Applied Machine Learning: Real-World Data (how to pre-process them?)

NAWWAF KHARMA

Real-World Data

- This chapter uses a real-world machine-learning example: 'churn' prediction. In business, churn refers to the act of a customer canceling or unsubscribing from a paid service. An important, high-value problem is to predict which customers are *likely* to churn in the *near* future.

- 2.1 Getting started: data collection

Questions that are well suited for a supervised ML approach include:

- Which of my customers will churn this month?
- Will this user click my advertisement?
- Is this user account fraudulent?
- Is the sentiment of this tweet negative, positive, or neutral?
- What will demand for my product be next month?

Cont.

- You'll notice a few **commonalities** in these questions:
- First, well-defined instances: they all require making assessments on one or several instances of interest.
- Second, well-defined prediction target.
- Third, well-defined input features.
- Finally, prediction allows action – better or faster.
  - The role of the ML algorithm is to use the training set to determine how the set of input features can most accurately predict the target variable.
  - Let's put all this in the context of the churn prediction problem. Imagine that you work for a telecom company and that the question of interest is, "Which of my current cell-phone subscribers will unsubscribe in the next month?" ..

Cont.

- Training instances

Features									Target
Cust. ID	State	Acct length	Area code	Int'l plan	Voicemail plan	Total messages	Total mins.	Total calls	Churned?
502	FL	124	561	No	Yes	28	251.4	104	False
1007	OR	48	503	No	No	0	190.4	92	False
1789	WI	63	608	No	Yes	34	152.2	119	False
2568	KY	58	606	No	No	0	247.2	116	True

The following subsections provide a **practical** guide to addressing four of the most common data-collection questions:

1. Which input features should I include?
2. How do I obtain known values of my target variable?
3. How much training data do I need?
4. How do I know if my training data is good enough?

Cont.

- 1.2.1 Which features should be included?

Only two practical restrictions exist on whether something may be used as an input feature:

(1) The value of the feature must be known at the time predictions are needed

(2) The feature must be numerical or categorical in nature

Relevant features + potentially relevant features – not everything under the sun. Gene rally,

1 Include all the features that you suspect to be predictive of the target variable.
Fit an ML model. If the accuracy of the model is sufficient, stop.

2 **Otherwise**, expand the feature set by including other features that are less obviously related to the target. Fit another model and assess the accuracy. If performance is sufficient, stop.

3 **Otherwise**, starting from the expanded feature set, run an ML feature selection algorithm to choose the best, most predictive subset of your expanded feature set.

Cont.

- 1.2.2 How can we obtain ground truth for the target variable?

Consider the following training-data collection processes for a few selected ML use cases

* *Ad targeting*—You can run a campaign for a few days to determine which users did/didn't click your ad and which users converted.

* *Fraud detection*—You can pore over your past data to figure out which users were fraudulent and which were legitimate.

* *Demand forecasting*—You can go into your historical supply-chain management data logs to determine the demand over the past months or years.

* *Twitter sentiment*—Getting information on the true intended sentiment is considerably harder.

Cont.

- Other ways of obtaining ground-truth values of the target variable include

(A) Dedicating analysts to manually look through past or current data
(B) Using crowdsourcing to use the "wisdom of crowds" in order to attain estimates
(C) Conducting follow-up interviews with customers or
(D) Running controlled experiments (for example, A/B tests)

Each of these strategies is **labor-intensive**, but you can accelerate the learning process and shorten the time required to collect training data by collecting only target variables. One example of this is a method called **active learning**. Given an existing (small) training set and a (large) set of data with unknown response variable, active learning identifies the subset of instances from the latter set whose inclusion in the training set would yield the most accurate ML model.

Cont.

- 2.1.3 How much training data is required?

These factors determine the amount of training data needed:

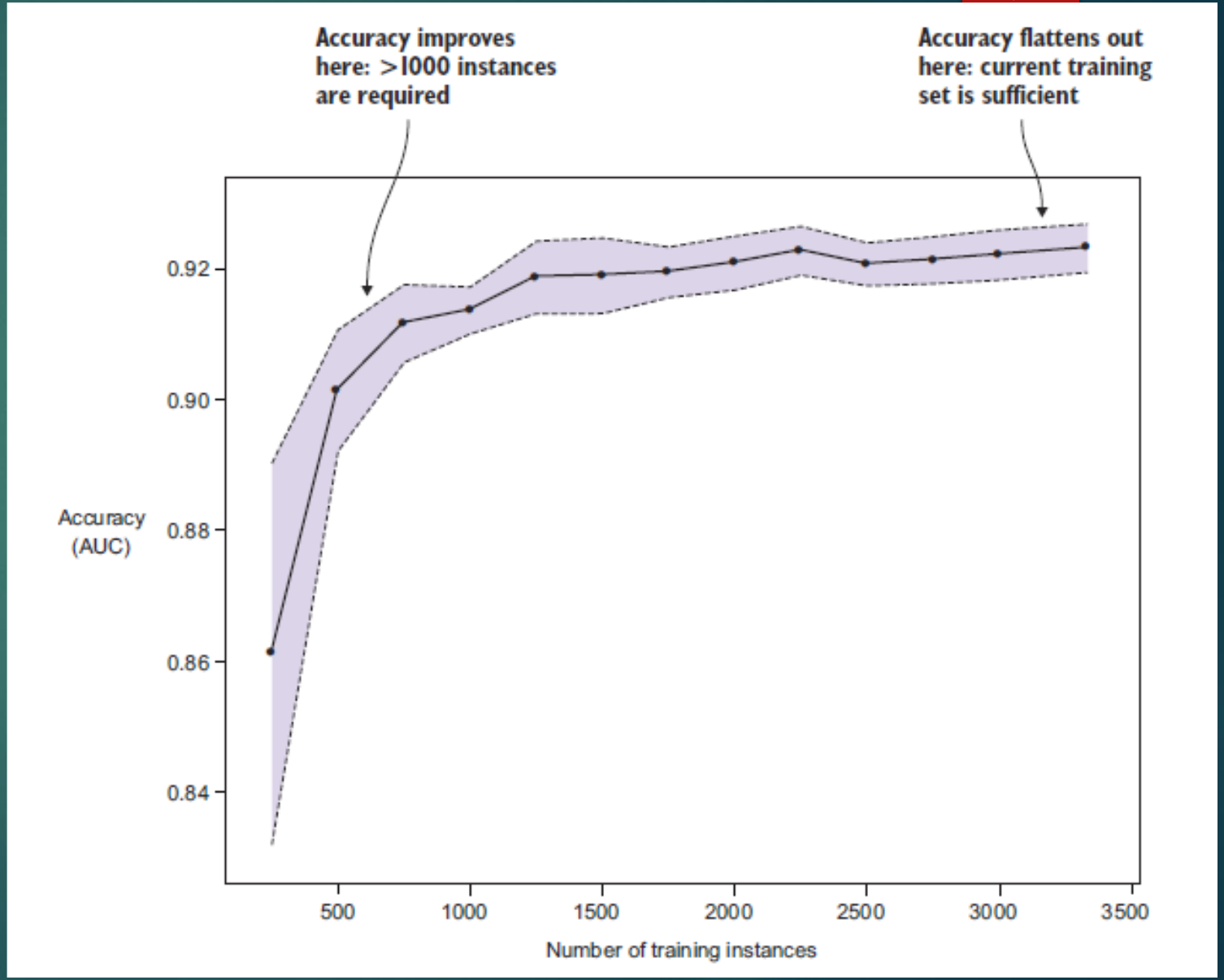
(A) The **complexity of the problem**. Does the relationship between the input features and target variable follow a simple pattern, or is it complex and non-linear?

