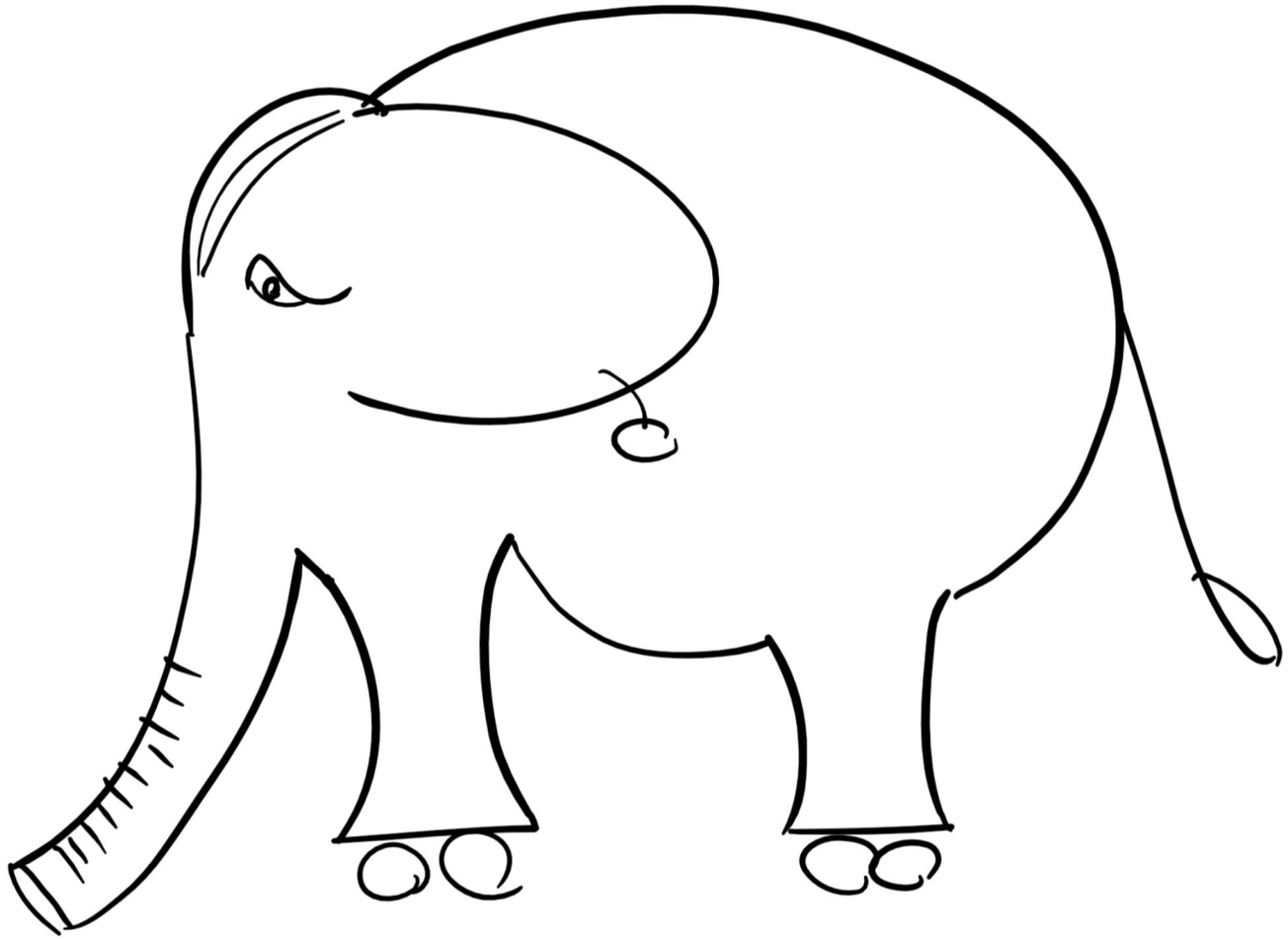


Doing Machine Learning with Azure ML

Seth Mottaghinejad

Data Scientist

sethmott@microsoft.com

www.linkedin.com/in/sethmott/

I believe over the next decade computing will become even more ubiquitous and intelligence will become ambient… This will be made possible by an ever-growing network of connected devices, incredible computing capacity from the cloud, insights from big data, and intelligence from machine learning.

Chapter 1

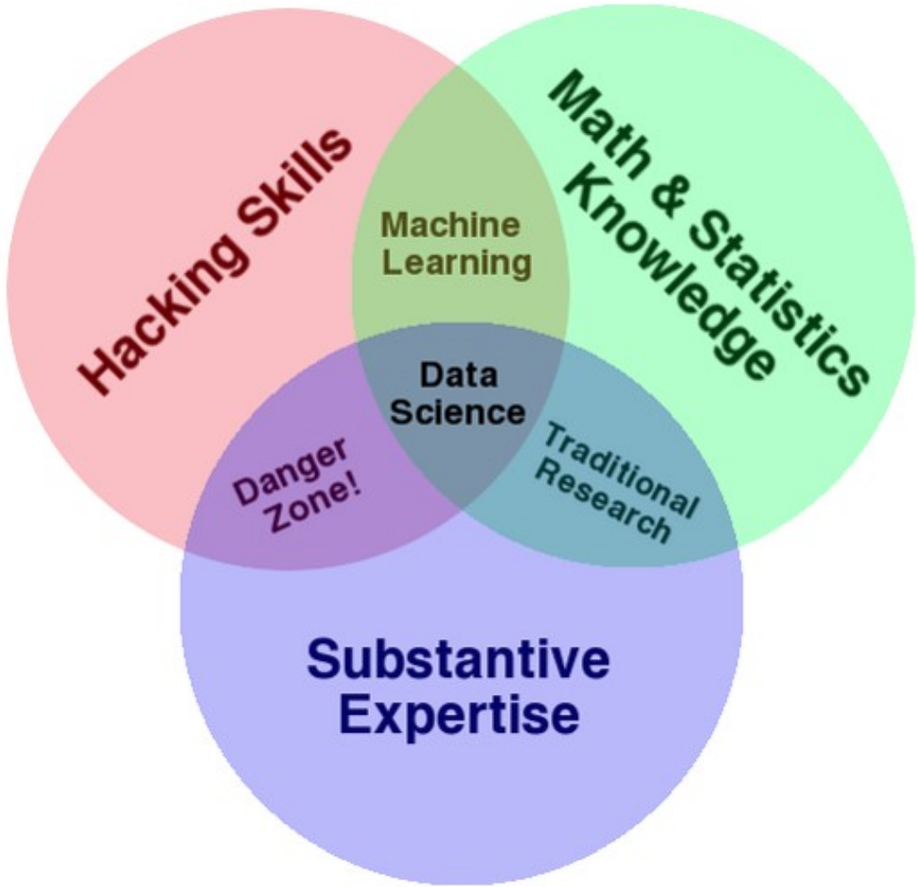
Basic overview

Here is what we are going to cover:

- statistics and data visualization
- machine learning concepts and examples
- Azure ML Studio
- data science process and workflow

There will be vocabulary and concepts, with **quizzes** for review and lots **hands-on labs** in the AzureML Studio.

Three essential skills of data scientists



Drew Conway

www.dataists.com/2010/09/the-data-science-venn-diagram/

What do data scientists do?

business understanding

- **ask**: identify key business variables
- **measure**: define success metrics

data acquisition and understanding

- **data ingestion**: ingest the data into analytic environment
- **data exploration**: explore the data to check quality and adequacy

modeling

- **pre-processing**: set up data pipeline to prepare data
- **feature engineering**: create features from raw data
- **model evaluation**: how well does the model fit?
- **model selection**: explore and find the "right" model

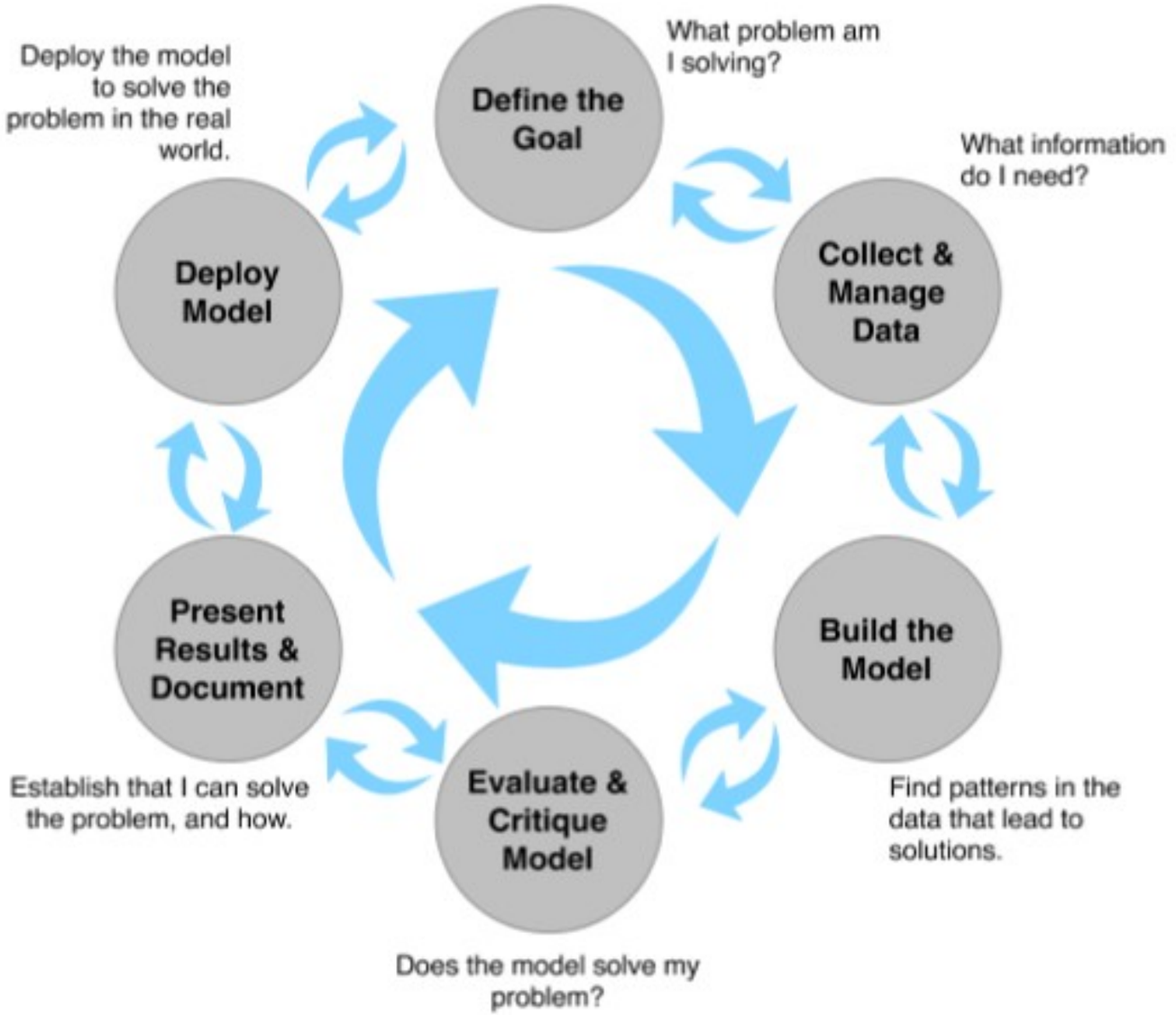
deployment

- **operationalization**: deploy model to production
- **model consumption**: use model to make predictions (scoring)
- **business validation**: are business requirements met?

last but not least

iterate, iterate, iterate...

What do data scientists do?

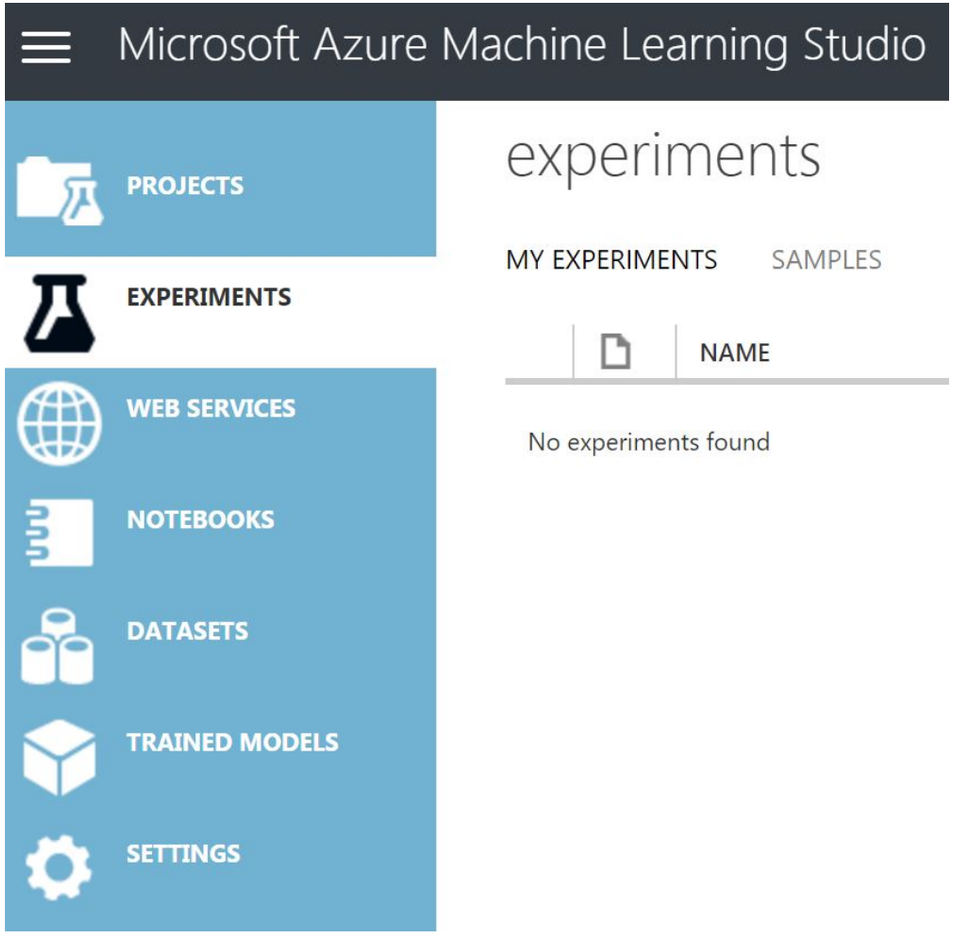


Azure Machine Learning

Our goal is to make machine learning **accessible** to every enterprise, data scientist, developer, information worker, consumer, and device anywhere in the world.

An Azure ML **experiment** is a blank canvas connecting **datasets to modules**, each module performing some action on the data.

You build a model in a training experiment and convert it to a predictive experiment to publish it as a web service so that your model can be consumed. An ML **project** is a collections of experiments, datasets, notebooks, and other resources.


Microsoft Azure Machine Learning Studio
PROJECTS
EXPERIMENTS
WEB SERVICES
NOTEBOOKS
DATASETS
TRAINED MODELS
SETTINGS
experiments
MY EXPERIMENTS
SAMPLES
NAME
No experiments found


Search experiment items
Saved Datasets
Trained Models
Transforms
Data Format Conversions
Data Input and Output
Data Transformation
Feature Selection
Machine Learning
OpenCV Library Modules
Python Language Modules
R Language Modules
Statistical Functions
Text Analytics
Time Series
Web Service

LAB 1

Our first experiment

Expected lab duration: 15 minutes.

About the labs

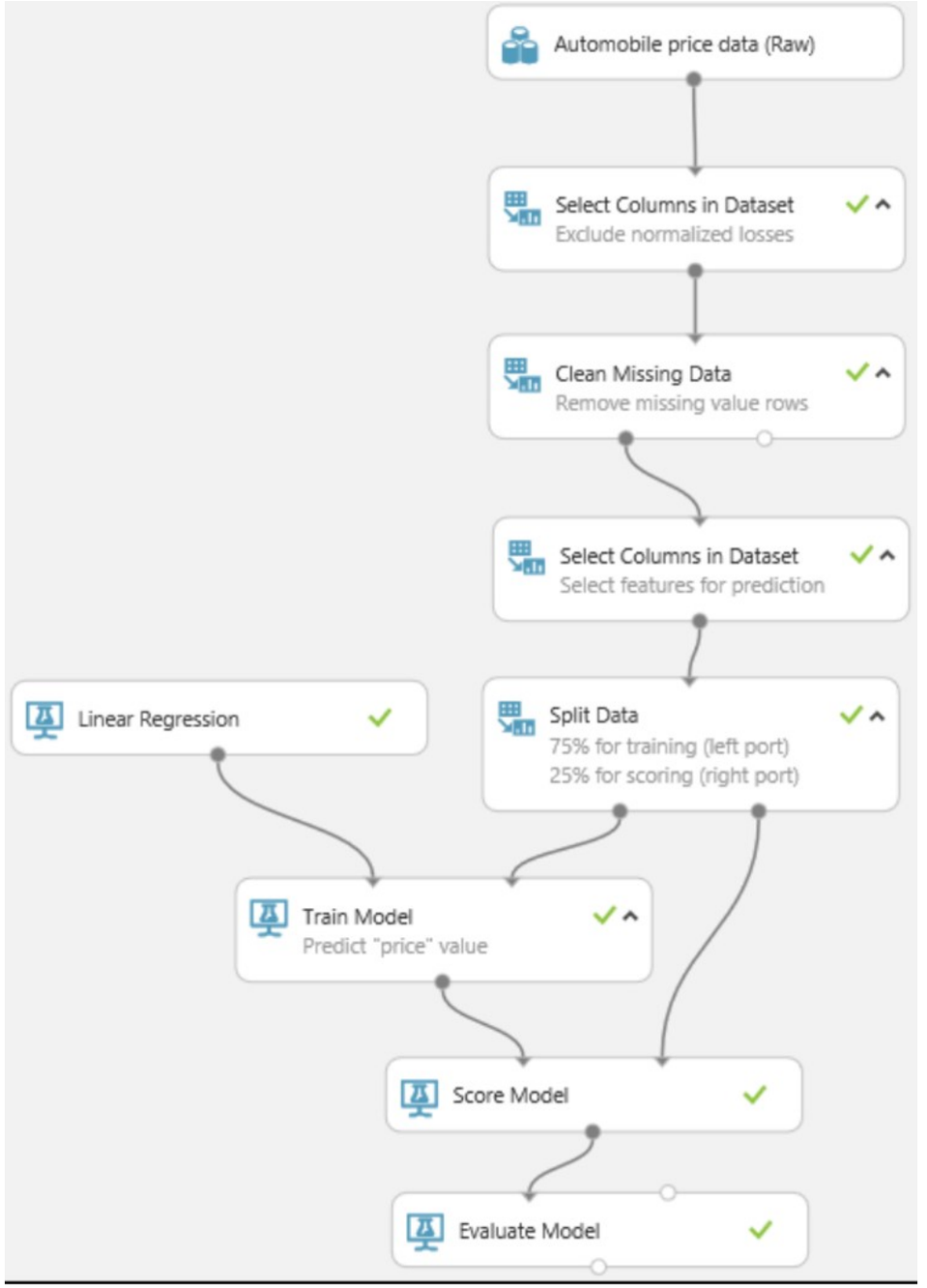
To make it easier to understand the instructions

- Names of `modules` are in a grey box.
- `Column` names and code have a grey background and monospace font.
- A screenshot of the finished experiment is sometimes provided as a visual guide to see the final arrangement. You can click on the screenshot to go back to reading instructions.
- Instructions are not too detailed because the interface can sometimes change and because users can become more familiar with the interface by looking around.

In this lab, we create and run the experiment shown in this screenshot.

1. Create a new experiment called **Automobile price prediction**.
2. Drag in `Automobile price data (Raw)` and visualize it.
3. Drag `Select Columns in Dataset` and remove `normalized-losses`.
4. Double-click on the module and entere text saying "column was removed due to too many NAs".
5. Drag `Clean Missing Data` and remove entire row if any column is missing.

6. Drag Select Columns in Dataset and paste in the following columns: `make, body-style, wheel-base, engine-size, horsepower, peak-rpm, highway-mpg, price`
7. Run the Split Data to split the data 75% to 25%.
8. Connect Split Data to a Train Model and connect that to a Linear Regression module in turn.
9. Connect the second split to a Score Model module.
10. Drag Evaluate Model to evaluate the model.


