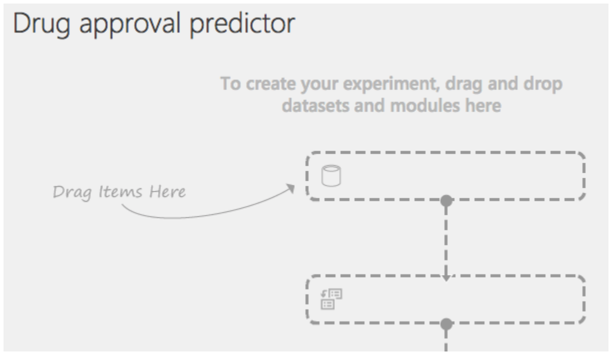
In this lab, we are going to apply machine-learning techniques using Microsoft's Azure Machine Learning Studio to predict drug approvals. We will use Citeline data provided by Informa Pharma Intelligence to analyze a clinical trial dataset with around 1,300 observations and 40 features. If you have not already done so, please sign-up for a free Microsoft Azure account by visiting https://studio.azureml.net, clicking on "Sign up here", and selecting the "Free Workspace" option (you will also need to sign up for a free Microsoft account if you do not already have one).

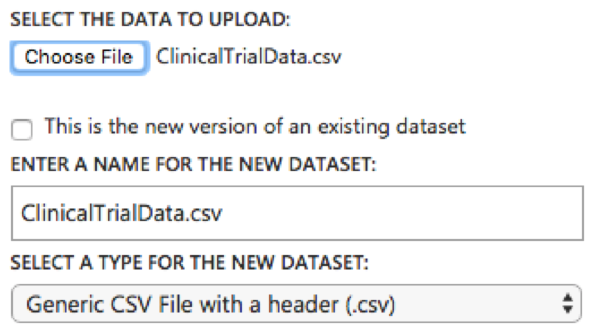
1 Creating a new experiment in Azure ML Studio

Sign-in to the Microsoft Azure Machine Learning Studio so you can create your first experiment. At the bottom of the screen select **+NEW**, then select **+Blank Experiment**. Change the title at the top of the screen from 'Experiment created on MM/DD/YYYY' to 'Drug approval predictor.'


Drug approval predictor
To create your experiment, drag and drop datasets and modules here
Drag Items Here

2 Loading the data set

Download the clinical trial dataset, "ClinicalTrialData.csv" to your local machine by clicking here. Definitions for each column are available in the file "Features.csv", which is available here. Once you have downloaded the dataset, you will need to upload it to the Azure Machine Learning Studio. Select **+NEW** at the bottom of the screen, and then select **DATASET** > **FROM LOCAL FILE**. Enter a description of the dataset, and then click the checkmark to upload the data.



SELECT THE DATA TO UPLOAD:

Choose File ClinicalTrialData.csv

This is the new version of an existing dataset

ENTER A NAME FOR THE NEW DATASET:

ClinicalTrialData.csv

SELECT A TYPE FOR THE NEW DATASET:

Generic CSV File with a header (.csv)

Expand **Saved Datasets** > **My Datasets** and drag your newly created clinical trial dataset into the experiment.

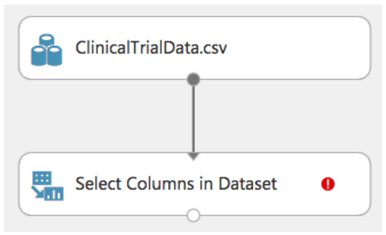
In any machine learning experiment, it is important to be familiar with your data. Right-click on your dataset module, and select **dataset** > **Visualize**. The dataset contains information about clinical trial designs and trial outcomes. Explore the dataset by clicking on the different columns.

What percentage of trials in the dataset have successful outcomes?

