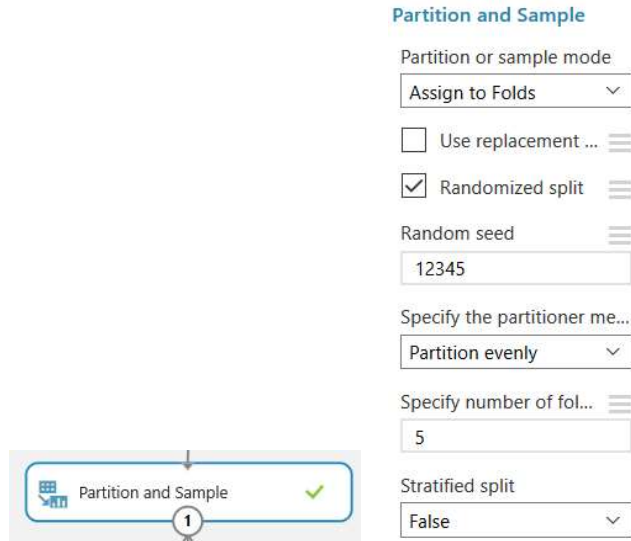
Machine learning can be classified into supervised, unsupervised and reinforcement learning algorithms. Supervised algorithms make predictions based on a set of labeled data – or examples. For instance, historical property sale prices can be used to predict future sale prices. Each example used for training is labeled with the value of interest—in this case the sales price. A supervised learning algorithm looks for patterns in that target variable. It can use any information that might be relevant—the size, effective age, number of bed rooms of the property—and each algorithm looks for different types of patterns. After the algorithm has found the best pattern, it uses that pattern to make predictions for unlabeled testing data—tomorrow's sales price that is unknown.

Given that we're predicting future sales price, we will use a class of supervised algorithms called Regression.

Partition and Sampling data into 5 evenly distributed datasets

Partition and Sampling module is used to first partition our data into 5 evenly distributed folds. Sampling is an important tool in machine learning because it lets us reduce the size of a dataset while maintaining the same ratio of values. This module supports several related tasks that are important in machine learning, such as dividing our data into multiple subsections of the same size. To facilitate use of the partitions for cross-validation.

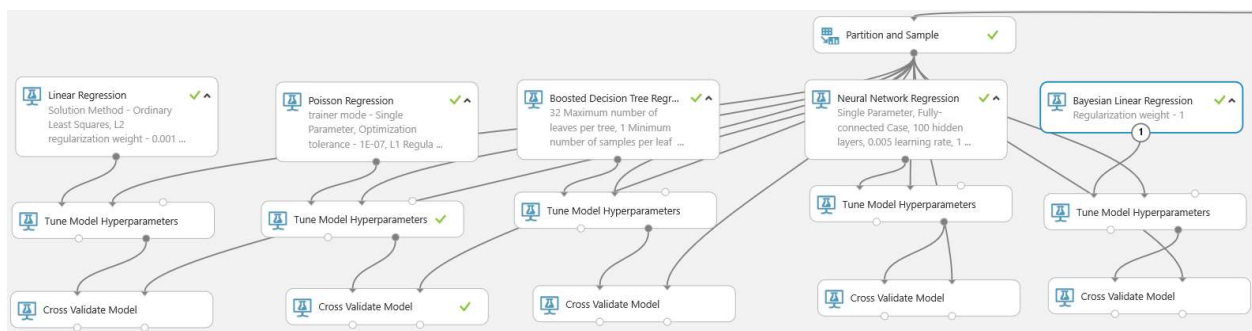
The Partition and Sample module is given below.



The terms parameter and hyperparameter can be confusing. The model's parameters are what you set in the properties pane. Basically, this module performs a parameter sweep over the specified parameter settings, and learns an optimal set of hyperparameters, which might be different for each specific decision tree, dataset, or regression method. The process of finding the optimal configuration is sometimes called tuning.

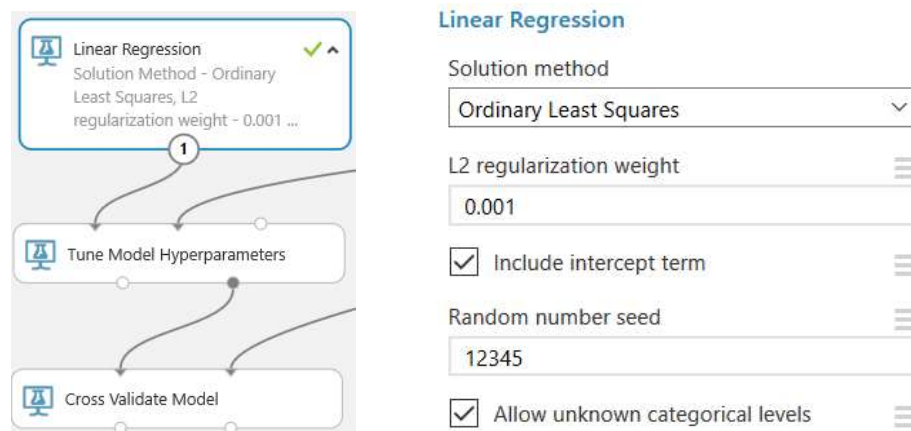
Random sweep trains a model using a set number of iterations, we specify a range of values to iterate over, and the module uses a randomly chosen subset of those values. Values are chosen with replacement, meaning that numbers previously chosen at random are not removed from the pool of available numbers. Thus, the chance of any value being selected remains the same across all passes. We choose a maximum of 5 runs of random sweep with a random seed of 12345.

Cross validation is used to evaluate the predictive performance of the model as well as that performance's stability regarding new data. Cross validation will build ten different models on the same algorithm but with different and non-repeating subsets (using a randomized seed of 12345) of the same dataset. The evaluation metrics on each of the ten models will be averaged and a standard deviation will infer the stability of the average performance.

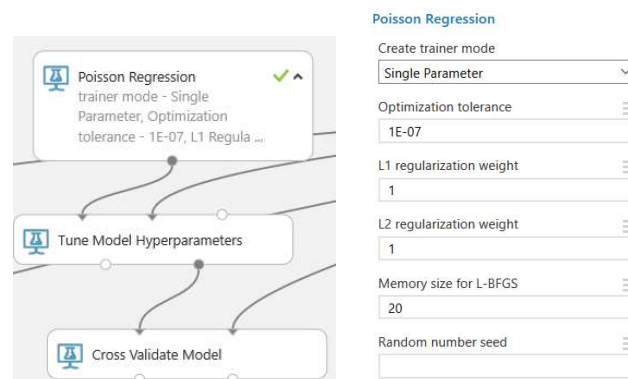


As given above 5 different machine learning algorithms were selected for regression. Regression is a supervised learning algorithm and we evaluate the Coefficient of Determination (COD) and the Root Mean Squared Error (RMSE) as key indicators.

