

Azure Machine Learning

Lecture 13

Deep Azure@McKesson

Zoran B. Djordjević

Need for Machine Learning System

1. The scale of data at large companies means that humans-only analysis of data is impractical. Humans are also expensive and difficult to deal with.
2. Data analysis has to be delegated to machines.
3. Model-driven approaches such as machine learning and statistics can often uncover patterns that cannot be seen by humans
4. Model-driven approaches can avoid human's emotional biases (as long as the correct processes are carefully applied)

For small data sets can use other packages

- If your datasets are small or moderate in size, you are better off with one of many Machine Learning toolkits:
- [Scikit-learn](#) is very comprehensive Python Machine Learning toolkit. Tool for hackers.
- [Matlab](#) can do anything that smacks of Math and Statistics. Large number of implemented ML algorithms, excellently integrated graphing utilities.
- [R](#) – your favorite. Does not scale as well outside of Azure ML but has a large number of implemented algorithms and can be used on your laptop as well as the Cloud.
- [WEKA](#) is Java
- [Rapid Miner](#) is Java
- [Apache Mahout](#) is Java and uses Spark in the background. Used to be very popular.
- [Accord.MachineLearning](#) is a .Net package
- [Vopal Wabbit](#) is C++
- [MultiBoost](#) and [Shogun](#) are C++
- [Microsoft Azure ML](#) is an excellent platform that scales well and works for small

When to use Azure ML

- If you are dealing with small samples classical API-s are faster and typically easier to deal with.
- Matlab or R are fully integrated testing and modelling suites. If you use Matlab or R on small data sets you will finish your job much more quickly and can do it on any machine, Windows, Mac, Linux. There is a free version of Matlab called Octave.
- If you are processing large volumes of data in Spark Batch or Spark Streaming mode and need to perform some fitting or classification (any ML algorithm) as an addition to other processing, it is convenient to add that particular part of analysis in Spark ML API. You deal with the same API, you do not need separate infrastructure and so on.
- For large volumes of data Azure ML scales infinitely, so it is a good choice.

Some Public Datasets

- In order to train your ML algorithms you need large datasets. Many are freely available on the Internet:
- University of California Irvine: <http://archive.ics.uci.edu/ml/>
This is a collection of almost 300 datasets of various types and sizes for tasks including classification, regression, clustering, and recommender systems.
- Amazon AWS public datasets: <http://aws.amazon.com/public-data-sets/>
Some of available datasets are:
 - [Landsat on AWS](#): An ongoing collection of moderate-resolution satellite imagery of all land on Earth produced by the Landsat 8 satellite
 - [NASA NEX](#): A collection of Earth science data sets maintained by NASA, including climate change projections and satellite images of the Earth's surface
 - [Common Crawl Corpus](#): A corpus of web crawl data composed of over 5 billion web pages
 - [1000 Genomes Project](#): A detailed map of human genetic variation
 - Google Books Ngrams: A data set containing Google Books n-gram corpuses
 - US Census Data: US demographic data from [1980](#), [1990](#), and [2000](#) US Censuses

Some Public Datasets

- *Kaggle*, a collection of datasets used in machine learning competitions. Areas include classification, regression, ranking, recommender systems, and image analysis. These datasets can be found under the *Competitions* section at: <http://www.kaggle.com/competitions>
- *Europeana Linked Open Data* contains open metadata on 20 million texts, images, videos and sounds gathered by Europeana - the trusted and comprehensive resource for European cultural heritage content: <http://labs.europeana.eu/api/linked-open-data/introduction/>
- *MIT Cancer Genomics gene expression datasets and publications* from MIT Whitehead Center for Genome Research:
- *KDnuggets*: <http://www.kdnuggets.com/datasets/index.html> contains a large number of links to very large and interesting datasets and code sources.
- Azure Machine Learning Studio comes with a moderately large set of illustrative datasets.

Install numpy, scipy, sklearn

- For Machine Learning you must have linear algebra and other Math libraries.
- NumPy is and an excellent library. Another is SciPy. You should install both.

```
$ sudo yum install numpy scipy
```

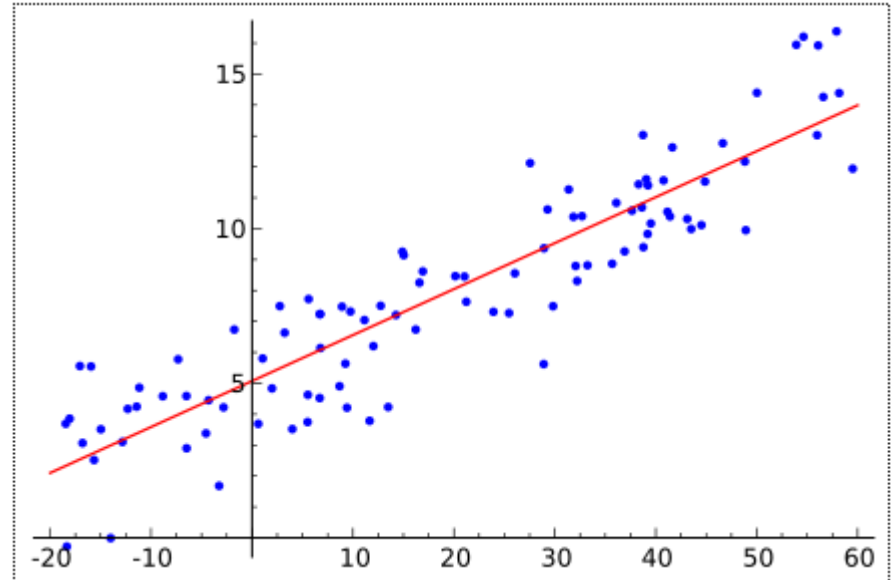
```
$ sudo yum install sklearn
```

```
$ sudo yum install seaborn
```

Regression

Regression Models

- Regression is concerned with predicting **target variables** that can take any real value. In statistics, **linear regression** is an approach for modeling the relationship between a scalar **target (dependent) variable(s) y** and one or more **explanatory variables (independent variables, predictors or features)** denoted X .
- The case of one explanatory variable is called **simple linear regression**. For more than one explanatory variable, the process is called **multiple linear regression**.



- How is regression related to learning and predicting.
- Regression presumes that there exists a relationship (a model) between features and the target variables. Learning process is the process of determining that relationship.
- Once we have the relationship, we could use it to estimate (predict) new target values, give new predictors.

Experimental Data with n independent variables (features)

Collect prices of house. There are multiple features (variables).

Size (feet ²)	Number of bedrooms	Number of floors	Age of home (years)	Price (\$1000)	
2104	5	1	45	460	$i = 1$
1416	3	2	40	232	$i = 2$
1534	3	2	30	315	$i = 3$
852	2	1	36	178	
...	
1	2	3	$n = 4$		$i = m$

Notation:

n = number of features

$x^{(i)}$ = input (features) of i^{th} training example.

$x_j^{(i)}$ = value of feature j in i^{th} training example.

There are m training examples

Hypothesis and the Cost Function

- First, we make a hypothesis that the best fit to our cloud of experimental points is a linear function of independent variables, i.e. features.
- If we had one feature, the hypothesis would be a simple line:

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

- When we have many feature the hypothesis is a multidimensional “plane”

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

- For convenience of notation, we define $x_0 = 1$ and write the hypothesis as:

$$h_{\theta}(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

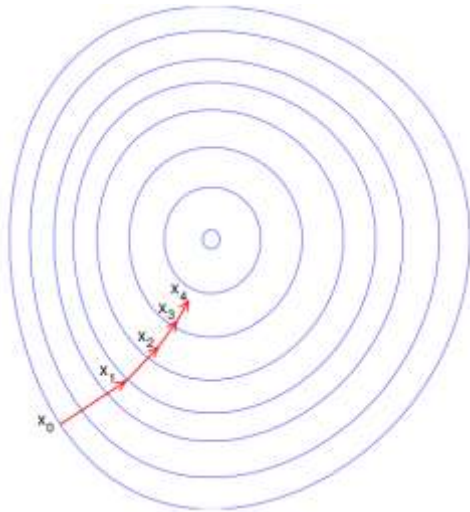
- We are after n unknown parameters $\theta_0, \theta_1, \dots, \theta_n$
- We determine those by minimizing a cost function, usually defined as:

$$J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2n} \sum_{i=1}^n (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

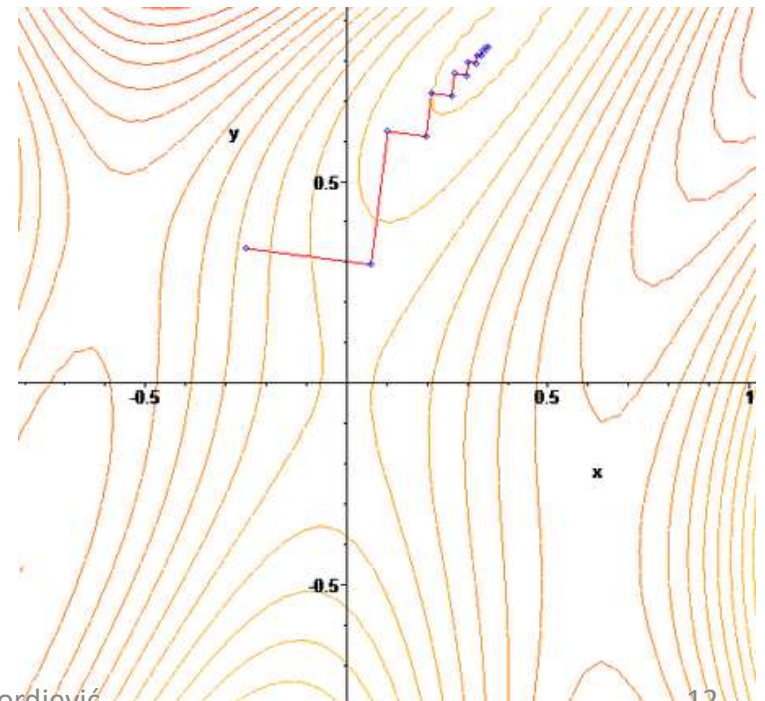
Iterative Solution: Gradient Descent

- For a small number of features and measurements the above "optimization" problem could be solved exactly. In most instances, when we have large number of features and a very large number of measurements, the optimal value of the set of parameters $\{\theta_i, i = 0, \dots, n\}$ has to be determined numerically.
- To that end we usually use an iterative method called Gradient Descent

Gradient descent: Repeat $\left\{ \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \dots, \theta_n) \right\}$
(simultaneously update for every $j = 0, \dots, n$)



- For simple cost functions this method converges rapidly and accurately. For complex cost functions it might not.