Automobile price data (Raw)
Select Columns in Dataset
Exclude normalized losses
Clean Missing Data
Remove missing value rows
Select Columns in Dataset
Select features for prediction
Linear Regression
Split Data
75% for training (left port)
25% for scoring (right port)
Train Model
Predict "price" value
Score Model
Evaluate Model

END OF LAB

Resources

- A listing of all the modules: https://msdn.microsoft.com/library/azure/dn906033.aspx
- Go to Cortana Intelligence Gallery and examine some pre-existing experiments: http://gallery.cortanaintelligence.com/
- Learn this basic infographic
- Azure Machine Learning Documentation Center: https://azure.microsoft.com/services/machine-learning/

End of chapter

Chapter 2

Just enough statistics

What we will learn

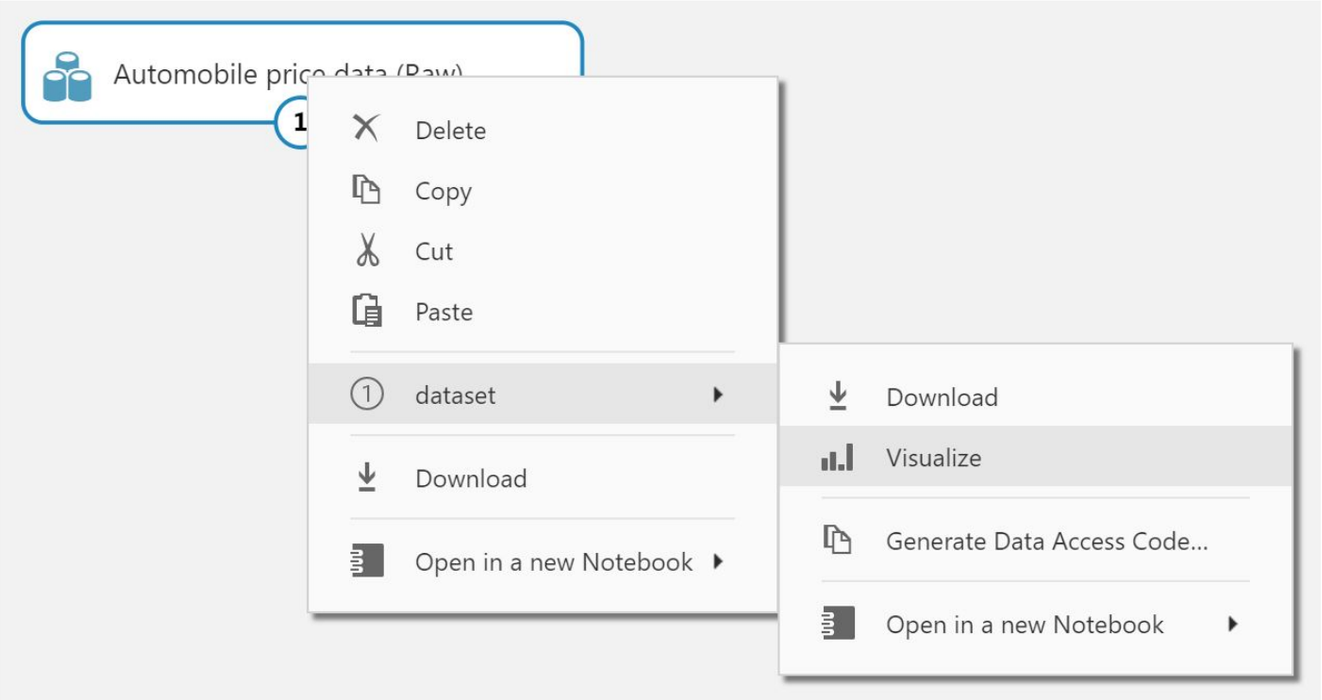
- variables: dependent, response, independent, explanatory, numeric, categorical, nominal, ordinal
- features and target
- exploratory data analysis
- mean, median, standard deviation, range, percentiles or quantiles, univariate vs bivariate summaries and visualizations, histograms and density plots

Exploratory data analysis (EDA)

We have a new dataset, so we *slice and dice* the data to get a feel for it, for example we

- use statistical summaries to see if the data makes sense or if outliers are present
- examine missing values
- visualize the data looking for any patterns
- bivariate statistics and plots can hint at relationships between variables

Visualize the **automobile price prediction** data from the lab



rows	columns
205	26

normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
								
	alfa-romero	gas	std	two	convertible	rwd	front	88.6
	alfa-romero	gas	std	two	convertible	rwd	front	88.6
	alfa-romero	gas	std	two	hatchback	rwd	front	94.5
164	audi	gas	std	four	sedan	fwd	front	99.8
164	audi	gas	std	four	sedan	4wd	front	99.4
	audi	gas	std	two	sedan	fwd	front	99.8
158	audi	gas	std	four	sedan	fwd	front	105.8
	audi	gas	std	four	wagon	fwd	front	105.8
158	audi	gas	turbo	four	sedan	fwd	front	105.8

Tabular data is made up of rows and columns. Statisticians sometimes call them observations and variables.

Some variables are numeric. Numeric variables may be **discreet** or **continuous**.

Some variables are categorical. Categorical variables may be **nominal** when there is no order or **ordinal** when there is an order. Categorical data with only two categories is sometimes called **binary**.

A numeric variable can sometimes be treated (by us and our models) as if it's categorical.

Can you describe this dataset?

rows	columns
205	26

normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
								
	alfa-romero	gas	std	two	convertible	rwd	front	88.6
	alfa-romero	gas	std	two	convertible	rwd	front	88.6
	alfa-romero	gas	std	two	hatchback	rwd	front	94.5
164	audi	gas	std	four	sedan	fwd	front	99.8
164	audi	gas	std	four	sedan	4wd	front	99.4
	audi	gas	std	two	sedan	fwd	front	99.8
158	audi	gas	std	four	sedan	fwd	front	105.8
	audi	gas	std	four	wagon	fwd	front	105.8
158	audi	gas	turbo	four	sedan	fwd	front	105.8

You can describe each variable using **summary statistics** and **visualizations**.

statistic	describes	useful because...
mean	average	single-number representation of a column
median	number in the middle	like mean, but robust to outliers
min	lowest value	check for data quality issues
max	highest value	check for data quality issues
standard deviation	average deviation from the mean	measure of spread or variability
unique values	number of distinct numbers or categories	find number of categories or distcreet numeric variables
missing values	how much of the variable is missing	check for data quality issues

Another common statistic is the **percentile**: The pth percentile is the value such that p *percent of the data falls below it* (or $1-p$ percent of the data is above it). We get percentiles by first sorting the data from small to large.

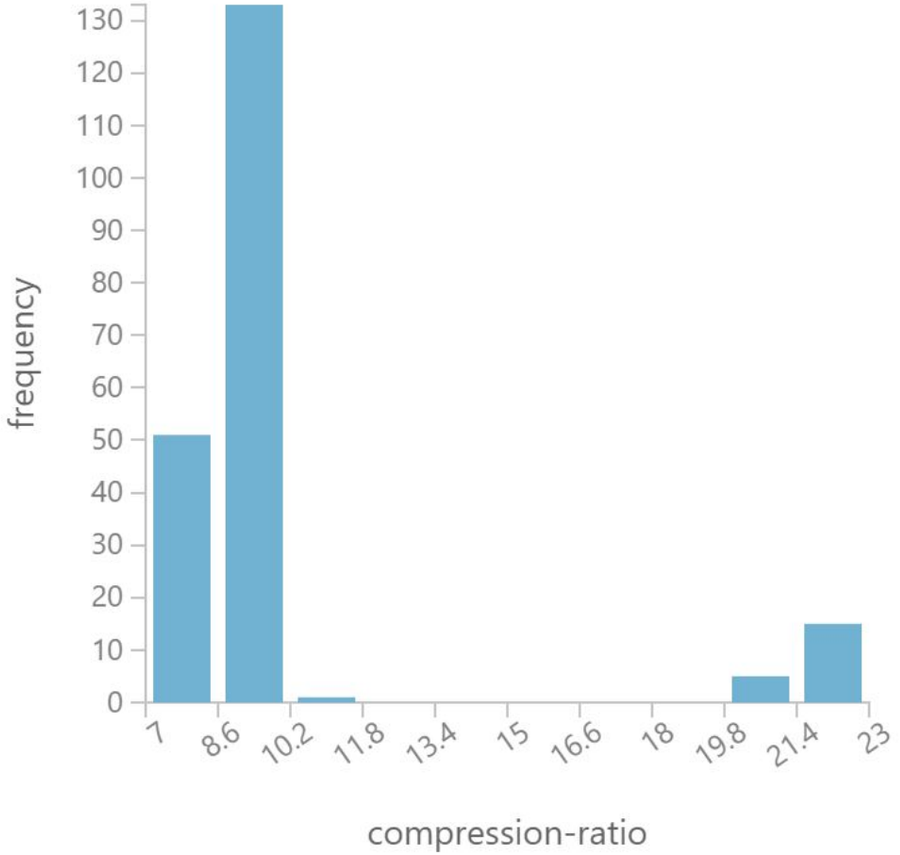
- the median is the 50th percentile, so 50% of the data is below the median (and 50% above it)
- the minimum is the 0th percentile and the maximum is the 100th percentile
- the 23rd precentile is the number that's bigger than 23% of the observations

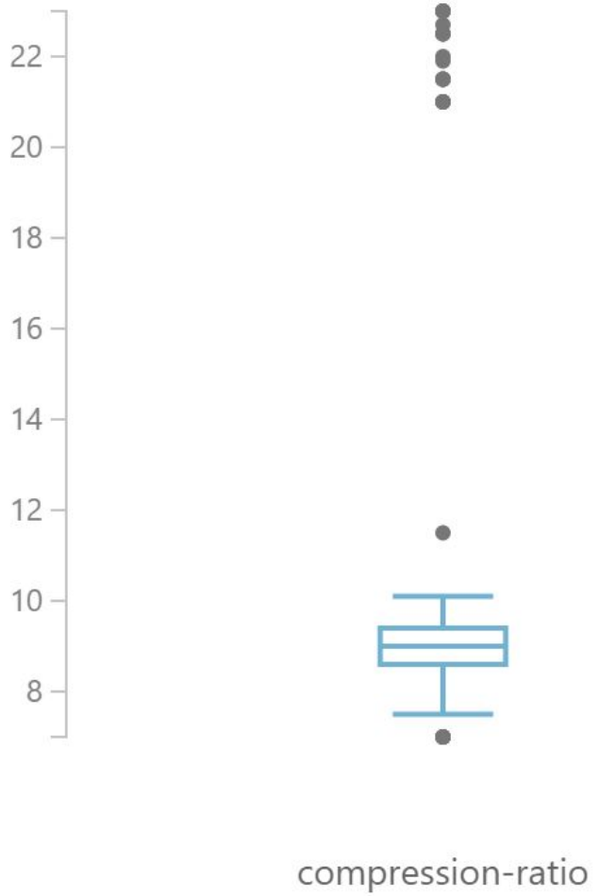
statistic	describes	useful because...
$q1$	first quartile	cut-off for bottom 25 percent of a variable
$q3$	third quartile	cut-off for top 25 percent of a variable
$p1$	1st percentile	numbers below may be outliers (compare it to the minimum)
$p5$	5th percentile	cut-off for bottom 5 percent of a variable
$p95$	95th percentile	cut-off for top 5 percent of a variable
$p99$	99th percentile	numbers above may be outliers (compare it to the maximum)

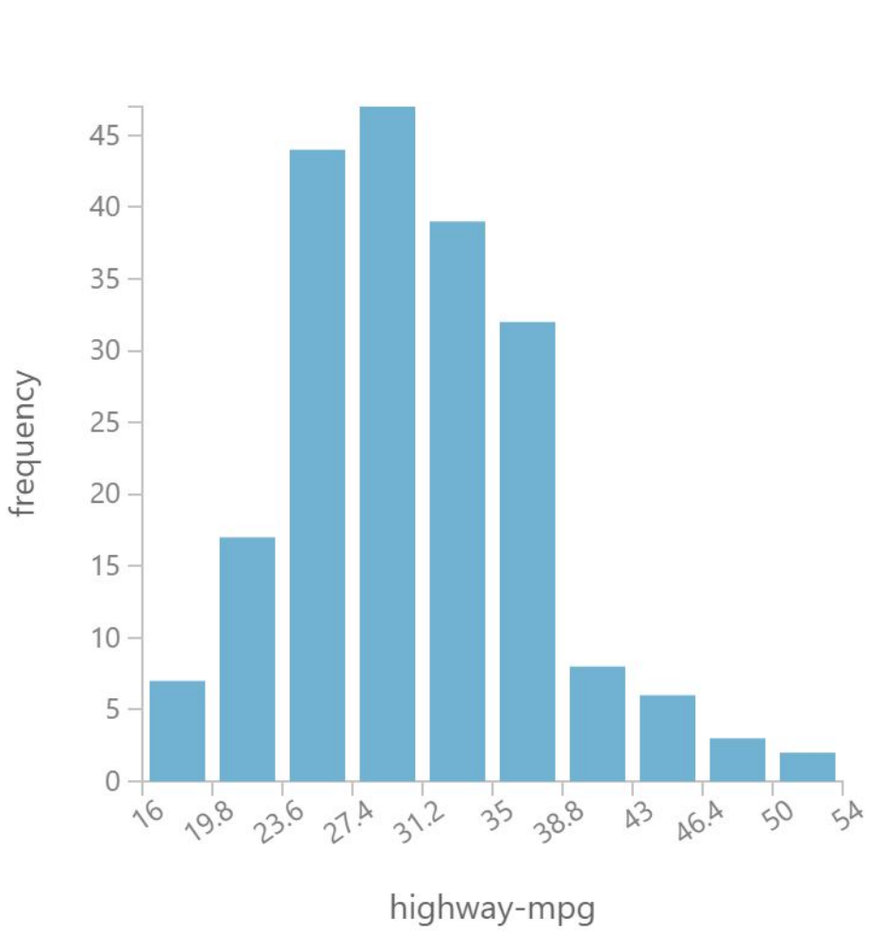
Visualizing data can reveal a lot of interesting trends. A histogram and boxplot are examples of univariate visualizations, because they involve only one variable and answer:

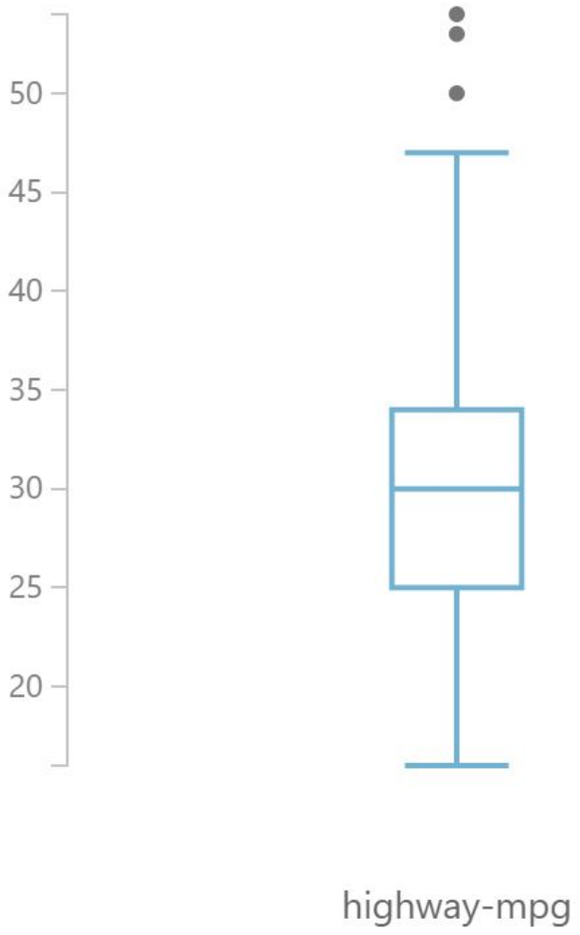
- is there a single peak or more than one peak?
- are the values tight or spread-out?
- is the data symmetric or heavy-tailed?
- are there any outliers (subjective)?

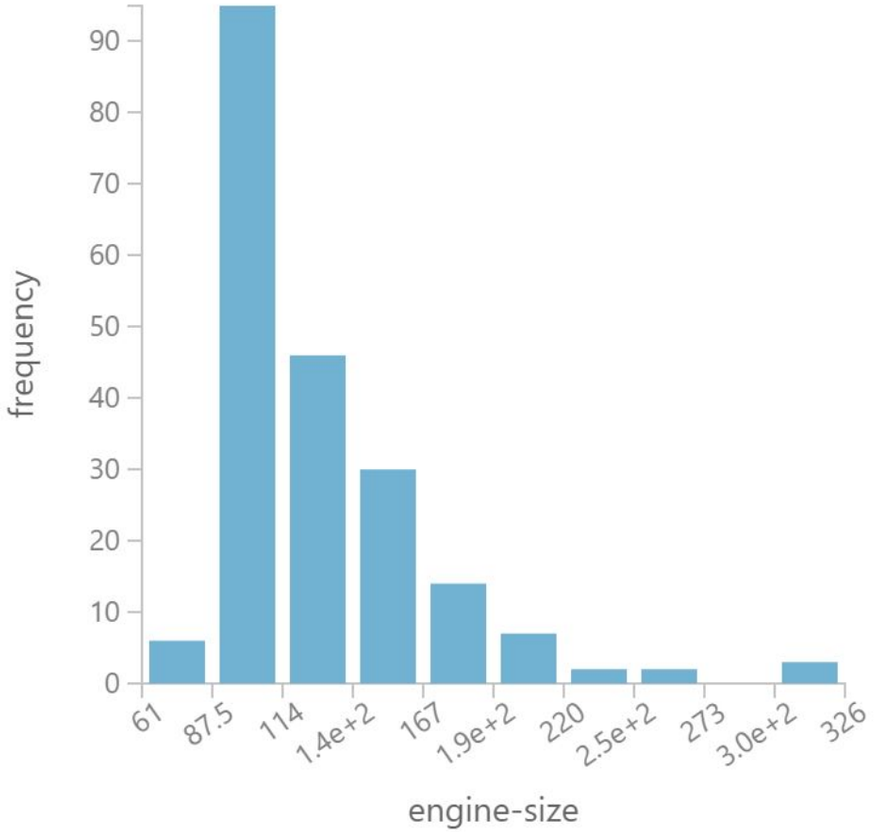
A **scatter plot** is an example of a bivariate visualization: it shows the relationship between two variables.