(B) The **requirements for accuracy**. If you require only a 60% success rate for your problem, less training data is required than if you need to achieve a 95% success rate.

(C) The **dimensionality of the feature space**. If only two input features are available, less training data will be required than if there were 2,000 features.

Cont.

- Using this data, it's straightforward to assess whether you need to collect more data. Do the following:
- Randomly draw different instances for each sample size and see how accuracy changes (stabilizes?) as sample size increases.



Cont.

- 2.1.4 Is the training set representative enough?

A training sample could be non-representative for several reasons:

(A) **Bias**. It was possible to obtain ground truth for the target variable for only a certain, biased subsample of data. *For example, if instances of fraud in your historical data were detected only if they cost the company more than $1,000, then a model trained on that data will have difficulty identifying cases of fraud that result in losses less than $1,000.*

(B) **Process**. The properties of the instances have changed over time. *For example, if your training example consists of historical data on medical insurance fraud, but new laws have substantially changed the ways in which medical insurers must conduct their business, then your predictions on the new data may not be appropriate.*

(C) **Features**. The input feature set has changed over time. *For example, say the set of location attributes that you collect on each customer has changed; you used to collect ZIP code and state, but now collect IP address. This change may require you to modify the feature set used for the model and potentially discard old data from the training set.*

it's important to attempt to make the training set as representative

of future data as possible!

Cont.

- 2.2 Preprocessing the data for modeling

2.2.1 Categorical Features

An example is a feature representing the day of the week, which could validly be encoded as either numerical (number of days since Sunday) or as categorical (the names Monday, Tuesday, and so forth).

Person	Name	Age	Income	Marital status
1	Jane Doe	24	81,200	Single
2	John Smith	41	121,000	Married

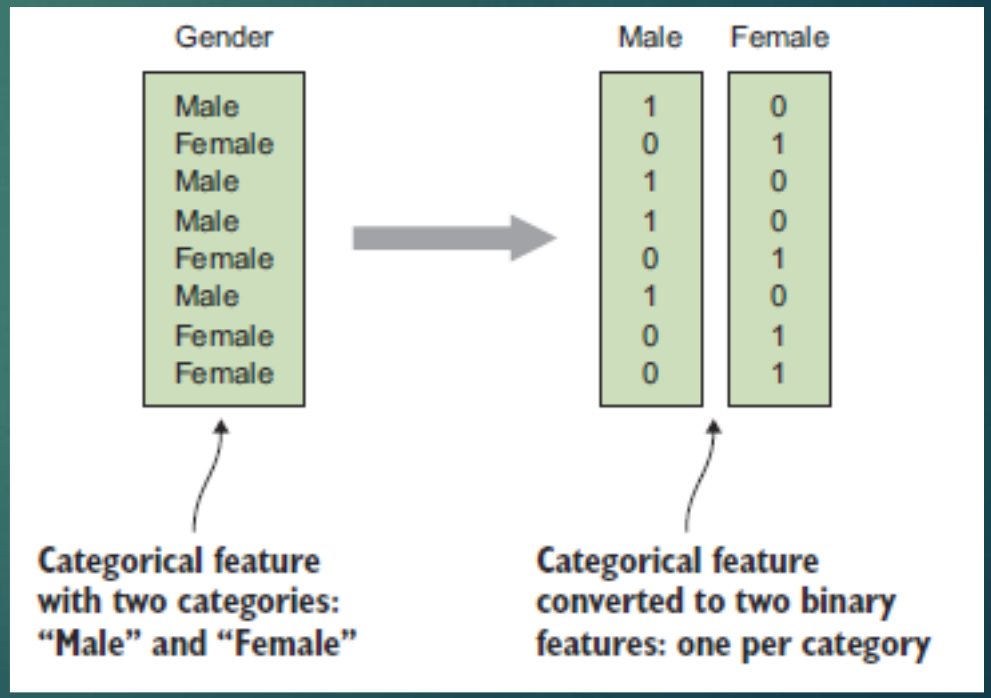
Categorical features

PassengerId	Survived	Pclass	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Male	22	1	0	A/5 21171	7.25		S
2	1	1	Female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Female	26	0	0	STON/02. 3101282	7.925		S
4	1	1	Female	35	1	0	113803	53.1	C123	S
5	0	3	Male	35	0	0	373450	8.05		S
6	0	3	Male		0	0	330877	8.4583		Q

Cont.

