Applied Machine Learning: Modeling and Prediction (the heart of ML)

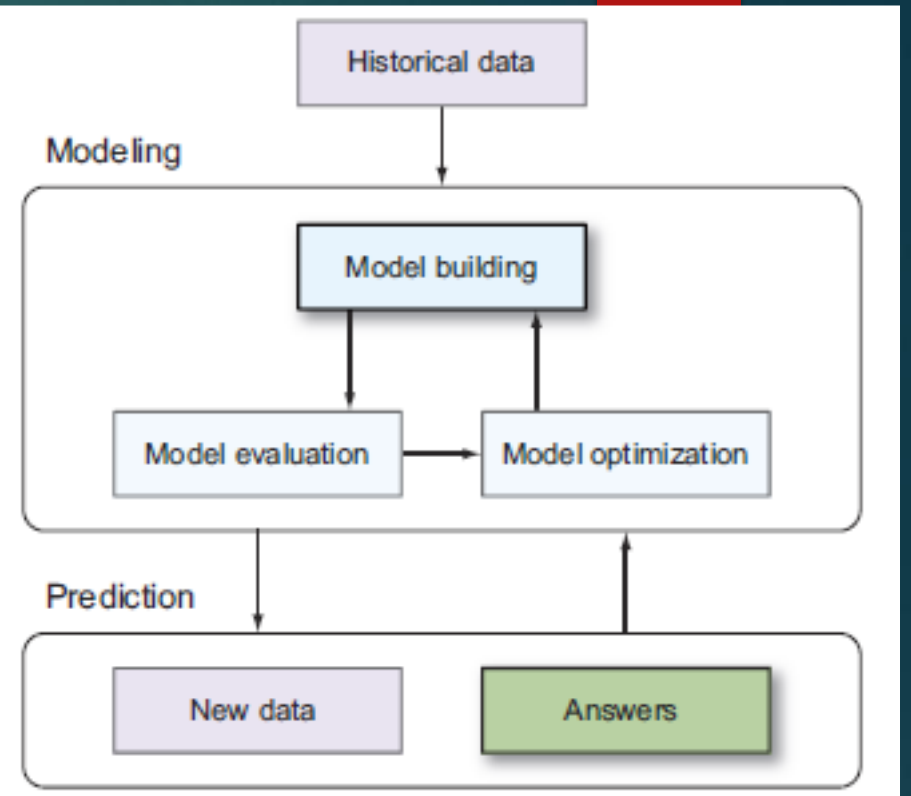
NAWWAF KHARMA

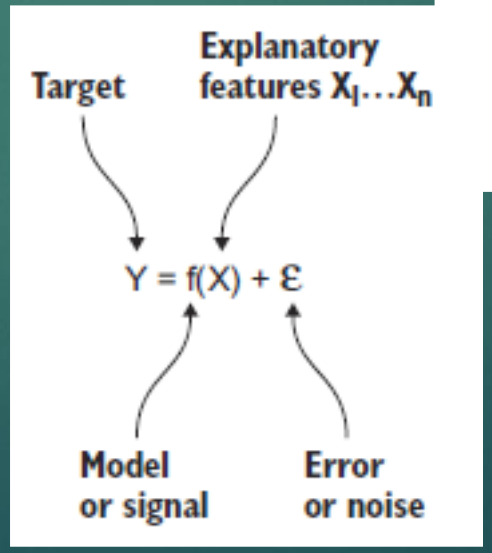
Modeling and Prediction

- The next step in the machine-learning workflow is to use that data to begin exploring and uncovering the relationships that exist between the input features and the target.
- 3.1 Basic machine-learning modeling
- 3.1.1 Finding the relationship between input and target

> **Input** features are typically referred to using the symbol X

> Likewise, the target variable is typically referred to as Y.





Modeling and Prediction

- .. you could have numerous sources of noise, including (and certainly not limited to) the following:
  - **Imperfect measurement** of each vehicle's MPG rating caused by small inaccuracies in the measuring devices or process—measurement noise
  - **Variations in the manufacturing** process (of the measured object), causing each car in the fleet to have slightly different MPG measurements—manufacturing process noise
  - **Lack of access** to the broader set of features that would exactly determine MPG; insufficiency of features for complete characterization of the output

Cont.