Gradient Descent

One independent variable ($n=1$):

Repeat {

$$\theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})}_{\frac{\partial}{\partial \theta_0} J(\theta)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x^{(i)}$$

(simultaneously update θ_0, θ_1)

}

Many independent variables:

Repeat { ($n \geq 1$)

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update θ_j for
 $j = 0, \dots, n$)

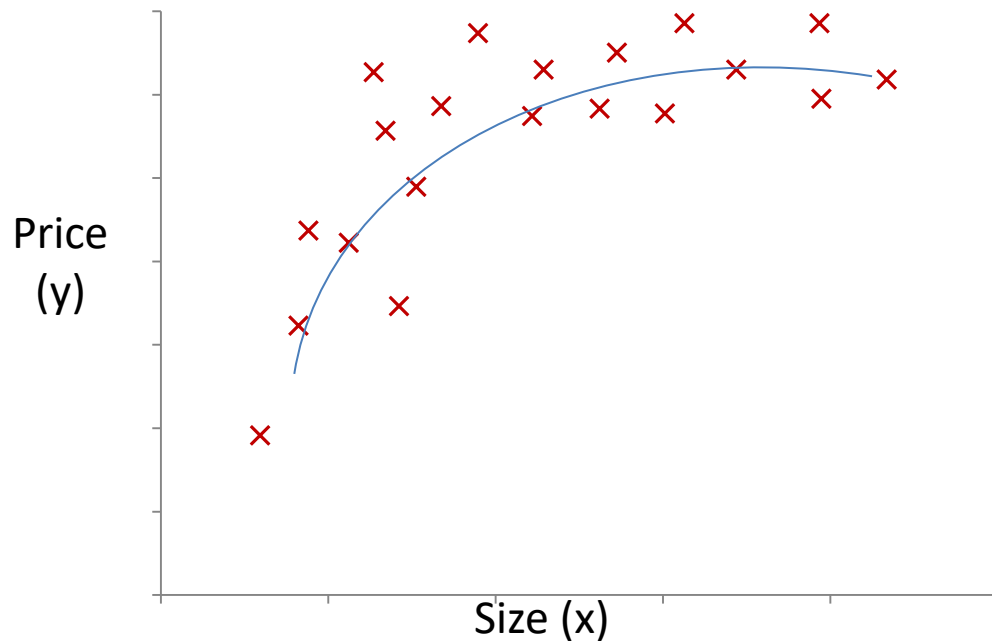
}

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

Polynomial Regression



- Quite often it is obvious from the experimental data that a "linear" model is a poor fit.
- A simple way to extend the regression analysis is to add "new (derived) features". Those could be higher powers of the independent variables.

$$\theta_0 + \theta_1 x + \theta_2 x^2$$

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

- The problem is still a linear optimization or search for the best parameters $\{\theta_i, i = 0, \dots, n\}$ which minimize the cost function $J(\theta_i)$.

$$\begin{aligned} h_{\theta}(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \\ &= \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2 + \theta_3(\text{size})^3 \end{aligned}$$

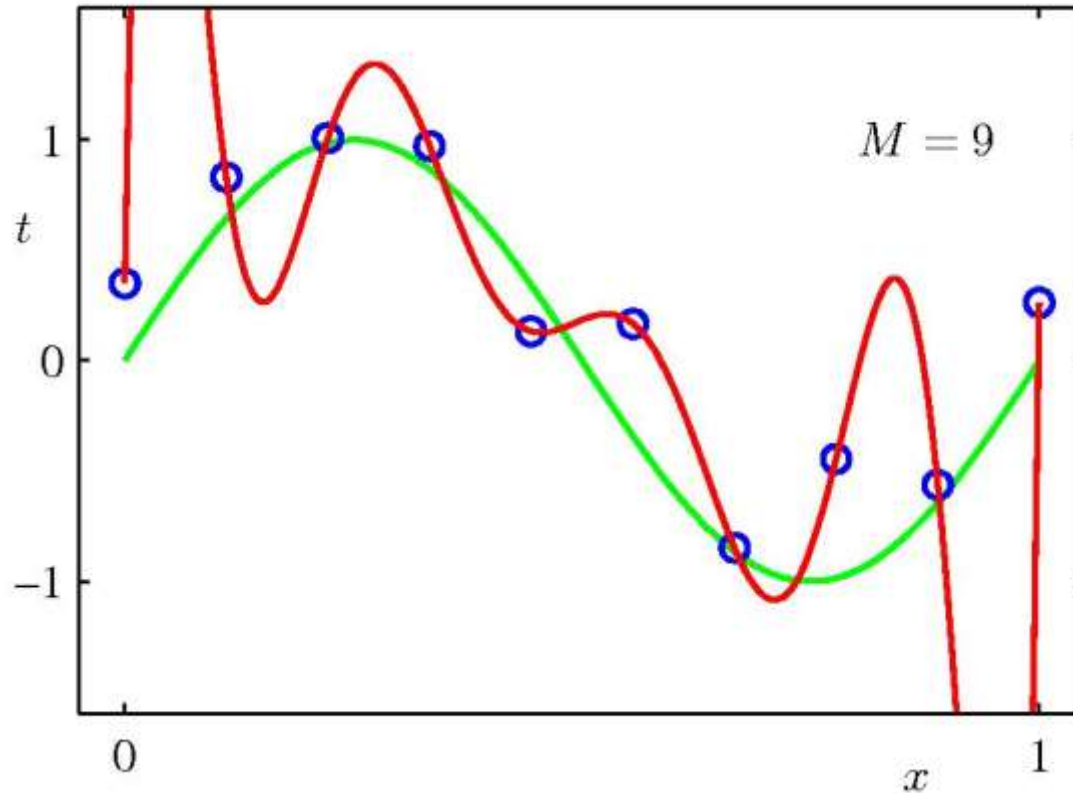
$$x_1 = (\text{size})$$

$$x_2 = (\text{size})^2$$

$$x_3 = (\text{size})^3$$

Avoid Overfitting

- For a small number of experimental points n one might be tempted to say: If I have n points, I do not have to look for an approximate solution, I could find a polynomial of order $n-1$ which passes exactly through those points.
- Curiously, this would result in a very poorly “fitting” function, which is unsuitable for predicting future results. Such polynomial would “over-fit” the solution.



Regularization

- One “automatic” way of avoiding overfitting is to add “regularization” term to our cost function.
- We add a nonlinear term to the cost function which prohibits large values of parameters and then choose vector θ to minimize the overall cost function:

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right]$$

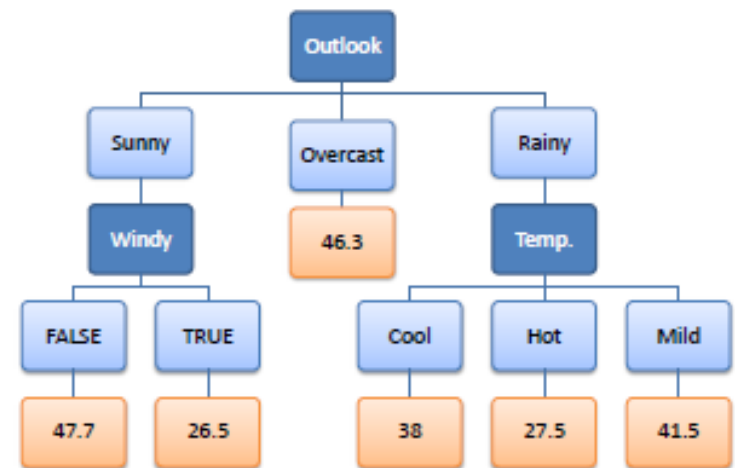
- $\lambda \sum_{j=1}^n \theta_j^2$ is the regularization term. We vary λ as well.
This quadratic regularization is referred to as L2 regularization
- Another popular regularization term uses absolute values: $\lambda \sum_{j=1}^n |\theta_j|$ of unknown parameters. This term is referred to as L1 regularization. You choose one or the other or some different regularization which works the best with your concrete problem.

Decision Trees

Decision Tree - Regression

- We are trying to predict the number of hours bikes will be rented based on weather condition.
- Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed.
- The final result is a tree with **decision nodes** and **leaf nodes**. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the feature tested. Leaf node (e.g., hours bikes are rented) represents the numerical target.


Predictors				Target
Outlook	Temp.	Humidity	Windy	Hours
Rainy	Hot	High	False	26
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	62
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	36
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	48
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	62
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30



Decision Tree Algorithm

- The core algorithm for building decision trees (J. R. Quinlan: <http://hunch.net/~coms-4771/quinlan.pdf>) employs a top-down, greedy search through the space of possible branches with no backtracking.
- The algorithm constructs a decision tree thru *Standard Deviation Reduction*.

Hours
25
30
46
45
52
23
43
35
38
46
48
52
44
30

$$S = \sqrt{\frac{\sum (x - \mu)^2}{n}}$$


Standard Deviation

S = 9.32

- Standard deviation is a measure of the spread of data values.
- Small value of S (standard deviation) means that data values are fairly uniform.
- Value of S=0 means that all values are identical.
- Large S means that data are spread over a large range and are not uniform.
- The algorithm seeks islands (groups of data points) of high uniformity.

Standard Deviation for Target Values Grouped by Features

- Consider data from the previous slide broken by the value of feature Outlook (Weather outlook)
- We see that by grouping target values (Hours) by the feature (Outlook) we created 3 groups, some of which have a lower standard deviation S , and therefore a higher uniformity than overall data.
- Standard deviation calculated as the mean of variations for all groups is apparently reduced when compared to the overall standard variation $S = 9.32$.
- Reduction is $9.32 - 7.66 = 1.66$

$$S(T, X) = \sum_{c \in X} P(c) S(c)$$

		Hours (StDev)	Count
Outlook	Overcast	3.49	4
	Rainy	7.78	5
	Sunny	10.87	5
			14



$$\begin{aligned} S(\text{Hours, Outlook}) &= P(\text{Sunny}) * S(\text{Sunny}) + P(\text{Overcast}) * S(\text{Overcast}) + P(\text{Rainy}) * S(\text{Rainy}) \\ &= (4/14) * 3.49 + (5/14) * 7.78 + (5/14) * 10.87 \\ &= 7.66 \end{aligned}$$

Standard Deviation Reduction, Process

- The standard deviation reduction is based on the decrease in standard deviation after a dataset is split on a feature.
- Constructing a decision tree is all about finding features that returns the highest reduction in standard deviation (i.e., the most homogeneous branches).
- Step 1:** The standard deviation of the whole target is calculated.
- Step 2:** The dataset is then split on different features. The standard deviation for each branch is calculated. The resulting standard deviation for each branch is subtracted from the standard deviation before the split. The result is the standard deviation reduction for the branch (feature).

		Hours (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
		SDR=1.66

		Hours (StDev)
Temp.	Cool	10.51
	Hot	8.95
	Mild	7.65
		SDR=0.17

		Hours (StDev)
Humidity	High	9.36
	Normal	8.37
		SDR=0.28

		Hours (StDev)
Windy	False	7.87
	True	10.59
		SDR=0.29

$$SDR(T, X) = S(T) - S(T, X)$$

$$\begin{aligned} SDR(\text{Hours}, \text{Outlook}) &= S(\text{Hours}) - S(\text{Hours}, \text{Outlook}) \\ &= 9.32 - 7.66 = 1.66 \end{aligned}$$

Standard Deviation Reduction, Process

Step 3: The attribute with the largest standard deviation reduction is chosen for the decision node: Outlook in our case.

Step 4a: Dataset is divided based on the values of the selected root feature (node).