3 Selecting features for the ML experiment

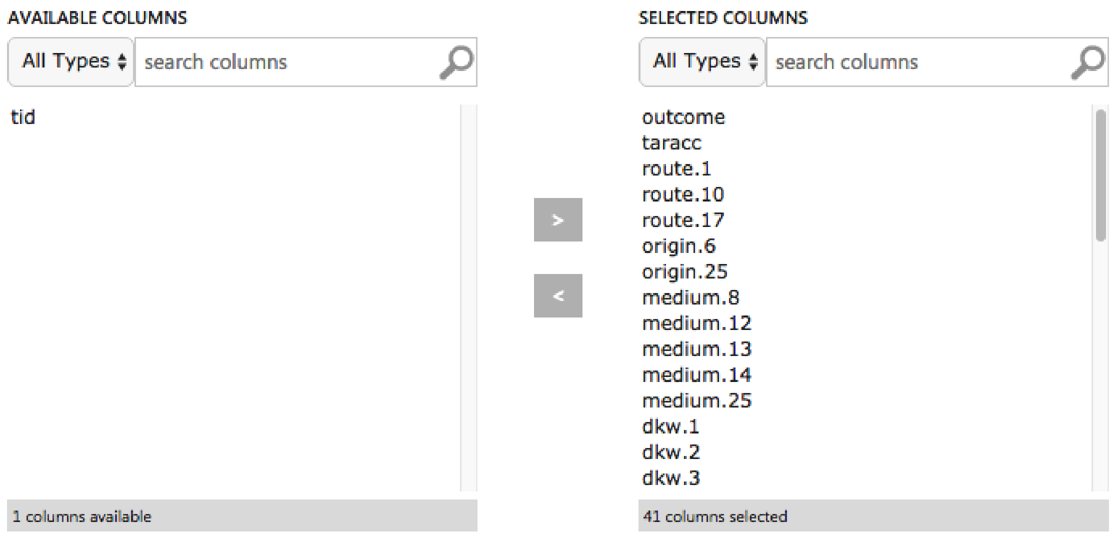
Some of the features in the dataset may not be meaningful for predicting clinical trial success. For example, Trial ID ("tid") is just a number assigned to each trial, and so it isn't going to help us predict the outcome.

Type "**Select**" into the search bar and drag the **Select Columns in Dataset** module into the experiment. Connect the output of your dataset to the column selector input.



The **Select Columns in Dataset** module allows you to specify which columns in the data set you think are significant to a prediction (i.e. your features). You need to look at the data in the dataset and decide which columns represent data that you think will affect whether or not a trial will be successful. You also need to include the column you want to predict. In this case we are going to try to predict the value of "outcome." This is a 0/1 column that indicates whether or not a trial was successful.

Click on the **Select Columns in Dataset** module. In the properties panel on the right hand side, select **Launch column selector**. Let's select all columns except Trial ID.



AVAILABLE COLUMNS

tid

1 columns available

SELECTED COLUMNS

outcome
taracc
route.1
route.10
route.17
origin.6
origin.25
medium.8
medium.12
medium.13
medium.14
medium.25
dkw.1
dkw.2
dkw.3

41 columns selected

4 Feature scaling

Since the range of values of raw data varies widely, some machine learning algorithms will not work properly without normalization. Therefore, it is common practice to normalize the range of all features so that they are approximately on the same scale. The simplest method, known as Min-Max scaling, rescales features to range between [0, 1]. The formula is given as,

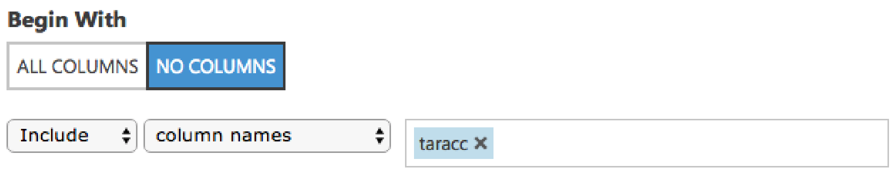
$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{1}$$

where x are the original values, and x' are the normalized values.

The only feature in our dataset that needs to be rescaled is target accrual ("taracc"), which provides the clinical trial's targeted sample size (i.e., number of patients in the study).

Type "**Normalize**" into the search bar and drag the **Normalize Data** module into the experiment. Connect the output of **Select Columns in Dataset** to the **Normalize**

Data input. In the properties window, use the "MinMax" transformation method, and use the column selector to choose "taracc."