- 3.1.2 The purpose of a good model

A. **Prediction**: Prediction is the most common use of machine-learning systems, including these cases ..

- Deciphering handwritten digits or *images* contents or voice recordings
- Predicting the stock *market*
- Forecasting *weather* (near-term or future climate)
- Predicting which users are most likely to click, convert, or buy
- Predicting which users will need product support and which are likely to *unsubscribe*
- Determining which transactions are *fraudulent*
- Making *recommendations*

Cont.

B. **Inference**: better understand the relationships between the input features and the output target.

- Which input features are most **strongly related** to the target variable?
- Are those relationships **positive or negative**?
- Is f a simple relationship, or is it a function that's more nuanced and **nonlinear**?

Returning to the Auto MPG example, you can use inference to answer questions such as these: Does manufacturer region have an effect on MPG? Which of the inputs are most strongly related to MPG?

Cont.

- 3.1.3 Types of modeling methods

(A) Parametric methods

The simplest example of a parametric approach is linear regression.

$$f(\mathbf{X}) = \beta_0 + \mathbf{X}_1 \times \beta_1 + \mathbf{X}_2 \times \beta_2 + \dots$$

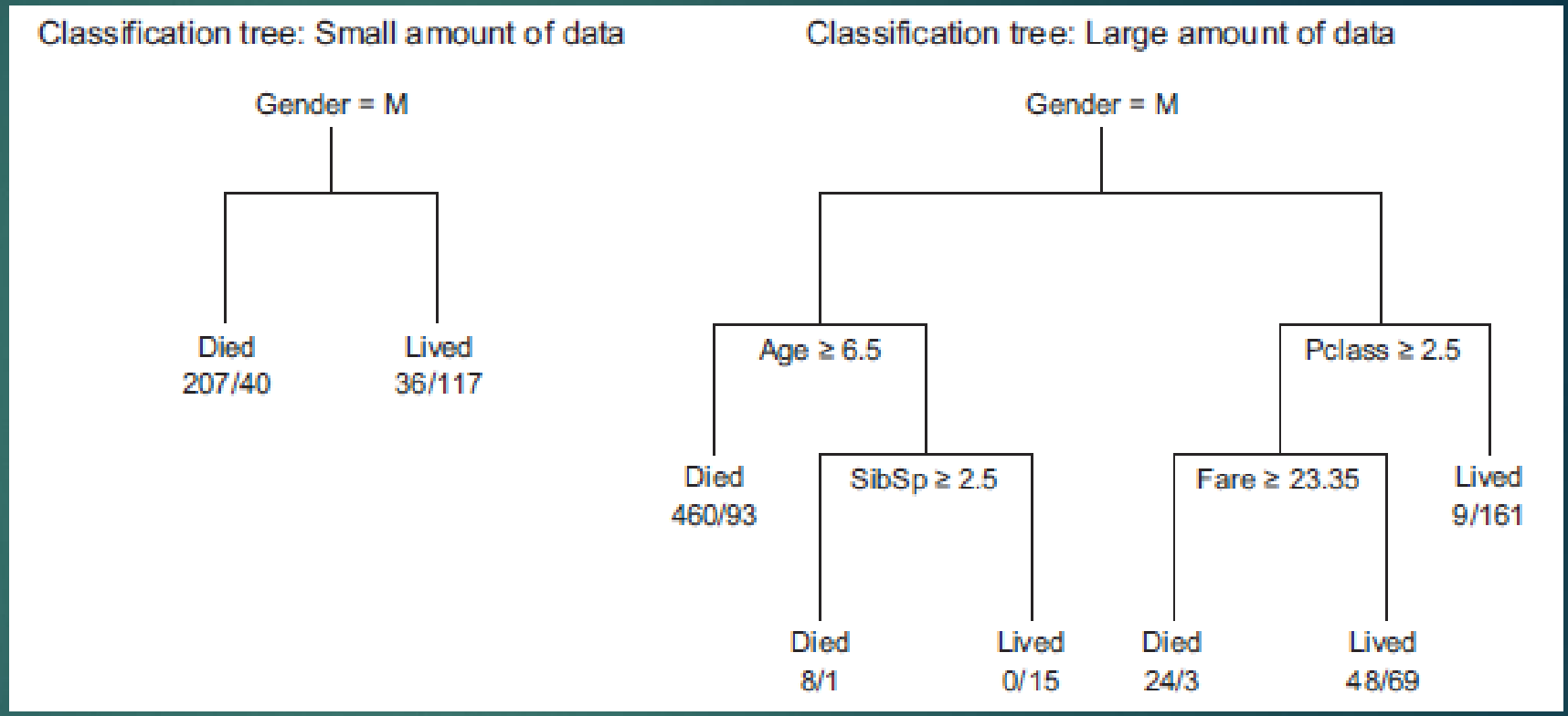
- Other examples of commonly used parametric models include logistic regression, polynomial regression, linear discriminant analysis, quadratic discriminant analysis, (parametric) mixture models, and naïve Bayes.
- The drawback of parametric approaches is that they make strong assumptions about the true form of the function f. In most real-world problems, f doesn't assume such a simple form .. So most real-world models use nonparametric ML methods