- Some machine-learning algorithms use categorical features natively, but generally they need data in numerical form.
- Instead, you can convert each of the categories into a separate binary feature that has value 1 for instances for which the category appeared, and value 0 when it didn't.

```
def cat_to_num(data):
    categories = unique(data)
    features = []
    for cat in categories:
        binary = (data == cat)
        features.append(binary.astype("int"))
    return features
```



Cont.

- 2.2.2 Dealing with missing data

There are two main types of missing data, which you need to handle in different ways:

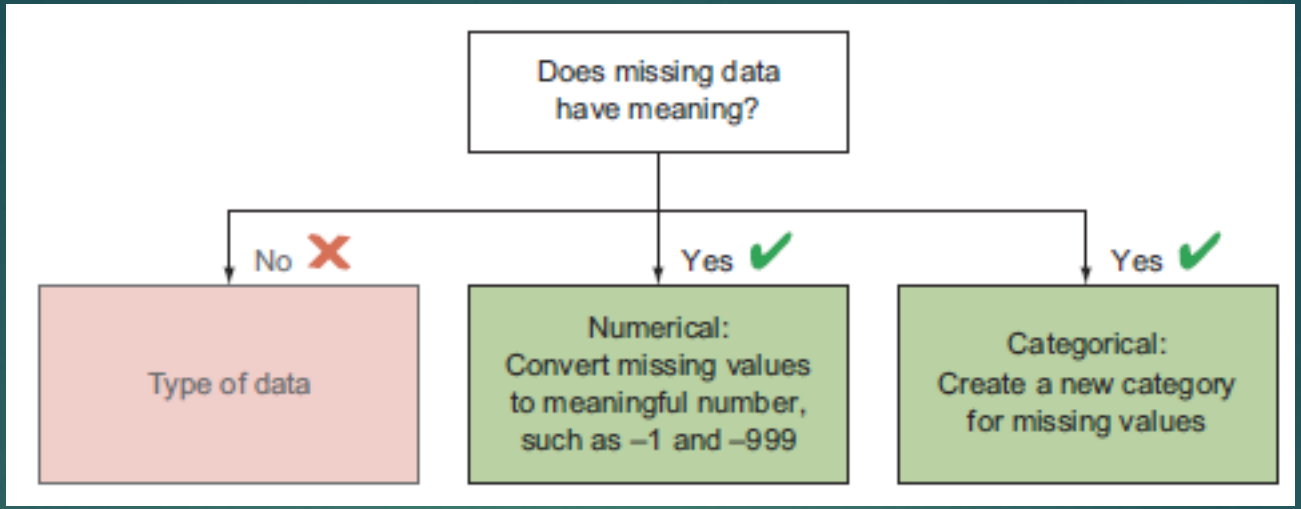
A. First, for some data, the fact that it's missing can carry meaningful information that may be useful for the ML algorithm.

B. In other cases, measurement was impossible

PassengerId	Survived	Pclass	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Male	22	1	0	A/5 21171	7.25		S
2	1	1	Female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Female	26	0	0	STON/02. 3101282	7.925		S
4	1	1	Female	35	1	0	113803	53.1	C123	S
5	0	3	Male	35	0	0	373450	8.05		S
6	0	3	Male		0	0	330877	8.4583		Q