Begin With

ALL COLUMNS | NO COLUMNS

Include | column names | taracc

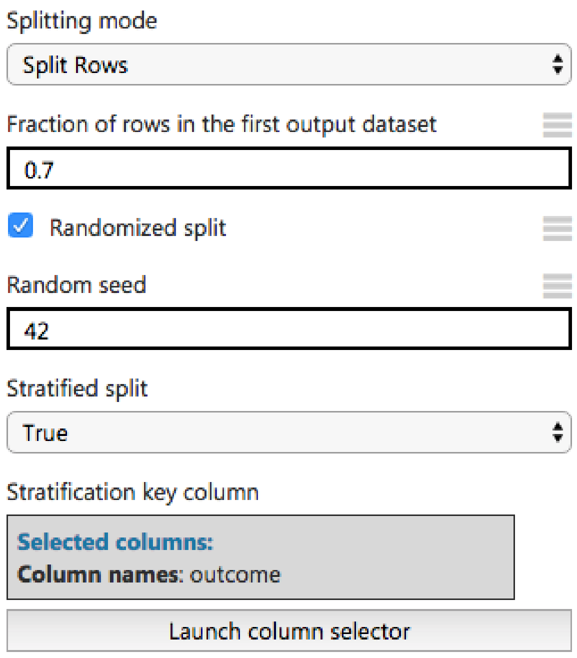
5 Setting aside data for testing

In machine learning experiments, we use some of the data to train the model (training set), and set some of the data aside to test the predictive capability of the model out-of-sample (test set). In the Azure Machine Learning Studio, you can accomplish this task using the **Split Data** module.

Type "split" into the search bar, and drag the **Split Data** module into the experiment workspace. Connect the left output of the **Normalize Data** module to the input of the **Split Data** module. Traditionally data is split 70/30 into a training and test set, respectively. Click on the **Split Data** module to bring up its properties, and specify 0.7 as the **Fraction of rows in the first output**.

Also, make sure **Randomized split** is checked, and the **Random seed** is set to "42". This allows us to use the random same split whenever we re-run the experiment, and will make sure your answers match the answers on MITx. (The default value is 0, meaning that the random split is generated based on the system clock. This specification can lead to slightly different results each time you run the experiment.)

Finally, change **Stratified spilt** to "True," and select "outcome" using the **Launch column selector**. This ensures that the same fraction of successes are included in both the training and test sets.



Splitting mode: Split Rows

Fraction of rows in the first output dataset: 0.7

✓ Randomized split

Random seed: 42

Stratified split: True

Stratification key column

Selected columns:
Column names: outcome

6 Training the model

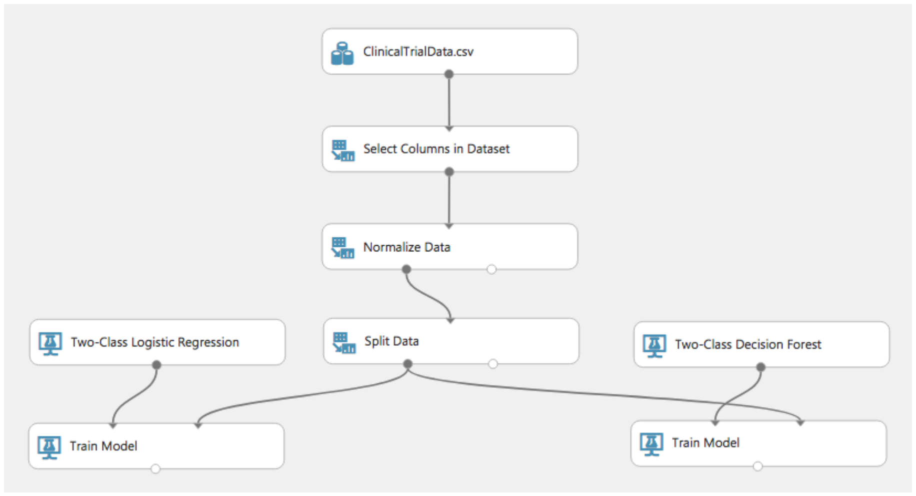
Now we can get Azure Machine Learning Studio to train our models so we can find the patterns in the clinical trial data to make predictions for new trials.