Cont.

(B) Nonparametric Models

> A simple example of a nonparametric model is a classification tree. Classification trees are nonparametric because the depth and complexity of the tree isn't fixed in advance, but rather is learned from the data.

> examples of nonparametric approaches to machine learning include k-nearest neighbors, splines, basis expansion methods, kernel smoothing, generalized additive models, neural nets, bagging, boosting, random forests, and support vector machines.



Cont.

- 3.1.4 Supervised versus unsupervised learning

In unsupervised learning, you have access to only input features, and don't have an associated target variable. They fall under two broad categories:

A. *Clustering*—Use the input features to discover natural groupings in the data and to divide the data into those groups. Methods: k-means, Gaussian mixture models, and hierarchical clustering.

B. *Dimensionality reduction*—Transform the input features into a small number of coordinates that capture most of the variability of the data. Methods: principal component analysis (PCA), multidimensional scaling, manifold learning.

Cont.

- 3.2 **Classification**: predicting into buckets
- 3.2.1 Building a classifier and making predictions

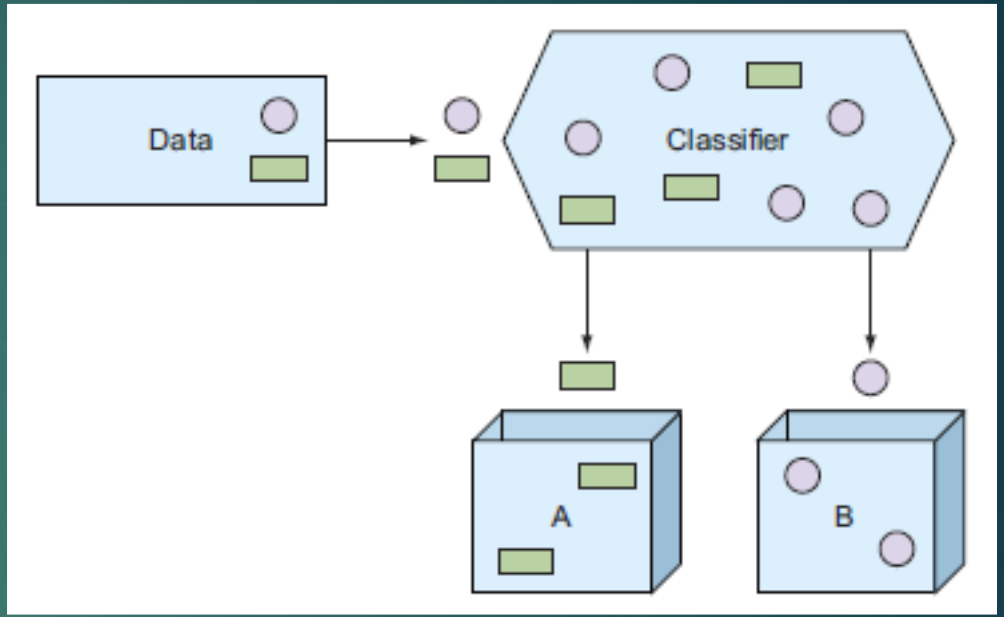
a simple classification algorithm: logistic regression;
For logistic regression, you need to do the following:

1. **Impute** missing values.