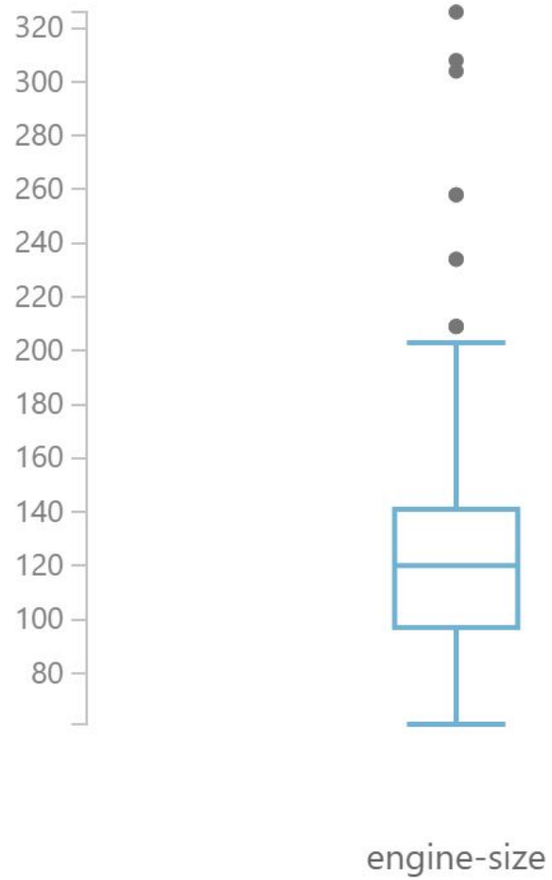

frequency
130
120
110
100
90
80
70
60
50
40
30
20
10
0
7
8.6
10.2
11.8
13.4
15
16.6
18
19.8
21.4
23
compression-ratio

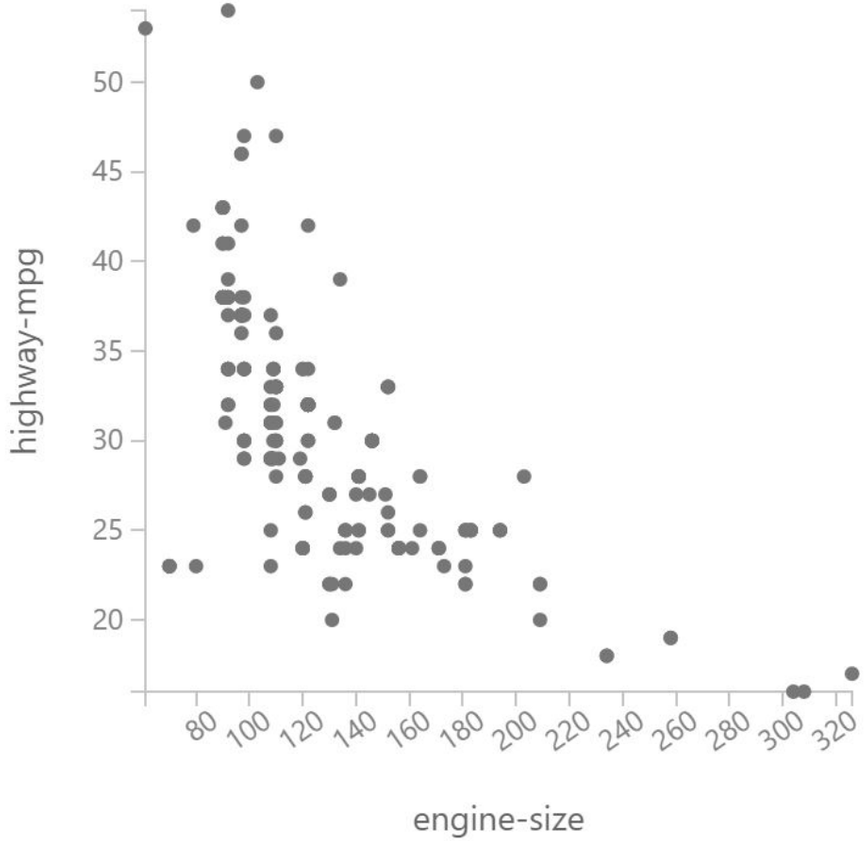

22
20
18
16
14
12
10
8
compression-ratio

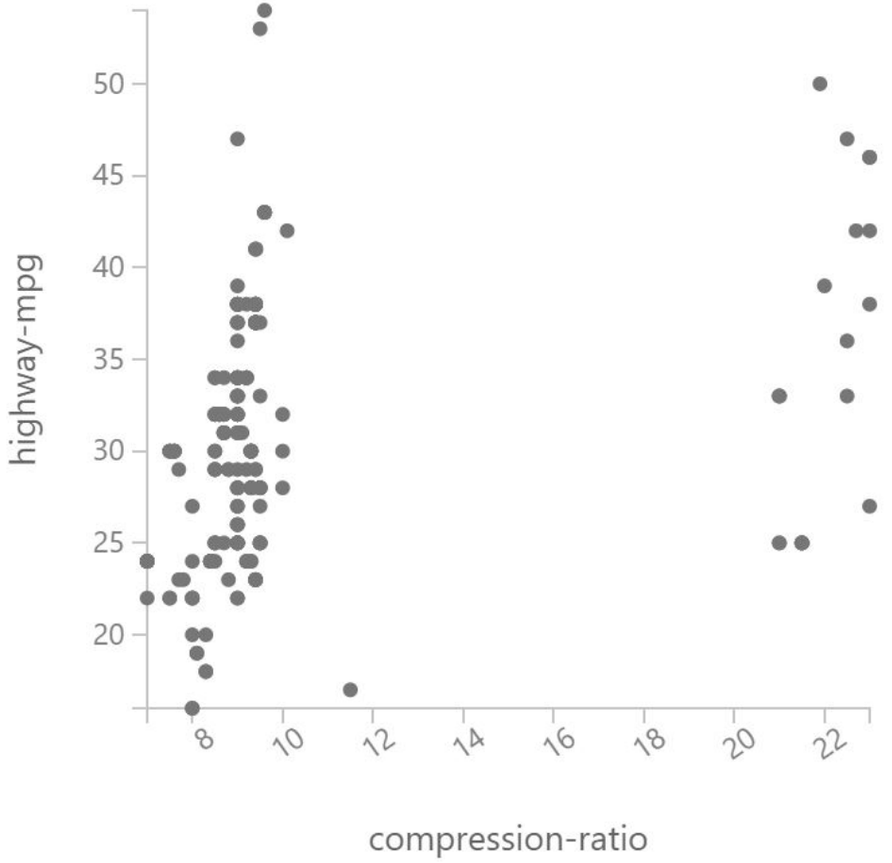

frequency
0
5
10
15
20
25
30
35
40
45
16
19.8
23.6
27.4
31.2
35
38.8
43
46.4
50
54
highway-mpg


20
25
30
35
40
45
50
highway-mpg


frequency
90
80
70
60
50
40
30
20
10
0
61
87.5
114
1.4e+2
167
1.9e+2
220
2.5e+2
273
3.0e+2
326
engine-size


320
300
280
260
240
220
200
180
160
140
120
100
80
engine-size


highway-mpg
50
45
40
35
30
25
20
80
100
120
140
160
180
200
220
240
260
280
300
320
engine-size


highway-mpg
50
45
40
35
30
25
20
8
10
12
14
16
18
20
22
compression-ratio

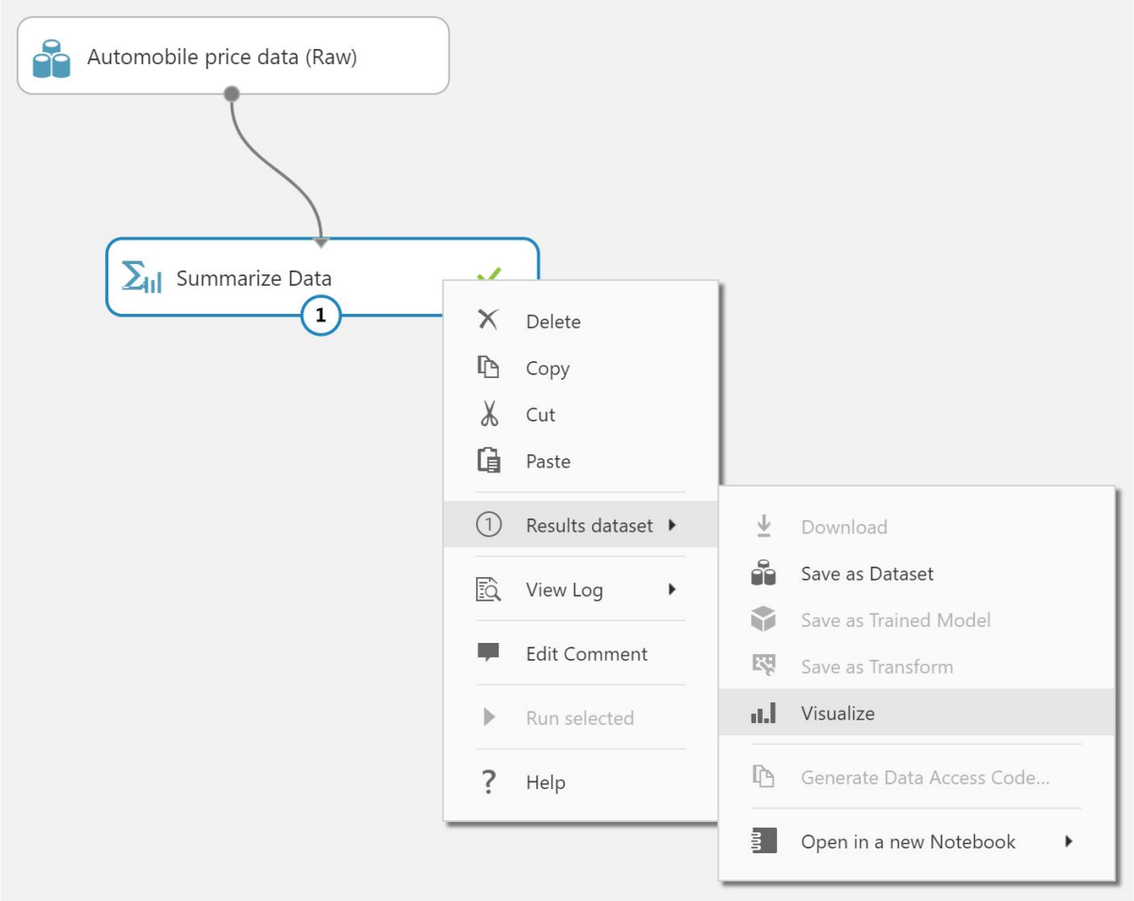
visualization	describes...	useful because...
histogram	distribution of a variable	find the peaks and tails
density plot	(continuous) fine-grained histogram	identify smaller peaks or tails the histogram missed
box plot	simplified distribution of a numeric variable showing quantiles	identify outliers and range
scatter plot	relationship between two numeric variables	look for trends (linear or non-linear)

For numeric variables with a heavy-tailed distributions, it helps to apply a transformation like the log to resize the visualization and make trends more apparent.

LAB 2

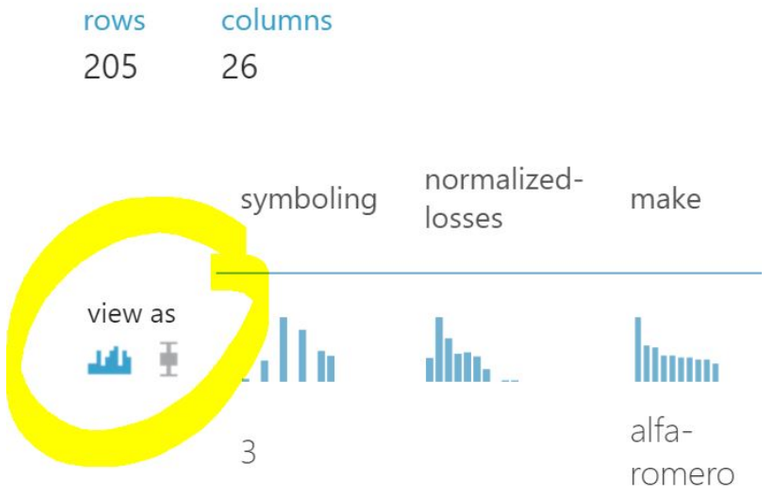
More summary statistics

Expected lab duration: 30 minutes.


Automobile price data (Raw)
Summarize Data
1
Delete
Copy
Cut
Paste
Results dataset
View Log
Edit Comment
Run selected
Help
Download
Save as Dataset
Save as Trained Model
Save as Transform
Visualize
Generate Data Access Code...
Open in a new Notebook

1. Find the summary statistics for the numeric column `price` and describe what you see.
2. Do the same for the variable `make`. Why are there fewer summary statistics for `make` than `price`?
3. Look at a histogram (bar chart) for `make`. What are the most and least common values for `make`?
4. Look at the histogram for `price`. Describe the distribution of `price`.
5. In the "compare to" drop-down box, choose `horsepower` to see a scatter plot of `price` against `horsepower`. Describe what you see.

6. In the "compare to" drop-down box, select "None" again to see the histogram for `price`. Since `price` is numeric, you can change the histogram to a boxplot by changing "view at" (next to where the number of rows and columns are reported).



7. Describe the boxplot for `price`. What sort of information does the boxplot convey better than the histogram? And vice versa?

8. In the "compare to" drop-down box, choose `make` to see a separate boxplot plot of `price` for each value of `make` (top 5 by default). Describe what you see.
9. Now drag in Summarize Data, run it and visualize it.
10. Interpret the 1st and 99th percentiles for `price` and compare them to the min and max. Does it seem like `price` has outliers? Does this confirm what we saw in the histogram for `price`?
11. Which columns in the data have a symmetric distribution and which ones have a skewed distribution?

END OF LAB

The importance of data visualization

Anscombe's quartet

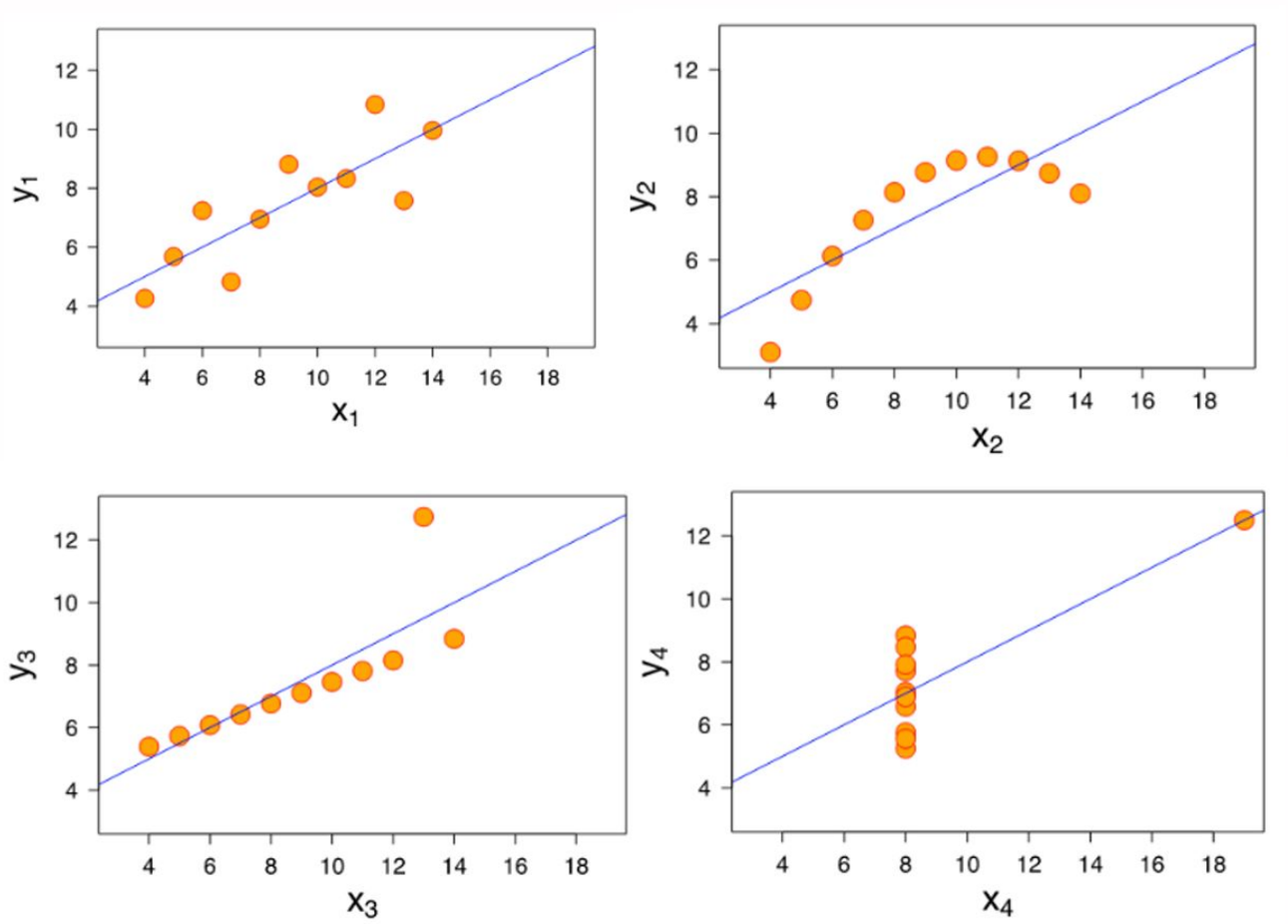
I		II		III		IV	
x	y	x	y	x	y	x	y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

https://en.wikipedia.org/wiki/Anscombe's_quartet

The four datasets seem very similar

- The average x value is 9 for each dataset.
- The average y value is 7.50 for each dataset.
- The variance for x is 11 and the variance for y is 4.12.
- Are these 4 datasets similar? What does it look like if we plot this data?

Until you plot them, that is.



Let's recap

- exploratory data analysis
- variable, column, row, observation
- numeric variables, categorical variables (including binary)
- nominal and ordinal variables
- mean, median, standard deviation, range, percentiles and quantiles
- univariate and bivariate summaries and visualizations
- histograms, boxplots, scatter plots

Quiz

True or false: a feature is a row in the data, a variable is a column in the data.

Answer

False.

> Features and variables usually refer to the same thing. Generally speaking we can think of a feature as a column in the data conveying some kind of information. Note that we usually have to do a lot of pre-processing to create useful features for modeling from the raw data.

Quiz

The average of a set of numbers is 200, but their median is 10. How can you explain this discrepancy?

Answer

> The data must be right-skewed (have some "outliers" in the positive direction). The mean (average) is dragged in the direction of outliers and therefore it's high. The median is not affected by outliers and remains low.

Quiz

What are some common reasons to perform EDA?

Answer

> Examine the distribution of each variable (skew, outliers, single peak, etc.).
> Examine correlations between variables.
> Find and try to account for missing values.
> Find more obvious trends and ask if they make sense.
> Know what kinds of pre-processing steps may be needed prior to any modeling.

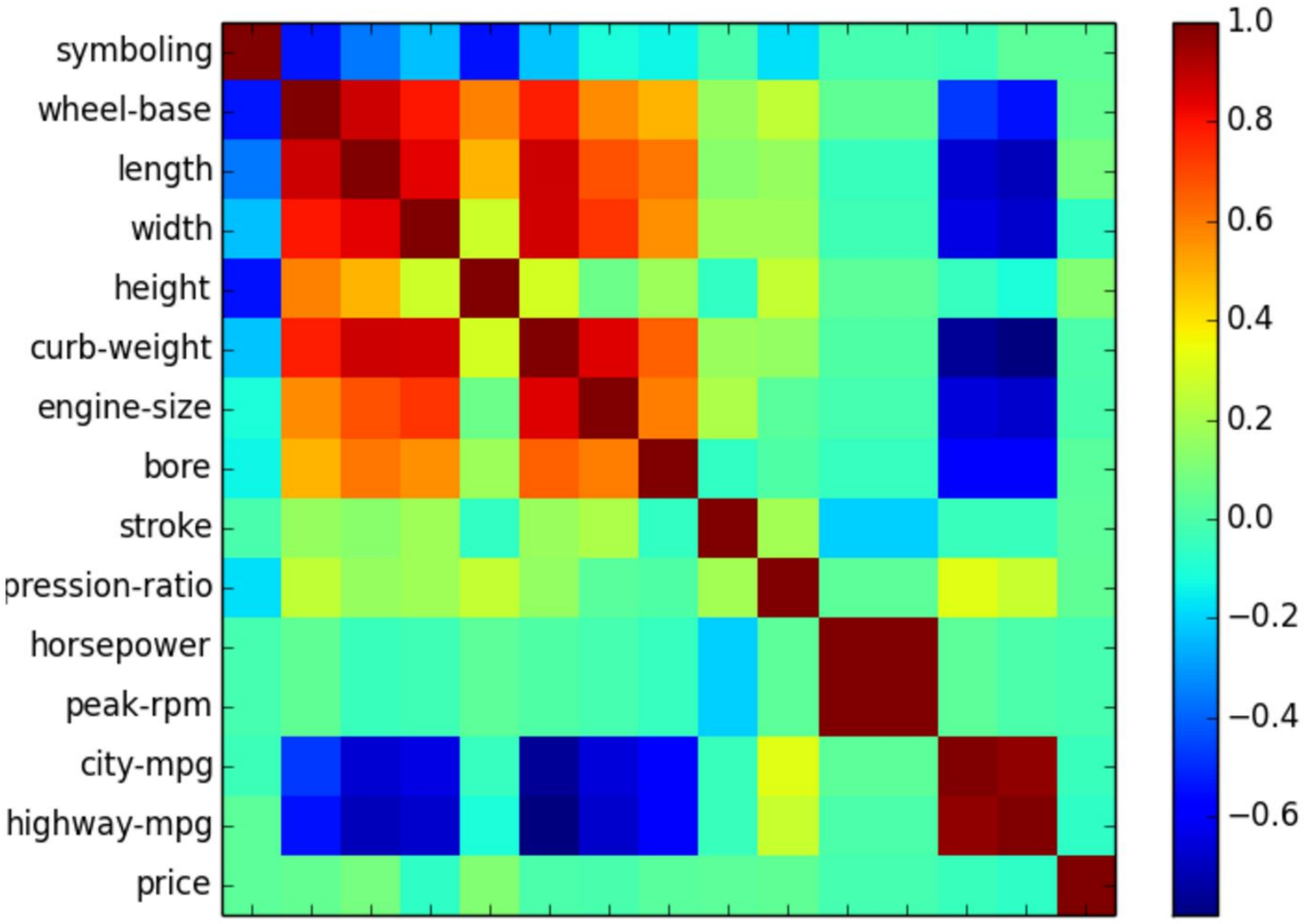
LAB 3

Creating a correlation matrix

Expected lab duration: 15 minutes.