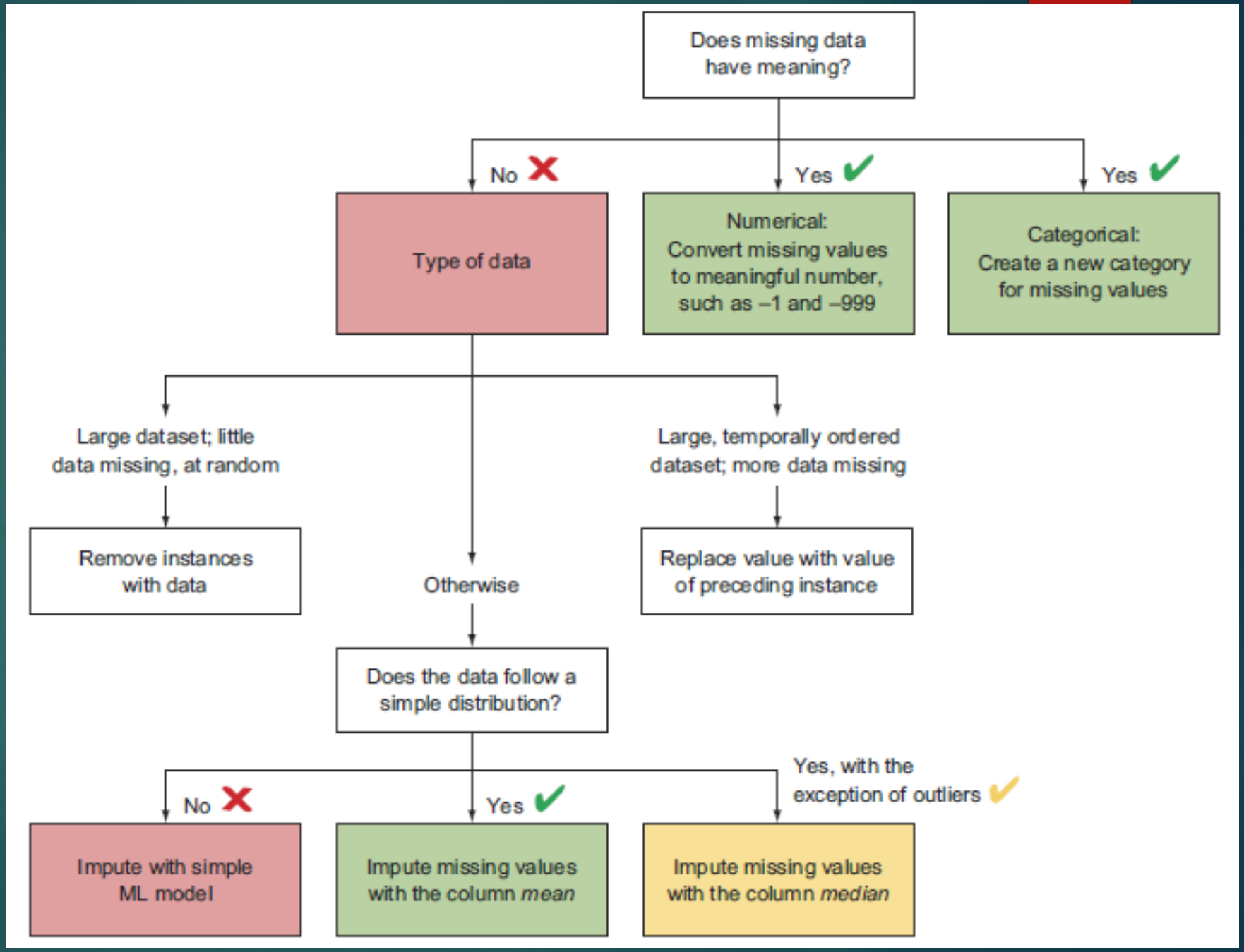
Missing values

Cont.



- This concept of replacing missing data is called **imputation**.
- If you have a large dataset and only a handful of missing values, dropping the observations with missing data is the easiest approach.
- Another simple approach is to assume some temporal order to the data instances and replace missing values with the column value of the preceding row.
- When possible, it's usually better to use a larger portion of the existing data to guess the missing values (e.g., mean , media ..), assuming some form of distribution

Cont. (summary of handling missing data)



Cont.

- 2.2.3 Simple feature engineering

- You'll create three new features from the Cabin feature.

- By now it should be no surprise what we mean by feature engineering: using the existing features to create new features that increase the value of the original data by applying our knowledge of the data or domain in question.

PassengerId	Survived	Pclass	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Male	22	1	0	A/5 21171	7.25		S
2	1	1	Female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Female	26	0	0	STON/02. 3101282	7.925		S
4	1	1	Female	35	1	0	113803	53.1	C123	S
5	0	3	Male	35	0	0	373450	8.05		S
6	0	3	Male		0	0	330877	8.4583		Q

Cont.

- Data Normalization

To make sure all features are considered equally, you need to normalize the data. Often data is normalized to be in the range from 0 to 1, or from –1 to 1.

```
def normalize_feature(data, f_min=-1.0, f_max=1.0):
    d_min, d_max = min(data), max(data)
    factor = (f_max - f_min) / (d_max - d_min)
    normalized = f_min + (data - d_min)*factor
    return normalized, factor
```

Note that you return both the normalized data and the factor with which the data was normalized. You do this because any new data (for example, for prediction) will have to be normalized in the same way in order to yield meaningful results.

Standardized residual = [e – mean(e)] / SD(e), where e = observ. – predict.

Cont.