Linear Regression uses Ordinary Least Squares solution method, L2 regularization weight - 0.001 and a Random No. Seed Of 12345. Using the Tuned Model Hyperparameters the Coefficient of Determination was 0.734144 or 73% and the Root Mean Squared Error was 0.258381.



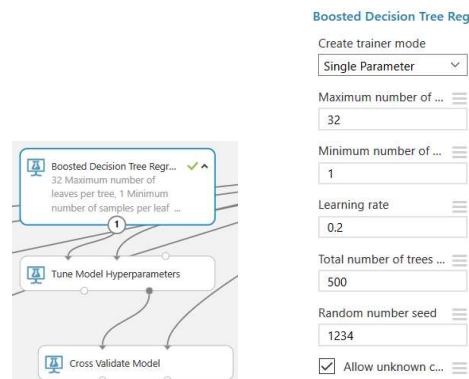
Poisson Regression uses Single Parameter trainer mode, Optimization tolerance - 1E-07, L1 Regularization weight - 1, L2 Regularization weight, Memory size for L-BFGS – 20. Using the Tuned Model Hyperparameters the Coefficient of Determination is 0.69041 or 69% and the Root Mean Squared Error is 0.278825.



Boosted Decision Tree Regression uses the default of 32 Maximum number of leaves per tree, 1 Minimum number of samples per leaf node, 0.2 learning rate, 500 trees were constructed. Using the Tuned Model Hyperparameters with 17 leaves per tree, 6 minimum samples per leaf node, 0.05039 learning rate, 239 trees were constructed gave the highest Coefficient of Determination of 0.884467 or 88% and the Root Mean Squared Error was 0.17.

This is marginally better than using a split data of 80% and test data of 20% and a COD of 0.862982 86.2% and RSME of 0.20. Which uses hyperparameters of 20 Maximum number of leaves per tree, 10 Minimum number of samples per leaf node, 0.2 learning rate, 100 trees constructed.

Upon making the recommended changes from the Tuned Model Hyperparameters, the new COD was 0.893643 or 89.3% with a RSME of 0.17. Hence the Boosted Decision Tree Regression algorithm was selected as the machine learning algorithm as it gave the highest COD and lowest RSME.



Boosted Decision Tree Reg

Create trainer mode
Single Parameter

Maximum number of ...
32

Minimum number of ...
1

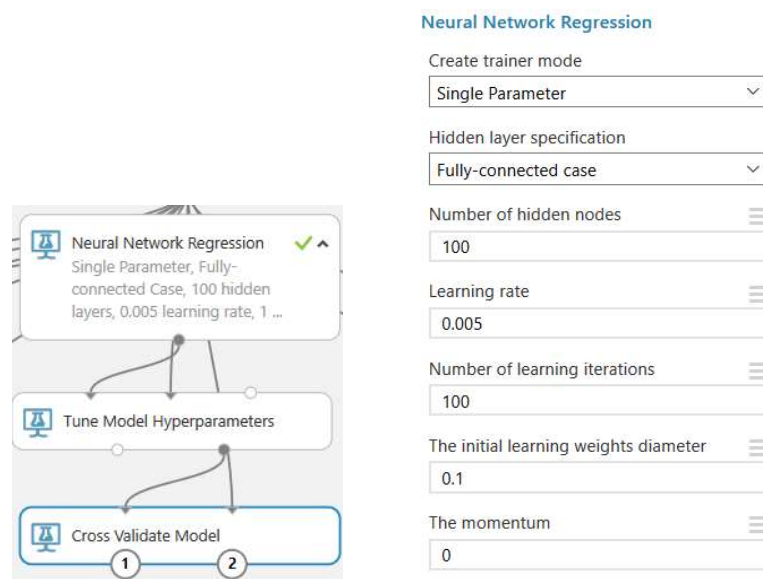
Learning rate
0.2

Total number of trees ...
500

Random number seed
1234

☒ Allow unknown c...

Neural Network Regression uses a Single Parameter for trainer mode, Fully-connected Case for the hidden layer, 100 hidden layers, 0.005 learning rate, 100 learning iterations, 0.1 initial learning weights, min-max normalizer. Using the Tuned Model Hyperparameters the Coefficient of Determination was 0.700239 or 70% and the Root Mean Squared Error was 0.274363.



Neural Network Regression

Create trainer mode
Single Parameter

Hidden layer specification
Fully-connected case

Number of hidden nodes
100

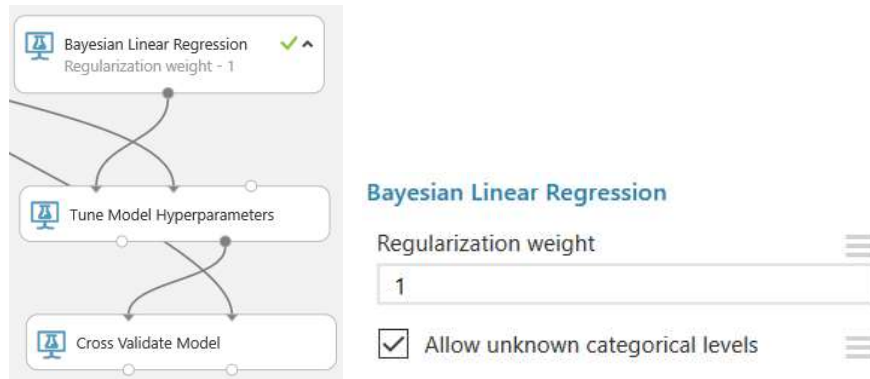
Learning rate
0.005

Number of learning iterations
100

The initial learning weights diameter
0.1

The momentum
0

Bayesian Linear Regression uses a Regularization weight of 1. Using the Tuned Model Hyperparameters the Coefficient of Determination was 0.732446 or 73% and the Root Mean Squared Error was 0.259205.



Summary

Linear Regression COD was 73% and the RSME was 0.26.

Poisson Regression COD was 69% and the RSME was 0.28.

Boosted Decision Tree Regression COD was 89.3% and RSME was 0.17 (Winner)

Neural Network Regression COD was 70% and RSME was 0.27.

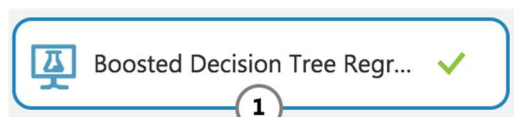
Bayesian Linear Regression COD was 73% and RSME was 0.26.

End of Step 4 – Choosing a Machine Learning Algorithm

Step 5 – Machine Learning Model Evaluation

Based on the Step 4 – Choosing a Machine Learning Algorithm results, let's choose “Boosted Decision Tree Regression” as our final model for machine learning with the optimized parameters identified from “Tune Model Hyperparameters” module.