Type "train model" into the search bar. Drag the **Train Model** module to the workspace. Connect the first output (the one on the left) of the **Split Data** module to the right-most input of the **Train Model** task. This will take 70% of our data and use it to train our model to make predictions. Click on the **Train Model** module. In the properties window, select "outcome" using the **Launch Column Selector**.

Different types of machine learning tasks use different algorithms. Since we are trying to predict if an output has one of two values we want to use a two-class classification algorithm to train our model. Two-class classification algorithms are used to predict outcomes that can only have two possible values. In our case a value of 1 or 0 which indicates a successful trial.

Type "two-class" into the search bar. You will see a number of different classification algorithms listed. Each algorithm has its advantages and disadvantages. Check out the Azure Machine Learning Studio Documentation for a quick reference guide to algorithm selection. We are going to use logistic regression to train our model. Select **Two-Class Logistic Regression** module and drag it into the workspace. Connect the output of the **Two-Class Logistic Regression** module to the leftmost input of the **Train Model** module.

We would like to compare the performance of the logistic regression algorithm to the decision forest algorithm. Therefore, let's include the **Two-Class Decision Forest** module in our experiment. At the end of this step, your workspace should look as follows:

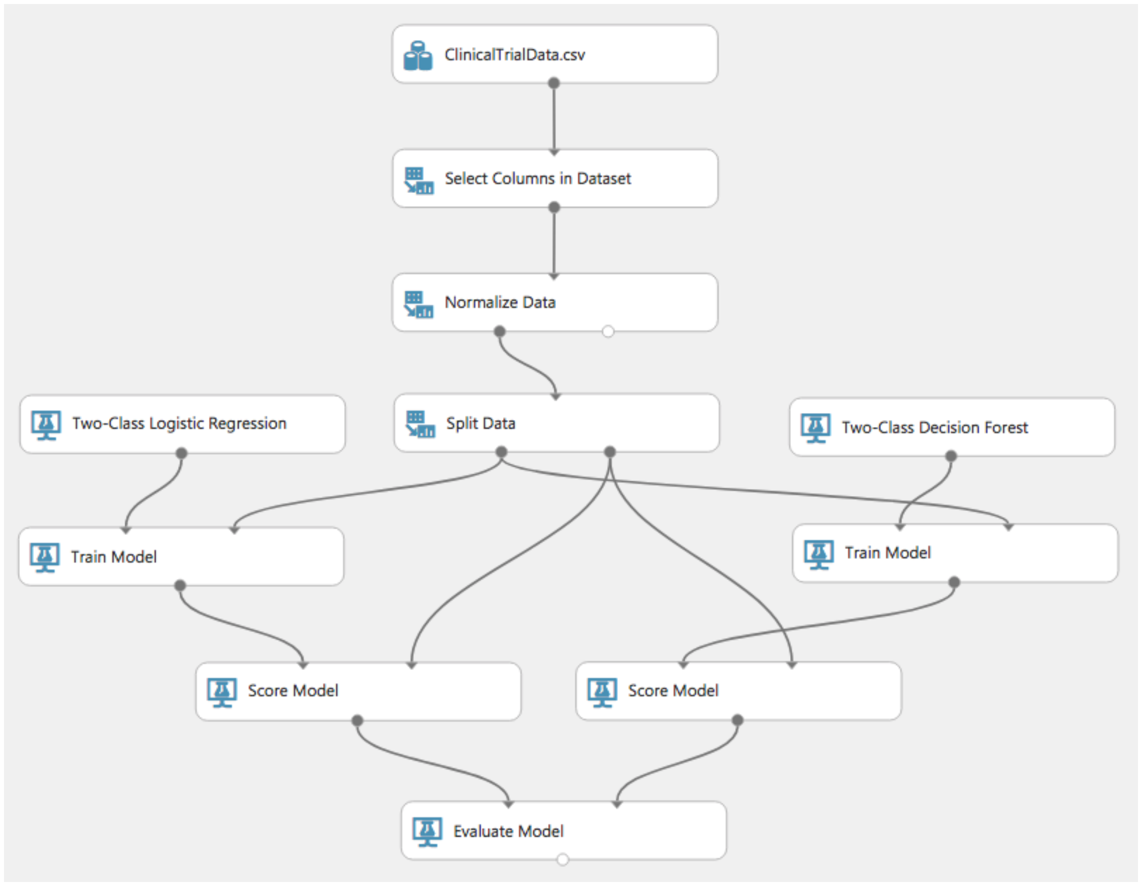

ClinicalTrialData.csv
Select Columns in Dataset
Normalize Data
Two-Class Logistic Regression
Split Data
Two-Class Decision Forest
Train Model
Train Model

7 Testing the model

After the model is trained, we need to see how well it predicts outcomes. Therefore we need to score each model by having it test against the 30% of data we split to our second output using the **Split Data** module.