★		Hours (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		



Outlook	Temp	Humidity	Windy	Hours
Sunny	Mild	High	FALSE	45
Sunny	Cool	Normal	FALSE	52
Sunny	Cool	Normal	TRUE	23
Sunny	Mild	Normal	FALSE	46
Sunny	Mild	High	TRUE	30
Rainy	Hot	High	FALSE	25
Rainy	Hot	High	TRUE	30
Rainy	Mild	High	FALSE	35
Rainy	Cool	Normal	FALSE	38
Rainy	Mild	Normal	TRUE	48
Overcast	Hot	High	FALSE	46
Overcast	Cool	Normal	TRUE	43
Overcast	Mild	High	TRUE	52
Overcast	Hot	Normal	FALSE	44

Standard Deviation Reduction, Process

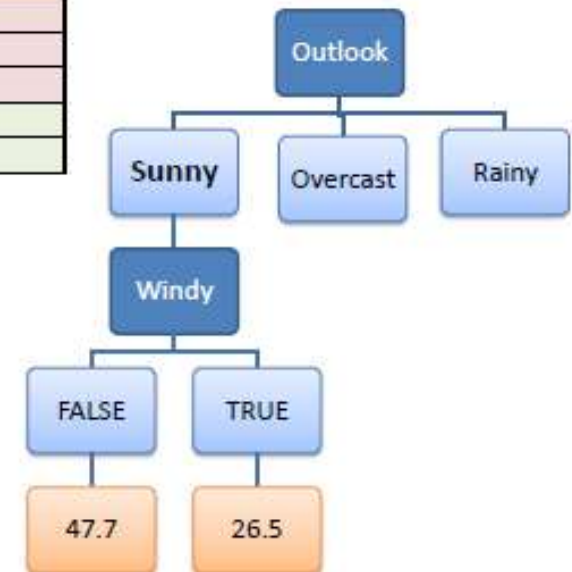
- **Step 4b:** A branch set with standard deviation more than 0 needs further splitting.
- In practice, we need some termination criteria. For example, when standard deviation for the branch becomes smaller than a certain fraction (e.g., 5%) of the standard deviation for the full dataset OR when too few instances remain in the branch (e.g., 3).
- **Step 5:** The process is run recursively on the non-leaf branches, until all data is processed.
- When the number of instances is more than one at a leaf node, we calculate the average as the final value for the target.

- In three subsets corresponding to 3 different Outlooks we find that grouping by feature Windy results in the greatest reduction of the standard deviation.
- Therefore the next decision node is feature Windy

Temp	Humidity	Windy	Hours
Mild	High	FALSE	45
Cool	Normal	FALSE	52
Mild	Normal	FALSE	46
Cool	Normal	TRUE	23
Mild	High	TRUE	30

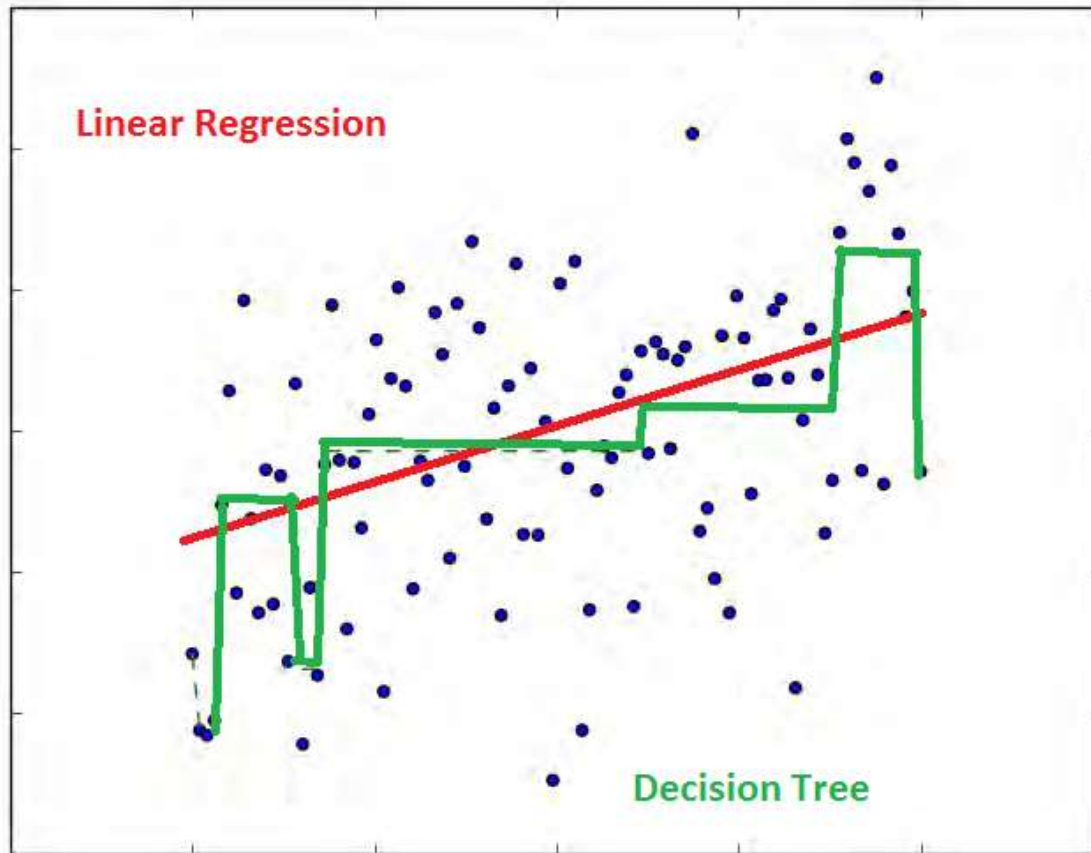
★		Hours (StDev)
Windy	False	3.09
	True	3.50
SDR= 7.62		

$$SDR = 10.87 - ((3/5) * 3.09 + (2/5) * 3.5)$$



Regression vs. Decision Tree

- Every method/algorithm has its advantages.
- Regression produces smooth boundaries/predictions.
- Decision Trees could give very non-linear predictions.
- The following image is just a sketch to illustrate the difference:



Azure ML Studio

Azure ML Studio

- Azure Machine Learning (ML) Studio is the primary tool that you will use to develop predictive analytic solutions in the Microsoft Azure cloud.
- The Azure Machine Learning environment is completely cloud-based and self-contained. It features a complete development, testing, and production environment for quickly creating predictive analytic solutions.
- Azure ML Studio gives you an interactive, visual workspace to easily build, test, and iterate on a predictive analysis model. You drag and drop datasets and analysis modules onto an interactive canvas, connecting them together to form an experiment, which you can then run inside Azure ML Studio.
- You can then iterate on your predictive analytics model by editing the experiment, saving a copy if desired, and then running it over and over again.
- When you're ready, you can publish your experiment as a web service, so that your predictive analytics model can be accessed by others over the Web.
- Highest organizational unit within Azure ML Studio is a project.

Workspace, ML Experiments

- Azure Machine Learning workspaces Represent a discrete “slice” of the Azure Machine Learning tool set that can be partitioned by the following criteria:
 - **Workspace name** is required to be unique and is the primary method for identifying a Machine Learning workspace.
 - **Workspace owner** is a valid Microsoft account that will be used to manage access to this Azure Machine Learning workspace
 - **Storage account** that is used to store all of the data and artifacts related to this Azure Machine Learning workspace.
 - **Data center location** Defines the physical Azure Data Center location for hosting the Azure Machine Learning workspace.
- **Azure Machine Learning experiments** are subproject within Azure Machine Learning workspaces and represent the primary method of enabling an iterative approach to rapidly developing Azure Machine Learning solutions.
- Within each Azure Machine Learning experiment, Azure ML Studio gives you an interactive, visual workspace to easily build, test, and iterate on a predictive analytic experiment.
- These experiments can then be submitted to Azure ML Studio for execution. Azure Machine Learning experiments are highly iterative. You can easily create, edit, test, save, and rerun experiments.

Web Services

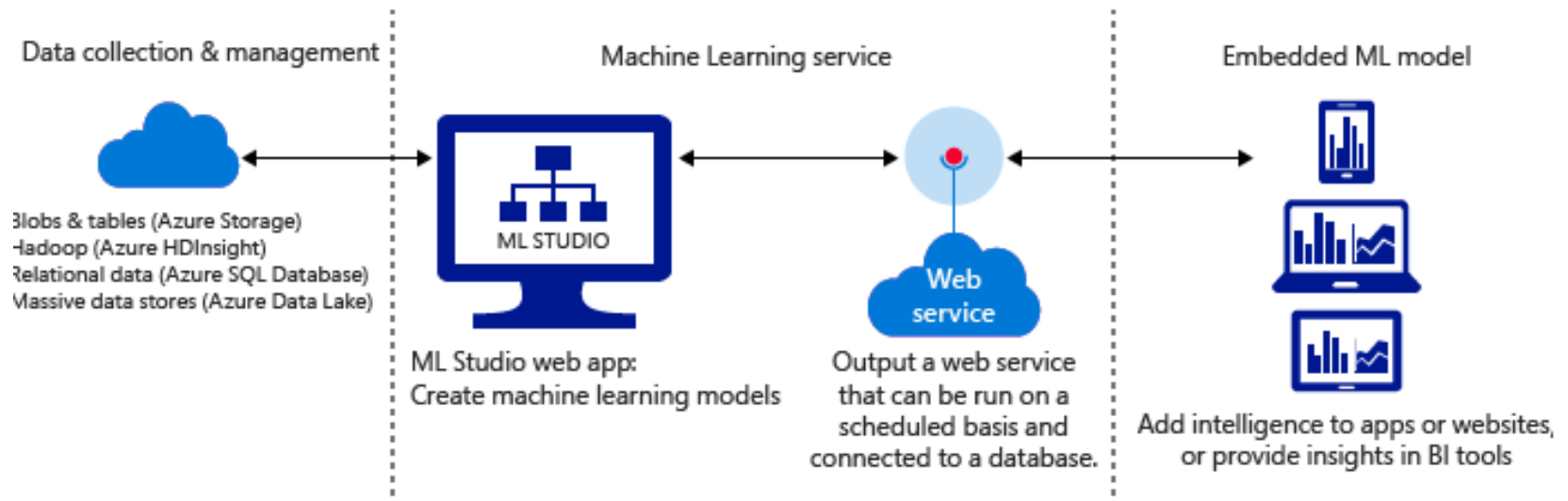
- **Azure Machine Learning web services** represent Azure Machine Learning models established in experiments that are exposed as public APIs over the Internet in the form of the Azure Machine Learning REST API.
- These services are generally exposed as a simple web service, or as an OData endpoint. The API provides two types of RESTful web interfaces:
 - Request Response Service (RRS) For individual, low-latency, synchronous uses, for making predictions.
 - Batch Execution Service (BES) For asynchronous scoring of a batch of data records. The input for BES is a batch of records from a variety of sources such as blobs, tables, SQL Azure, HD Insight (as a result of a Hive Query), and HTTP sources.
- **Datasets** are data that has been uploaded to Azure ML Studio so that it can be used in the prediction modeling process. A number of sample datasets are included with Azure ML Studio for you to experiment with, and you can upload more datasets as you need them.

Azure ML Modules

- **Modules** are various procedures and ML algorithms that you can apply to your data.
- ML Studio has a number of modules ranging from data ingress functions to training, scoring, and validation processes. Some examples are:
- **Convert to ARFF** Converts a .NET serialized dataset to ARFF format. ARFF is a common machine learning construct and stands for Attribute-Relation File Format. It is commonly defined as an ASCII text file that describes a list of instances sharing a set of attributes.
 - **Elementary Statistics** Calculates elementary statistics such as mean, standard deviation, and so on.
 - **Linear Regression** Creates an online gradient, descent-based, linear regression model.
 - **Score Model** Scores a trained classification or regression model.
- A module might have a set of parameters that you can use to configure the module's internal algorithms.
- When you select a module on the canvas, its parameters are displayed in the pane to the right of the canvas. You can modify the parameters in that pane to tune your model.