2. **Expand** categorical features.

3. From chapter 2, you know that the Fare feature is heavily skewed. In this situation,

it's advantageous (for some ML models) to **transform** the variable to make the feature *distribution* more symmetric and to reduce the potentially harmful impact of *outliers*. Here, you'll choose to transform Fare by taking the square root.



Cont.

- Original dataset

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

- Transformed data set

The first five rows of the Titanic Passengers dataset after processing categorical features and missing values, and transforming the Fare variable by taking the square root

Pclass	Age	SibSp	Parch	sqrt_Fare	Gender = female	Gender = male	Embarked = C	Embarked = Q	Embarked = S
3	22	1	0	2.692582	0	1	0	0	1
1	38	1	0	8.442944	1	0	1	0	0
3	26	0	0	2.815138	1	0	0	0	1
1	35	1	0	7.286975	1	0	0	0	1
3	35	0	0	2.837252	0	1	0	0	1

Cont. (code for logistic regression)


```
from sklearn.linear_model import LogisticRegression as Model

def train(features, target):
  model = Model()
  model.fit(features, target)
  return model

def predict(model, new_features):
  preds = model.predict(new_features)
  return preds

# Assume Titanic data is loaded into titanic_feats,
# titanic_target and titanic_test
model = train(titanic_feats, titanic_target)
predictions = predict(model, titanic_test)
```

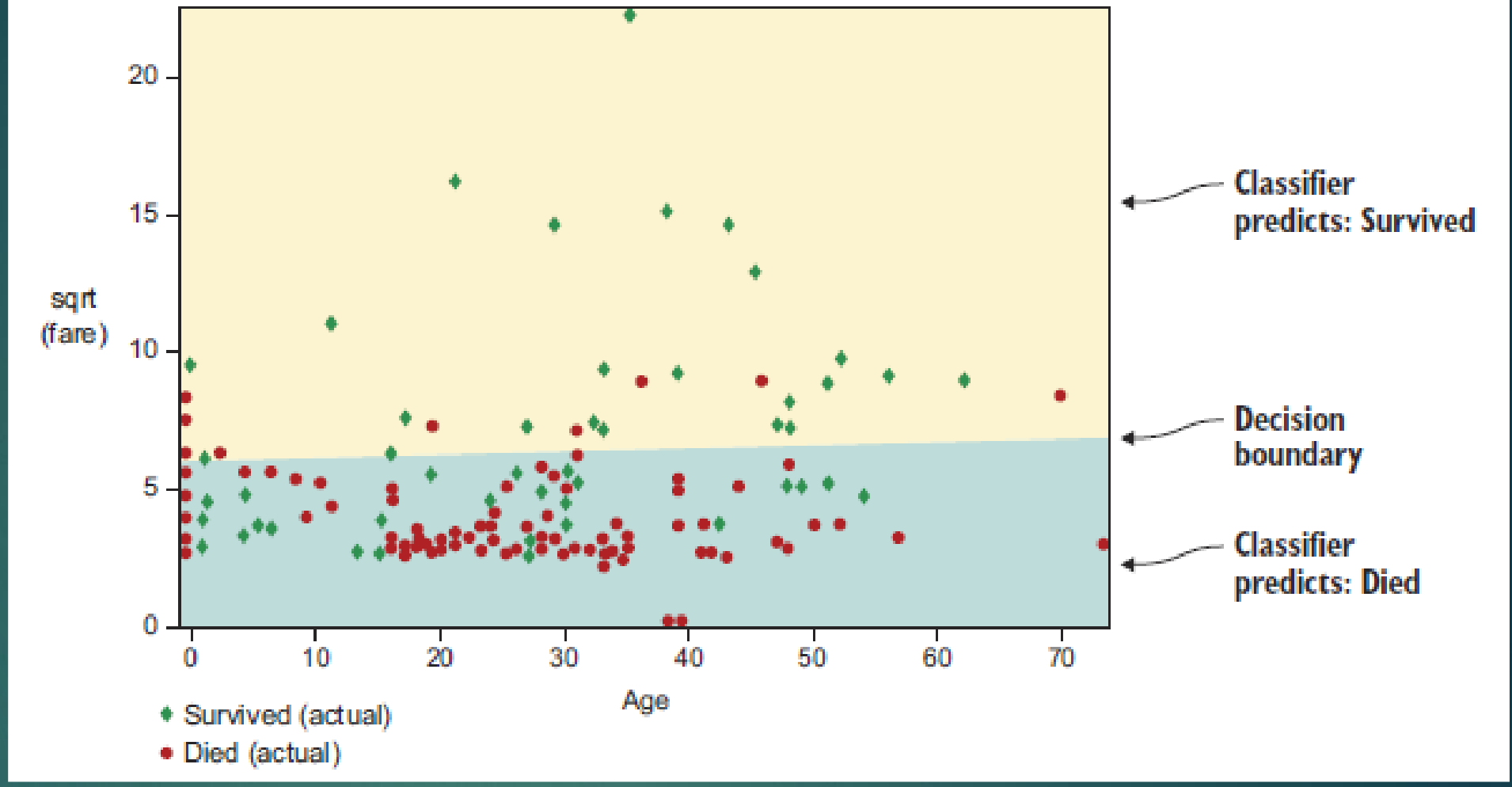
Imports the logistic regression algorithm