Type "score" into the search bar and drag two **Score Model** modules to the workspace. Connect the output of **Train Model** to the left input of a **Score Model** module for each classification algorithm. Connect the right output of the **Split Data** module to the right input of both **Score Model** modules.

Finally, we need to report our test results. Type "evaluate" into the search bar and drag the **Evaluate Model** module to the bottom of the workspace. Connect the outputs of the **Score Model** modules to the inputs of the **Evaluate Model** module. The final experiment set-up should look as follows:



We are now ready to run the experiment! Press **RUN** on the bottom toolbar. You will see green checkmarks appear on each task as it completes.

8 How to interpret your results

To see your test results, right click on the **Evaluate Model** module and select **Evaluation results** > **Visualize**.

The closer the ROC curve is to a straight diagonal line the more your model is guessing randomly. You want ROC curve to get as close to the upper left corner as possible. If you scroll down you can see the detailed results. AUC (Area Under Curve) is a great overall indicator of your model performance. The closer AUC is to 1, the better your model is making predictions. You can learn more about ROC curves and AUC by watching this video.

You can also see the number of false and true positive and negative predictions for different values of the decision criteria threshold (which can be changed using the sliding scale).

- True positives are how often your model correctly predicted a trial would succeed
- False positives are how often your model predicted a trial would succeed, when they did not succeed (i.e. your model predicted incorrectly)
- True negatives indicate how often your model correctly predicted a trial would not succeed
- False negatives indicate how often your model predicted a trial would not succeed, when in fact it did succeed (i.e. your model predicted incorrectly)

You want high values for true positives and true negatives, and you want low values for false positives and false negatives.

What is the AUC for the logistic regression model? What is the AUC for the decision forest model?

9 Visualizing the model

A trained machine learning model can often be a black box. To gain insights, it's helpful to know how a variable influences a model.

THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Source: xkcd

For our logistic regression model, we can look at the magnitude and sign of the weights our model applies to each feature. Roughly speaking, features with large weights, in terms of absolute values, are more important for making predictions. Moreover, a positive (negative) weight on a feature indicates that the predicted probability of success increases (decreases) as the feature value increases.

Right-click the **Train Model** module for the logistic regression model, and select **Trained model** > **Visualize**.

Ignoring the "Bias" feature, which is simply the intercept in our regression, what are the 5 most important features that this model uses for making predictions?

10 Tuning the model

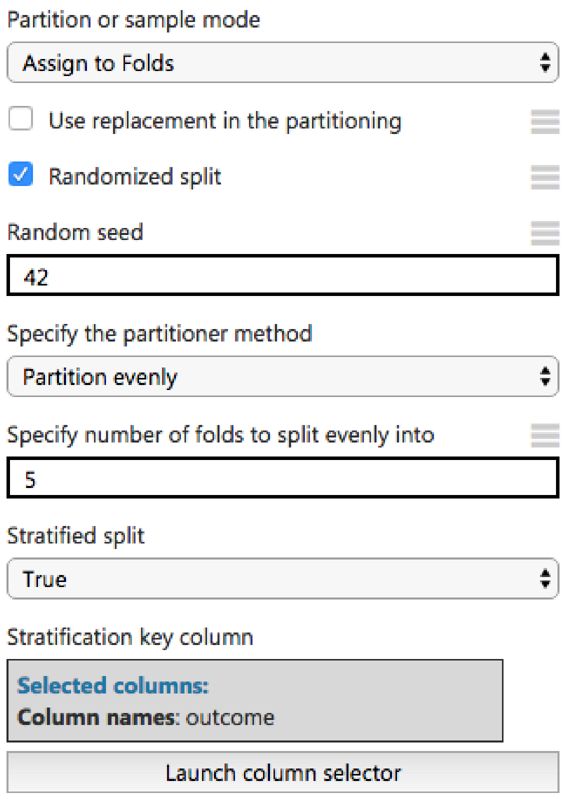
In the previous section, we used the default hyperparameters for each algorithm. In this section, we will tune the algorithm's hyperparameters using cross-validation to see if we can achieve better performance.