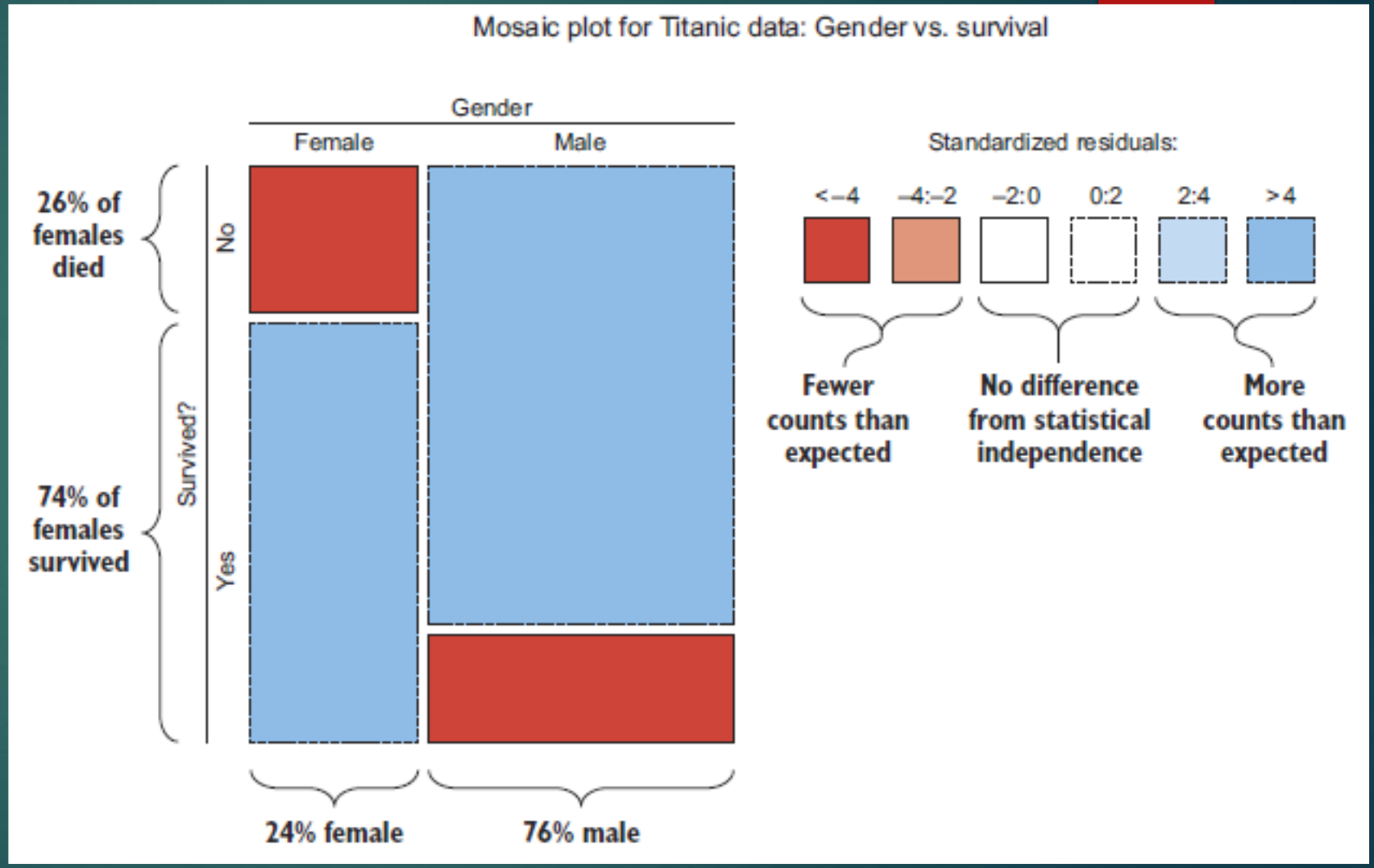
- 2.3 Data visualization

This section focuses on methods for visualizing the association between the target variable and the input features.

- 2.3.1 Mosaic plots

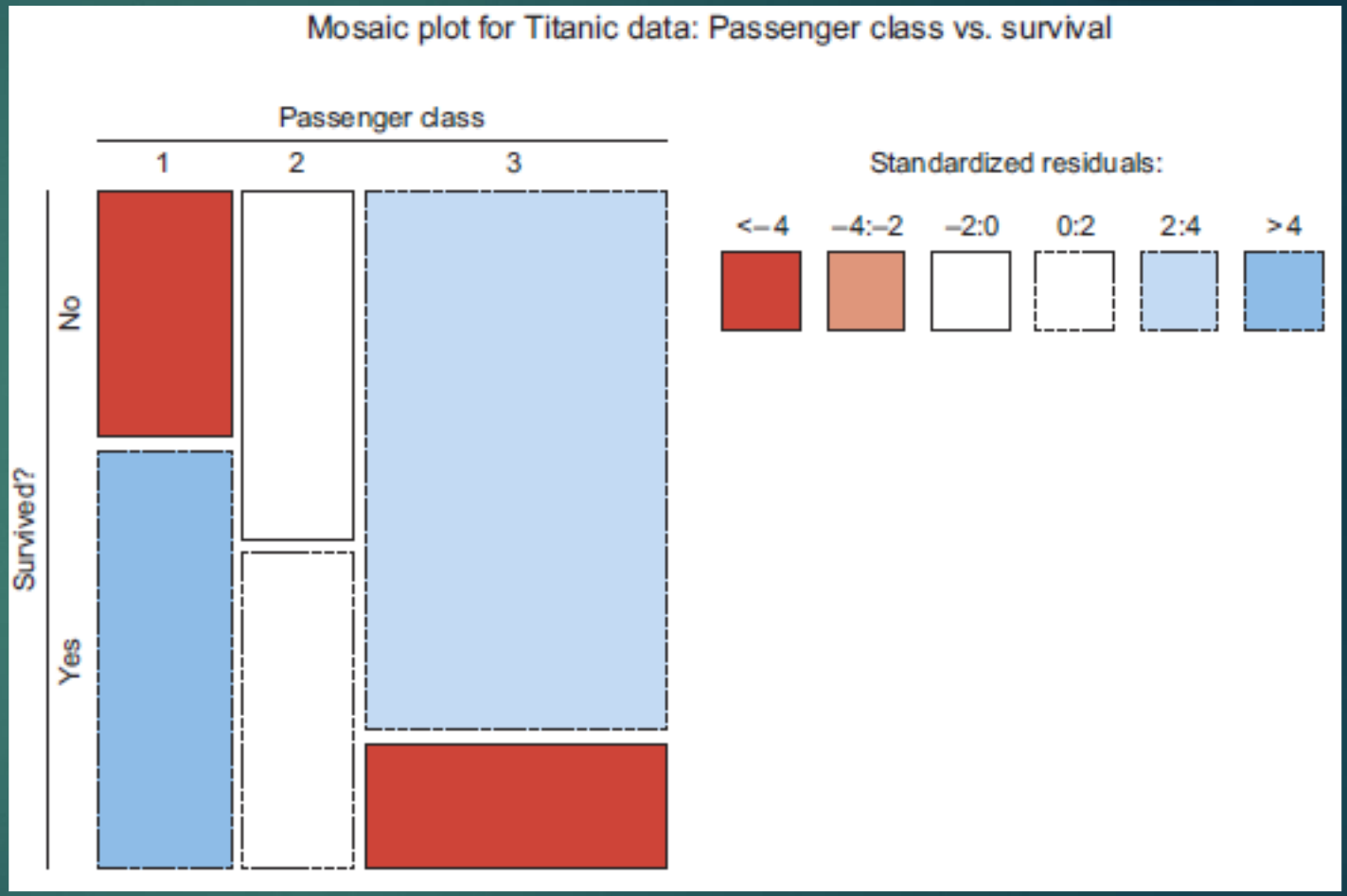
To demonstrate the utility of mosaic plots, you'll use one to display the relationship between passenger gender and survival in the Titanic Passengers dataset.