Number of leaves	Minimum leaf instances	Learning rate	Number of trees	Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Relative Squared Error	Coefficient of Determination
17	6	0.05039	239	0.118444	0.17033	0.329687	0.115533	0.884467



Boosted Decision Tree Regressi...

Create trainer mode

Single Parameter

Maximum number of leav...

17

Minimum number of sam...

6

Learning rate

0.05039

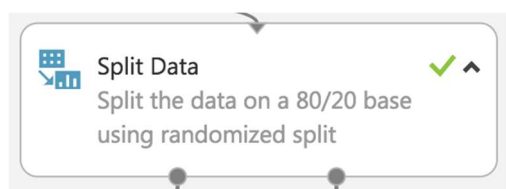
Total number of trees con...

239

Random number seed

12345

We also split our dataset into training and test set. As a rule of thumb, 80% of the data is split between training data with 20% as test data.



Split Data

Splitting mode

Split Rows

Fraction of rows in the first output...

0.8

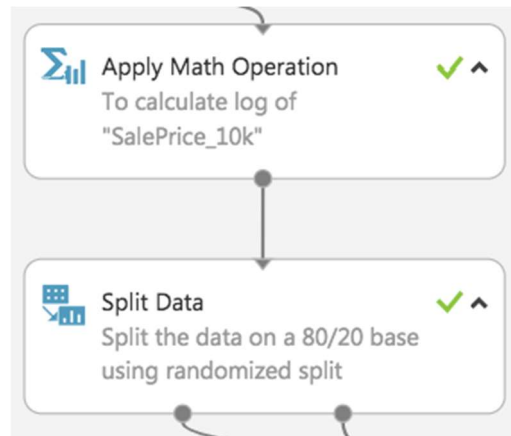
☒ Randomized split

Random seed

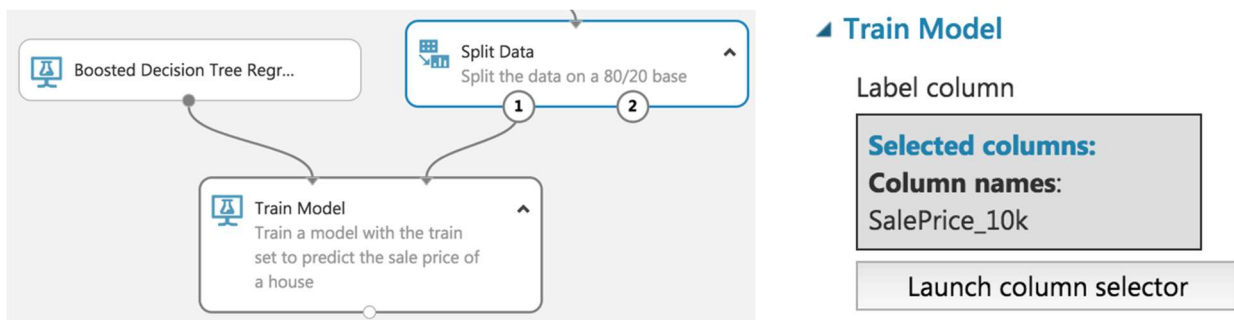
1234

Stratified split

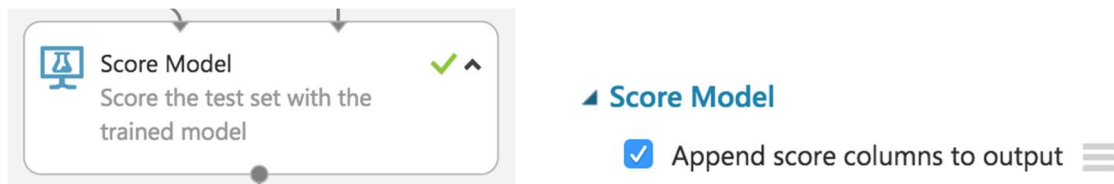
False



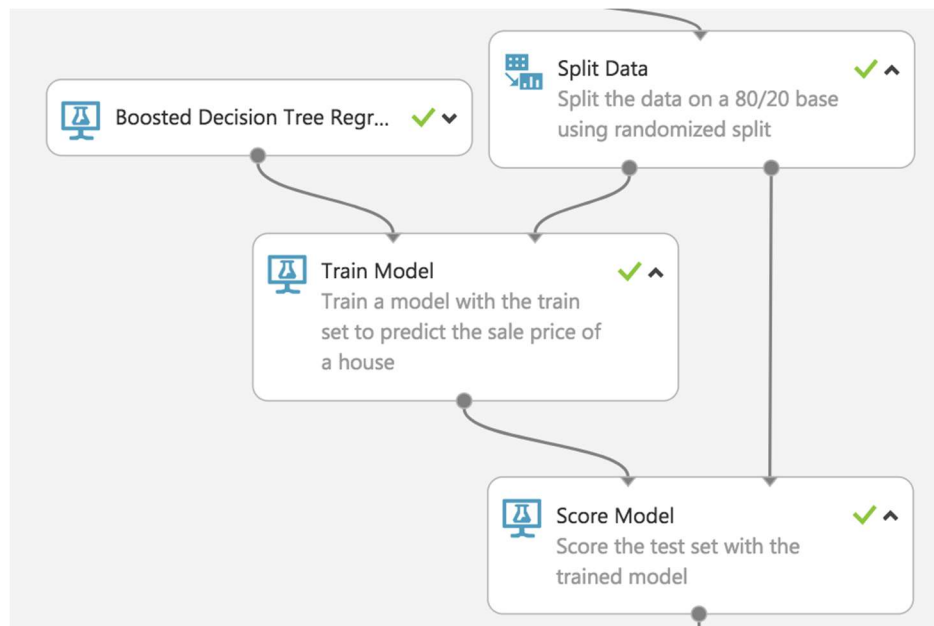
Connect the Boosted Decision Tree Regression module and Split Data module to “Training Model” module. Select [SalePrice_10k] in the property of “Train Model” module.