Cross-validation is a technique to train predictive models by partitioning the original training set further into sub-training sets to train different versions of the model with different parameters, and validation sets to select the one with the best parameters. The model with the best parameters is then evaluated on the test set to measure its out-of-sample performance.

In k-fold cross-validation, the training set is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for evaluating the model, and the remaining k-1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged to produce a single estimation. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once. For classification problems, we typically use stratified k-fold cross-validation, in which case the folds are selected so that each fold contains roughly the same proportions of class labels (in our case successes and failures).

Let's create a new experiment that implements stratified k-fold cross-validation. Select all modules in our **Drug approval predictor** workspace, right-click and choose "Copy", then open a new blank experiment and paste the copied modules into it. Let's call this new experiment **Drug approval predictor with cross-validation**.

Remove the **Train Model** modules from the experiment by right-clicking on them and selecting **Delete**. Next, add the **Partition and Sample** module to your experiment, and connect the training data. In the **Partition and Sample** module's properties window change the **Partition or sample mode** to "Assign to Folds", and set the **Random seed** to "42." Also set **Stratified split** to "True", and use the **Launch column selector** to select "outcome." This ensures that each of the 5 folds contains roughly the same proportion of successful outcomes.


Partition or sample mode: Assign to Folds

Use replacement in the partitioning: ☐

Randomized split: ✓

Random seed: 42

Specify the partitioner method: Partition evenly

Specify number of folds to split evenly into: 5

Stratified split: True

Stratification key column

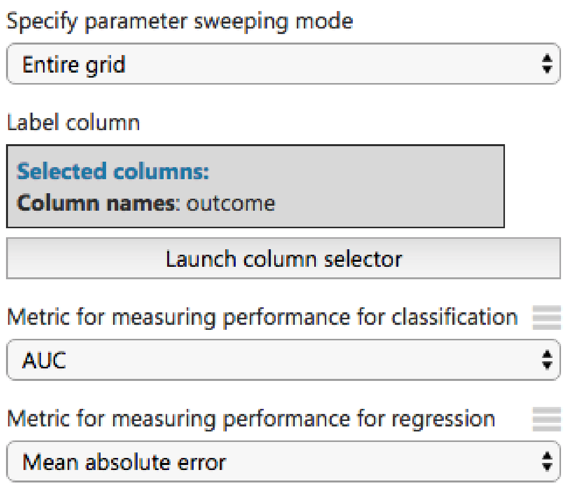
Selected columns:
Column names: outcome

Using the search bar, add two **Tune Model Hyperparameters** modules to your experiment (one for each classification algorithm). Connect the classification algorithms to the left input of the **Tune Model Hyperparameters** modules, and connect the output of the **Partition and Sample** module to the middle input of the **Tune Model Hyperparameters** modules.