1. Connect the data to Select Columns in Dataset, launch the "column selector" and select all the numeric columns except `normalized-losses`.
2. Connect Execute Python Script to it and paste the python script (in the **next slide**) in the script box. Careful with tabs!
3. Run the experiment and right-click on Execute Python Script, select "Python Device" and then "Visualize".
4. Describe what you see. A high correlation implies a strong **linear** relationship between two variables. What does a low correlation imply?

```
def azureml_main(dataframe1 = None):
    import numpy as np
    import pandas as pd
    import matplotlib
    matplotlib.use("agg")
    import matplotlib.pyplot as plt
    cm=dataframe1.corr()
    fig=plt.figure()
    plt.imshow(cm,interpolation='nearest')
    plt.xticks(list(range(0,len(cm.columns))),
               list(cm.columns.values), rotation=45)
    plt.yticks(list(range(0,len(cm.columns))),
               list(cm.columns.values))
    plt.colorbar()
    fig.savefig("CorrelationPlot.png")
    return pd.DataFrame(cm)
```


symboling
wheel-base
length
width
height
curb-weight
engine-size
bore
stroke
pression-ratio
horsepower
peak-rpm
city-mpg
highway-mpg
price
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

END OF LAB

End of chapter

Chapter 3

Machine learning through basic examples

What we will learn

supervised learning algorithms

- regression and classification
- prediction and scoring
- RMSE, MAE, R and R-squared
- accuracy, precision, recall
- training and test set

unsupervised learning algorithms

- k-means clustering
- principal component analysis (PCA)

What is machine learning?

An algorithm is a self-contained set of **rules** used to *solve problems* through data processing, math, or automated reasoning.

Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed, using data (experience).

The *problems* ML algorithms try to solve are usually (1) prediction and (2) finding structure in data, so the algorithms that do them are called supervised learning and unsupervised learning algorithms respectively.

Two main types of ML problems

supervised learning

- look at some examples (labeled data) and find a way to predict future (unlabeled) examples
- the target variable ("labels") contains the ground truth we want to **predict**
- by comparing predictions with the ground truth, we know how well we're doing

unsupervised learning

- look at unlabeled data and find general patterns
- more subjective and difficult to interpret

Supervised learning

We are trying to **predict** a variable (called labels, target variable, response variable or dependent variable) using other variables (called features, explanatory variables, covariates, attributes or independent variables).

- **regression** algorithms predict a *number* (numeric target)
- **classification** algorithms predict a *category* (categorical target)

Sometimes **regression** refers to a family of ML algorithms. For example, *linear regression* is a regression algorithm but *logistic regression* is a classification algorithm!

Supervised learning algorithms

Common examples of algorithms used include

- **tree-based algorithms** such decision tree, random forest, boosted trees
- **regression models** such as linear regression, logistic regression, lasso regression and elastic net
- **neural networks** including deep learning
- **support vector machines**
- **k-nearest neighbors**

Most of these algorithms can be used for *both* classification and regression, although the implementation is different in each case, and some algorithms are more appropriate than others.

Supervised learning: training

An ML algorithm is sometimes called a **model**. When we build a model on data we say we **train** or **fit** a model.

For example, we say we trained a decision tree on the data, or fitted a decision tree model to the data. The result is called a trained model. A trained model is also often referred to as a **model**, which can be confusing.

Sometimes, people use the word **model** for a trained model to distinguish it from the algorithm itself (which does not depend on data).

Supervised learning: an analogy to training

Think of written language as a framework or a **model** for communication: it is a simplification of thoughts and emotions such that we can convey them to others (in written form).

- A written language can be an model (an abstraction, a framework).
- Written language can have different "flavors" (hieroglyphs, alphabets, emojis...).
- Some flavors are more complex than others (can express more complex thoughts).
- There are other frameworks for language: spoken language, miming, etc.

Supervised learning: scoring and evaluation

Once you have a trained model, you can use it to get predictions on *any* data (as long as it has the features needed by the model to run the predictions). This is called **scoring**.

If the data that you scored also has the target (or labels), we can compare scores (the predictions) to the target (the true values) to see how well the model predicts. This is called evaluating a model.

Regression evaluation metrics

evaluation metric	definition	interpretation
Root Mean Squared Error	$\sqrt{\sum \frac{(observed - predicted)^2}{n}}$	average prediction error
R-squared	R^2 where R is the correlation between observed and predicted	percentage of variation *explained by* the model

Binary classification

Binary (2 categories) classification is the most common kind of classification, because it can be used to answer (predict) **yes/no** or **true/false** questions.

A model makes predictions (positive or negative) which we compare to the actual answers (true or false). When the answers disagree we get a misclassification. This can happen in two ways.

Confusion matrix

	we actually observe positive case	we actually observe negative case
the model predicts a positive case	TP	FP
the model predicts a negative case	FN	TN

Binary classification evaluation metrics

evaluation metric	definition	interpretation
accuracy	$\frac{TP+TN}{TP+FP+FN+TN}$	percentage of correctly classified cases
ROC curve	a visual representation of the model's performance	refer to this great article
AUC	area under the ROC curve	close to 0.5 is bad close to 1 is good

Supervised learning: recap

term	what is needed	results in
training (a model)	appropriate ML algorithm + labeled data	trained model
scoring (data)	trained model + data (labeled or unlabeled)	scores (predictions)
evaluating a model	scoring labeled data	evaluation metrics

More about scoring and model evaluation in the next chapter.

the machine learning community calls it…	the community of statisticians calls it…
learning algorithm (or model)	model
supervised learning algorithm	predictive model
trained model (or just model)	fitted model
supervised learning	prediction problem
unsupervised learning	data-mining or pattern recognition
features or attributes	explanatory or independent variables
target or labels	response or dependent variables
training	fitting
scoring	predicting

Quiz

True or false: In supervised learning, we have a target variable, but in unsupervised learning we only have features.

Answer

True

> The word "supervised" refers to the fact that we know the "ground truth" and therefore the predictions from the predictive model need to get as close as possible to this ground truth. In unsupervised learning there is no ground truth and results are much more subjective and difficult to evaluate.

Quiz

What is needed to train a model? Score data? Evaluate a model?

Answer

> We need labeled data to train a model. We also need to pick an ML algorithm that works for the problem at hand.
> We need a trained model and new data to score. The new data doesn't need to have labels.
> We need score labeled data to evaluate it.

Quiz

True or false: Labeled data is needed to train a (predictive) model, but not to evaluate it.

Answer

False.

> Labeled data is needed to both train a model and evaluate it. This is because when we evaluate a model, we compare the model's predictions to the ground truth.

Quiz

True or false: To score data using a trained model, the data must be labeled.

Answer

False.

> Once we have a trained model we can score data, labeled or unlabeled, to obtain predictions. But if the data is labeled, we can also evaluate the predictions.

LAB 4

Regression

Expected lab duration: 45 minutes.

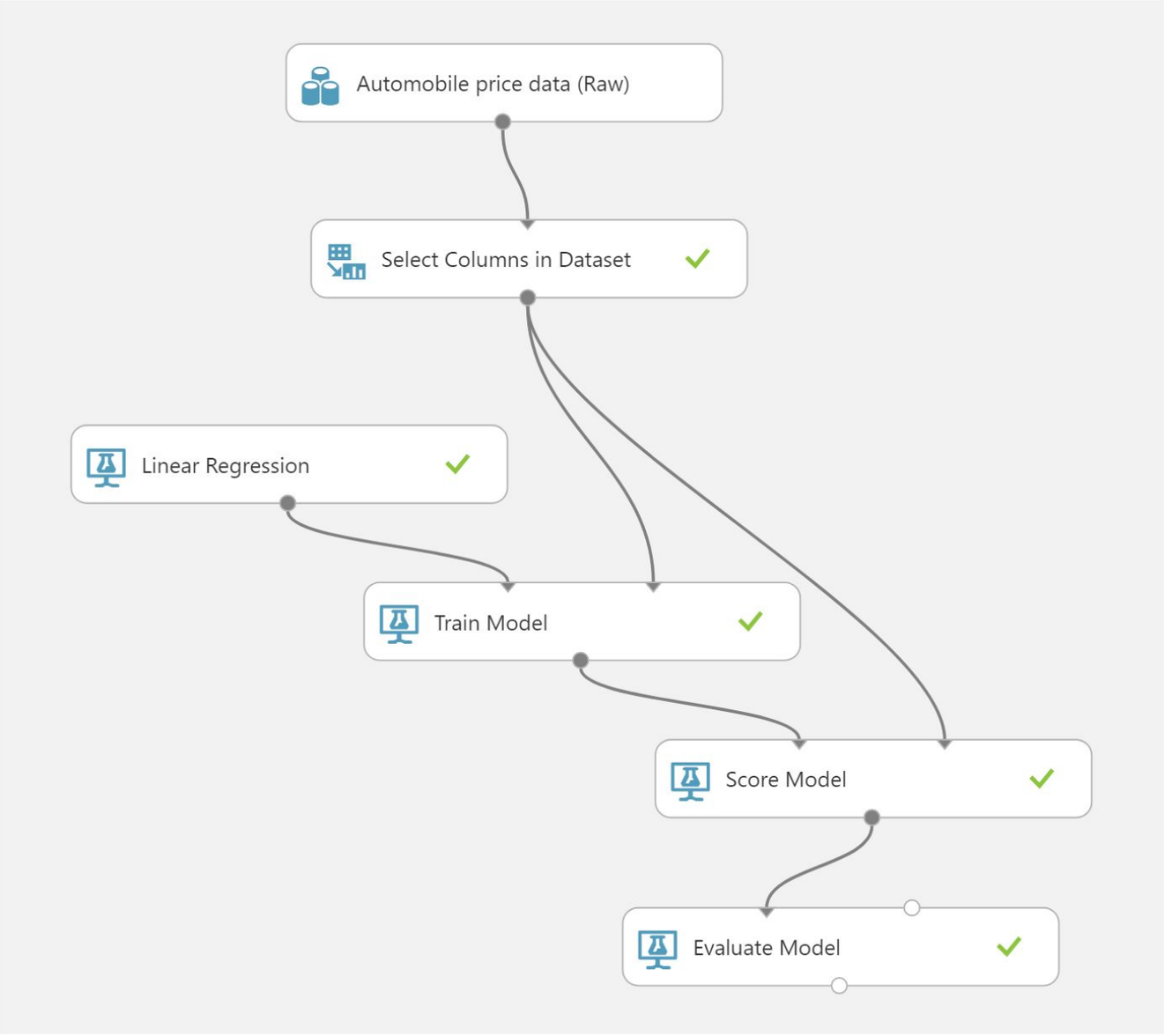
Here is a screenshot of the finished experiment. Let's say you want to predict `price` using `horsepower`, `city-mpg`, and `body-style`.

1. Is this a supervised or unsupervised learning problem?
2. Is it a classification or regression problem?
3. What are the target and features?
4. What are the response and explanatory variables?

You have a few choices of algorithms you can try, and you will try **linear regression** and **decision tree** because they are fundamental building blocks for other more complex algorithms, and because they are easier to interpret which can help you build intuition around how ML algorithms work.

Create a linear regression model for predicting price and run it (a screenshot of the final experiment is shown in the next slide).

1. Connect `Select Columns in Dataset` to the data and select only the columns we want to keep (the target and features).
2. Drag in `Linear Regression` and uncheck the box that says "Allow unknown level in categorical features".
3. Connect it to `Train Model` and select the target.
4. Run the experiment.


Automobile price data (Raw)
Select Columns in Dataset
Linear Regression
Train Model
Score Model
Evaluate Model

5. Right-click on Train Model and choose "Trained model" then "Visualize". The numbers you are looking at translate into the following prediction equation:

```
predicted price = - 364.378
                  + 156.012 * horsepower
                  - 52.7053 * city-mpg
                  - 3510.22 (if body-type is "hatchback")
                  + 2793.26 (if body-type is "convertible")
                  + 1519.70 (if body-type is "hardtop")
                  - 1116.29 (if body-type is "station wagon")
                  - 50.8291 (if body-type is "sedan")
```

6. Take the third row of the data and pass it through the above equation (you can use Excel to calculate predicted price). The difference between the observed price and predicted price for each row is called the **residual**. What is the residual for the third row?

For this model, we could compute the predicted values for every row by simply creating a new column (using `Execute R Script` or `Execute Python Script`) because the calculation is straight-forward.

However imagine a linear model with dozens of features and you can see that hard-coding the above equation can get tedious. For more complex models such as neural networks and *ensemble models* hard-coding it would be a nightmare. Foretunately, we have `Score Model` to run predictions on the data without our having to hard-code it.

7. Connect `Train Model` and `Select Columns in Dataset` to `Score Model` to run predictions on the data.
8. To check your predictions by right-click on `Score Model` and choose "Score dataset" and "Visualize". Check that your prediction for the third row of the data matches what we found in (6).

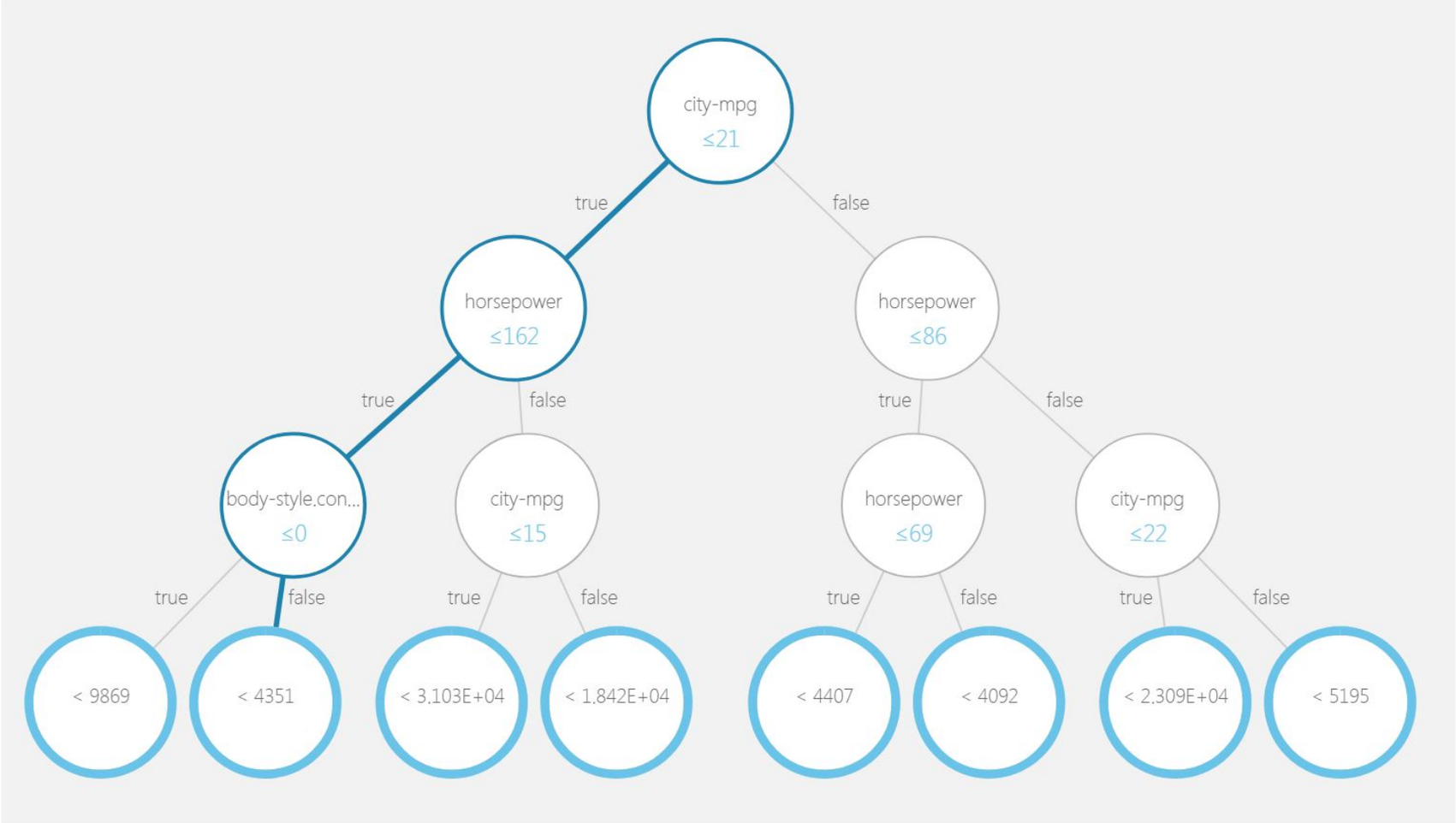
8. Connect `Score Model` to `Evaluate Model` and run it to evaluate the predictions. Note that we can do this here because the data we scores the ordinal data used to train the model (so it is labeled). We evaluate models so we make sure they are a good fit.
9. Right-click on `Evaluate Model` and select "Evaluation results" and "Visualize" to see the evaluation metrics. How is your overall fit for this model?

More on model evaluation in the next chapter.

A **decision tree** is a very different kind of algorithm from linear regression. Linear regression describes the relationship between the features and the target a single equation. A decision tree keeps splitting the data into subsets until within each subset (the leaves of the tree) the target has similar values, and between the subsets the target has different values on average.

Sometimes when we use a decision tree to predict a number, we refer to it as a **regression tree** and reserve the word **decision tree** for use in classification.

10. Change Linear Regression to Decision Forest Regression and set "Number of decision trees" to 1 and "Maximum depth of decision tree" to 4 and uncheck "Allow unknown values for categorical features". Run the whole experiment.
11. Right-click on Train Model and choose "Trained model" and "Visualize".
12. Right-click on Score Model and choose "Scored dataset" and "Visualize". The predictions are shown in the column called `Scored Label Mean`. Why is it common to see the same set of numbers for the predictions?


city-mpg
≤21
true
false
horsepower
≤162
horsepower
≤86
true
false
true
false
body-style.con...
≤0
city-mpg
≤15
horsepower
≤69
city-mpg
≤22
true
false
true
false
true
false
true
false
< 9869
< 4351
< 3.103E+04
< 1.842E+04
< 4407
< 4092
< 2.309E+04
< 5195

END of LAB.

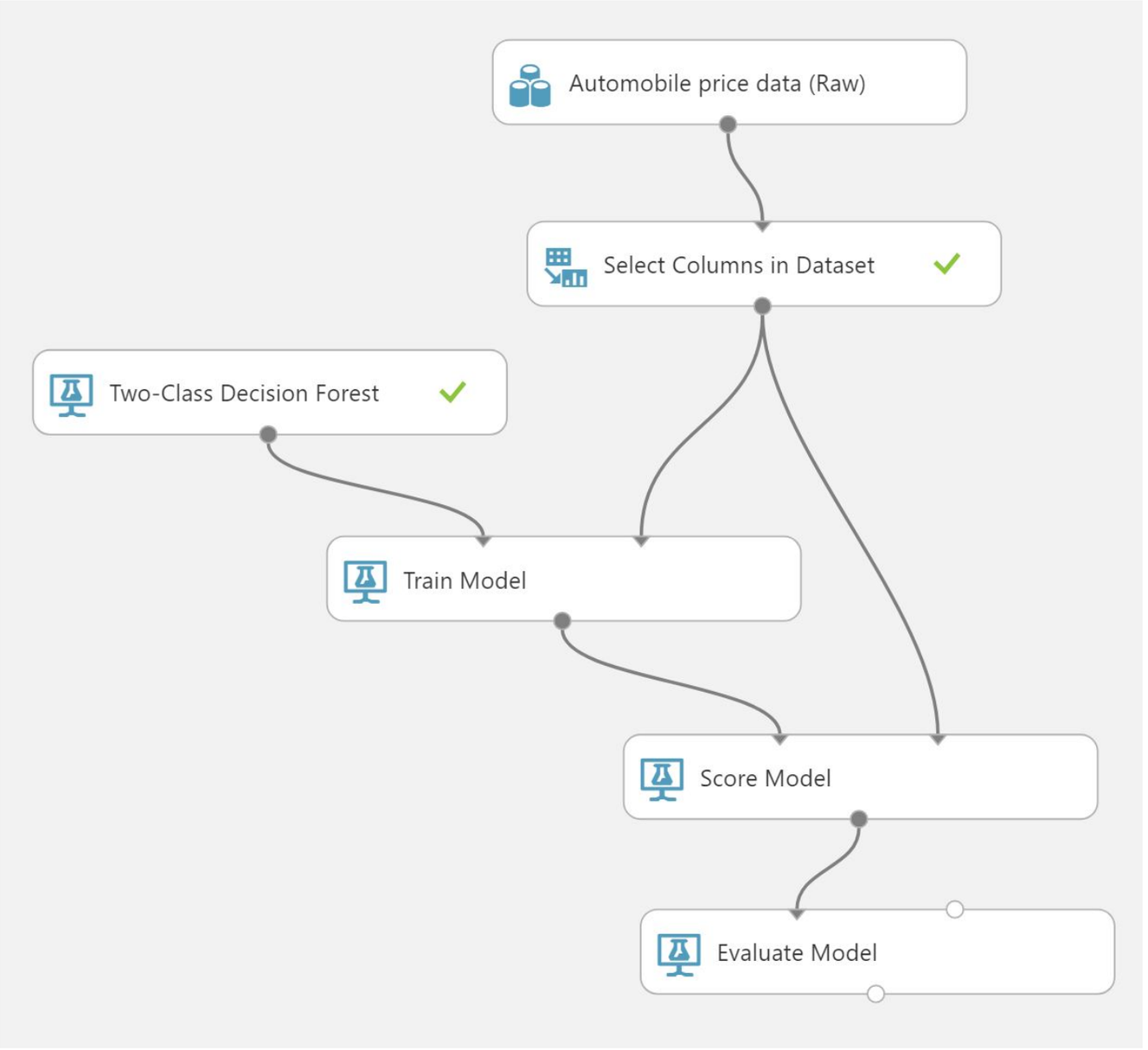
LAB 5

classification

Expected lab duration: 30 minutes.