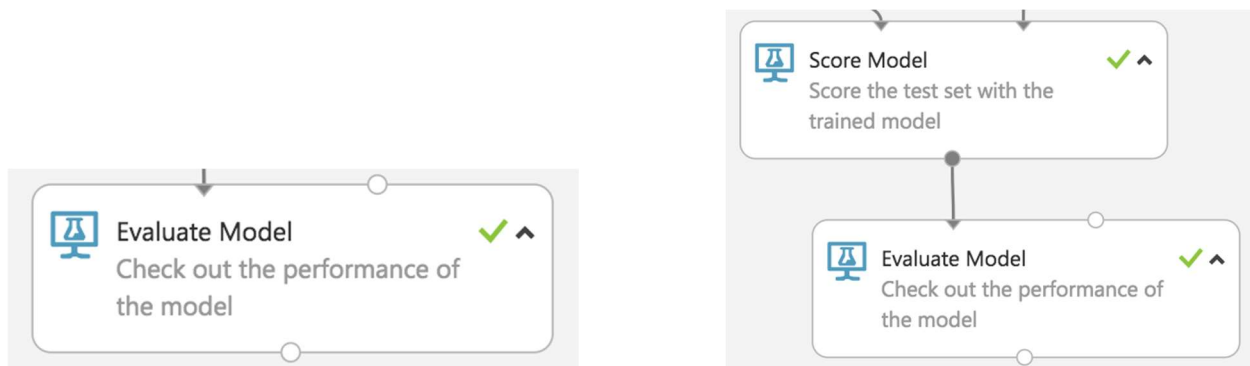
Use “Score Model” to score the test set with the train model. Connect the output of “Train Model” and test dataset from “Split Data” module.



Connect the output of the “Train Model” and test dataset to “Score Model” module.



Use “Evaluate Model” to check out the performance of the model. Connect the output of the “Score Model” module to “Evaluate Model” module.




Visualize the evaluation results that RMSE and COD are 0.17 and 0.89 respectively.

Metrics

Mean Absolute Error	0.125995
Root Mean Squared Error	0.176737
Relative Absolute Error	0.325627
Relative Squared Error	0.106357
Coefficient of Determination	0.893643

Use “Apply Math Operation” to calculate the exponential of [SalePrice_10k] and [Scored Labels]. The exponential of [Scored Labels] is the predicted sale price.



Apply Math Operation
Compare the true SalePrice and predict SalePrice

▲ Apply Math Operation

Category
Basic


Basic math function
Exp

Column set
Selected columns:
Column names: Scored Labels, SalePrice_10k

Launch column selector

Output mode
Append

Use “Permutation Feature Importance” module to check out the importance of the features.



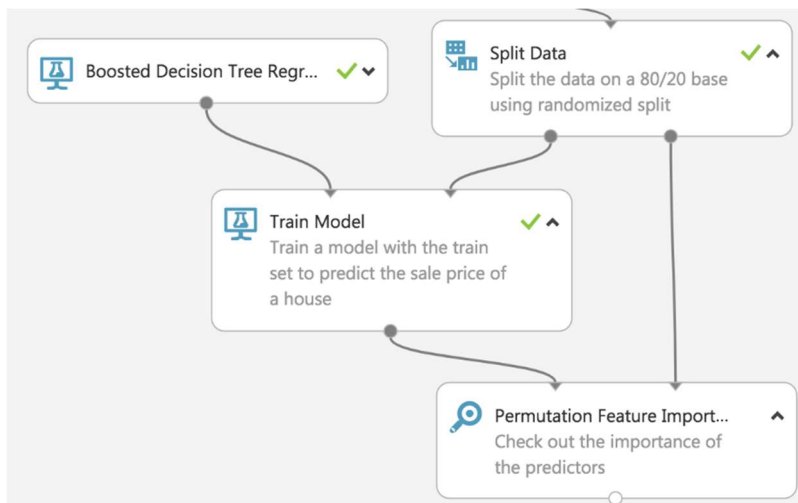
Permutation Feature Import...
Check out the importance of the predictors

▲ Permutation Feature Importance



Random seed
1234

Metric for measuring performance
Regression - Coefficient of Determina

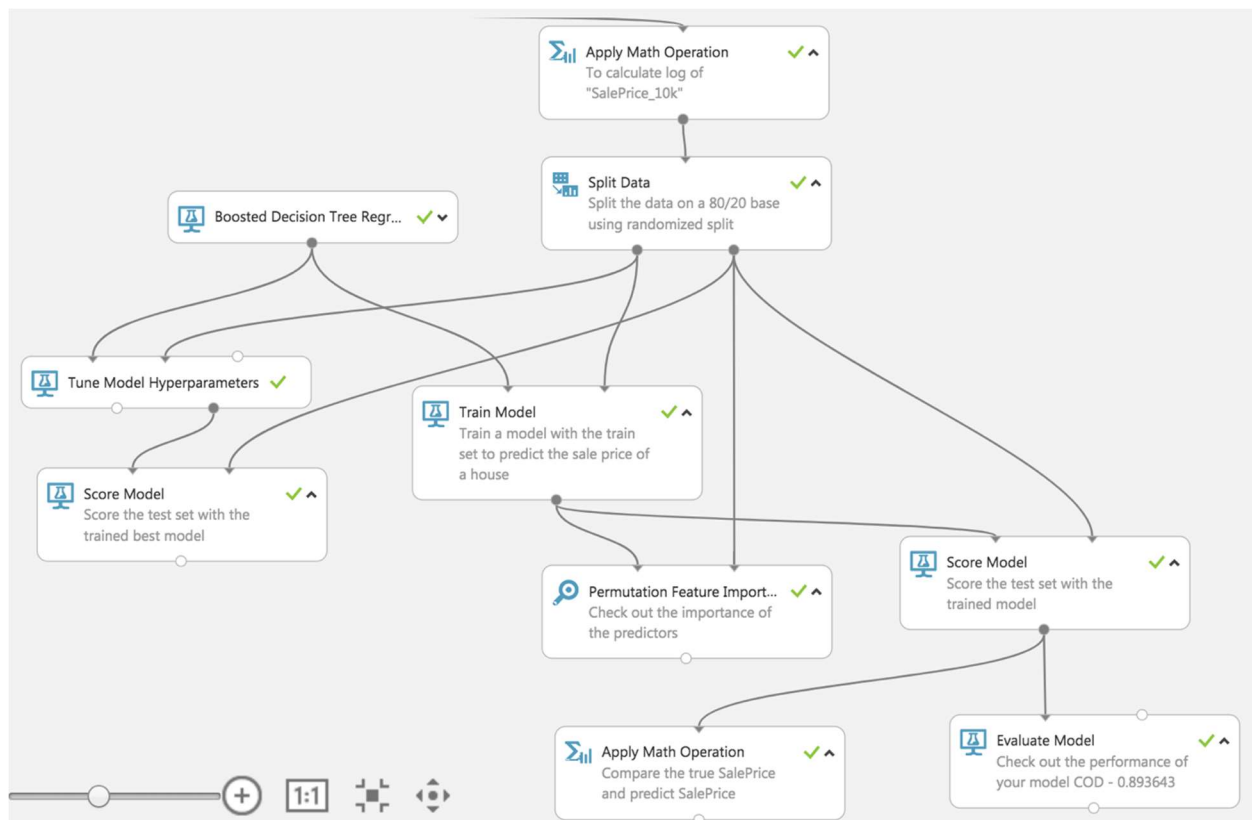
Connect the output of the “Train Model” and test dataset to “Permutation Feature Importance” module.