Azure ML Basic workflow

- Azure ML presumes a workflow with several steps depicted in the diagram below. You build a model from data and publish it as an ML solution



Free Trial Access to Azure ML

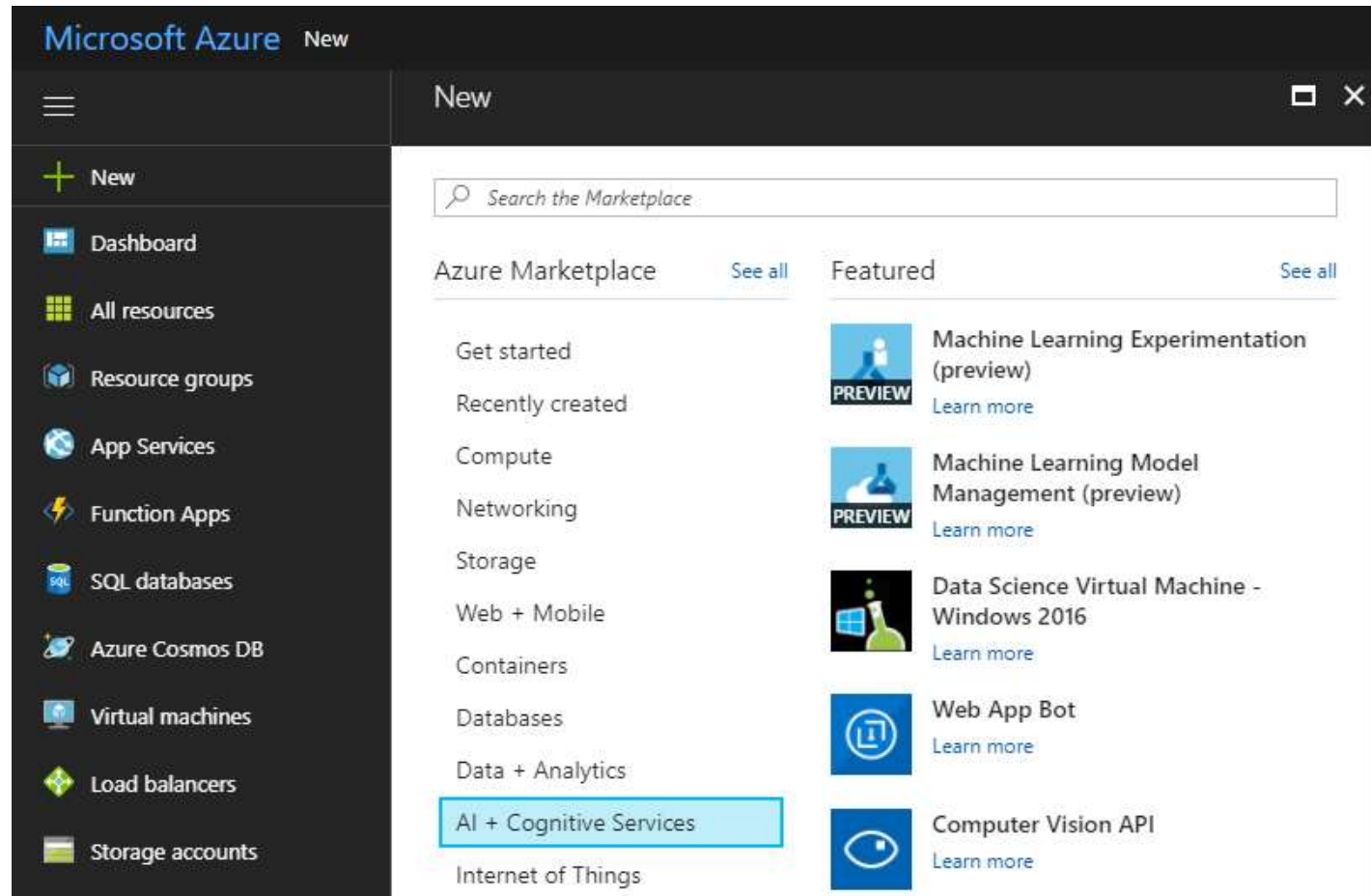
- Besides free Azure access there is a special Azure Machine Learning trial which can be reached at: <https://studio.azureml.net/Home>
- This is a free Azure offer that is feature-specific and therefore only allows you access to the Azure Machine Learning environment.
- This is an extremely low-friction option for new adopters: All you need is a valid Microsoft account to get started. We do not know whether it works with existing McKesson subscriptions.
- Note that if you opt to take advantage of the free Azure Machine Learning trial offer, you will only have access to the Azure Machine Learning features, not access to the full Azure environment.
- If you are not sure whether you are using the free trial, pay attention to costs.
-

Start Work with ML Studio

- To use Azure Machine Learning (ML) Studio go to: <https://studio.azureml.net/>
- *Azure Machine Learning Studio is a powerfully simple browser-based, visual drag-and-drop authoring environment where little coding is necessary.*
- Please note that Azure offers a newer tool called Azure ML Workbench that you need to install on your machine. The tool requires integration with Visual Studio or PyCharm and has many other dependencies that might be difficult to arrange for (for now and on Windows 7 machines).
- The developer's experience with the Workbench is practically identical to your experience with the ML Studio. We decided not to introduce the Workbench in this lecture.

Create ML Workspace in Portal

- Login into azure.com with an existing Azure account.
- Select + New > AI + Cognitive Services > Machine Learning Experimentation:



Create Experimentation Account, Select Free Pricing Tier

- Provide names for the experimentation account, work group, workspace and select the Dev/Test pricing tier.
- Once resources are create, log out

ML Experimentation
Machine Learning Experimentation

Experimentation account name

zdzordjeexperimentation

Subscription

Pay-As-You-Go

Resource group

Create new

Use existing

zdzordjegrp

Location

East US 2

Number of seats. First two are free.

2

Storage account

Create new

Use existing

zdzordjeexperimenstorage

Workspace for Experimentation account

zdzordjemlworkspace

Assign owner for the workspace

c344c50d-2b70-4de6-9772-34...

Create Model Management account

Account name

zdzordjeexperimentationModelMgmt

Model Management pricing tier

No pricing tier selected

☐ Pin to dashboard

Create Automation options

Select your pricing tier
Browse the available plans

Click the link to learn more about the Model Management account

DEVTEST	S1
20 Models	100 Models
2 Deployments	10 Deployments
4 Cores	16 Cores
0.00 USD/DAY (ESTIMATED)	1.60 USD/DAY (ESTIMATED)

S3

10,000 Models

1,000 Deployments

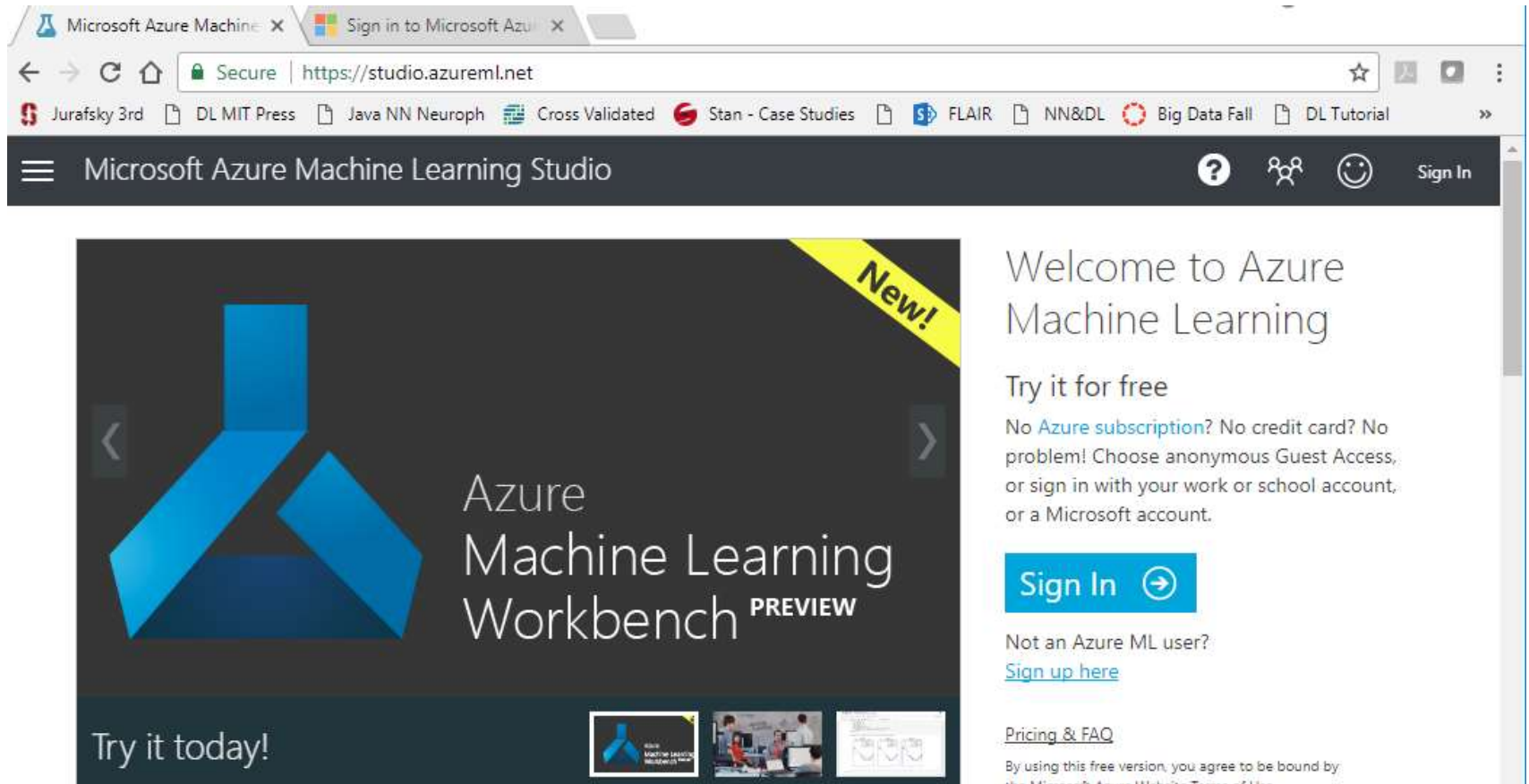
800 Cores

80.63
USD/DAY (ESTIMATED)

Select

Sign In

- Go back to Azume ML Studio Page <https://studio.azureml.net> and Sign In



The screenshot shows a web browser window with the URL <https://studio.azureml.net>. The browser's address bar shows the page is secure. The page header includes the Microsoft Azure Machine Learning Studio logo and a 'Sign In' button. The main content area features a large blue 3D logo for 'Azure Machine Learning Workbench PREVIEW' with a 'New!' banner. Below the logo, it says 'Try it today!'. To the right, the text 'Welcome to Azure Machine Learning' is displayed, followed by 'Try it for free'. A message states: 'No Azure subscription? No credit card? No problem! Choose anonymous Guest Access, or sign in with your work or school account, or a Microsoft account.' A prominent blue 'Sign In' button with a right arrow is visible. Below this, there is a link for 'Sign up here' for non-users, and a link for 'Pricing & FAQ'. At the bottom, a small disclaimer states: 'By using this free version, you agree to be bound by the Microsoft Azure Website Terms of Use.'