> What results is a quick visualization of the relationship between gender and survival.



Cont.

- This tells you that when building a machine-learning model to predict survival on the Titanic, gender is an important factor to include.
- mosaic plot for survival versus passenger class (first, second, and third).

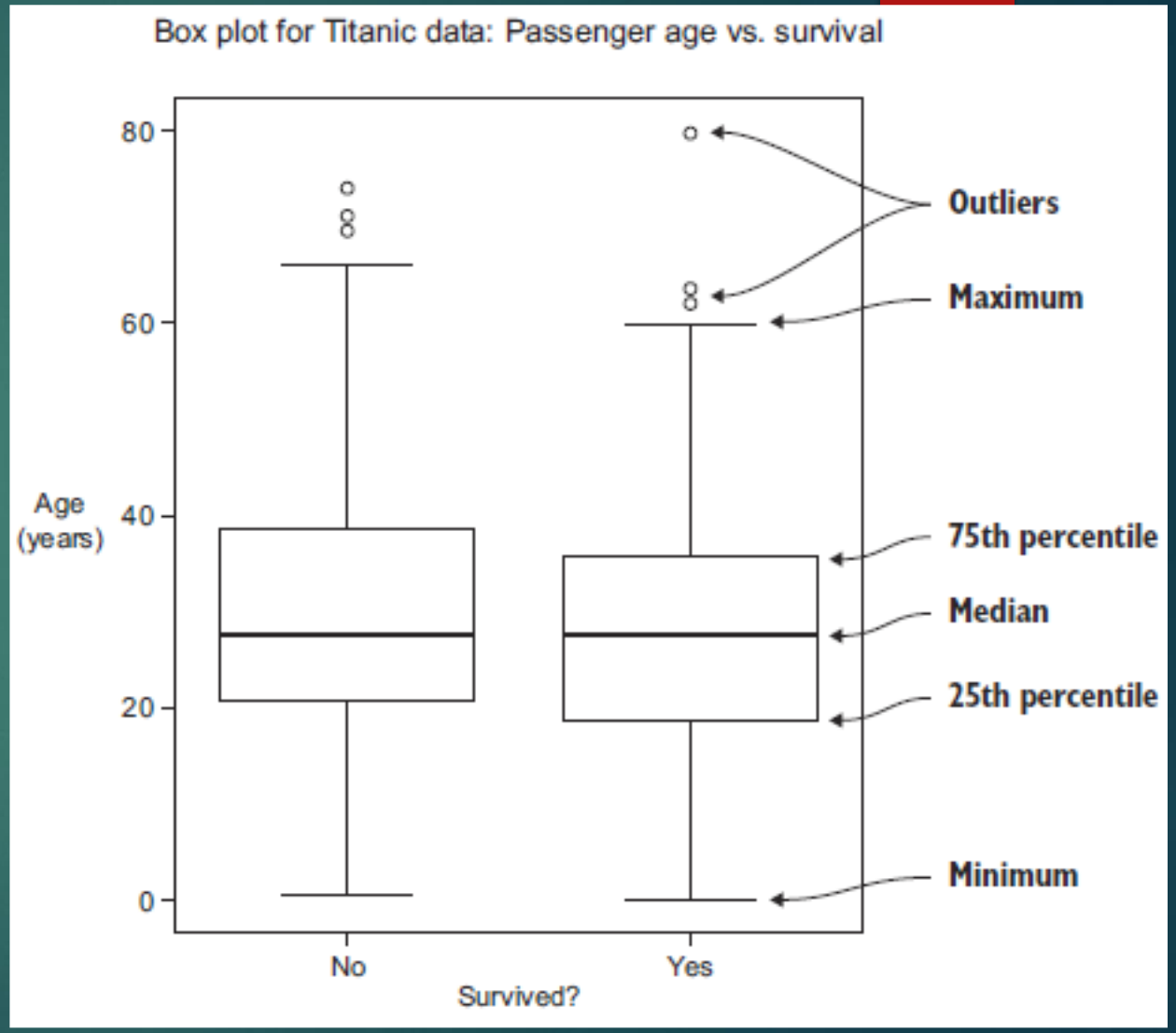


Cont.