Visualize the Feature importance to identify the features with their score.

Feature	Score
	
SaleableArea	0.858545
Eff_Build_Ages	0.166877
Bus_Route	0.076519
Parks	0.019011
Floor_(H/M/L)	0.015987
Edu_Inst	0.00912
Roof	0.00192
Bed_Room	0.001571
Estate_Size	0.000229

Below shows the Azure Machine Learning experiment on step 5 – Machine Learning model evaluation.



End of Step 5 –Machine Learning Model Evaluation

Step 6 – Deploy Machine Learning Model As A Web Service

From the above step 1 to 5, we have established a machine learning model with an acceptable score on COD and RMSE. In the step 6, we updated our training experiment to predictive experiment and then deploy to web service.

Recall the purpose of our machine learning model and identify the features for input and the predicted output.

The features for input: [Bed_Room], [Roof], [SaleableArea], [Parks], [Bus_Route], [Floor], [Kindergarten], [Primary_Schools], [Secondary_Schools], [Estate_Size], [Build_Ages], [Rehab_Year]

The feature [SalePrice_10k] is used in our training experiment but it should be represented as our predicted output. So, the [SalePrice_10k] will not be included in our features for input.

The predicted output: [Exp(Scored Labels)]

Review the training experiment and remove the necessary modules. In our training experiment:

the feature [Tower] is removed and it is not included in our machine learning model. So, we should remove that “Clean Missing Data” module.

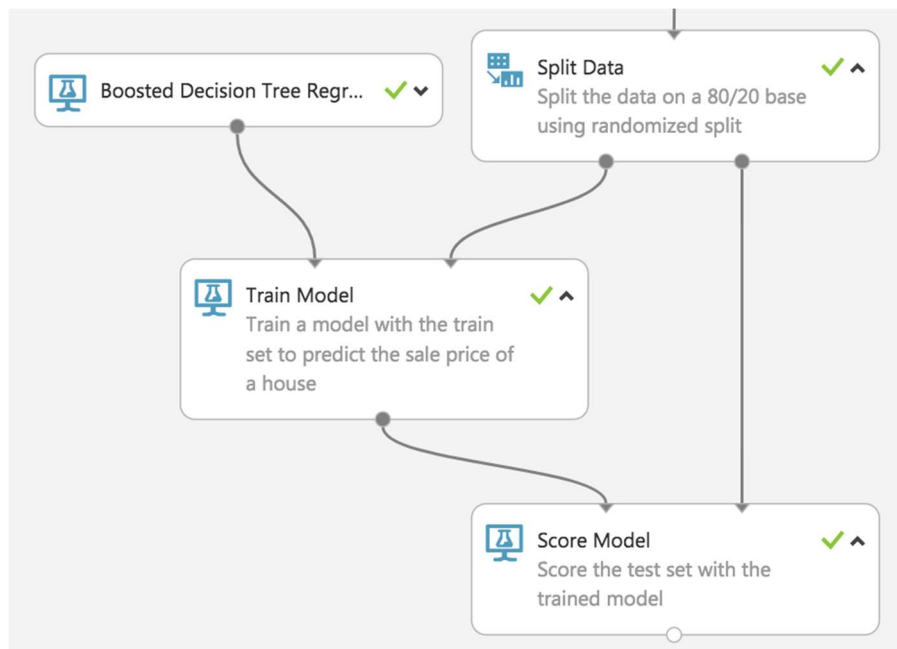
The observations with missing value in the feature [SaleableArea] were removed for machine learning model, but we must input [SaleableArea] for predict output. So, we should remove that “Clean Missing Data” module.

The step 2 – Data cleaning is updated from 4 modules to 2 modules.



The step 4 – Choosing ML Algorithm is used to select the best algorithm and related parameters. For our predictive experiment, all the modules in step 4 are not relevant and they should be removed.

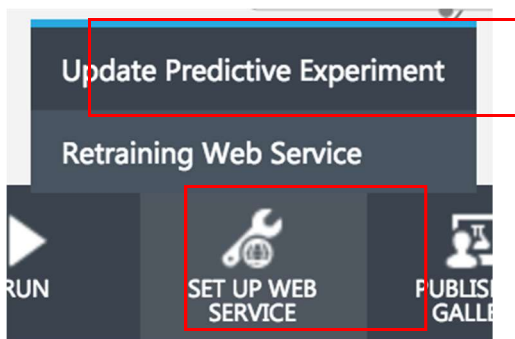
The step 5 - ML Model Evaluation is used to build our training model and score the predict output. For our predictive experiment, only 4 modules “Split Data”, “Boosted Decision Tree Regression”, “Train Model” and “Score Model” should be remained and the rest module should be removed.



Run the whole training experiment to make sure no error in the experiment. The training experiment should look like below:



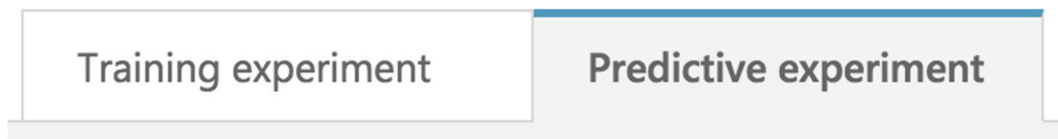
Select "Update Predictive Experiment" from "SET UP WEB SERVICE"



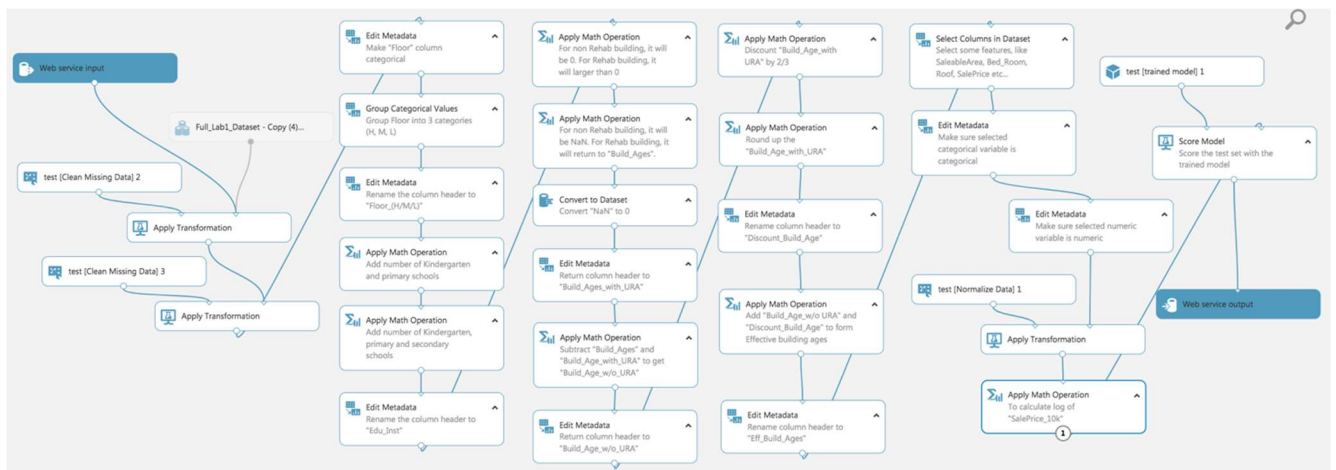
When update complete, click “CLOSE”