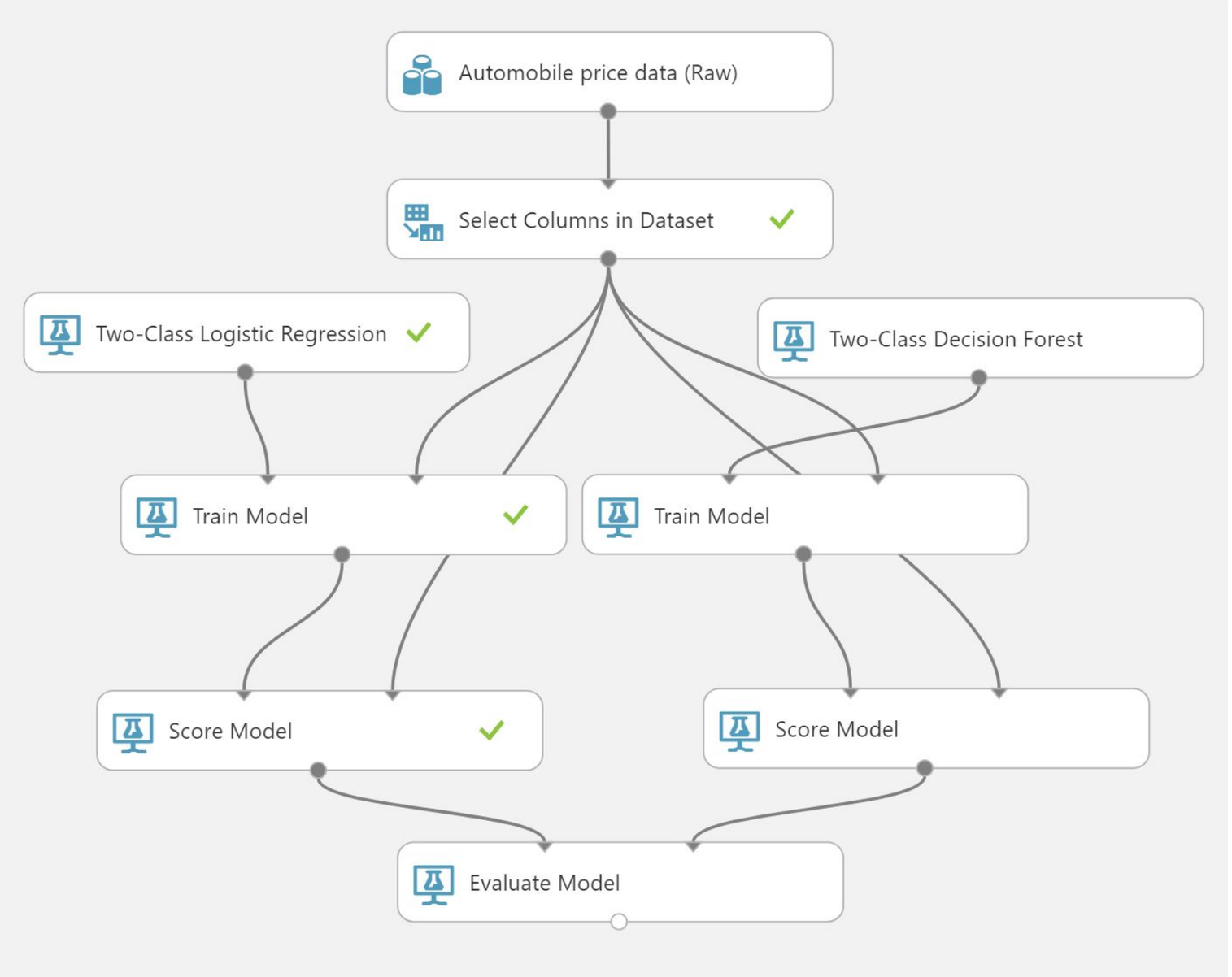
1. Return to the experiment from the previous lab and click on Select Columns in Dataset and add the column `num-of-doors` to the columns selected.
2. Click on Train Model and change the target variable to `num-of-doors`.
3. Replace the algorithm with a Two-Class Decision Forest and set "Number of decision trees" to 1.
4. Re-run the experiment.

A screenshot of the experiment is shown in the here.


Automobile price data (Raw)
Select Columns in Dataset
Two-Class Decision Forest
Train Model
Score Model
Evaluate Model

5. Right-click on Score Model and choose "Scored dataset" and "Visualize".
6. The predictions are shown in a column called `Scored Labels`. Each prediction is associated with a probabilty in the column `Scored Probabilties`. Find a row where a misclassification occured and see if you can guess why.
7. Right-click on Evaluate Model and choose "Evaluation results" and "Visualize". Report the accuracy and the AUC for this model.
8. What do the false negatives and false positives represent?

9. Replace `Two-Class Decision Forest` with `Two-Class Logistic Regression` and re-run the experiment.
10. Is the new model better or worse than the old one? Justify your answer.
11. We can compare two models side by side in Azure ML. To do so, we copy and paste `Train Model` and `Score Model` and connect `Two-Class Decision Forest` to `Train Model`. See the screenshot here.
12. Re-run the experiment and examine the side-by-side evaluation.


Automobile price data (Raw)
Select Columns in Dataset
Two-Class Logistic Regression
Two-Class Decision Forest
Train Model
Train Model
Score Model
Score Model
Evaluate Model

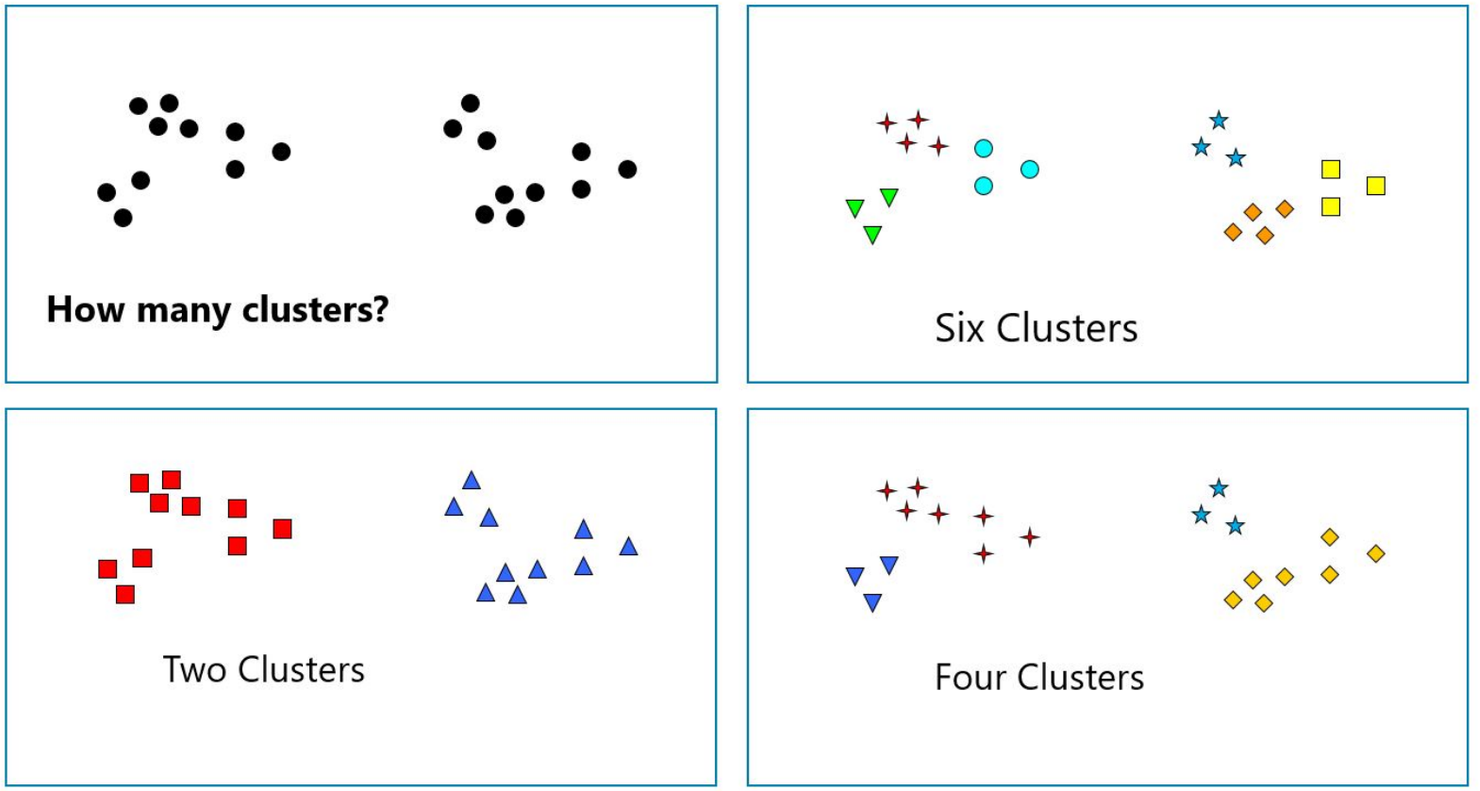
END OF LAB

Unsupervised learning

Unsupervised learning is about *finding structure* (natural groupings) in the data. There is no target, only features.

- The k**-means** algorithm attempts to find clusters in the observations (rows), such that two observations in the same cluster have similar features.
- **Principal component analysis** attempts to find clusters in the variables (columns) such that two variables within the same cluster contain similar (redundant) information.

Choosing the right k



k-means clustering

- The k-**means clustering** is an algorithm that attempts to find grouping in the rows of the data.
- It finds similar data points (observations) when we compare their features.
- So k-means clustering finds **redundancy** in the data across rows.
- We choose k (the number of clusters) and k-means gives us a new column showing the cluster assignments for each row.

Quiz

In k-means clustering, what are the advantages and disadvantages of choosing a higher k?

Answer

> A higher k means we have more clusters. The advantage is that we can capture more complex patterns in the data (in this case think of it as more *niche* clusters", if this is desirable. The disadvantage is that having more clusters means we have to do more work to understand and explain differences between them. A much larger k also means longer run-time for the algorithm.

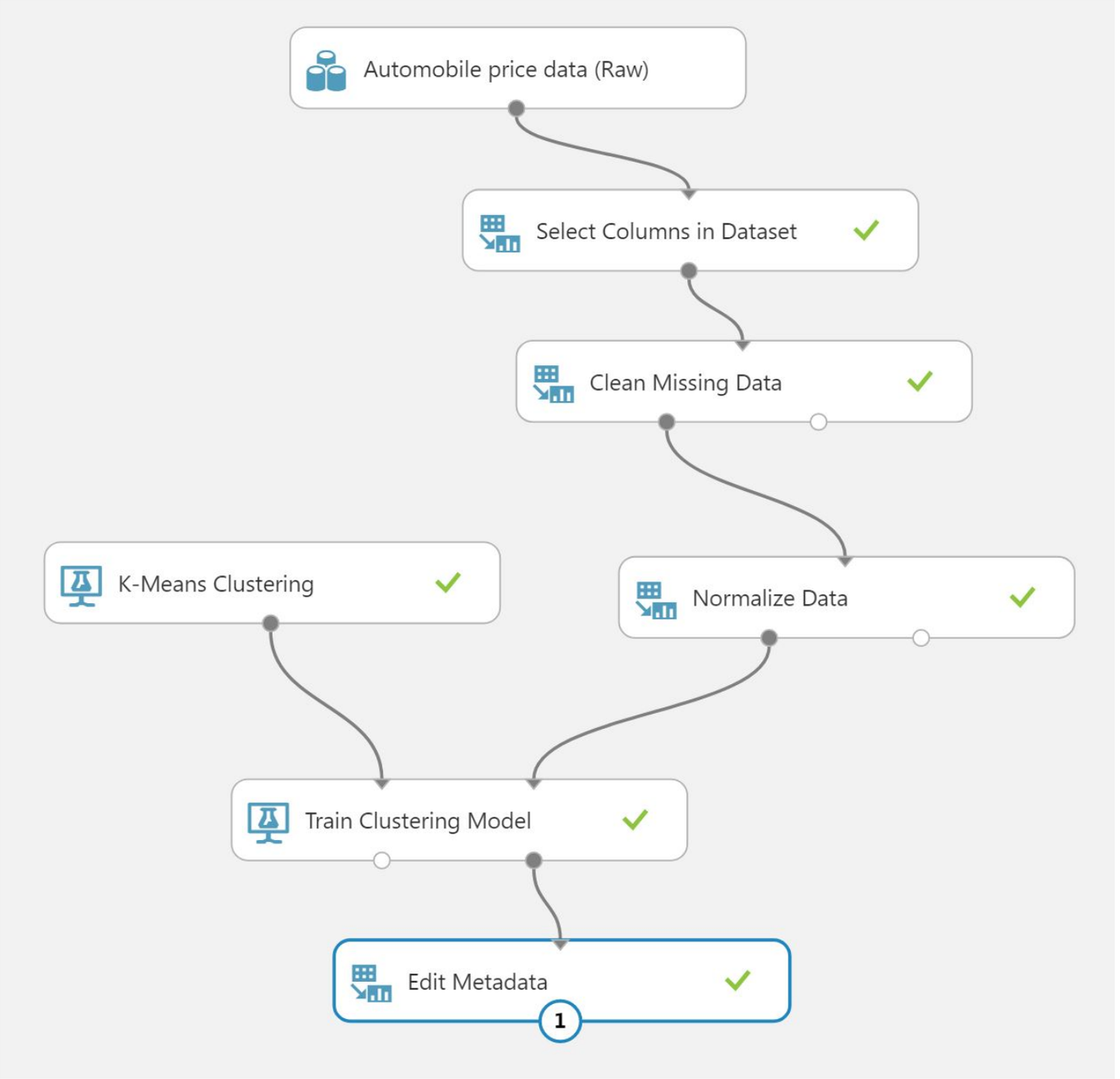
LAB 6

k-means clustering

Expected lab duration: 30 minutes.

We are going to run the k-means clustering algorithm to see how it can divide the cars into groups. A screenshot of the final experiment is shown here.

1. Create a new experiment with Automobile price data (Raw) and use Select Columns in Dataset to drop the columns `symboling`, `normalized-losses`, `make`, `fuel-type`, `engine-location`, `bore`, `stroke` and `compression-ratio`.
2. Use Clean Missing Data to replace any missing data from numeric columns with the average of the non-missing data for that column.
3. Drag in Normalize Data and use a MinMax transformation and apply it to all the numeric columns.


Automobile price data (Raw)
Select Columns in Dataset
Clean Missing Data
K-Means Clustering
Normalize Data
Train Clustering Model
Edit Metadata
1

Normalizing a variable means that we change its scale by applying a mathematical transformation to it. The **MinMax** transformation replaces the variable X with the following rescaled version of X:

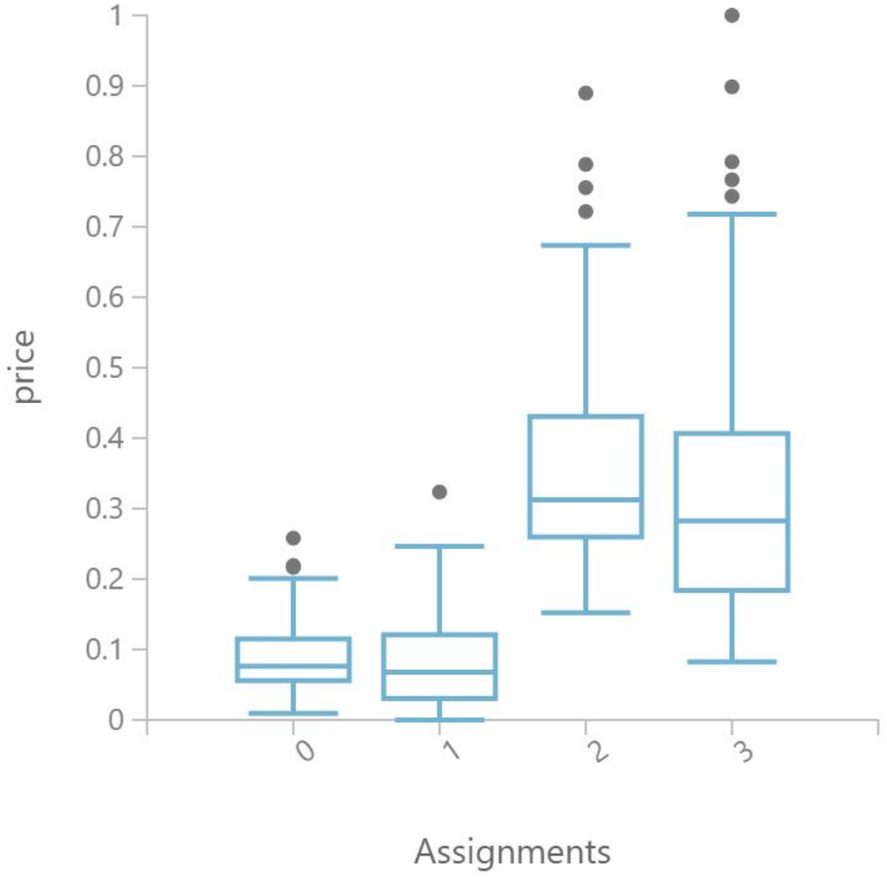
$$\frac{X - min(X)}{max(X) - min(X)}$$

This puts the variable X on a scale of 0 to 1. Some ML algorithms require us to normalize variables before passing them to the algorithm otherwise variables on a larger scale skews the results. Some algorithms have normalization built into them, so we don't have to use `Normalize Data`.

Normalizing is also sometimes called standardizing or rescaling.

4. Drag in `K-Means Clustering` and set "Number of controids" to 4.
5. Drag in `Train Clustering Model` and select all the columns in the data.
6. Drag in `Edit Metadata` so we can look at the data with a column containing the cluster assignments.
7. Run the experiment and right-click on `Edit Metadata` and select "Results dataset" and "Visualize". Scroll to the right to find the column with the cluster assignment.
8. Right-click on `Edit Metadata` and select "Results dataset" and "Save as Dataset". Call it `Automobile data (with clusters)`. We will return to this data in the next lab.

9. Now begins the hard part of understanding what makes the clusters different from one another. Click on the `Assignment` column to see a bar plot of the clusters. The numbers 0-3 (one for each cluster) are only nominal. Use the "compare to" drop-down list to create two-dimensional plots that compare the distribution a given column across the 4 clusters. Two such examples are shown in the next slide, one for `price` (numeric) and one for `fuel-type` (categorical). Compare the distribution of at least 3 variable across the 4 clusters and based on that put together a "profile" for each cluster.


price
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
1
2
3
Assignments


body-style
hardtop
wagon
sedan
hatchback
converti...
1
7
13
12
53
37
6
1
52
17
1
5
0
1
2
3

END OF LAB

Principal component analysis

- **Principal pomponent analysis** attempts to find groupings of the features.
- It finds features that are similar (relay similar information because they are highly correlated) and combines them into one feature called a principal component.
- PCA finds redundancy in the data across columns and each principal component (PC) tries to capture left-over variation that was not captured by PCs that came before it.
- PCA is an example of a dimensionality reduction: Usually, the first few principal components captures most of the variation in the data. So we replace our original p features with the first m principal components, where m should be much smaller than p.