- 2.3.2 Box plots

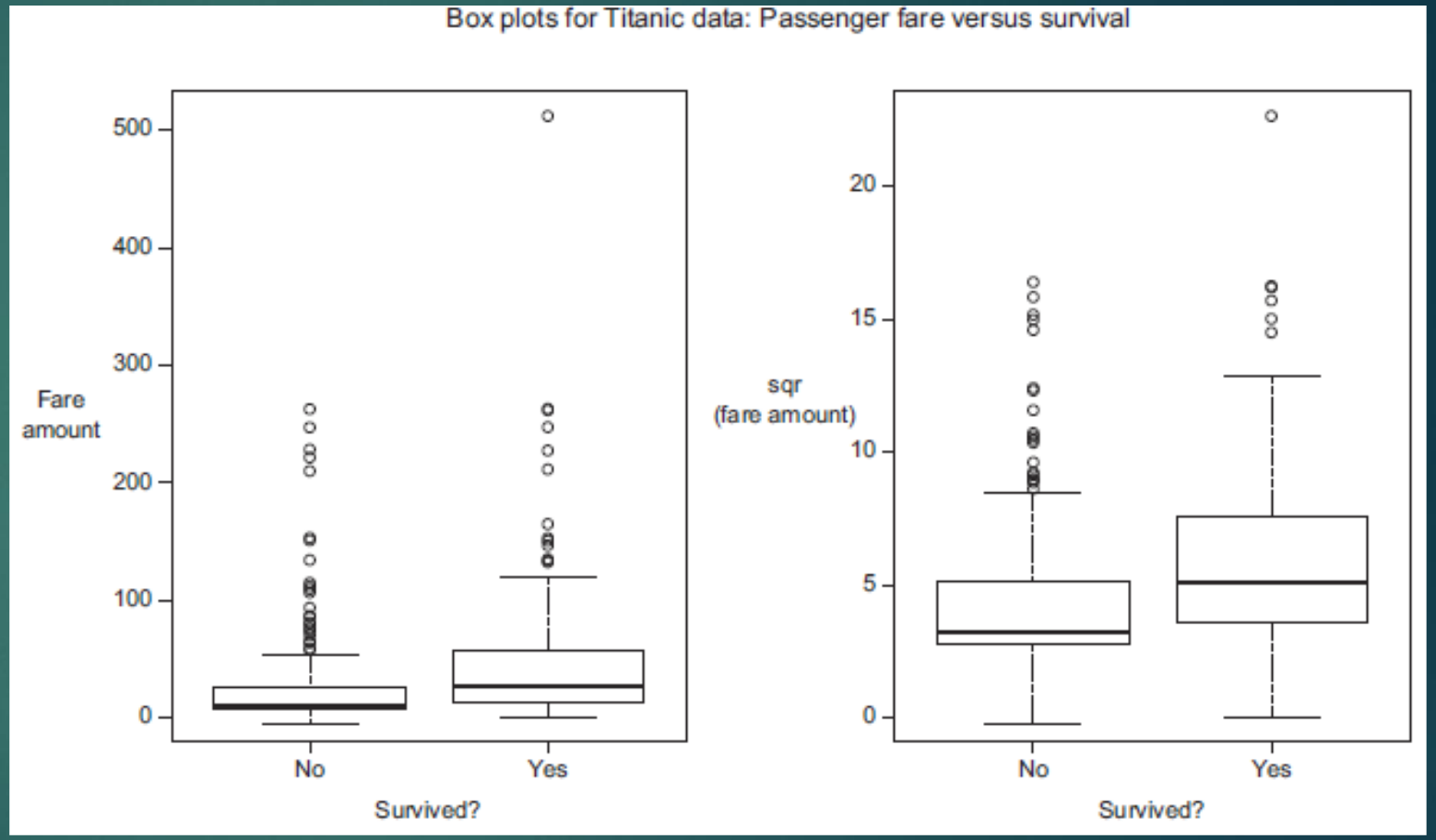
Box plots are a standard statistical plotting technique for visualizing the distribution of a numerical variable. For a single variable, a box plot depicts the quartiles of its distribution

> Visualizations aren't a substitute for ML modeling!



Cont.

- box plots exploring the relationship between passenger fare paid and survival outcome. In the left panel, it's clear that the distributions of fare paid are highly skewed ..
- This is remedied by a simple transformation of the fare (square root, in the right panel)

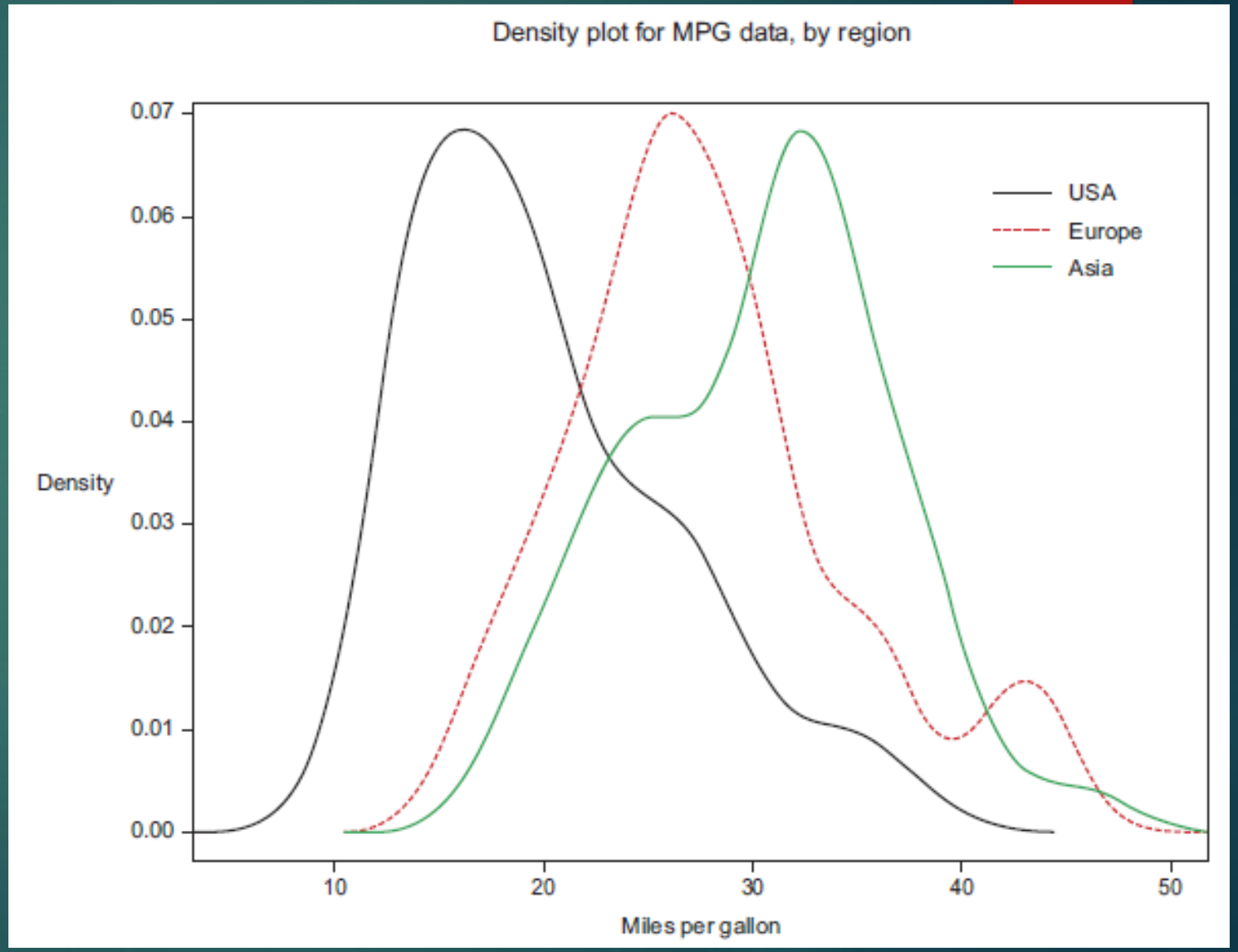


Cont.