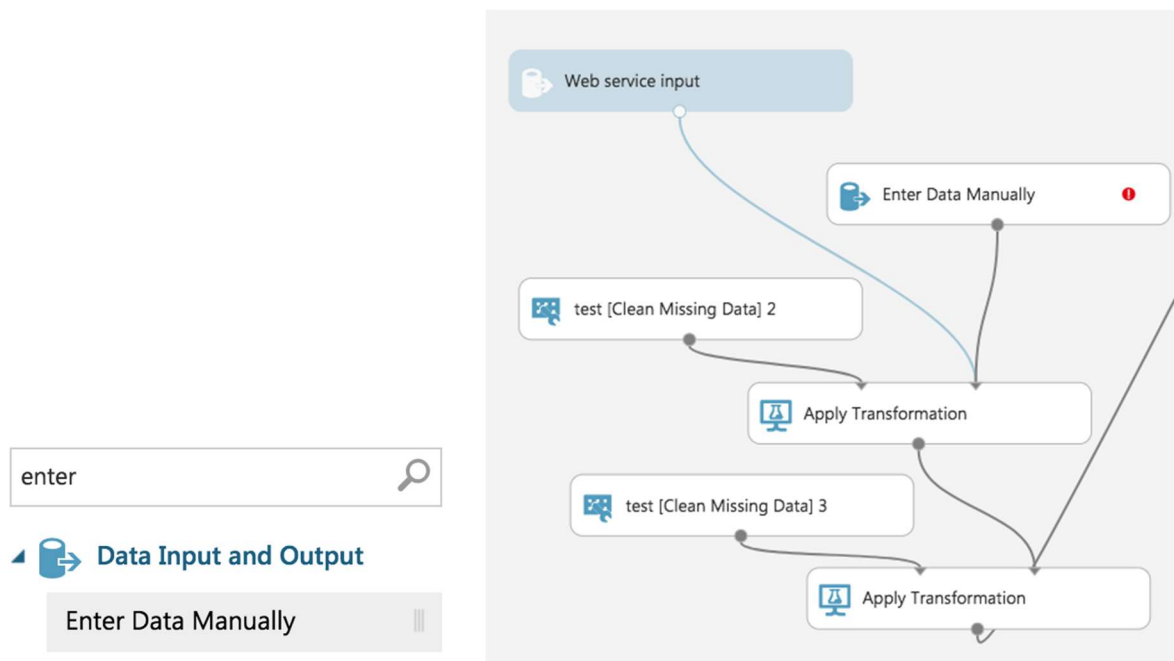
A new tab of Predictive experiment is created.



Rearrange the modules in the Predictive experiment and it should look like below.



As we mentioned before, the initial dataset “Full_Lab1_Dataset” should be replaced by another manual dataset that has the 12 features for input. Select “Enter Data Manually” module and replace “Full_Lab1_Dataset” into the predictive experiment.



The dataset for “Enter Data Manually” is prepared and named “Step 6 - Enter Data Manually.csv”. Open the csv file by text editor, select all data by click <Ctrl><a> and then copy the selected data by click <Ctrl><c>

```

Estate_Size,Floor,Bed_Room,Roof,Build_Ages,Rehab_Year,SaleableArea,Kindergarten,Primary_Schools,Secondary_Schools,Parks,Bus_Route
1,3,,N,40,,464,1,2,1,6,52
1,19,2,N,24,,333,1,2,0,4,29
1,6,2,N,23,,333,1,2,0,4,29
1,20,,N,39,,1,1,1,9,31
1,2,,N,51,,370,1,0,1,7,45
1,3,,N,51,,370,1,0,1,7,45
1,6,,N,39,,539,3,2,1,3,44
1,16,,N,38,,539,3,2,1,3,44
1,16,,N,38,,506,3,2,1,3,44
1,15,,N,38,,539,3,2,1,3,44
1,17,2,N,24,,511,3,2,1,3,46
1,6,2,N,23,,511,3,2,1,3,46
1,8,2,N,23,,511,3,2,1,3,46
1,27,2,N,23,,511,3,2,1,3,46
1,5,,N,30,,447,2,2,1,2,40
1,22,,N,23,,393,3,2,1,2,40
1,6,3,N,29,,481,3,2,1,2,48
1,8,3,N,31,,526,3,2,1,2,48

```

Switch to Predictive experiment and select the “Enter Data Manually” module, paste the data