ML Studio Dashboard

- ML Studio creates a free workspace named after your user (email)

The screenshot displays the Microsoft Azure Machine Learning Studio interface. The top navigation bar includes the 'Microsoft Azure Machine Learning Studio' title and a user profile dropdown for 'zoran.djordjevic0106-Free...'. The left sidebar contains navigation links: PROJECTS, EXPERIMENTS, WEB SERVICES, NOTEBOOKS, DATASETS, TRAINED MODELS, and SETTINGS. The main content area shows the 'projects' section with a table of existing projects:

NAME	AUTHOR
igor	zoran.djordjevic0106
bykes	zoran.djordjevic0106

A modal dialog is open on the right, titled 'REGION:' and 'WORKSPACE:'. The 'REGION:' dropdown is set to 'South Central US'. The 'WORKSPACE:' section shows a text input field containing 'zoran.djordjevic0106-Free...' and a 'Free' button.

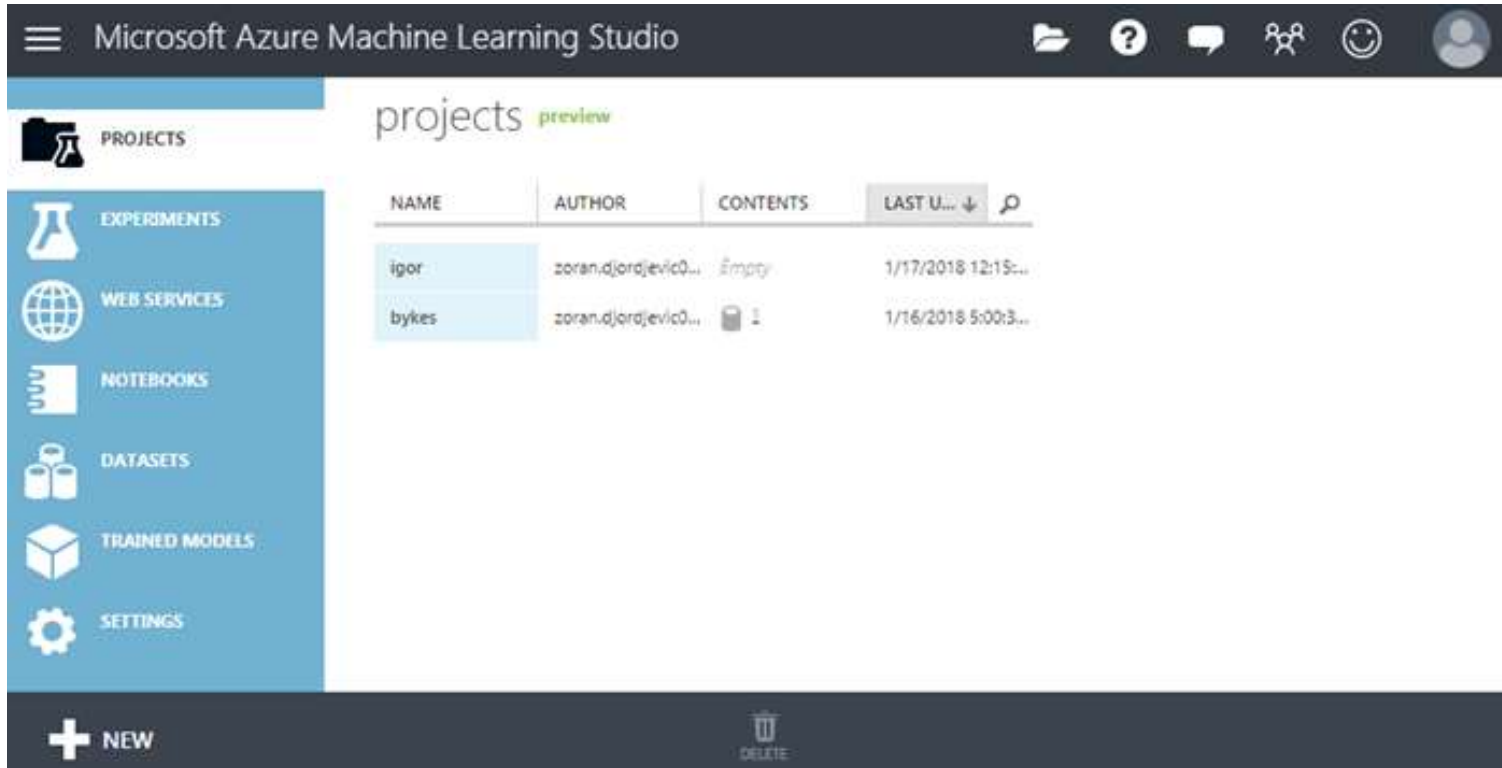
Left Navigation Bar

The left navigation bar has several links:

- **EXPERIMENTS** Experiments that have been created, run, and saved as drafts.
- **WEB SERVICES** A list of experiments that you have published.
- **DATASETS** Uploaded datasets that can be used in your experiments. Azure provides a large number of useful datasets. You can upload your own data as well.
- **TRAINED MODELS** New predictive models that have been “trained” using the built-in Azure ML Studio machine learning algorithms.
- **SETTINGS** A collection of settings that you can use to configure your account and resources.

Create New Project

- Go to Projects and click on NEW on the bottom of the screen.
- ML Studio will let you create experiments without first creating projects.



Select Empty Project, Provide Name

The screenshot shows the Microsoft Azure Machine Learning Studio interface. At the top, the title bar reads 'Microsoft Azure Machine Learning Studio'. Below it, a navigation bar shows 'PROJECTS' and 'projects preview'. A sidebar on the left lists options: 'NEW', 'DATASET', 'MODULE', 'PROJECT PREVIEW' (highlighted), 'EXPERIMENT', and 'NOTEBOOK PREVIEW'. The main area displays a large button with a plus sign and the text 'Empty Project'. A 'New Project' dialog is open on the right, with a close button (X) in the top right corner. The dialog has two input fields: 'NAME' with the text 'zdjordjeproject' and 'DESCRIPTION' with the text 'Sample project'. A checkmark button is located at the bottom right of the dialog.

- Select: Empty Project, provide name, description, select check mark.

Import Dataset

- In what follows we will use UC Irvine Machine Learning Repository data download location at <https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.data>
- We added the heading row manually based on the attached description file. Data are from 1987. The spreadsheet with the automobile data should look like the following:

	A	B	C	D	E	F	G	H	I	J	K	L
1	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width
2	3 ?		alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1
3	3 ?		alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1
4	1 ?		alfa-romero	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5
5	2	164	audi	gas	std	four	sedan	fwd	front	99.8	176.6	66.2
6	2	164	audi	gas	std	four	sedan	4wd	front	99.4	176.6	66.4
7	2 ?		audi	gas	std	two	sedan	fwd	front	99.8	177.3	66.3
8	1	158	audi	gas	std	four	sedan	fwd	front	105.8	192.7	71.4
9	1 ?		audi	gas	std	four	wagon	fwd	front	105.8	192.7	71.4
10	1	158	audi	gas	turbo	four	sedan	fwd	front	105.8	192.7	71.4
11	0 ?		audi	gas	turbo	two	hatchback	4wd	front	99.5	178.2	67.9
12	2	192	bmw	gas	std	two	sedan	rwd	front	101.2	176.8	64.8
13	0	192	bmw	gas	std	four	sedan	rwd	front	101.2	176.8	64.8
14	0	188	bmw	gas	std	two	sedan	rwd	front	101.2	176.8	64.8
15	0	188	bmw	gas	std	four	sedan	rwd	front	101.2	176.8	64.8
16	1 ?		bmw	gas	std	four	sedan	rwd	front	103.5	189	66.9
17	0 ?		bmw	gas	std	four	sedan	rwd	front	103.5	189	66.9
18	0 ?		bmw	gas	std	two	sedan	rwd	front	103.5	193.8	67.9
19	0 ?		bmw	gas	std	four	sedan	rwd	front	110	197	70.9
20	2	121	chevrolet	gas	std	two	hatchback	fwd	front	88.4	141.1	60.3
21	1	98	chevrolet	gas	std	two	hatchback	fwd	front	94.5	155.9	63.6
22	0	81	chevrolet	gas	std	four	sedan	fwd	front	94.5	158.8	63.6
23	1	118	dodge	gas	std	two	hatchback	fwd	front	93.7	157.3	63.8
24	1	118	dodge	gas	std	two	hatchback	fwd	front	93.7	157.3	63.8
25	1	118	dodge	gas	turbo	two	hatchback	fwd	front	93.7	157.3	63.8
26	1	148	dodge	gas	std	four	hatchback	fwd	front	93.7	157.3	63.8

Import Data

- Start by clicking Datasets in the left navigation bar of the Azure ML Studio screen,.
- Next click + New on the bottom left of the screen. Then select Dataset in the navigation bar on the left to upload a new dataset from a local file.
- Next, click From Local File. Browse to the location where you saved the Excel spreadsheet we modified for column headings and saved as Imports-85.data-hdrs.csv.
- Select the Generic CSV File With A Header (.csv) type
- Click the check mark when ready and your dataset will be uploaded to the Azure ML Studio Datasets repository

The screenshot displays the Azure ML Studio interface. On the left, a navigation pane shows 'DATASETS' selected. Below it, a 'NEW' button is visible, along with 'DATASET' and 'EXPERIMENT' options. The main area shows the 'Upload a new dataset' dialog. The dialog has a title 'Upload a new dataset' and a section 'SELECT THE DATA TO UPLOAD:' with a 'Choose File' button and the filename 'imports-85.csv'. Below this is a checkbox 'This is the new version of an existing dataset'. The next section is 'ENTER A NAME FOR THE NEW DATASET:' with a text input field containing 'imports-85.csv'. This is followed by 'SELECT A TYPE FOR THE NEW DATASET:' with a dropdown menu set to 'Generic CSV File with a header (.csv)'. The final section is 'PROVIDE AN OPTIONAL DESCRIPTION:' with a large text area. A checkmark icon is visible on the right side of the dialog.

Microsoft Azure Machine Learning | Home Studio Gallery PREVIEW

NEW

DATASET FROM LOCAL FILE Upload a new dataset

EXPERIMENT

Upload a new dataset

SELECT THE DATA TO UPLOAD:

imports-85.csv

☐ This is the new version of an existing dataset

ENTER A NAME FOR THE NEW DATASET:

imports-85.csv

SELECT A TYPE FOR THE NEW DATASET:

Generic CSV File with a header (.csv) ▼

PROVIDE AN OPTIONAL DESCRIPTION:

☒

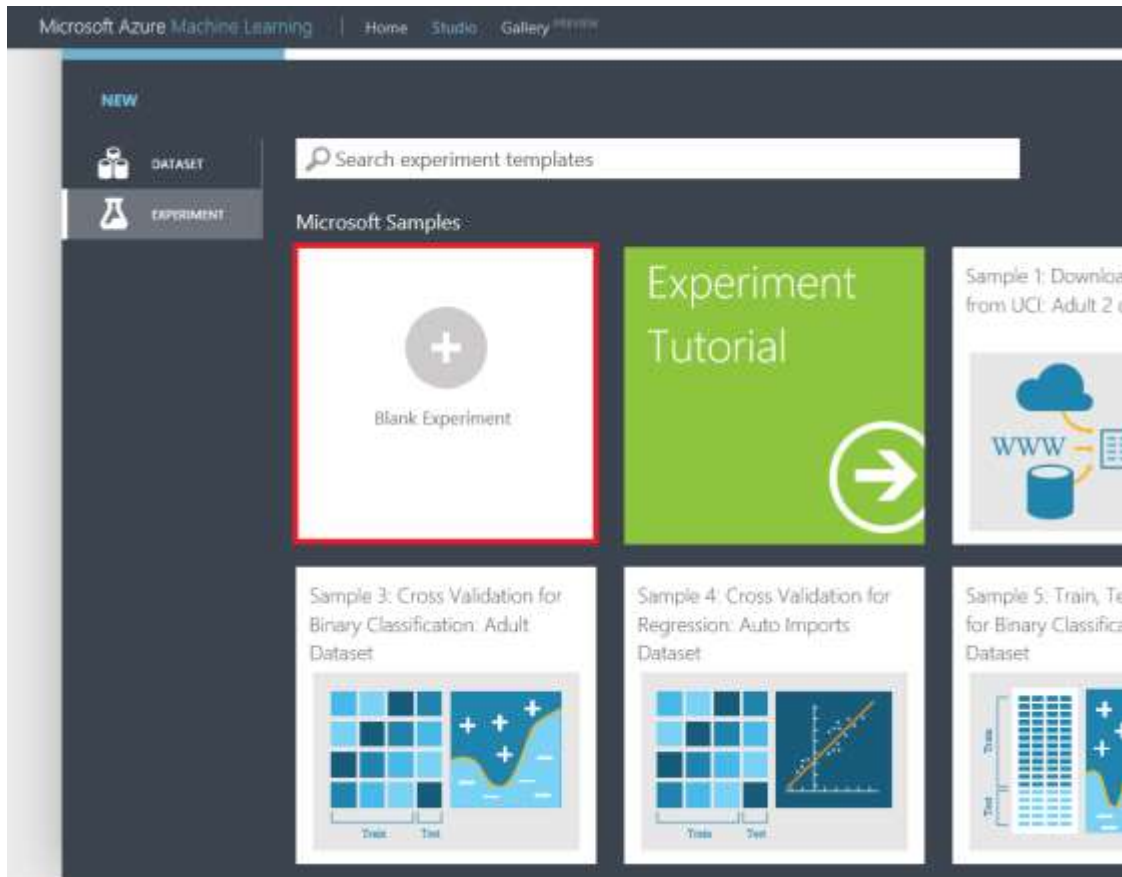
Automobile Price Prediction Experiment

- Now that we have uploaded our sample automobile dataset, it's time to create our linear regression sample experiment to predict the price of an automobile based on the selected features.
- To get started with creating a new experiment, select the Experiments tab on the left navigation bar. Then click the + New icon on the bottom left app bar



New Experiment

- You will then be presented with the option to start with a blank experiment or chose from a gallery of tutorials and other sample experiments. For our automobile price prediction experiment, start with a blank experiment .



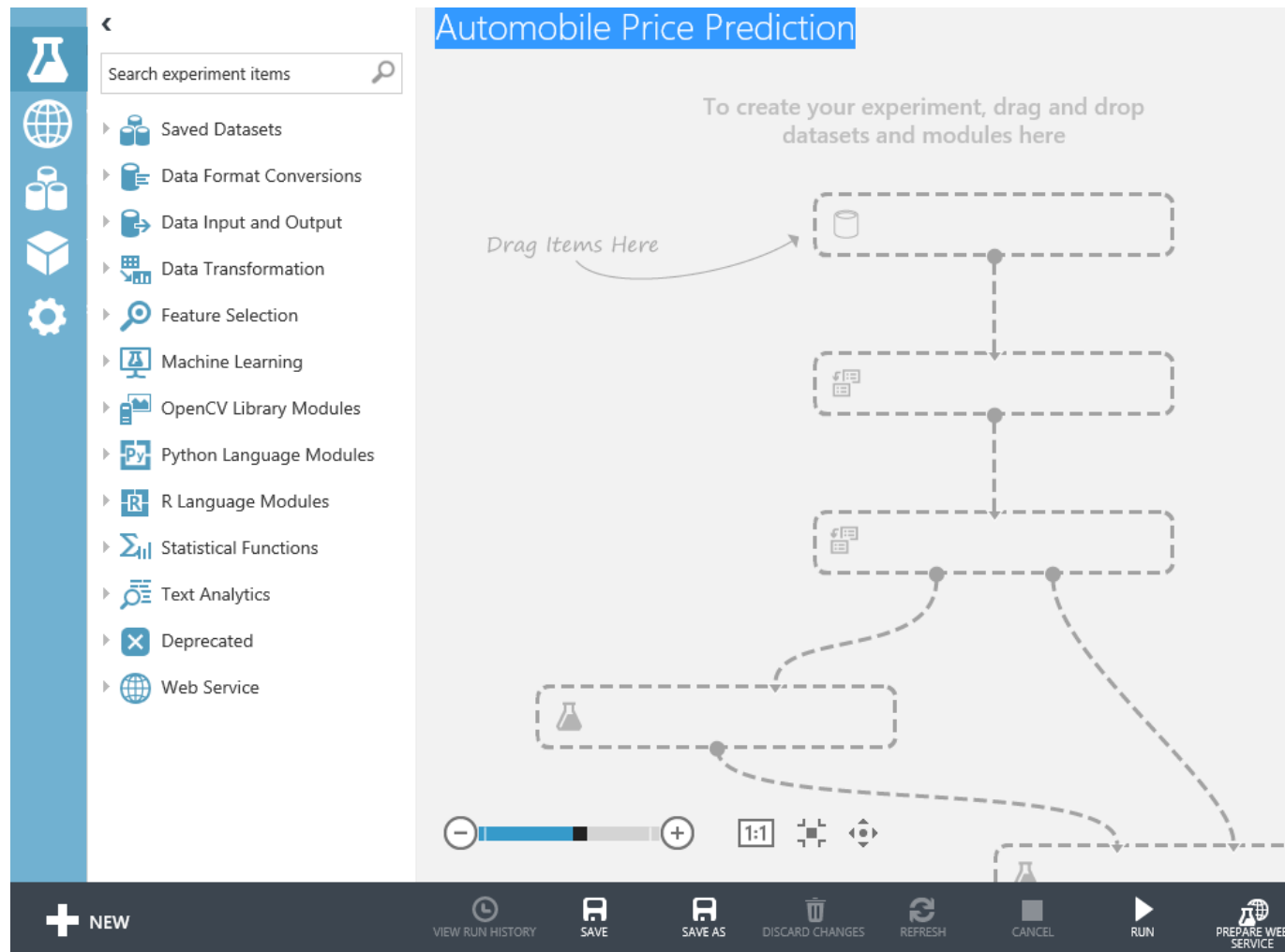
- The blank experiment opens with a drag-and-drop template for filling in the steps of your experiment processing logic. Once created, you can name your experiment Automobile Price Prediction by clicking the experiment's name and entering the new name

Regression Model with Azure ML

- Regression models are concerned with target variables that can take any real value. The underlying principle is very similar—we wish to find a model that maps input features to target variables. That mapping is the prediction. Like classification, regression is also a form of supervised learning.
- Regression models can be used to predict just about any variable of interest.
 - Gambling returns on stocks and other economic variables
 - Loss amounts for loan defaults (this can be combined with a classification model that predicts the probability of default, while the regression model predicts the amount in the case of a default)
 - Recommendations
 - Predicting customer lifetime value in a retail, mobile, or other business, based on user behavior and spending patterns
- There are many classes of regression models:
 - Linear regression models,
 - Decision tree regression models,
 - Bayesian Linear Regression,
 - Decision Forest Regressions,
 - Neural Network Regressions, etc.

Populate the Template

- The first step is to drag our new dataset onto the Azure ML Studio canvas. Click the Saved Datasets tab on the left navigation bar. Look for our freshly uploaded dataset named Imports-85.data-hdrs.csv. Drag it onto the top of the designer surface.



Visualize Data

- We can use the Visualize utility to examine our dataset by hovering over the bottom connector. Right-click and then select the Visualize command

Price Prediction > imports-85.data-hdrs.csv > dataset

columns

26

0	188	bmw	gas	std	four	sedan	rwd	fron
1		bmw	gas	std	four	sedan	rwd	fron
0		bmw	gas	std	four	sedan	rwd	fron
0		bmw	gas	std	two	sedan	rwd	fron
0		bmw	gas	std	four	sedan	rwd	fron
2	121	chevrolet	gas	std	two	hatchback	fwd	fron
1	98	chevrolet	gas	std	two	hatchback	fwd	fron
0	81	chevrolet	gas	std	four	sedan	fwd	fron
1	118	dodge	gas	std	two	hatchback	fwd	fron
1	118	dodge	gas	std	two	hatchback	fwd	fron
1	118	dodge	gas	turbo	two	hatchback	fwd	fron
1	148	dodge	gas	std	four	hatchback	fwd	fron
1	148	dodge	gas	std	four	sedan	fwd	fron
1	148	dodge	gas	std	four	sedan	fwd	fron
1	148	dodge	gas	turbo		sedan	fwd	fron
-1	110	dodge	gas	std	four	wagon	fwd	fron

Dealing with Missing Values

- A cursory inspection of the automobile pricing dataset using the Visualize tool reveals that we have a few missing values in our dataset. The answer to this problem is to use an Azure ML Studio module named Clean Missing Data to detect missing values. This module will supply default values instead.
- To find and implement the Clean Missing Data module in Azure ML Studio, simply type the name into the search bar on the top of the left navigation bar. Let the Azure ML Studio search function find it for you.

The screenshot displays the Azure ML Studio interface. On the left, a workflow diagram shows a data source 'imports-85.data-hdrs.csv' connected to a 'Clean Missing Data' module. On the right, the configuration panel for the 'Clean Missing Data' module is visible. The panel includes the following settings:

- Columns to be cleaned:** A button labeled 'Selected columns: All columns' and a button labeled 'Launch column selector'.
- Minimum missing value range:** A text input field containing '0'.
- Maximum missing value range:** A text input field containing '1'.
- Cleaning mode:** A dropdown menu with 'Custom substitution value' selected. This section is highlighted with a red rectangle.
- Replacement value:** A text input field containing '0'.
- Generate missing values:** An unchecked checkbox.

Missing Values

- This is a very handy module in the Azure ML Studio toolbox and allows us to easily handle cases where we have missing or invalid data in our initial dataset. We start the configuration by launching the column selector and including all columns. We can then set the minimum and maximum missing value ranges. Then set the Cleaning Mode. When you click the Cleaning Mode drop-down list, you will see that there are actually eight different cleaning mode options we could use to scrub our data.
- For our purposes in the automobile price prediction experiment, select the Custom Substitution option and designate the replacement value to be 0. This means any column in the initial dataset that has missing data will be automatically updated to have a 0 substituted for the missing column at run time.

Clean Missing Data

Columns to be cleaned

Selected columns:
All columns

Launch column selector

Minimum missing value ra... 

0

Maximum missing value ra... 

1

Replace using MICE
Custom substitution value
Replace with mean
Replace with median
Replace with mode
Remove entire row
Remove entire column
Replace using Probabilistic PCA

Select Columns

- The next step in building our automobile price prediction experiment will be to insert a Select Columns module to allow us to selectively include or exclude columns of data from our initial dataset. The Project Columns module is probably one of the most useful and widely used modules in the entire Azure ML Studio module collection.