Fits the logistic regression algorithm using features and target data

Makes predictions on a new set of features using the model

Returns the model built by the algorithm

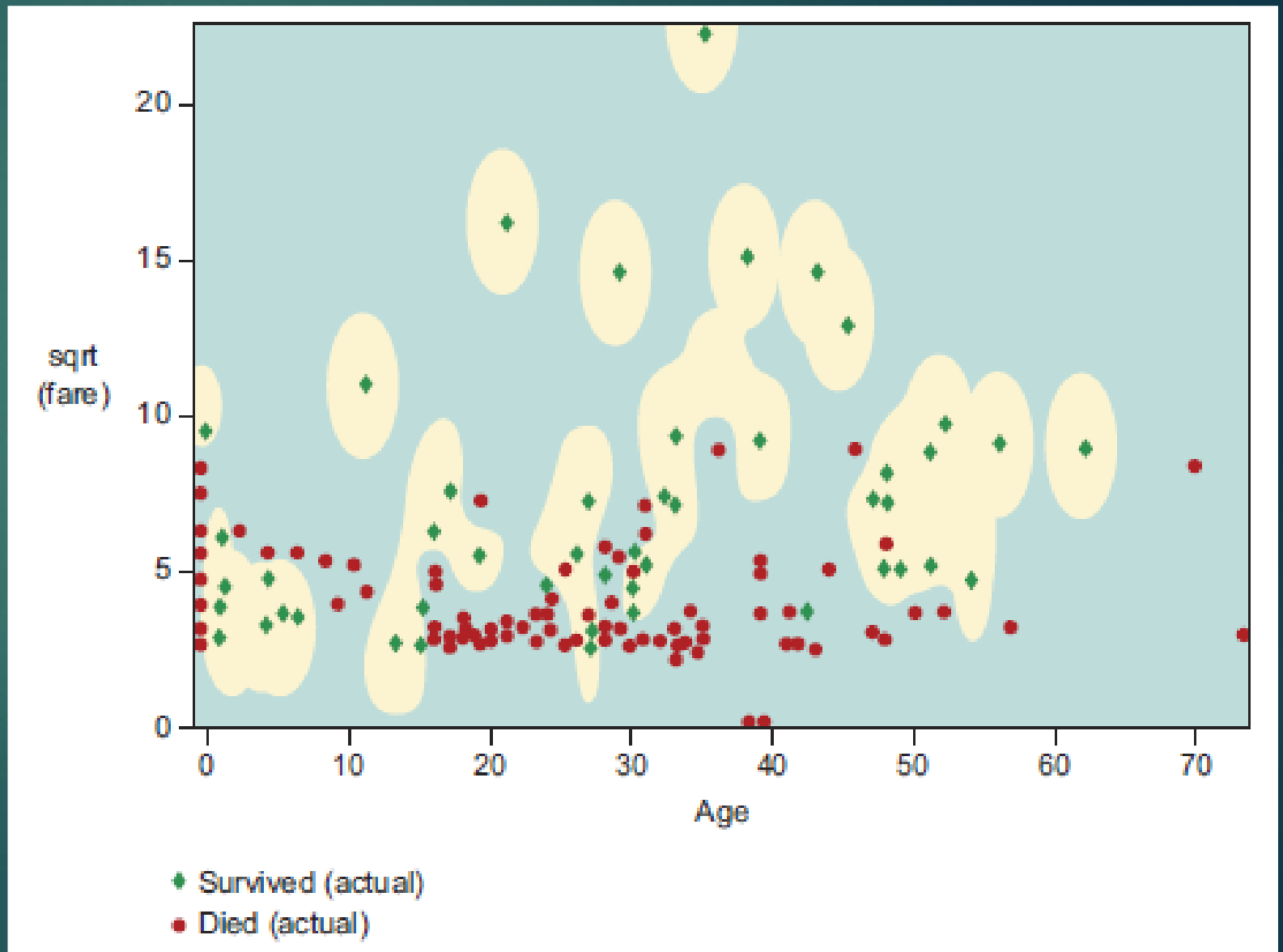
Cont.