Enter Data Manually

DataFormat: CSV

☒ HasHeader

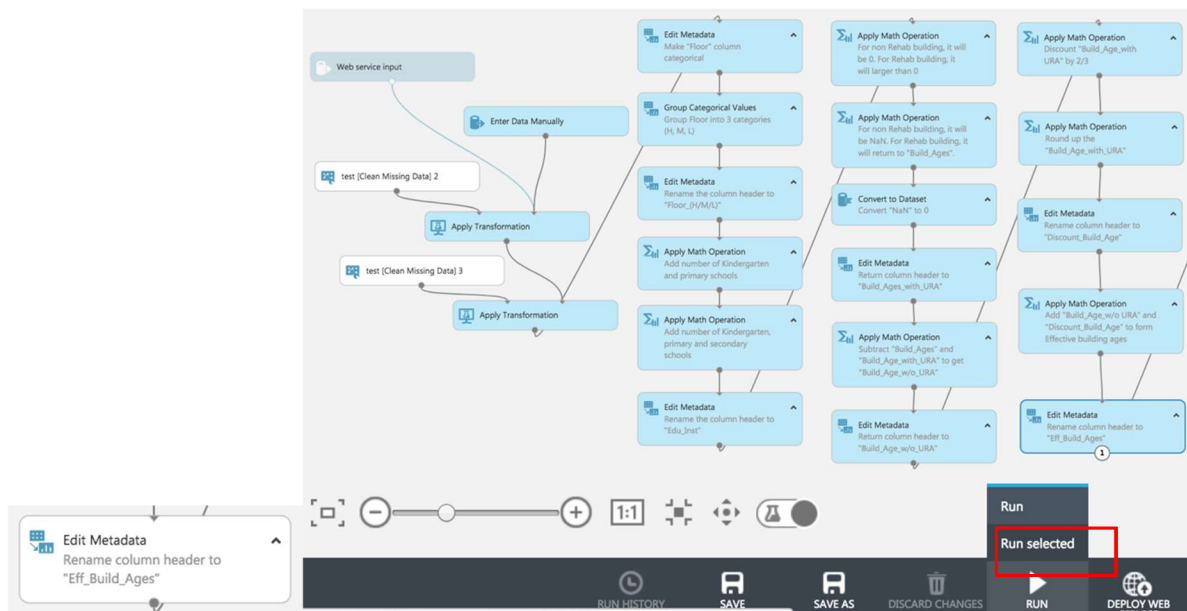
Data

```

583 1,28,,Y,26,,450,3,4,0,7,45
584 1,25,,N,25,,386,3,4,0,7,45
585 1,26,,N,25,,450,3,4,0,7,45
586 1,1,,N,25,,450,3,4,0,7,45
587 1,22,,Y,35,,269,3,3,0,6,42
588 1,10,,N,35,,280,3,3,0,6,42
589 1,1,,N,34,,260,3,3,0,6,42
590 1,20,,N,34,,269,3,3,0,6,42
591 1,4,,N,34,,260,3,3,0,6,42
592 1,17,,N,34,,397,3,3,0,6,42
593 1,2,,N,34,,260,3,3,0,6,42
594 1,4,,N,34,,260,3,3,0,6,42
595 1,10,,N,34,,280,3,3,0,6,42
596 1,12,,N,34,,260,3,3,0,6,42
597 1,11,,N,33,,280,3,3,0,6,42
598 1,3,,N,33,,280,3,3,0,6,42
599 1,2,,N,33,,280,3,3,0,6,42
600 1,5,,N,33,,260,3,3,0,6,42
601

```

Select the module below and click “run select”

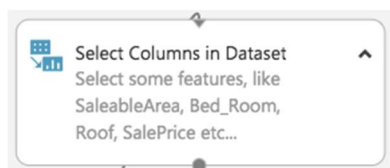


As we mentioned before, the [SalePrice_10k] should not be included in the “Enter Data Manually” module. So, if the [SalePrice_10k] is used in the modules of Predictive experiment, it should be removed.

Select the module below and click “Launch column selector” and remove [SalePrice_10k]

Select Columns in Dataset

Select columns



Selected columns:
Column names:
Bed_Room, Roof, SalePrice_10k

Launch column selector

Begin With

ALL COLUMNS

NO COLUMNS

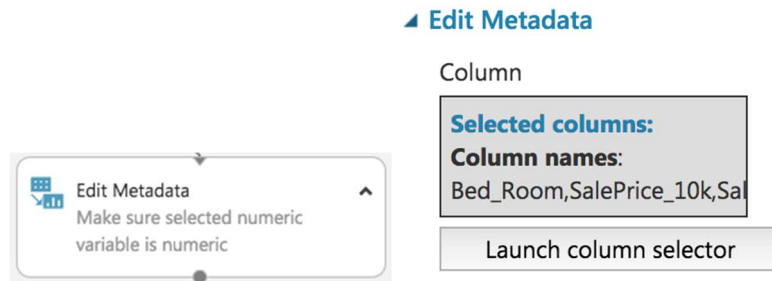
Include

column names

Bed_Room X Roof X SaleableArea X Parks X
Bus_Route X Floor_(H/M/L) X Edu_Inst X
Estate_Size X Eff_Build_Ages X SalePrice_10k X



Select the module below and click “Launch column selector” and remove [SalePrice_10k]