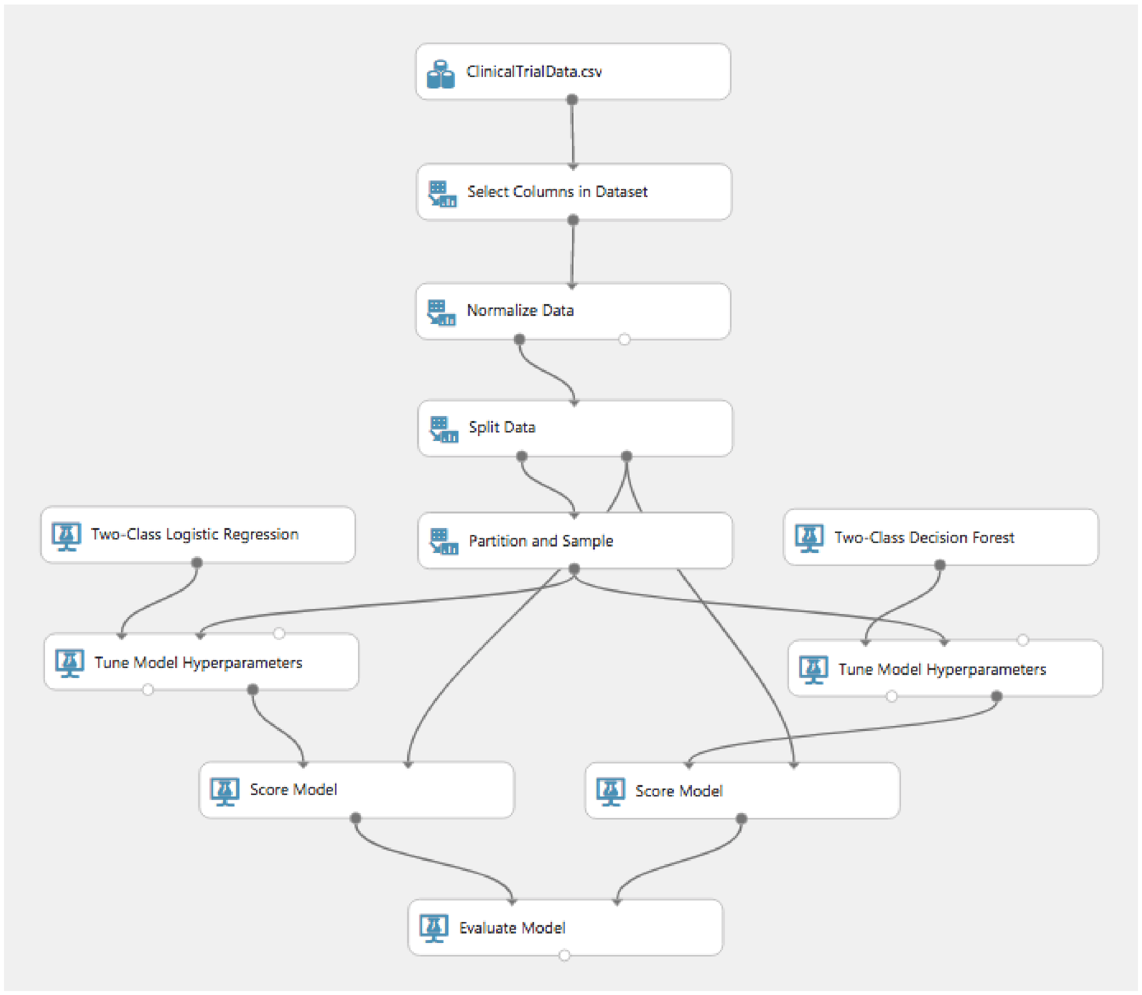
For both the **Two-Class Logistic Regression** module and the **Two-Class Decsion Forest** module, change **Create trainer mode** from "Single Parameter" to "Parameter Range."

In the **Tune Model Hyperparameters** modules, change **Specify parameter sweeping mode** from "Random sweep" to "Entire grid." This specifies that the model will be trained using each possible combination of parameter values. Finally, change the

Label column to "outcome," and change the **Metric for measuring performance for classification** to AUC.


Specify parameter sweeping mode
Entire grid
Label column
Selected columns:
Column names: outcome
Launch column selector
Metric for measuring performance for classification
AUC
Metric for measuring performance for regression
Mean absolute error

Connect the right output of the **Tune Model Hyperparameters** modules to the **Score Model** modules. If you have implemented everything correctly, your final experiment set-up should look as follows:


ClinicalTrialData.csv
Select Columns in Dataset
Normalize Data
Split Data
Two-Class Logistic Regression
Partition and Sample
Two-Class Decision Forest
Tune Model Hyperparameters
Tune Model Hyperparameters
Score Model
Score Model
Evaluate Model

We are now ready to run the experiment! Press **RUN** on the bottom toolbar. You will see green checkmarks appear on each task as it completes. When the entire experiment is completed (it may take a couple minutes), check how well your tuned models make predictions.

What is the AUC for the tuned logistic regression model? What is the AUC for the tuned decision forest model?