- The decision boundary for the Age and sqrt(Fare) features. The **diamonds** show passengers who survived, whereas circles denote passengers who died. The light background denotes the combinations of Age and Fare that are predicted to yield survival. Notice that a few instances overlap the boundary. The classifier isn't perfect!

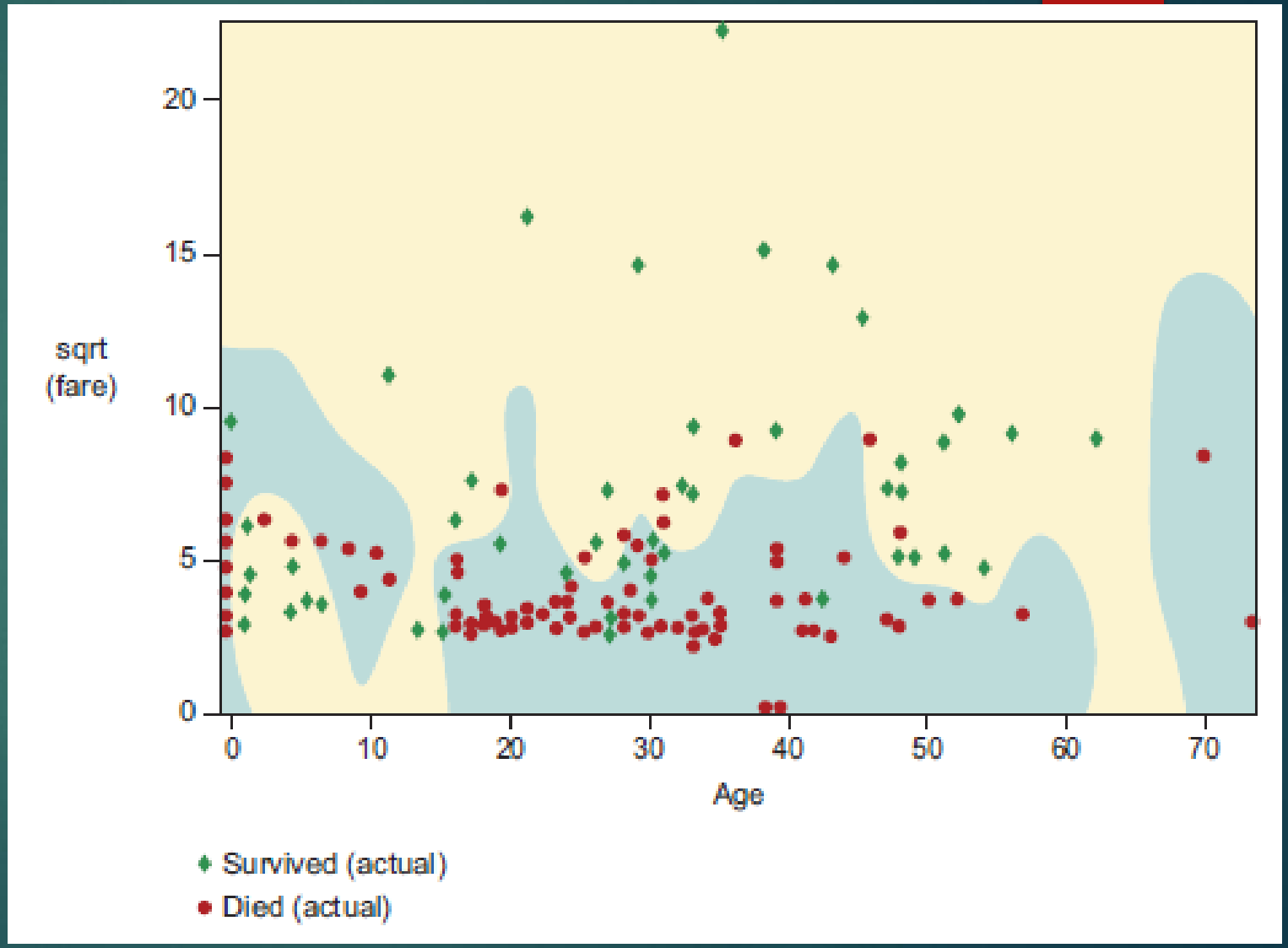
Cont.