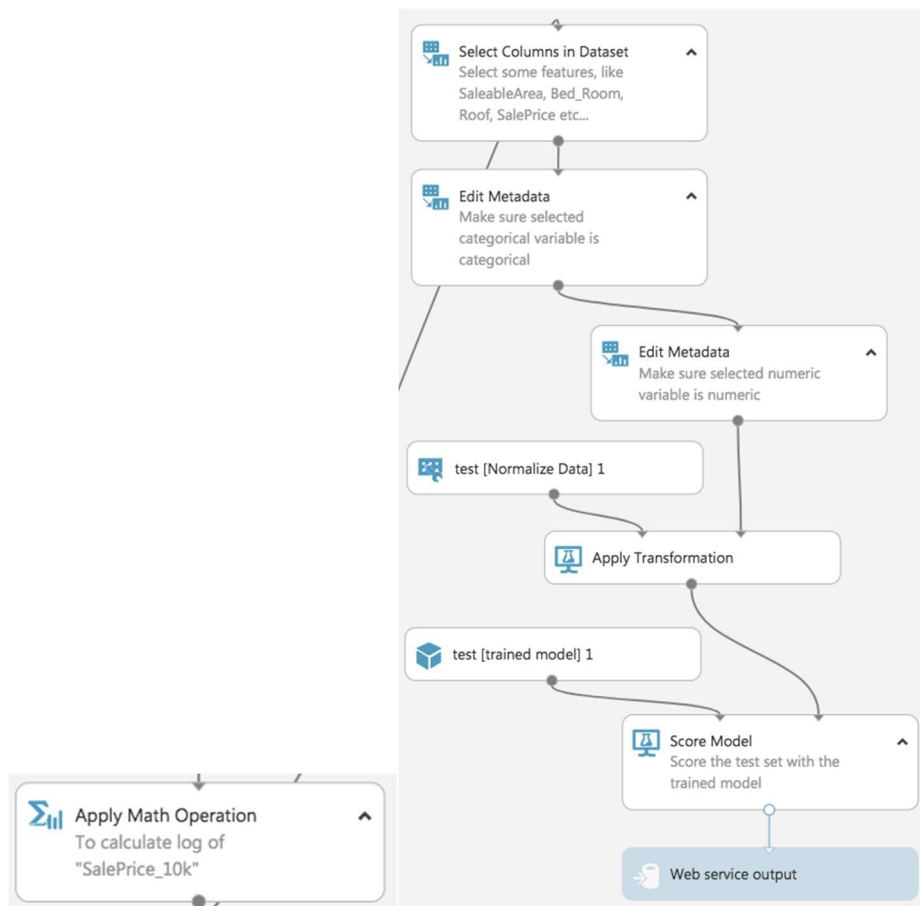
Begin With

ALL COLUMNS NO COLUMNS

Include column names

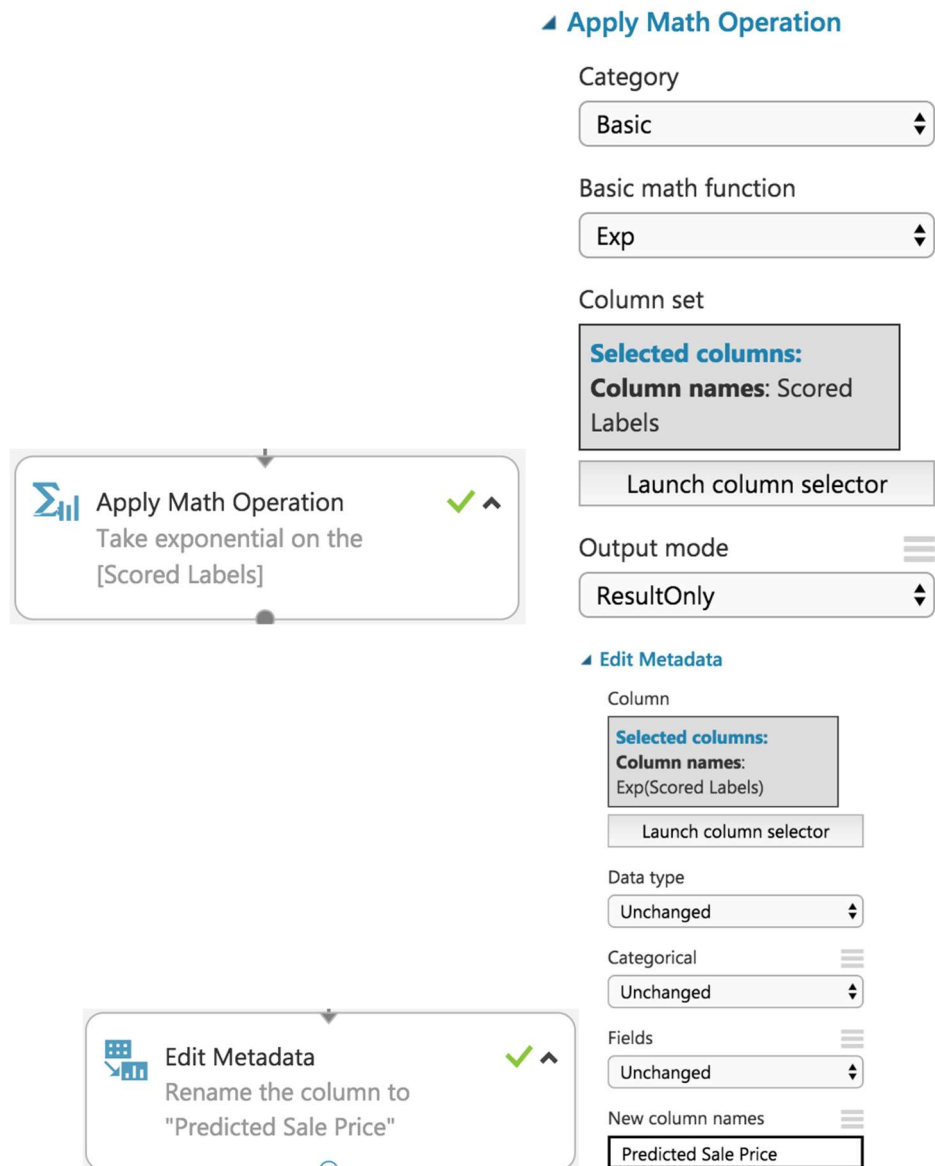
Bed_Room x SaleableArea x Parks x Bus_Route x
 Edu_Inst x Estate_Size x Eff_Build_Ages x
 SalePrice_10k x

Select the module below and delete it and connect the modules as shown in below.



Run the Predictive experiment to ensure no errors.

As we mentioned above, the [Scored Labels] from “Score Model” module is our predicted output but it is in ln mode. We need to take exponential on the [Scored Labels] and rename the column label.



Apply Math Operation

Category: Basic

Basic math function: Exp

Column set: Selected columns: Column names: Scored Labels

Launch column selector

Output mode: ResultOnly

Edit Metadata

Column: Selected columns: Column names: Exp(Scored Labels)

Launch column selector

Data type: Unchanged

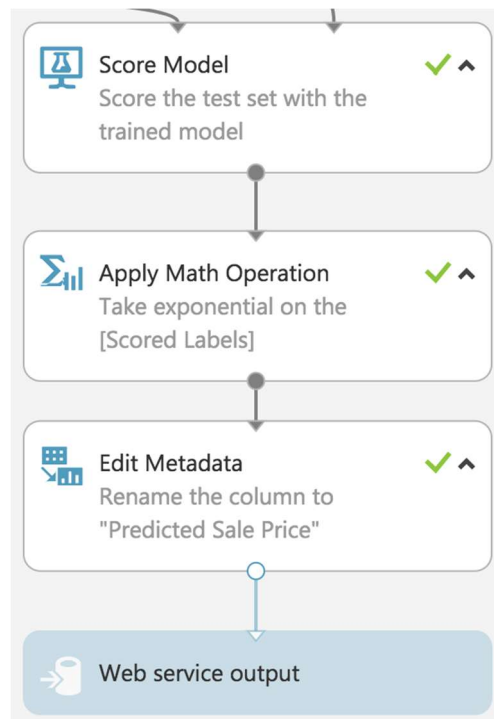
Categorical: Unchanged

Fields: Unchanged

New column names: Predicted Sale Price

Apply Math Operation
Take exponential on the [Scored Labels]

Edit Metadata
Rename the column to "Predicted Sale Price"

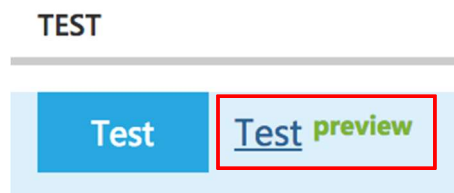


Run the whole Predictive experiment to ensure no error.

Select “DEPLOY WEB SERVICE” to convert Predictive experiment to Web Service.



Test the web service by click on the link.



Input the features and click “Test Request-Response” button to predict sale price.

Estate_Size	<input type="text" value="1"/>	Predicted Sale Price	784.714096175767
Floor	<input type="text" value="50"/>		
Bed_Room	<input type="text" value="1"/>		
Roof	<input type="text" value="N"/>		
Build_Ages	<input type="text" value="3"/>		
Rehab_Year	<input type="text" value="0"/>		
SaleableArea	<input type="text" value="313"/>		
Kindergarten	<input type="text" value="10"/>		
Primary_Schools	<input type="text" value="10"/>		
Secondary_Schools	<input type="text" value="10"/>		
Parks	<input type="text" value="10"/>		
Bus_Route	<input type="text" value="10"/>		

Test Request-Response