LAB 7

Principal component analysis

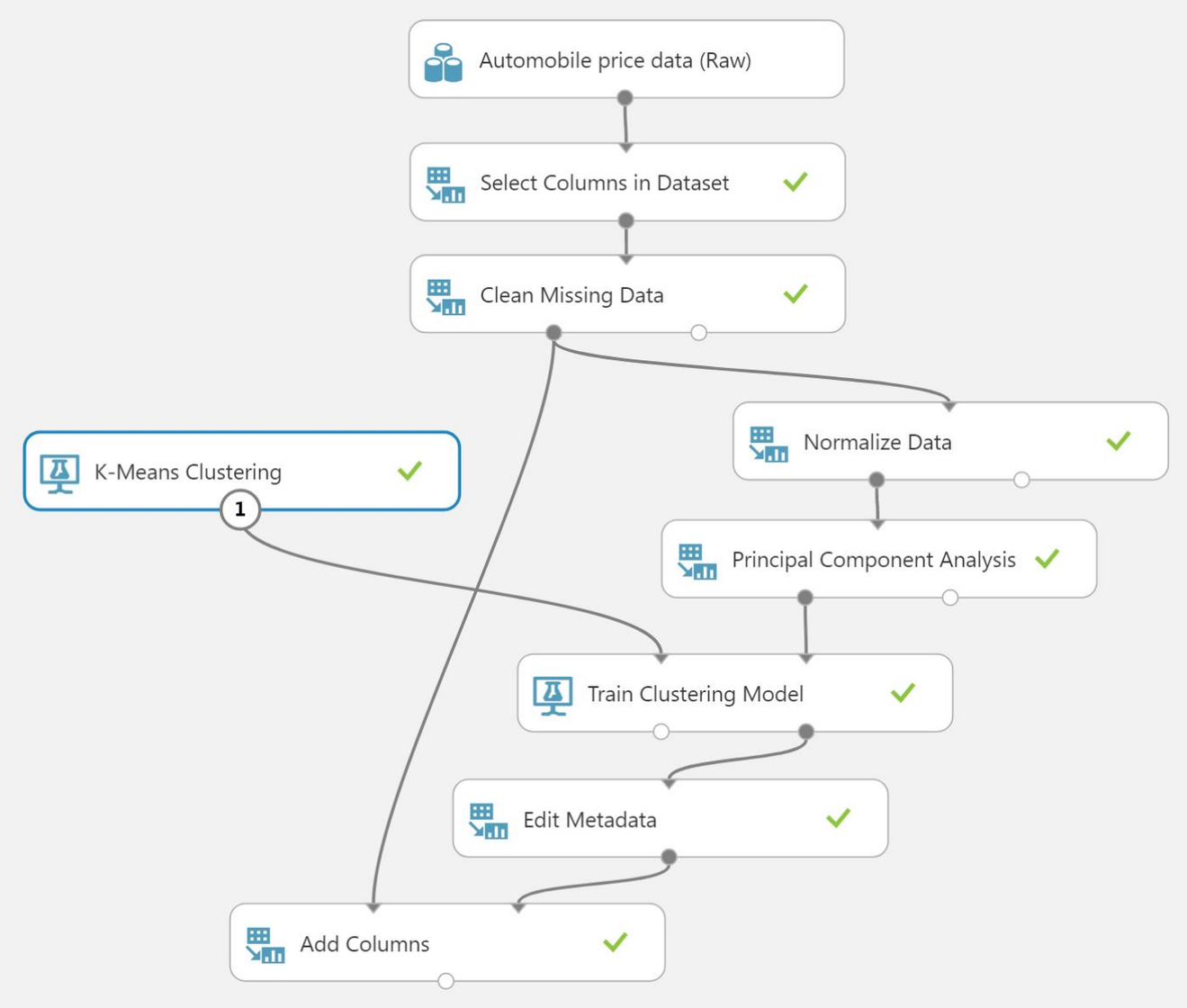
Expected lab duration: 30 minutes.

The k-means algorithm (and other algorithms that rely on "locality") suffer from two problems:

- Numeric variables with larger scale can influence it more. As we saw, the remedy was to rescale the variables.
- When there are lots of features, it becomes harder to find "similar" points, because even small differences are magnified over many dimensions. This is referred to as the curse of dimensionality. One remedy for this is to use principal component analysis to reduce the dimensionality of the data prior to clustering.

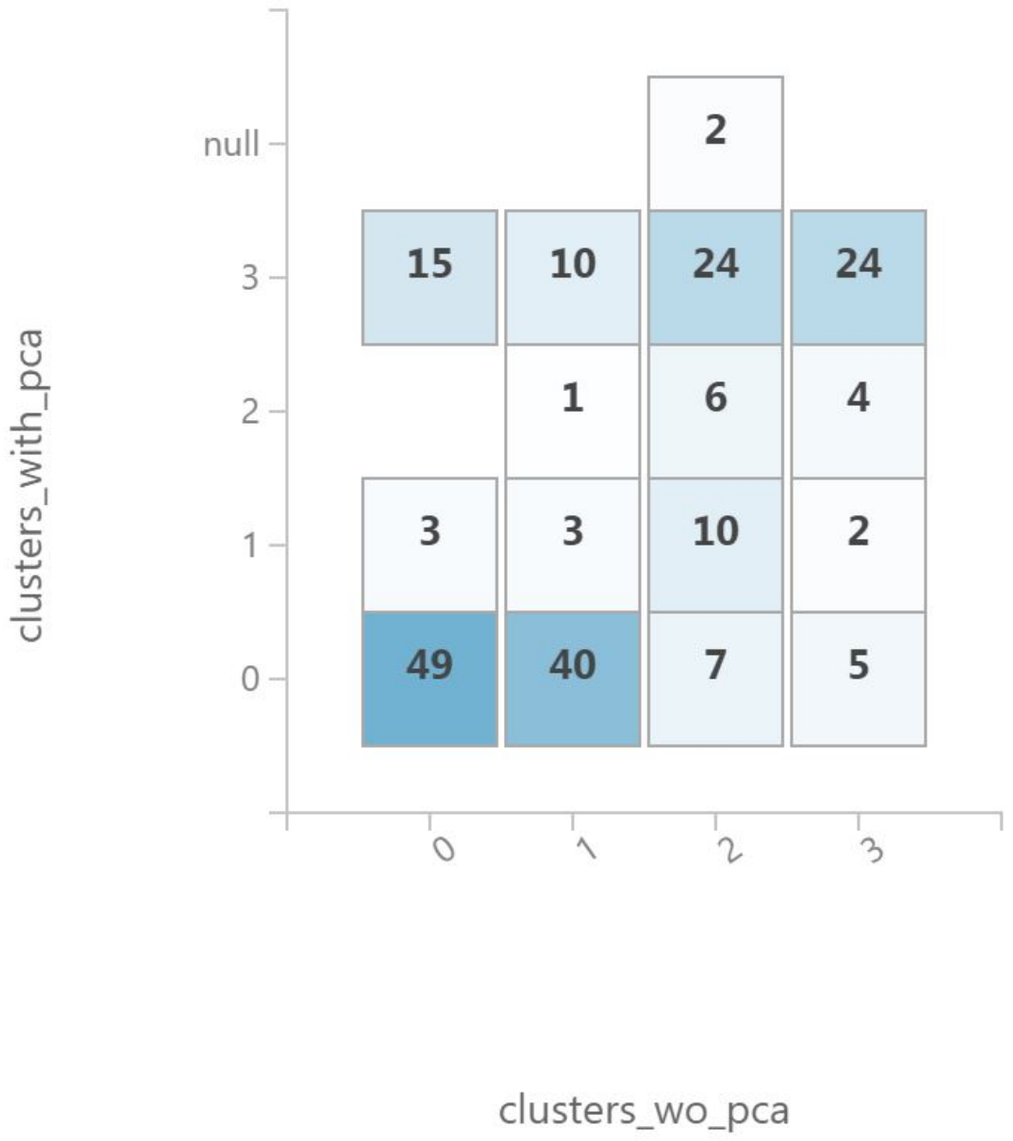
A screenshot of this lab can be seen here. Return to the clustering experiment from the last lab, but make the following changes:

1. Use Select Columns in Dataset to keep all numeric columns except `normalized-losses`.
2. Use Clean Missing Data to replace any missing data from numeric columns with the average of the non-missing data for that column.
3. Use Normalize Data and use a Z-Score transformation and apply it to all the numeric columns. A Z-Score transformation replaces the variable X with $\frac{X-mean(X)}{sd(X)}$.
4. Drag in Principal Component Analysis on all the columns and choose 5 for "Number of dimensions to reduce to" and uncheck the "Normalize dataset" box. Run the experiment.


Automobile price data (Raw)
Select Columns in Dataset
Clean Missing Data
Normalize Data
K-Means Clustering
1
Principal Component Analysis
Train Clustering Model
Edit Metadata
Add Columns

5. Right-click on Edit Metadata and choose "Results dataset" and "Visualize". Do you see your principal components? Which principal component does the best job of discriminating between the clusters?
6. Right-click on Edit Metadata and choose "Results dataset" and "Save as Dataset" and call it `Automobile data (with clusters using PCA)`.
7. Start a new experiment and drag in `Automobile data (with clusters)` and `Automobile data (with clusters using PCA)` and connect both to Add Columns so we combine them column-wise into one data.
8. Drag in Select Columns in Dataset and choose only the original columns in the data and the cluster assignments. The columns end up in the order you select them.

9. Right-click on Add Columns and choose "Results dataset" and "Visualize".
10. Use Edit Metadata to change the variable names for the cluster assignments to `clusters_wo_pca` and `clusters_with_pca` (Careful which is which!).
11. Create a two-way table comparing cluster assignments with and without PCA. What has changed?
12. Pick one cluster try examine its features to see what old "profile" with the new "profile". Did we do a better job of separating profiles with the new cluster assignments using PCA? Answering this question is not easy and takes a lot of "slicing and dicing".


clusters_with_pca
null
3
2
1
0
2
15
10
24
24
1
6
4
3
3
10
2
49
40
7
5
0
1
2
3
clusters_wo_pca

END OF LAB

k-means	principal component analysis
finds redundancy across rows	finds redundancy across columns
must choose k before	must choose m (the top PCs)
groups observations into distinct clusters	combines features into orthogonal PCs
create a column of cluster assignments	creates one column per principal components
each observation falls into a unique cluster	each PC is a weighted average of all features
clusters can still share similarities	PCs are orthogonal to each other
"profiling" cluster centroids can be hard	interpreting PCs is hard

Some additional observations

Because unsupervised learning deals with unlabeled data, we can still **"train"** a **"model"** on data and **"score"** data using a trained model (quotes are used to emphasize that the terms are used more loosely), but what makes it unsupervised is that evaluating a model is very difficult and subjective because no true labels are present.

It is best to avoid terms like training and scoring with unsupervised learning algorithms.

Although clustering and PCA are unsupervised learning algorithms, machine learning workflows often involve combining supervised and unsupervised learning algorithm. Examples:

- run k-means to build clusters and use clusters as one of the features in a predictive model
- run PCA and use the top few principal components as features in a linear regression model: this is called principal component regression
- run PCA on a dataset with lots of features and use only the top m to run k-means on the data: this is useful because k-means does not perform well on high-dimensional data.

Let's recap

supervised learning algorithms

- regression and classification
- prediction and scoring
- RMSE, MAE, R and R-squared
- accuracy, precision, recall
- training and test set

unsupervised learning algorithms

- k-means clustering
- principal component analysis (PCA)

End of chapter

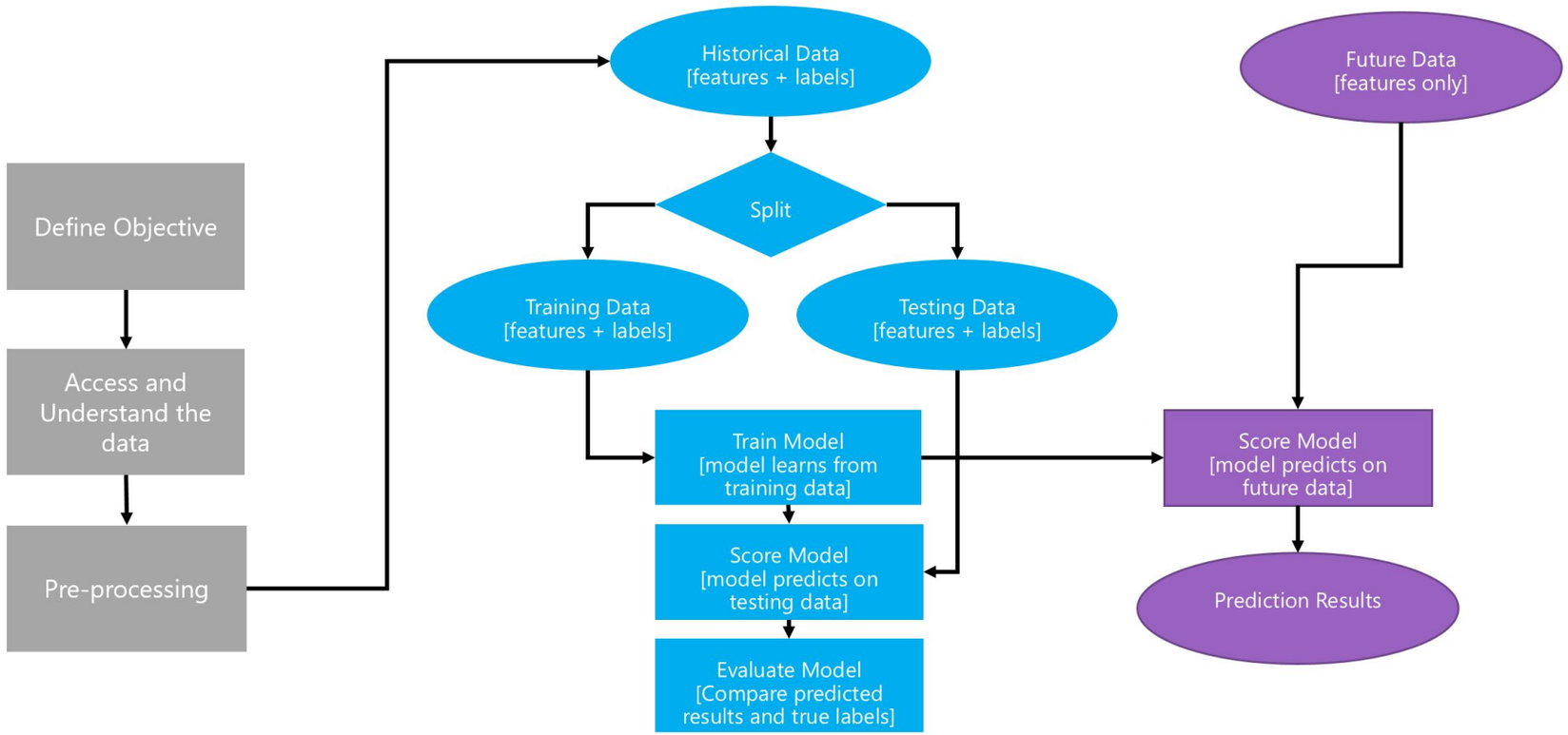
Chapter 4

Concepts in Machine Learning

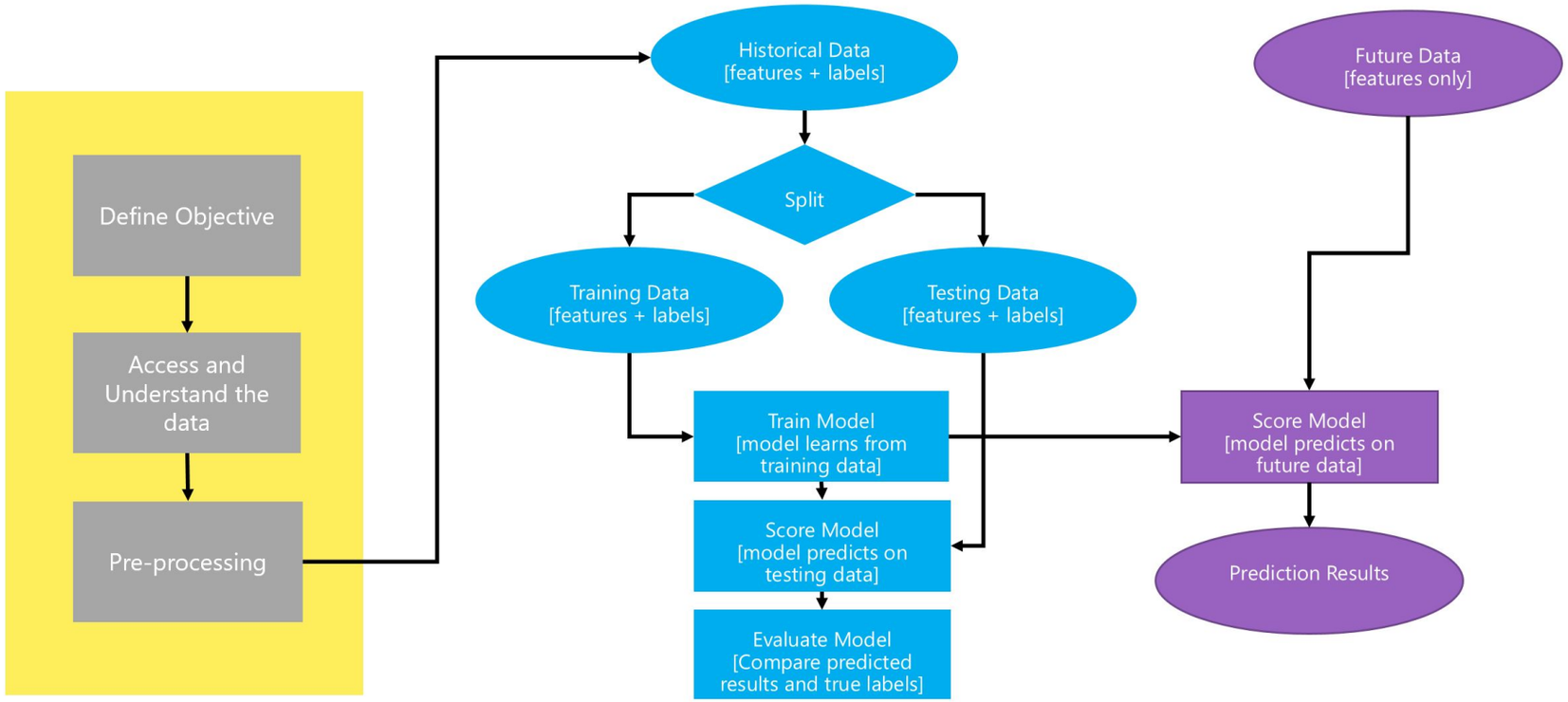
What we will learn

- Fitting, training, modelling
- Predicting, scoring
- Pre-processing:
  - Aggregating, cleaning, reshaping, feature extraction.
  - Handling missing values and outliers
  - Feature creation
- Model complexity, overfitting and underfitting
- Model interpretability vs prediction accuracy

The Machine Learning Process



Part 1: Pre-processing data


Define Objective
Access and Understand the data
Pre-processing
Historical Data [features + labels]
Split
Training Data [features + labels]
Testing Data [features + labels]
Train Model [model learns from training data]
Score Model [model predicts on testing data]
Evaluate Model [Compare predicted results and true labels]
Future Data [features only]
Score Model [model predicts on future data]
Prediction Results

Once we have a business questions in mind we need to get the data ready. Data in its raw form is almost never ready for modeling, so we begin by pre-processing data. Pre-processing data is everything we do to the raw data before we can feed it to one of the machine learning algorithms. Common pre-processing tasks involve:

- Aggregate, sample data
- Clean data by treating missing values and/or outliers
- Feature extraction and transformation
- feature engineering

Aggregate and/or sample

- Raw data is sometimes too granular for modeling, or biased toward more frequent observations.
- The granularity of the data affects the interpretation of our model and we can aggregate data to make it less granular.
- Aggregating data can also dampen the number of missing values and the effect of outliers.
- Sampling data can remove bias posed by more frequent observations in the raw data.

Treat missing values and outliers

- Some models are more "sensitive" to missing values and outliers than others.
- There are many ways to impute missing values, some get very fancy.
- In some cases, outliers are exactly what we're interested in, such as anomaly detection. In other cases, outliers can get in the way of building good models.
- What exactly should be an outlier is somewhat subjective.