- 3.2.2 Classifying complex, nonlinear data (using SVM)
- You can change a single line of code in last listing to use this new algorithm, and the decision boundary is plotted in figure below: from sklearn.svm import SVC as Model

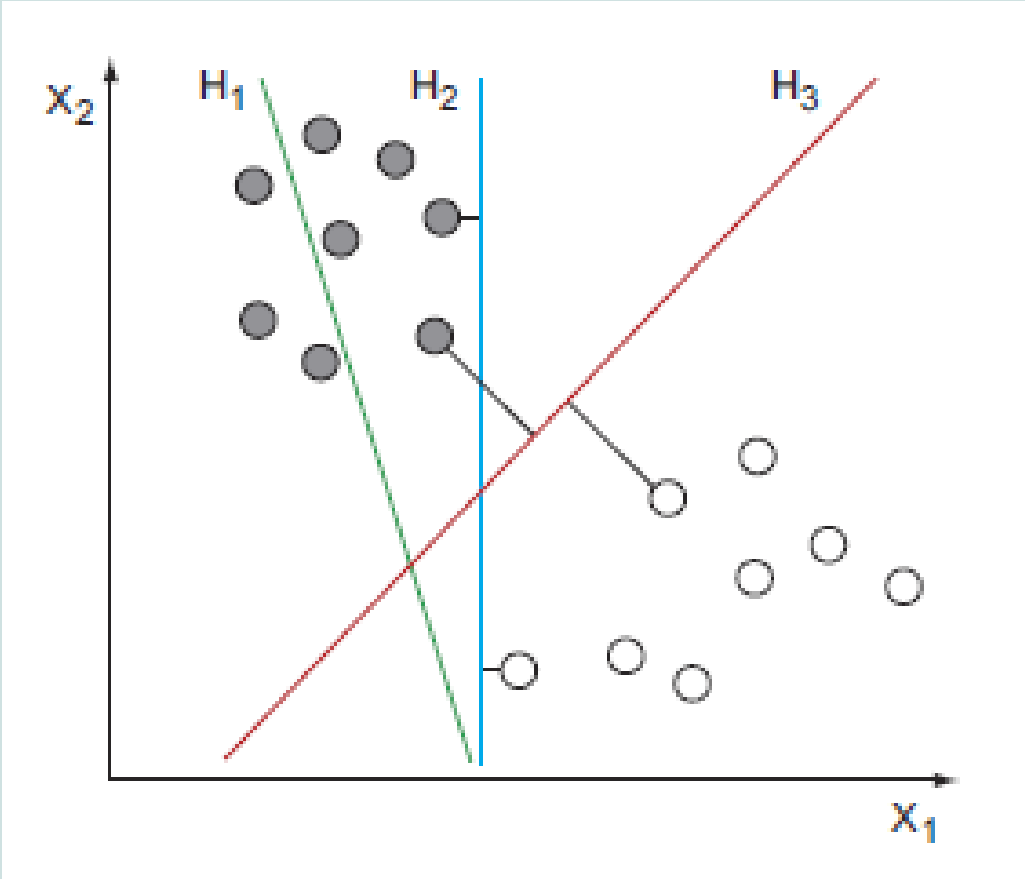


Cont.

- Usually, you can avoid overfitting a nonlinear model by using model parameters built into the algorithm.
- Decision boundary using SVM with gamma = 0.1



Cont. (SVM)



An SVM decision boundary (H_3) is often superior to decision boundaries found by other ML algorithms.

Cont.

- 3.2.3 Classifying with Multiple Classes
- The **images** are 28 × 28 pixels each, but we convert each image into 282 = 784 features, one feature for each pixel. In addition to being a **multiclass** problem, this is also a **high-dimensional** problem. The pattern that the algorithm needs to find is a complex combination of many of these features, and the problem is nonlinear in nature.