Select columns

☐ Allow duplicates and preserve column order in selection

Begin With

ALL COLUMNS NO COLUMNS

Include column names

make X fuel-type X aspiration X num-of-doors X

body-style X drive-wheels X engine-location X

wheel-base X length X width X curb-weight X

num-of-cylinders X engine-size X fuel-system X

peak-rpm X highway-mpg X city-mpg X

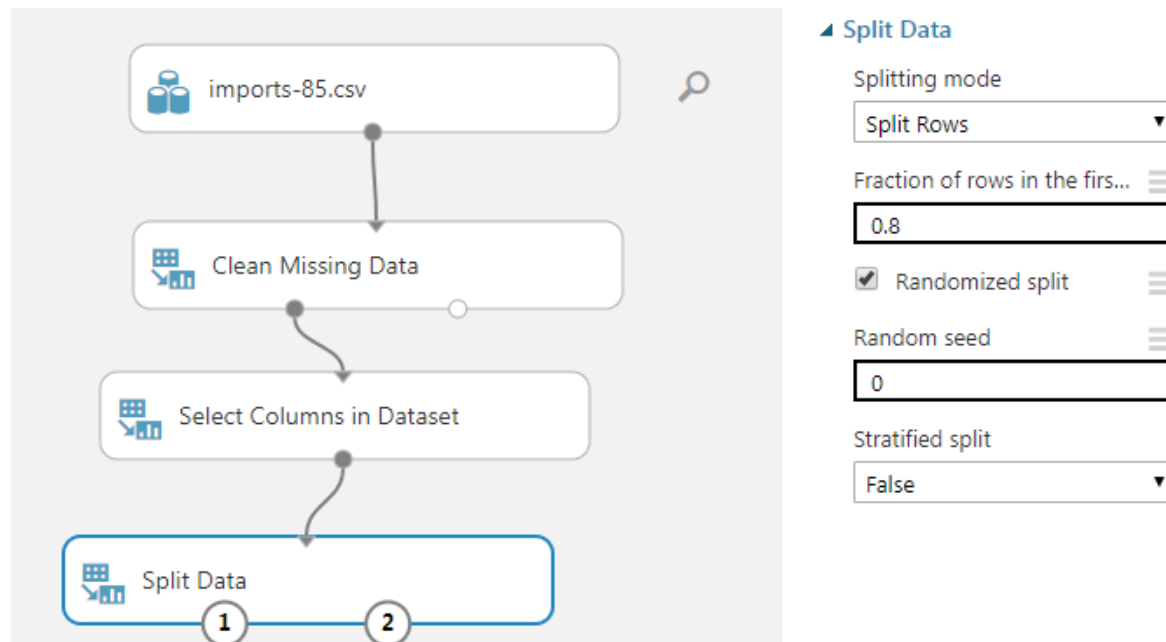
+ -

✓

- After adding the Select Columns module and configuring the module options, we have now chosen to effectively exclude two columns from our initial dataset. The assumption here is that the columns named bore and stroke will have no material impact on the prediction model and therefore do not need to be included in the input dataset. The addition and configuration of the Select Columns module will also have an impact when we configure a web service for our experiment. The input endpoint for our web service will respect this setting for the desired input columns. The endpoint will therefore not require these two columns when we submit an incoming web service request.

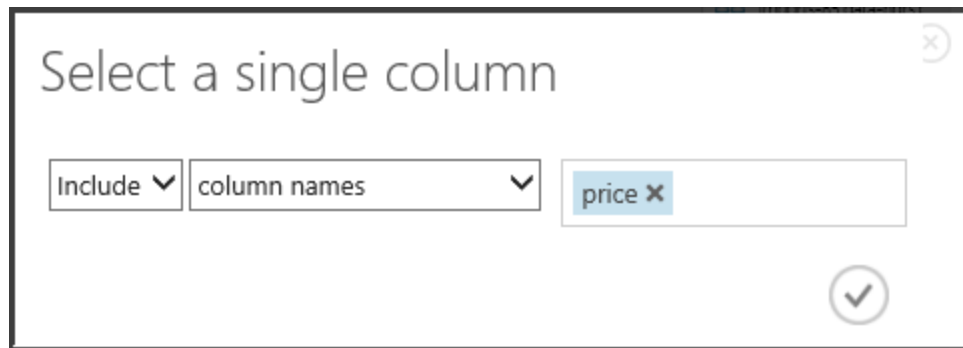
Split Data into Train and Test (Score)

- The next step is to add a Split module to our experiment so we can divert a certain percentage of our dataset for training purposes. The Split module will also designate another percentage for scoring and evaluating our new prediction model. As before, you can find this module by typing Split into the search bar on the top of the left navigation bar and let the Azure ML Studio search function find it for you.
- Once you find the Split module, drag it onto the Azure ML Studio design surface and place it under the Project Columns module. Next, drag and connect the Split module to the Project Columns module. Now, click the Split module and set the parameter for the Fraction Of Rows In The First Output Dataset to .80. In this context, .80 means that we use 80 percent of the data in the Imports-85.data.csv dataset for training our model.



Add Train Model Module

- The next step is to find a Train Model module and add it below the Split module. You can add this module by either searching for it or manually traversing the left navigation bar to the Azure Machine Learning tab and then the Train subtab. Connect the bottom left connector of the Split modules to the top right connector of the Train module.
- Again, you will notice a red exclamation mark on the Train module, which means we need to finish configuring the module. Do this by clicking the module and then launching the column selector in the properties section on the right side of the Azure ML Studio designer. The Train module needs to know which column in the dataset is to be used for the prediction. Configure the column selector by selecting Include, selecting Column Names, and adding the price column for training.
- Training column is also called Label Column.