Feature extraction and transformation

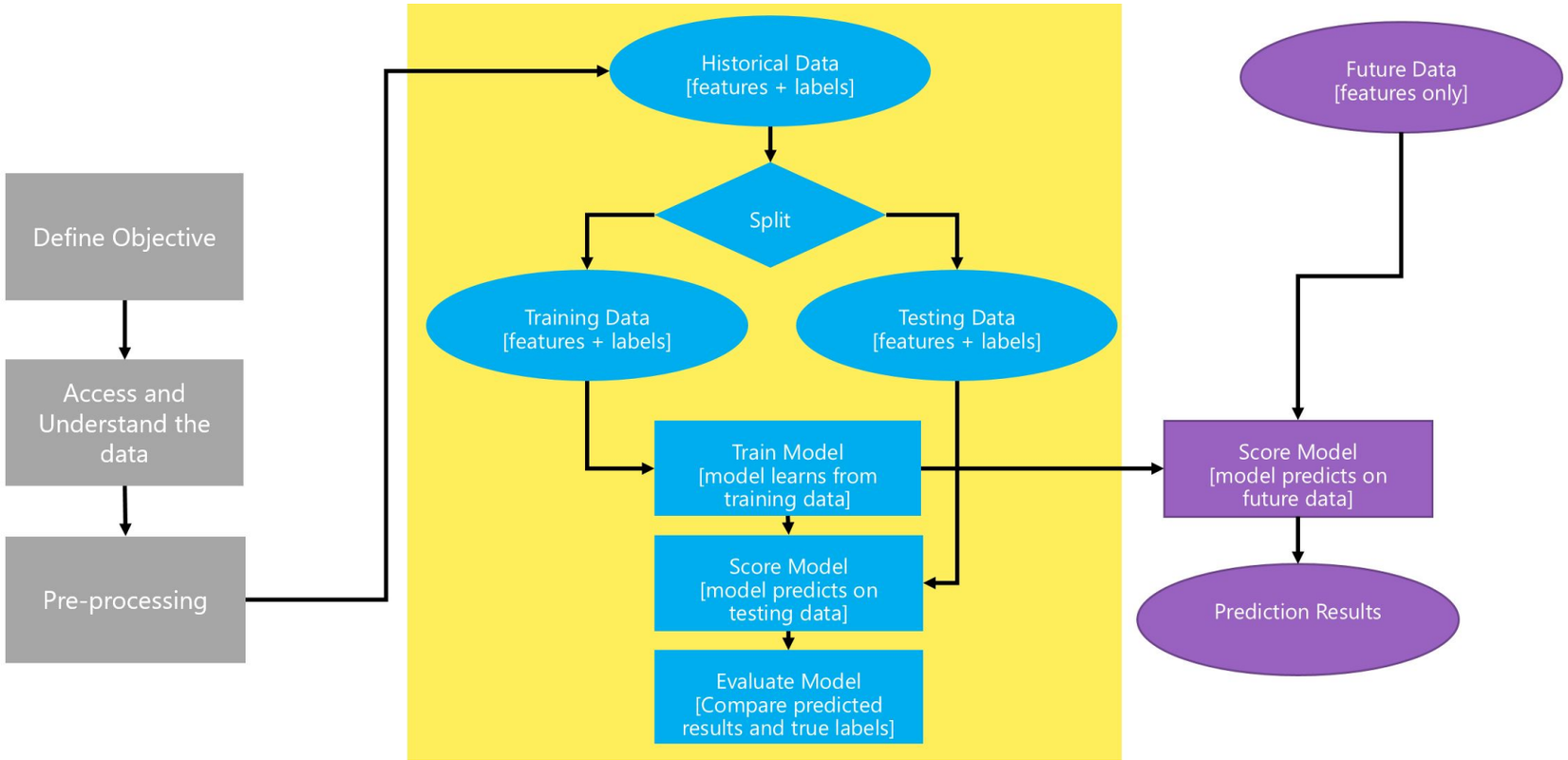
Depending on the model, transforming our features can make them easier to interpret or more accurate. Here are some common transformations we might consider:

- **Normalization or standardization** has to do with rescaling variables so they are more or less on the same scale.
- **Binning, bucketing, or discretizing** turns a variable with high-cardinality into one with low cardinality. It can be applied to both numeric and categorical features.
- A new feature can be a clever combination of existing features which can reduce redundancy in the data or bring out some of its more hidden aspects (feature engineering can do the same).

Feature engineering

- Feature engineering is the idea is that we can automatically extract new features from the raw data that give us more accurate models than the original features.
- But it can also can make it hard to interpret our models.
- Some models such as neural networks have feature engineering built into the algorithm. Others may require more trial-and-error (such as using principal components in regression).

Part 2: Model building and evaluation


Define Objective
Access and Understand the data
Pre-processing
Historical Data [features + labels]
Split
Training Data [features + labels]
Testing Data [features + labels]
Train Model [model learns from training data]
Score Model [model predicts on testing data]
Evaluate Model [Compare predicted results and true labels]
Future Data [features only]
Score Model [model predicts on future data]
Prediction Results

An analogy for training and testing

- You are going to take an exam (call it future exam) in a subject you know almost nothing about (like a driver's test)
- You have a practice exam sheet with lots of questions
- You can use it to study for the future exam
- How do you estimate (predict) your score on the future exam?

Answer

You can't use the training portion to test yourself because you could then "study" by just memorizing the answers.

To remedy this, we use the holdout method: *randomly* divide the practice exam into two parts

- Call the bigger part the training portion and use it to study
- Call the smaller part the testing portion and use it to test yourself by pretending it's the future exam

If you learned well from the training portion, your score on the future exam should be only slightly worse than your score on the testing portion.

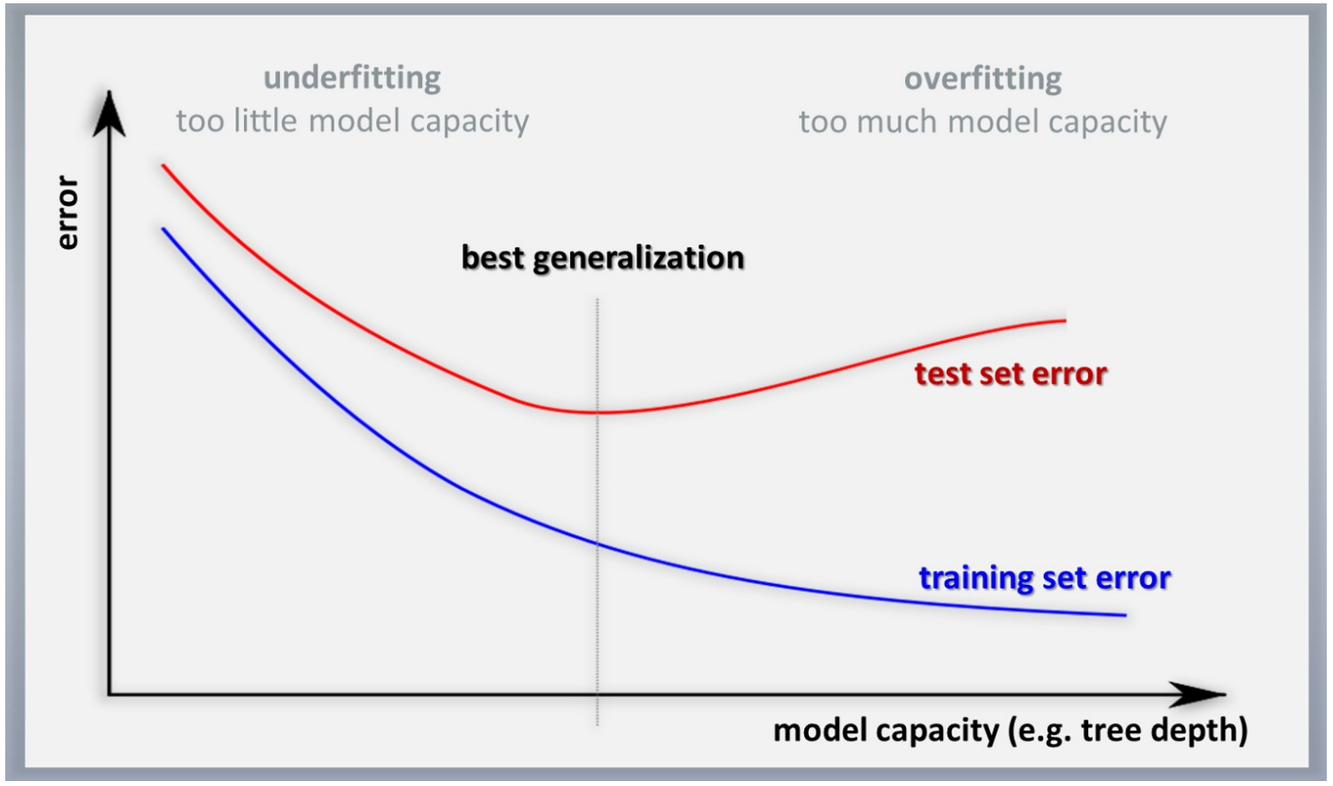
Overfitting

- A similar situation to "memorizing the answers" can happen with models.
- A good model should capture meaningful trends (signals) in the training data, meaning trends that generalize to data outside the training data (out-of-sample as statisticians call it).
- An apparent trend that fails to generalize to out-of-sample data (noise) should be ignored by the model. Otherwise the model is said to overfit.
- Without a test data, we can't distinguish the signal from the noise.
- The more "complex" models are more likely to overfit.

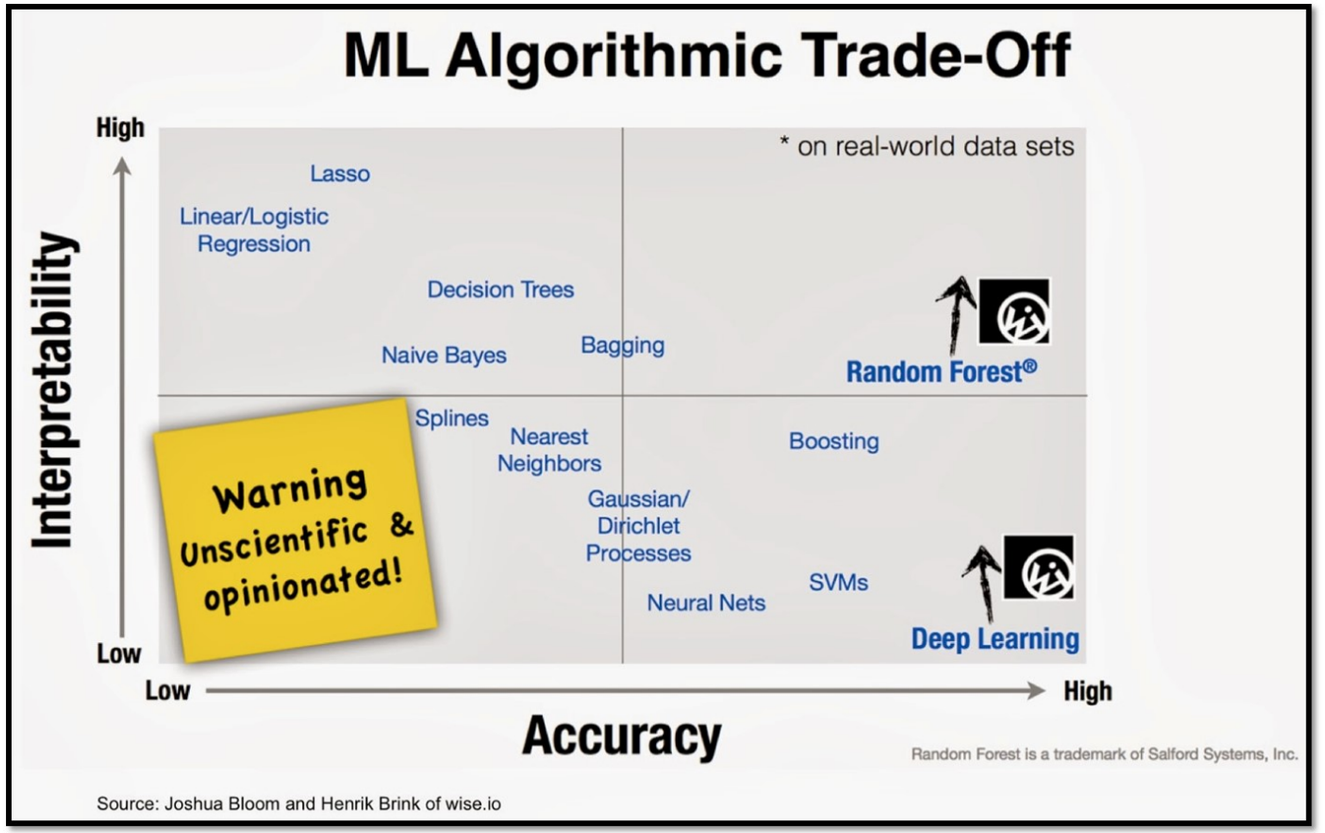
Model complexity

- The same model can be made more complex by including more features or by different choices for some of its hyperparameters (such asa tree's depth). Hyperparameters are inputs to the model that must be specified but can not be directly *learned*.
- Some models are just more complex than others (such as a random forest versus a decision tree, neural networks versus linear models, ...). More complex models are said to be high-variance, and more simple models are said to be high-bias.

Complexity for the same model



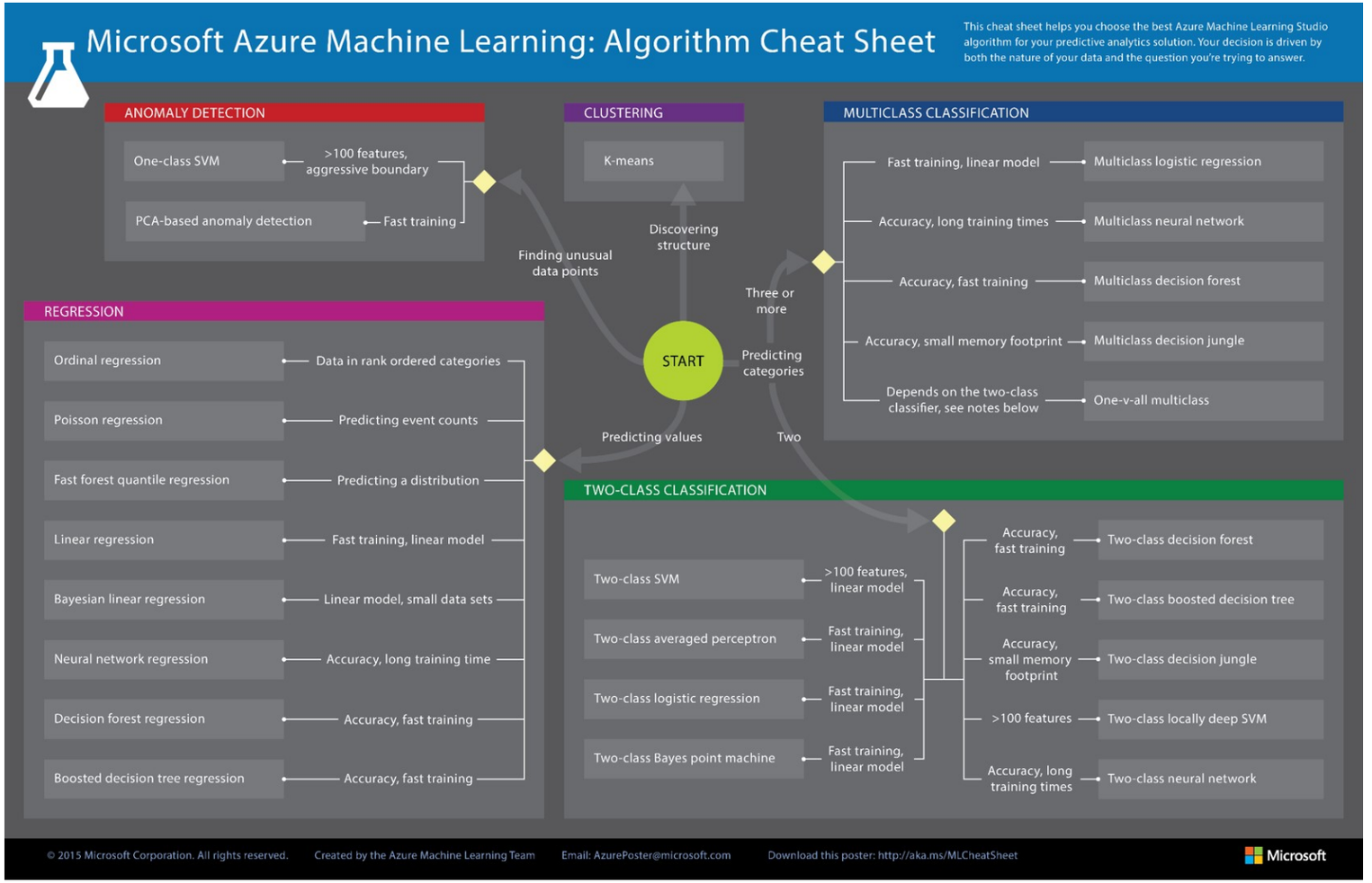
Complexity for different models



don't do this	why?	do this instead
simply throw all our variables at the model	multi-collinearity (feature redundancy)	variable selection or feature engineering
simply pick the most accurate model	more likely to overfit, harder to interpret, maybe less efficient	decide on the right trade-off between accuracy and complexity

"Models should be simple, but not simplistic."

Click on the image to download.



"Machine learning is part art part science."

Resources:

You can learn more here about how to choose a machine learning algorithm:

https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice#considerations-when-choosing-an-algorithm

Quiz

True or false: When building models, there is no point extracting less granular features from more granular features (for example, month from a datetime column). More information is always better.

Answer

False.

> More granularity is only good as far as it contributes more to the "signal" and less to the "noise". So while it's not easy to give a straight-forward answer (because it depends on the use-case) it is false to say that more granular features are always better.

Quiz

- What are the benefits of more complex models?
- What are the benefits of simpler models?

Answer

> More complex models can have higher predictive accuracy than more simple models, but this usually comes at the cost of longer training time and scoring time. More complex models are also need more data, otherwise predictions can have high variance. Finally, more complex models make it harder to explain how the predictions came about. Simple models require less data, are more explainable, usually train faster and they can be even easier to deploy.

Quiz

True or false: overfitting affects more simple models.

Answer

False.

> Complex models are more prone to overfitting than more simple models. We can offset that effect by using more data, but we may need *a lot* more data.

Quiz

A model shows very high error rate on both training and test data. What should be the next steps?

Answer

> If the error is high both on the training and the test set, then the model is failing to capture significant relationships. In this case we need to try a more complex model. We can also try to find more informative features.

Quiz

Why does having more features not necessarily result in better models?

Answer

> Throwing more features at an algorithm is usually only helpful if (1) the feature is a useful predictor for what we're trying to model and (2) the feature contains new information that other features so far don't already capture (otherwise, we will have too many correlated features which can increase the variance of our predictions).

LAB 8

Revisiting model evaluation

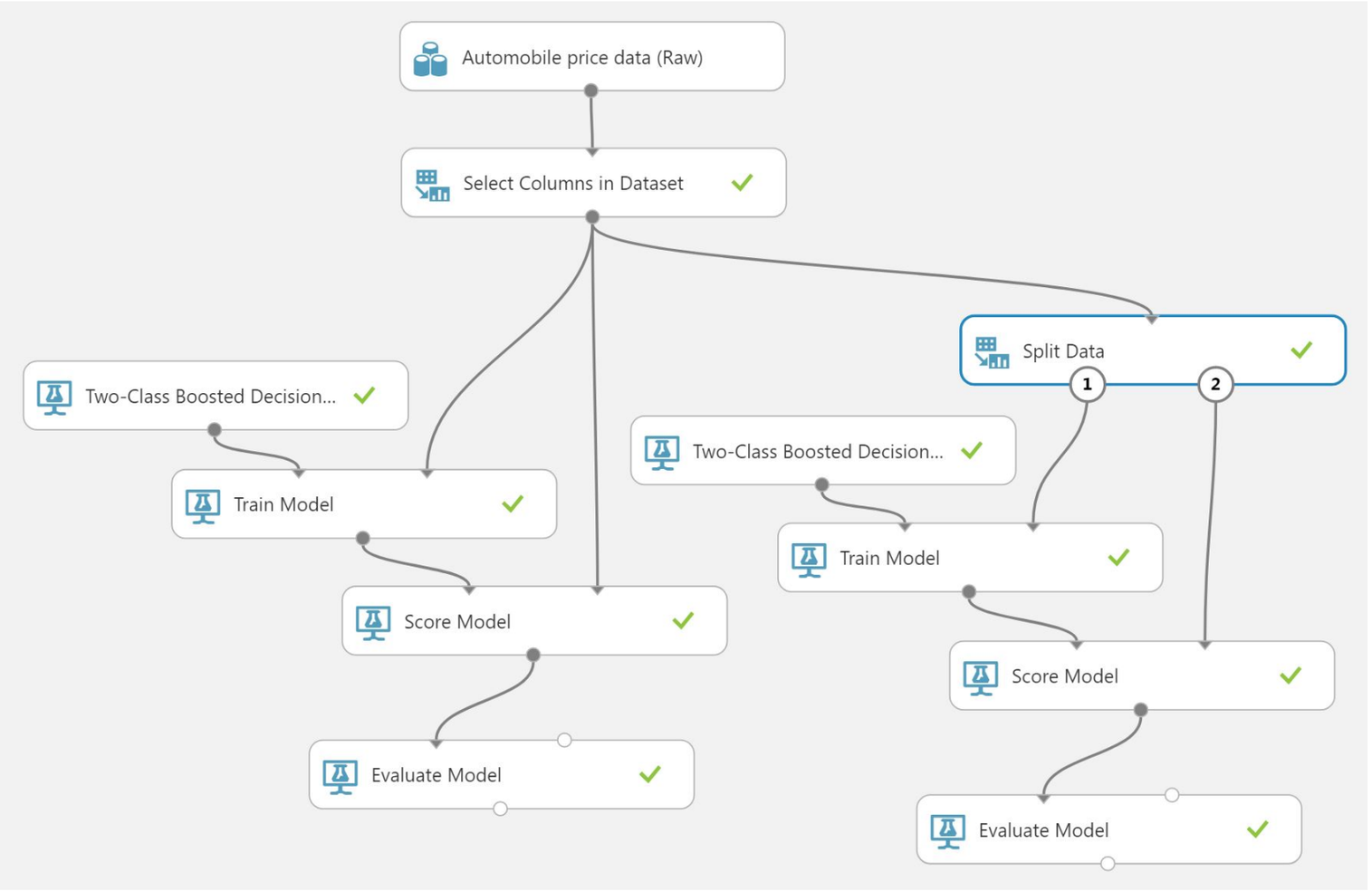
Expected lab duration: 30 minutes.