- 2.3.3 Density plots

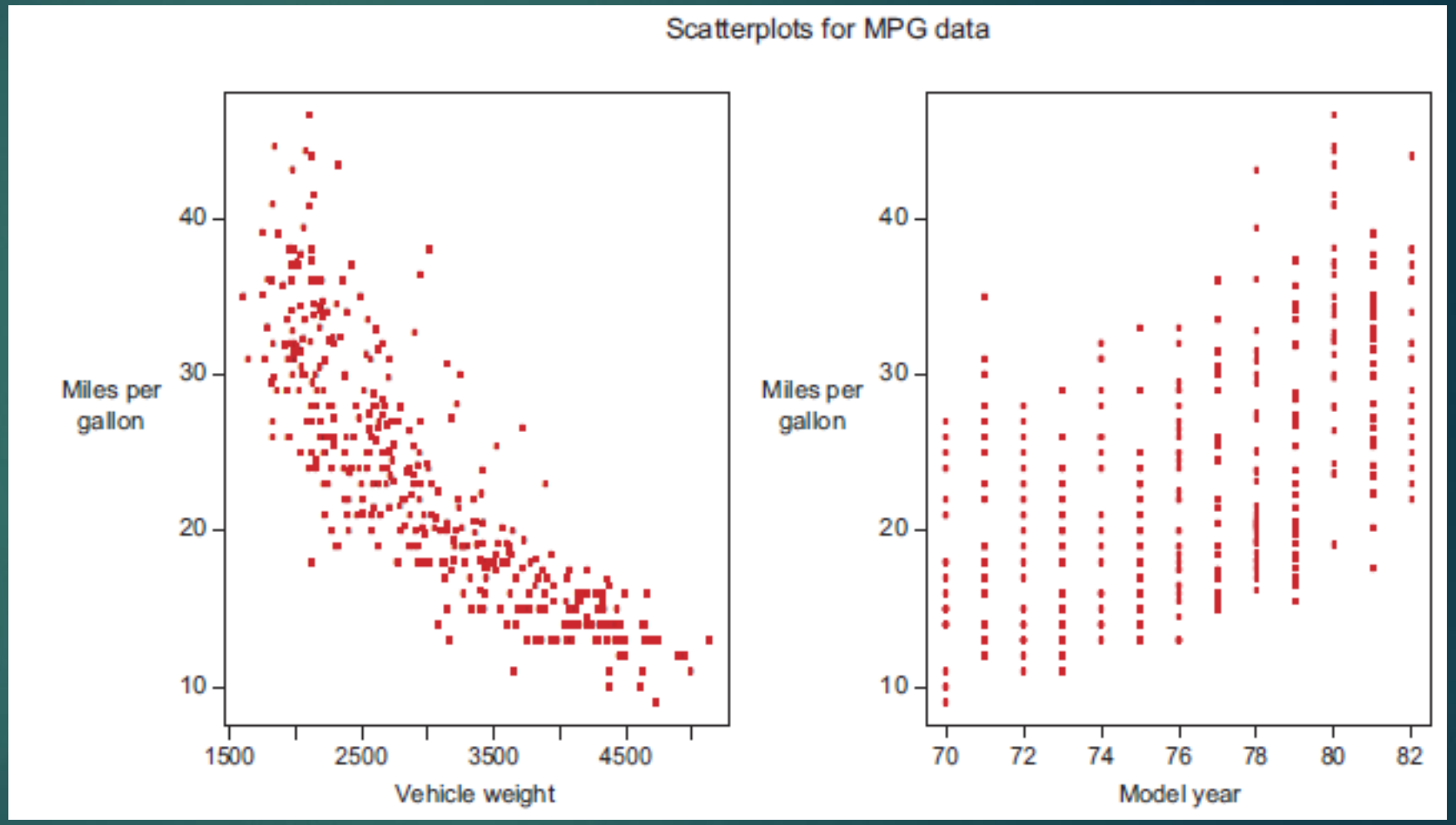
Density plots display the distribution of a single variable in more detail than a box plot.

- density plots are similar to histograms, but their smooth nature makes it much simpler to visualize multiple distributions in a single figure.
- Asian cars tend to have higher MPG, followed by European and then American cars.
- What about the bumps?



Cont.

- 2.3.4 Scatter plots
- A scatter plot is a simple visualization of the relationship between two numerical variables
- Though simple, scatter plots can reveal both linear and nonlinear relationships between the input and response variables.



Summary

> Steps in **compiling your training data** include the following:

- Deciding which input features to include
- Figuring out how to obtain ground-truth values for the target variable
- Determining when you've collected enough training data
- Keeping an eye out for biased or non-representative training data

> **Preprocessing steps** for training data include the following:

– Recoding categorical features
– Dealing with missing data
– Feature normalization (for some ML approaches)
– Feature engineering

> Four useful **data visualizations** are mosaic plots, density plots, box plots, and scatter plots:

		Input Feature	
		Categorical	Numerical
Response Variable	Categorical	Mosaic plots	Box plots
	Numerical	Density plots	Scatter plots