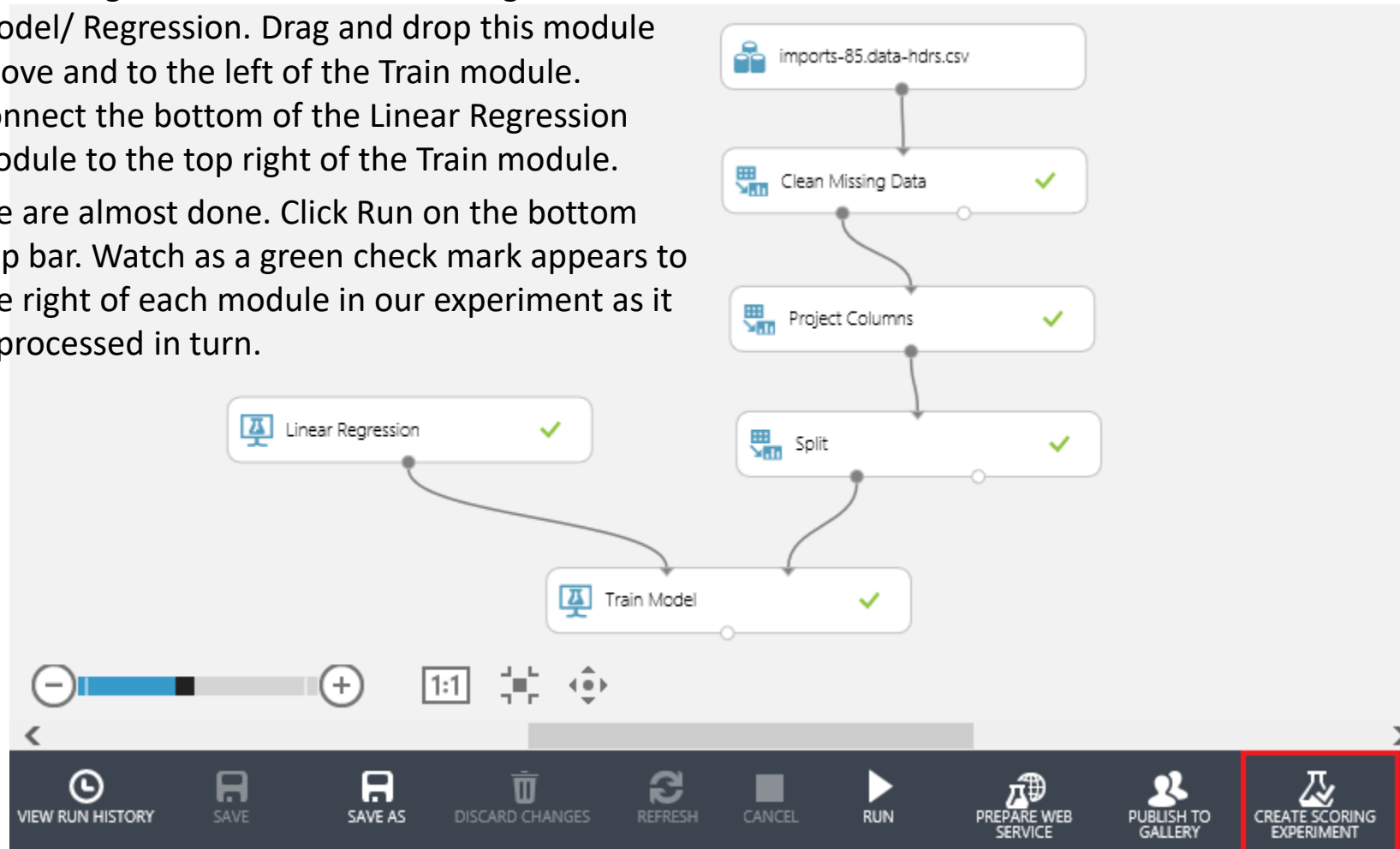
Select a single column

Include ▼ column names ▼ price x

✓

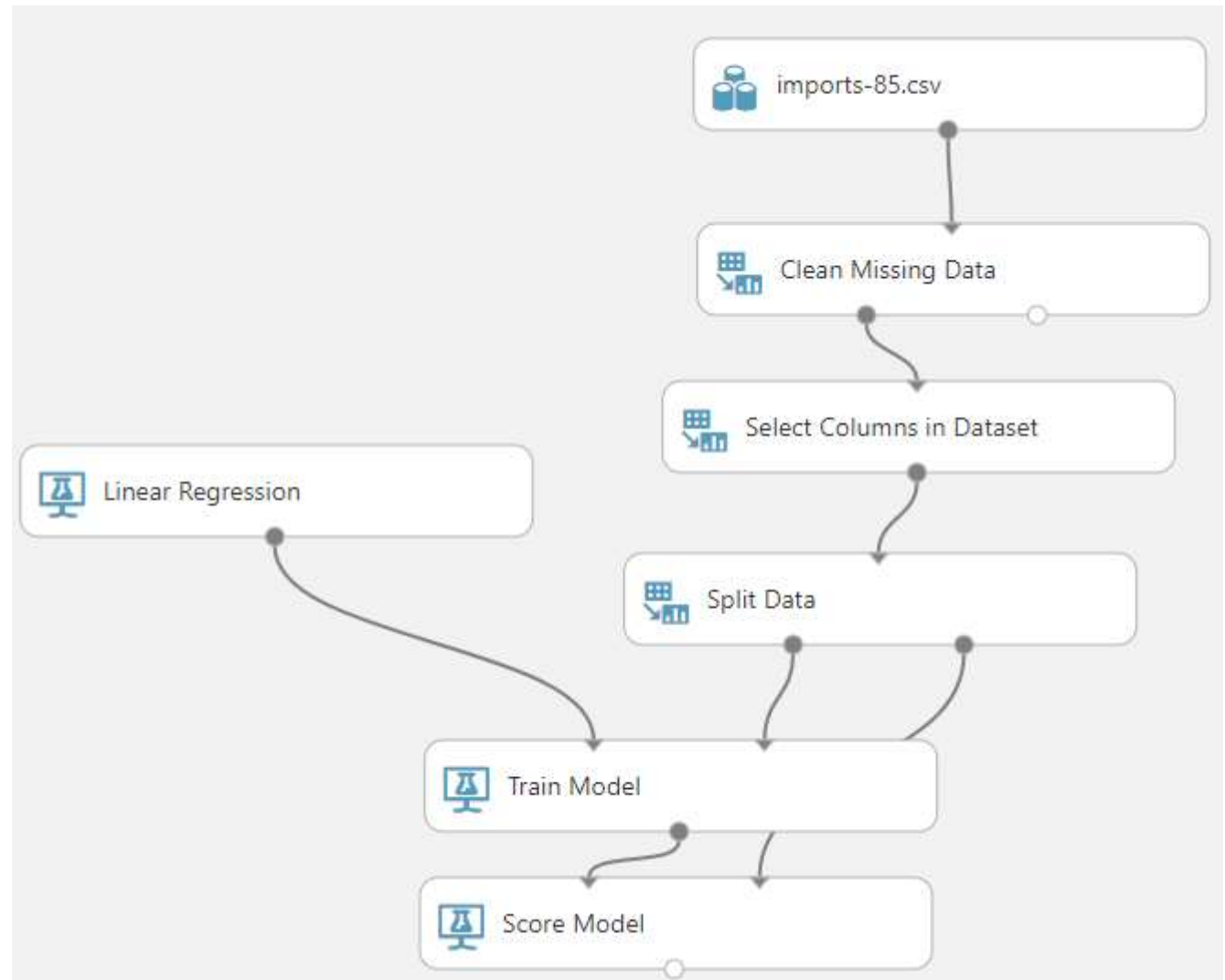
Add Algorithm (Regression) Module

- The next step is to find and add the Linear Regression module to the designer surface. You can find this model by using the search function or locating it under Machine Learning/Initialize Model/ Regression. Drag and drop this module above and to the left of the Train module. Connect the bottom of the Linear Regression module to the top right of the Train module.
- We are almost done. Click Run on the bottom app bar. Watch as a green check mark appears to the right of each module in our experiment as it is processed in turn.








Adding Score Module

- We want to be able to see how predictions of our trained model compare with actual prices. To do that we use remaining 20% of data and the Score Model. Our Experiment looks like:



Run the Experiment

- You can run upper portions of the graph above any selected module. You can run the entire experiment by hitting Run at the bottom tool bar. To see the effectiveness of the model, right click on the bottom of the score model. You will see the actual price and predicted price (Scored Label).
- You can argue that predictions are good or bad.
- What you actually do is try different models and algorithms until you find one which pleases you.
- Save your model.

peak-rpm	city-mpg	highway-mpg	price	Scored Labels
				
5200	19	25	13499	17472.599887
5400	38	43	0	3554.97592
4800	27	32	7898	3125.752471
5200	17	22	13499	18153.169089
4800	38	47	7738	6606.188435
4750	16	18	34184	40515.378841
5000	19	24	12629	12541.982188
4350	22	25	28176	25880.330453
5500	25	31	10345	11150.662809
5500	24	30	7689	7952.809827
4800	26	32	10245	11592.009387
5500	19	27	22018	-15556.982573
5500	31	38	6229	6501.288152
5500	31	38	6692	7448.283572
5200	17	23	19699	18690.257322

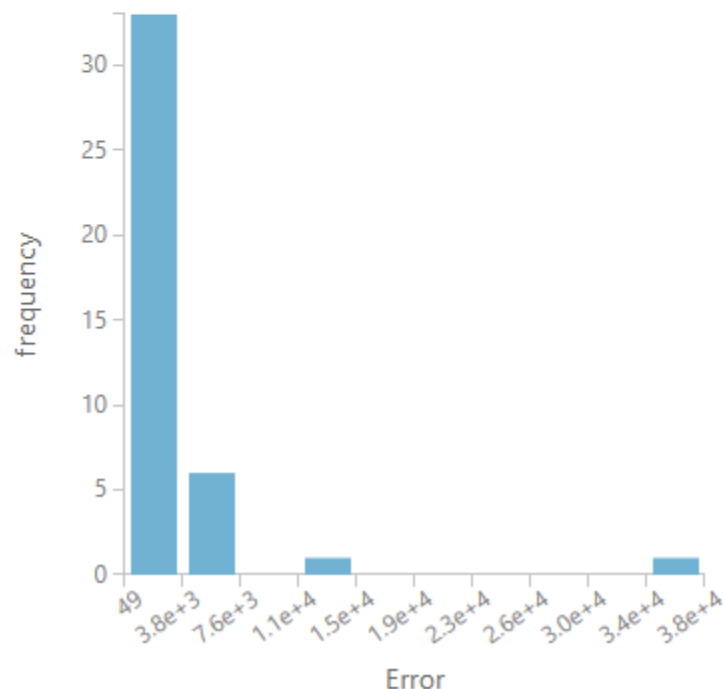
Evaluate Model

- We need more concise measures of success of our model. That is the purpose of Evaluate Model. Evaluate Model is added below the Score Model and fed the output of the Score Model.
- When we right click on the bottom of the Evaluate Model we get several Statistical Measures of the quality of our model.

Metrics

Mean Absolute Error	3170.102968
Root Mean Squared Error	6784.944583
Relative Absolute Error	0.451707
Relative Squared Error	0.540751
Coefficient of Determination	0.459249

Error Histogram



Evaluating the Performance of Regression Models

- When dealing with regression models, it is very unlikely that our model will accurately predict the target variable, because the target variable can take on any real value. However, we would naturally like to understand how far away our predicted values are from the true values, so will we utilize a metric that takes into account the overall deviation.
- Some of the standard evaluation metrics used to measure the performance of regression models include the **Mean Squared Error (MSE)** and **Root Mean Squared Error (RMSE)**, the **Mean Absolute Error (MAE)**, the R-squared coefficient, and many others.

Mean Squared Error and Root Mean Squared Error

- MSE is the average of the squared error that is used as the loss function for least squares regression

$$\sum_{i=1}^n \frac{(\theta^T x(i) - y(i))^2}{n}$$

- It is the sum, over all the data points, of the square of the difference between the predicted and actual target variables, divided by the number of data points.
- RMSE is the square root of MSE. MSE is measured in units that are the square of the target variable, while RMSE is measured in the same units as the target variable.
- Due to its formulation, MSE, just like the squared loss function that it derives from, penalizes larger errors more severely.
- In order to evaluate our predictions based on the mean of an error metric, we will first make predictions for each input feature vector in an RDD of `LabeledPoint` instances by computing the error for each record using a function that takes the prediction and true target value as inputs. This will return a `[Double]` RDD that contains the error values. We can then find the average using the mean method of RDDs that contain Double values.
- Let's define our squared error function as follows:

```
def squared_error(actual, pred):  
    return (pred - actual)**2
```

Mean Absolute Error, Root Mean Squared Log Error

- MAE is the average of the absolute differences between the predicted and actual targets:

$$\sum_{i=1}^n \frac{|x(i) - y(i)|^1}{n}$$

- MAE is similar in principle to MSE, but it does not punish large deviations as much. The function to compute MAE could be written as follows:

```
def abs_error(actual, pred):  
    return np.abs(pred - actual)
```

Root Mean Squared Log Error

- RMSLE is not as widely used as MSE and MAE, but it is used as the metric for the Kaggle competitions that use the bike sharing dataset. It is effectively the RMSE of the log-transformed predicted and target values. This measurement is useful when there is a wide range in the target variable, and you do not necessarily want to penalize large errors when the predicted and target values are themselves high. It is also effective when you care about percentage errors rather than the absolute value of errors. We calculate RMSLE as

```
def squared_log_error(pred, actual):  
    return (np.log(pred + 1) - np.log(actual + 1))**2
```

Classification vs. Clustering

Frequently used data science algorithms used in machine learning today which are also available in Azure ML are:

- **Classification algorithms** These are used to classify data into different categories, which can then be used to predict one or more discrete variables, based on the other attributes in the dataset.
- **Clustering algorithms** These determine natural groupings and patterns in datasets. They are typically used to predict grouping classifications for a given variable.
- Classification and regression are usually categorized as supervised machine learning.
- This is due to the fact that in both cases, we provided training datasets that allowed the Azure Machine Learning algorithms the ability to learn from the data. The supervised learning in these algorithms determines what the correct prediction should be.
- Clustering algorithms are used in situations where we want the machine to conduct its own analysis on the dataset, determine relationships, infer logical groupings, and generally attempt to make sense of chaos by determining the forests from the trees.
- Clustering algorithms are referred to as unsupervised learning

Cognitive Services

- Azure has organized various algorithms exposed in Azure Machine Learning in 20+ domain specific services. Each of those services comes with a programming API which allow you to incorporate those services in your applications and use full power of Azure Cloud Machine Learning.

Those services fall in 5 groups:

- **Vision**: Image-processing algorithms to smartly identify, caption and moderate your pictures.
- **Speech**: Convert spoken audio into text, use voice for verification, or add speaker recognition to your app.
- **Knowledge**: Map complex information and data in order to solve tasks such as intelligent recommendations and semantic search.
- **Search**: Add Bing Search APIs to your apps and harness the ability to comb billions of webpages, images, videos, and news with a single API call.
- **Language**: Allow your apps to process natural language with pre-built scripts, evaluate sentiment and learn how to recognize what users want.

Vision Services

Computer Vision API

- Distill actionable information from images

Face API

- Detect, identify, analyze, organize, and tag faces in photos

Content Moderator

- Automated image, text, and video moderation

Emotion API PREVIEW

- Personalize user experiences with emotion recognition

Custom Vision Service PREVIEW

- Easily customize your own state-of-the-art computer vision models for your unique use case

Video Indexer PREVIEW

- Easily extract insights from your videos and quickly enrich your applications to enhance discovery and engagement

Speech Services

Translator Speech API

- Easily conduct real-time speech translation with a simple REST API call

Speaker Recognition API PREVIEW

- Use speech to identify and authenticate individual speakers

Bing Speech API

- Convert speech to text and back again to understand user intent

Custom Speech Service PREVIEW

- Overcome speech recognition barriers like speaking style, background noise, and vocabulary

Language Services

Language Understanding (LUIS)

- Teach your apps to understand commands from your users

Text Analytics API

- Easily evaluate sentiment and topics to understand what users want

Bing Spell Check API

- Detect and correct spelling mistakes in your app

Translator Text API

- Easily conduct machine translation with a simple REST API call

Web Language Model API PREVIEW

- Use the power of predictive language models trained on web-scale data

Linguistic Analysis API PREVIEW

- Simplify complex language concepts and parse text with the Linguistic Analysis API

Knowledge Services

Recommendations API PREVIEW

- Predict and recommend items your customers want

Academic Knowledge API PREVIEW

- Tap into the wealth of academic content in the Microsoft Academic Graph

Knowledge Exploration Service PREVIEW

- Enable interactive search experiences over structured data via natural language inputs

QnA Maker API PREVIEW

- Distill information into conversational, easy-to-navigate answers

Entity Linking Intelligence Service API PREVIEW

- Power your app's data links with named entity recognition and disambiguation

Custom Decision Service PREVIEW

- A cloud-based, contextual decision-making API that sharpens with experience

Search Services

Bing Autosuggest API

- Give your app intelligent autosuggest options for searches

Bing Image Search API

- Search for images and get comprehensive results

Bing News Search API

- Search for news and get comprehensive results

Bing Video Search API

- Search for videos and get comprehensive results

Bing Web Search API

- Get enhanced search details from billions of web documents

Bing Custom Search API

- An easy-to-use, ad-free, commercial-grade search tool that lets you deliver the results you want

Bing Entity Search API PREVIEW

- Enrich your experiences by identifying and augmenting entity information from the web