In this exercise, we run an experiment similar to the ones from the last chapter, but using a more complex model. We then add a new module that splits the data into training and test set. We examine the effect of splitting the data on the model's evaluation metrics. Finally, we repeat all of this for a simpler model to see if we can see the same effect. A screenshot of this lab can be seen here.

1. Start a new experiment and drag in the Automobile price data (Raw).
2. Drag in Select Columns in Dataset and keep the following features: `body-style, num-of-doors, wheel-base, engine-size, horsepower, highway-mpg, price`.
3. Drag in Two-Class Boosted Decision Tree and in the drop-down for "Create trainer mode" select "Parameter range". This allows us to select different values for the two hyperparameters in th model, such as the maximum number of leaves, the minimum leaf size, the learning rate, and the number of trees to construct. Uncheck the box that says "Allow unknown levels..."

4. Train, score and evaluate a Two-Class Boosted Decision Tree for predicting `num-of-doors`. You should be familiar with this process now.
5. The AUC we obtained was on the training data, so it can make us over-confident in the model's predictive ability. Use Split Data to first split the data into training and test sets (use an 75-25 split). Notice that Split Data has two outlets. The one on the right is the training data, and the one on the left is the test data. Note that you can hold the CTRL key and select multiple modules, copy them and paste them back into the canvas.

6. Train the data on the training dataset, then score the test data and evaluate your scores.
7. Compare the AUC from evaluating the scores on the training data to the AUC from evaluating the scores on the test. Which one is higher? Why?
8. Replace `Two-Class Boosted Decision Tree` with `Two-Class Logistic Regression` and rerun the experiment. What do you see now?


Automobile price data (Raw)
Select Columns in Dataset
Split Data
1
2
Two-Class Boosted Decision...
Train Model
Score Model
Evaluate Model
Two-Class Boosted Decision...
Train Model
Score Model
Evaluate Model

END OF LAB

Ensemble models

A boosted decision tree is an example of an ensemble model. The idea behind ensemble models is that while any single model may not make great predictions, we improve our predictions by using the "wisdom of crowds": We can build multiple "variations" of the same model (algorithm) and combine their results (by averaging them if it's regression and using majority rule if it's classification). In addition to making better predictions, ensemble models usually also make more stable predictions: small changes in the data cause only small changes in the predictions.

Two of the most common ensemble modeling techniques are bagging and boosting.

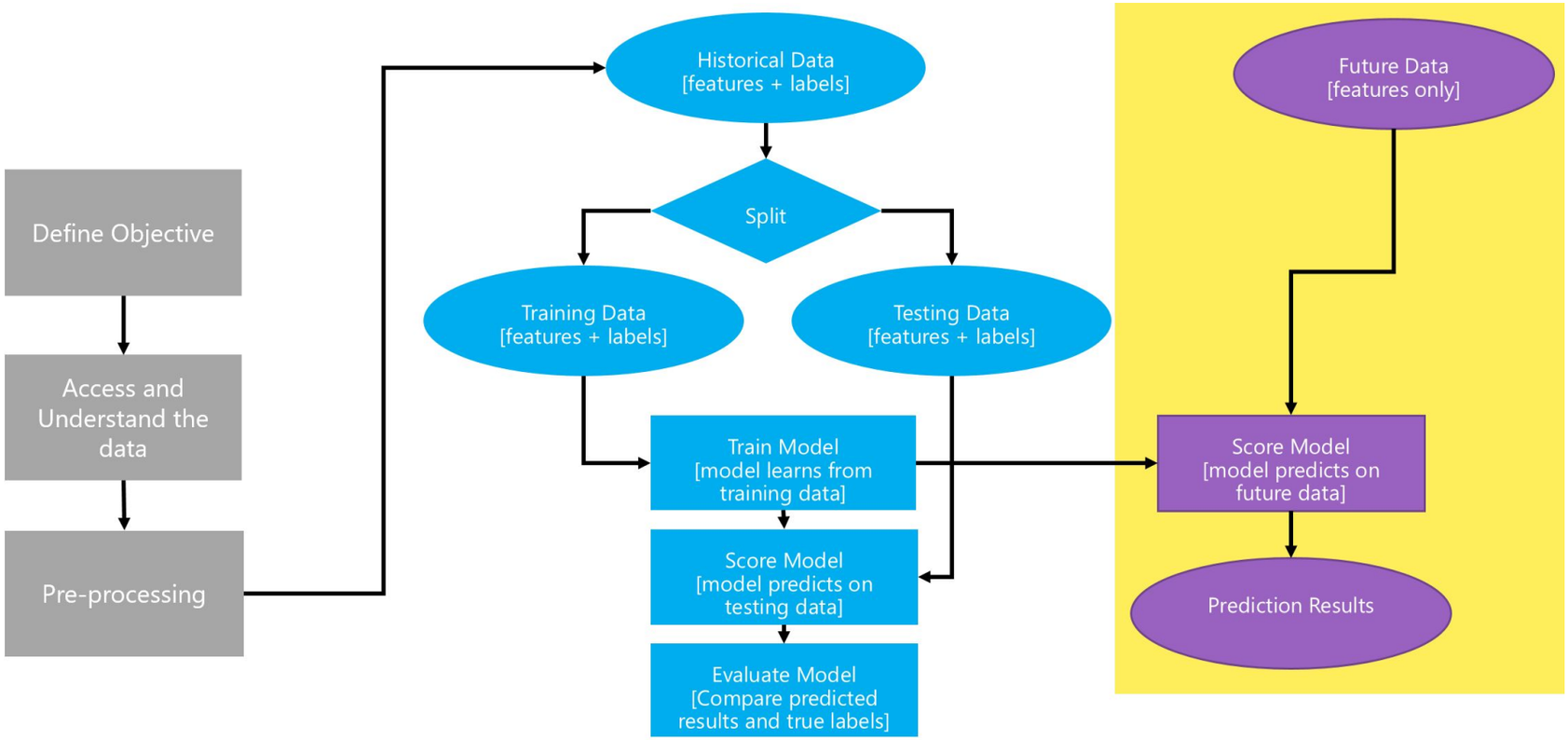
Bagging

- An ensemble where the *same* model is built multiple times on different samples of the data.
- An example of a **random forest**, which is just a collection of decision trees.
- Bagging can be done in parallel, since each model is built independently.
- Bagging reduces a model's variance, not its bias, so it's suitable for complex models.

Boosting

- A sequential ensemble method where each model is trying improve upon the previous model by putting more emphasis on those observations the previous model had difficulty predicting
- An example is a gradient boosted decision tree.
- Boosting reduces a model's bias, not its variance, so it's suitable for high-bias (simpler) models.

Part 3: Model deployment


Define Objective
Access and Understand the data
Pre-processing
Historical Data [features + labels]
Split
Training Data [features + labels]
Testing Data [features + labels]
Train Model [model learns from training data]
Score Model [model predicts on testing data]
Evaluate Model [Compare predicted results and true labels]
Future Data [features only]
Score Model [model predicts on future data]
Prediction Results

What we will learn

- Development and production
- Scaling and consumption
- The Azure Machine Learning Studio
- The data science process
- The data science workflow

Development vs production

- When we develop models, we want to be unhindered so we can focus on the analysis, but this mindset can get us trouble when taking our code to production.
- We should consider how the model is to be used in production. In a production environment we are not just concerned with model accuracy, but also how the model scales with data (scalability) and how the model is consumed (operationalization).
- The requirement imposed on us by the production environment can often send us back to the development phase to reiterate.

Scoring in production

In production models are usually used to make predictions, also known as scoring. Predictions can be made

- **in batch**: scoring future data in one go, done on a schedule (such as nightly or hourly) as future data accumulates, can be used to power up dashboards and reporting systems.
- **online**: scoring a single case with real-time response time, can be used to make intelligent applications.

Operationalizing a model to make batch vs online predictions usually involves very different considerations.

Model retraining in production

A model's accuracy may degrade over time, as the future data that the model scores starts to deviate from the training data that was supposed to represent it. If a sample of the data at the time the model was built looks "different" (statistically speaking) from a sample of the data now. So we occasionally retrain or refresh models to reflect these changes in the model and improve the accuracy of future predictions.

What can go wrong in production?

what happens	possible solutions
scoring is slow	choose a simpler model or scale out (parallel scoring is easy)
retraining (or refreshing) models fails on larger datasets	choose a simpler model or scale out (non-trivial)
predictions occasionally fail	check edge cases and make sure output is specific and predictable

Quiz

- What’s the most common way a dashboard consumes models?
- What’s the most common way a webapp consumes models?

Answer

> A dashboard can have models generate predictions in batch so the predictions are ready to be served by the dashboard. However, it's also possible to have real-time dashboards where models need to score new data as it streams in.
> A webapp usually needs to be able to score data *online* (as opposed to *batch*).

Quiz

For each of the terms below state if the concept is more relevant to development or to production:
Model evaluation

- Variable selection
- Scoring future data
- Scoring the test data
- How well a model scales

Answer

> Variable selection: we do this during training
> Scoring future data: we test this during development and implement it in production
> Scoring the test data: we do this so we can evaluate a model during training
> How well a model scales: this is a question that usually comes up in the context of scoring in production, but we can also ask this question during development when we have big data and longer training times

Quiz

State two reasons for why more accurate models are not always better.

Answer

> More accurate models are generally more complex, so they are less explainable and demand longer training time and longer scoring time (which can affect production).

In Azure Machine Learning, we can

- Deploy web services
- Retrain models through APIs
- Manage web service endpoints
- Scale a web service
- Consume web service

Both batch and online scoring is available in Azure Machine learning as a web service:

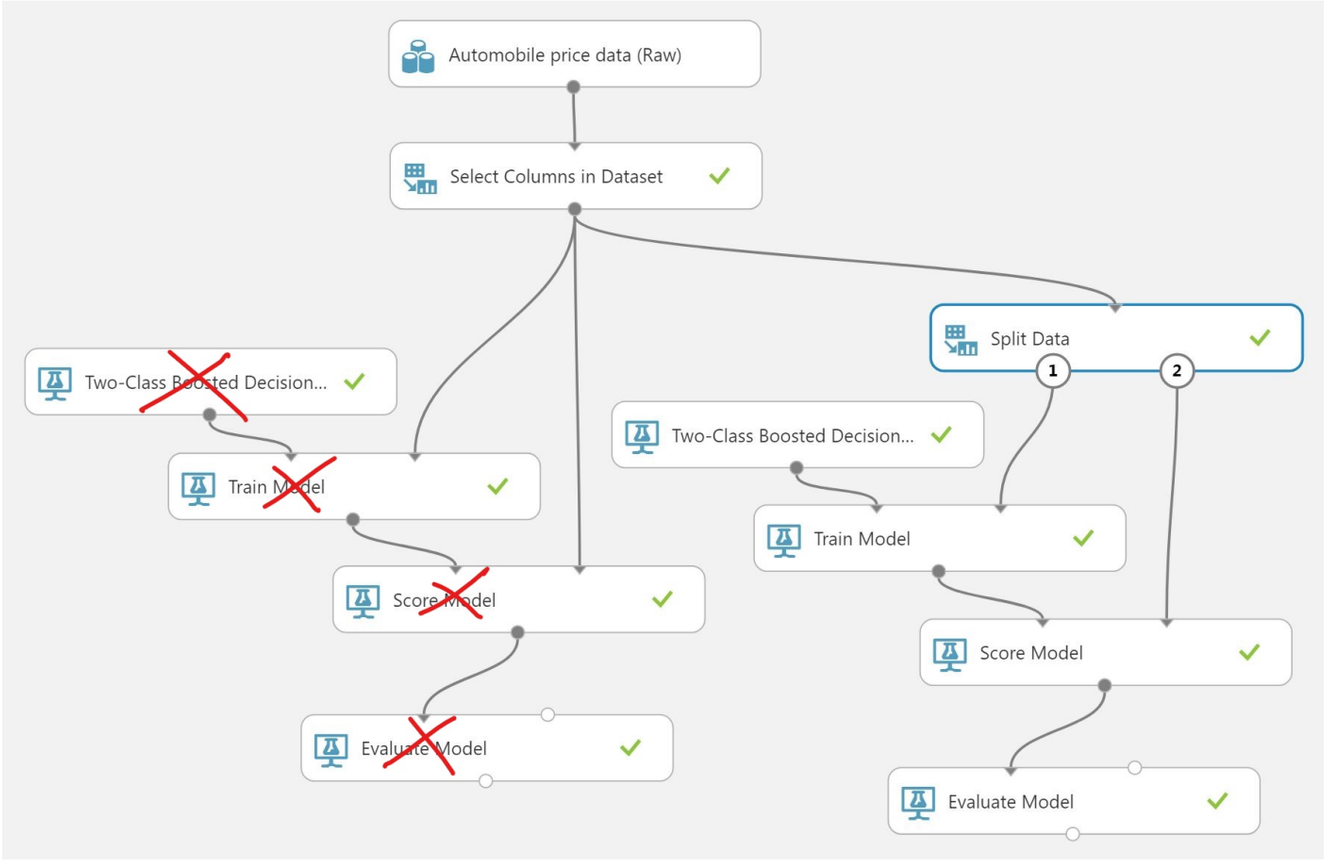
- **Request-Response Service (RRS)**: A low latency, highly scalable service that provides an interface to the stateless models created and deployed by using Machine Learning Studio.
- **Batch Execution Service (BES)**: An asynchronous service that scores a batch for data records.

LAB 9

Deploying a model in AzureML

Expected lab duration: 15 minutes.

1. Return to the last lab and delete all the modules associated shown in this screenshot, then rerun the experiment.
2. Click on "Set Up Web Service" and choose "Predictive Web Service" which is a scoring web service. This creates a new tab next to "Training experiment" called "Predictive experiment". You see some changes happening and modules being rearranged in the canvas. Describe what changed. Why did `Evaluate Model` disappear? Why is `Automobile price data (Raw)` greyed out? What are the two modules that were added in?


Automobile price data (Raw)
Select Columns in Dataset
Split Data
1
2
Two-Class Boosted Decision...
Train Model
Score Model
Evaluate Model
Two-Class Boosted Decision...
Train Model
Score Model
Evaluate Model

3. Run the experiment, then click on "Deploy Web Service". On the new page that opens click on "New Web Services Experience". On this page, we can test and monitor our web service.
4. Under "Basic" click on the link called "Test endpoint". Under "input1", fill in the following values for the inputs: `body-style = sedan`, `horsepower = 102`, and `city-mpg = 18`. Then scroll down and click on "Test Request Response". Under "output1" you should now see the predicted price in the field called "Scored Labels".
5. Is this an example of batch prediction or online? Why were the other input fields not required?
6. Browse around the other sections and see if you can guess what each section is used for.

Quickstart | Dashboard | Batch Request Log | Configure | Consume | Test | Swagger API

← Linear Regression: Automobile price [Predictive Exp.]

default

View in Studio

BASICS

Test endpoint
Configure endpoint
Use endpoint
Launch in Excel

MANAGE & MONITOR

Tutorial: Retrain web service

DEVELOP

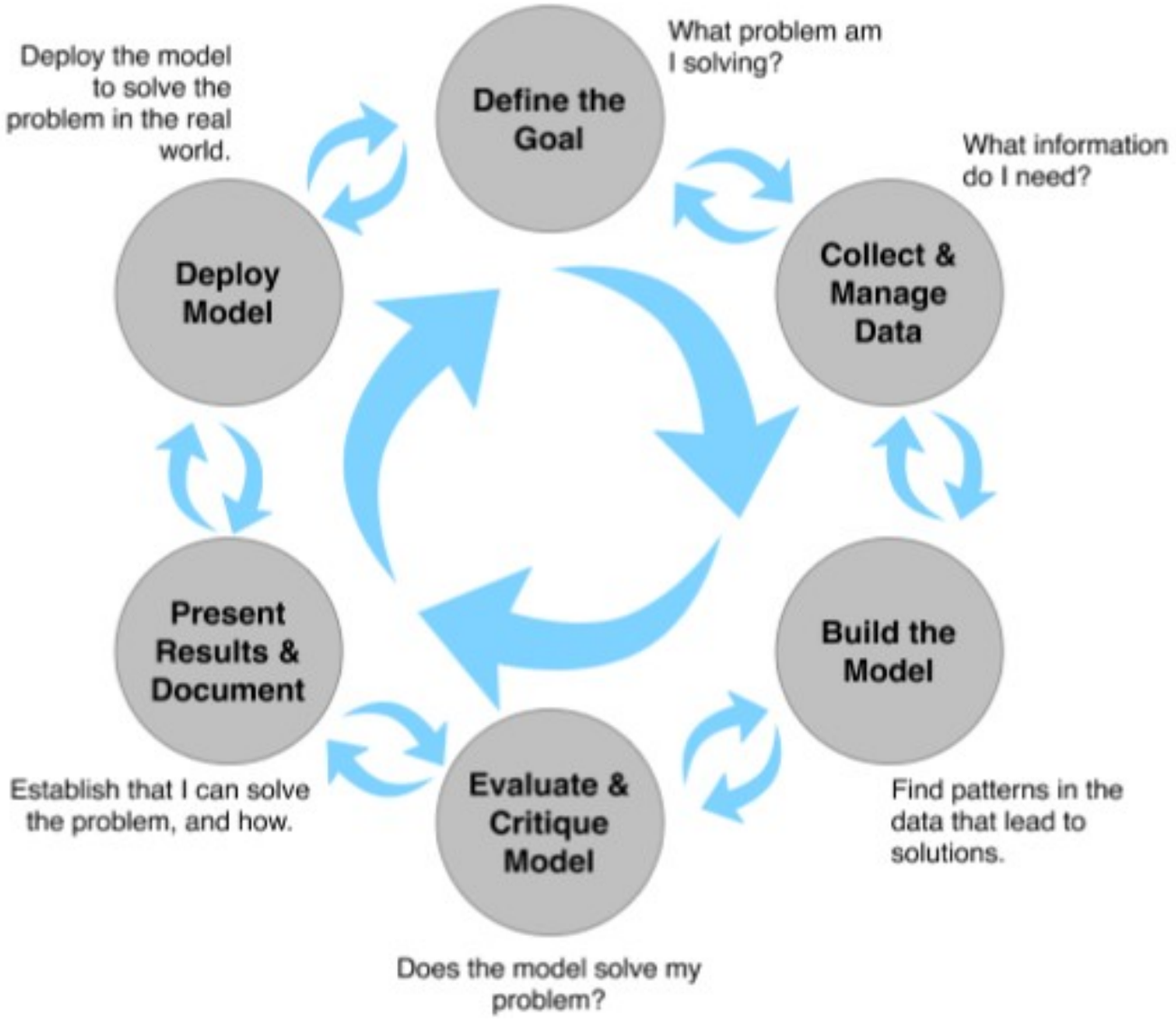
Tutorial: How to build apps
Tutorial: Debug web service
Tutorial: Best practices

END OF LAB

Let's recap

- Fitting, training, modelling
- Predicting, scoring
- Pre-processing:
- Aggregating, cleaning, reshaping, feature extraction.
- Handling missing values and outliers
- Feature creation
- Model complexity, overfitting and underfitting
- Model interpretability vs prediction accuracy

The data science workflow



Resources

- Data pipeline scenarios in AzureML: https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-data-science-plan-sample-scenarios
- Scaling an Azure Machine Learning web service by adding additional endpoints: https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-scaling-webservice

End of chapter

The end

thank you

www.linkedin.com/in/sethmott/


Microsoft