- **To build the classifier**, you first choose the algorithm to use from the appendix. The first nonlinear algorithm on the list that natively supports multiclass problems is the **k-nearest neighbors** classifier

```
from sklearn.neighbors import KNeighborsClassifier as Model

def predict_probabilities(model, new_features):
  preds = model.predict_proba(new_features)
  return preds
```

Cont.

- K-NN applied

$$d = \sqrt{n_1^2 + n_2^2 \cdots + n_n^2}.$$

	Actual value	0	1	2	3	4	5	6	7	8	9
Digit 1	7	0	0	0	0.0	0	0.0	0	1	0	0.0
Digit 2	3	0	0	0	0.7	0	0.2	0	0	0	0.1
Digit 3	9	0	0	0	0.0	0	0.0	0	0	0	1.0
Digit 4	5	0	0	0	0.0	0	0.9	0	0	0	0.1

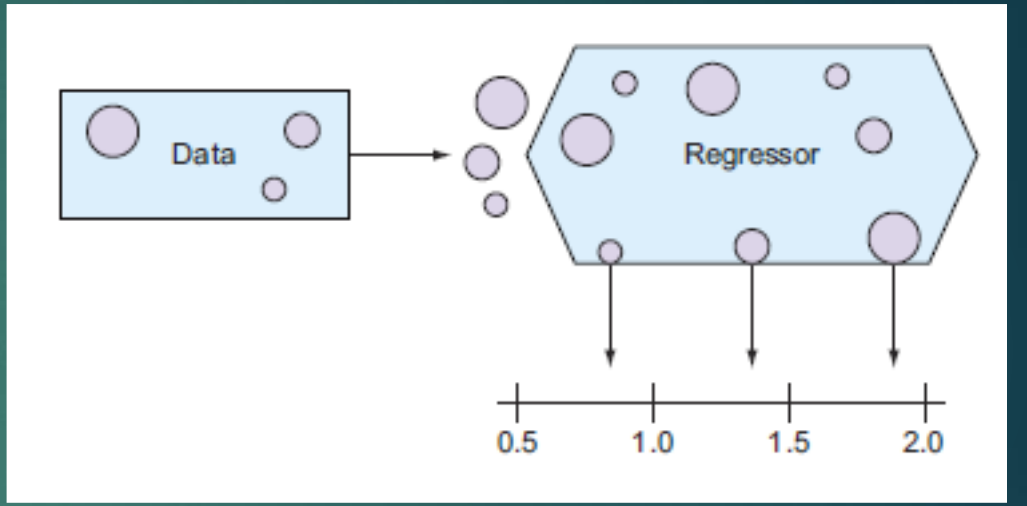
Predicted digit

Digit 2 has a probability of 0.2 of being 5.

Cont.

- 3.3 Regression: predicting numerical values

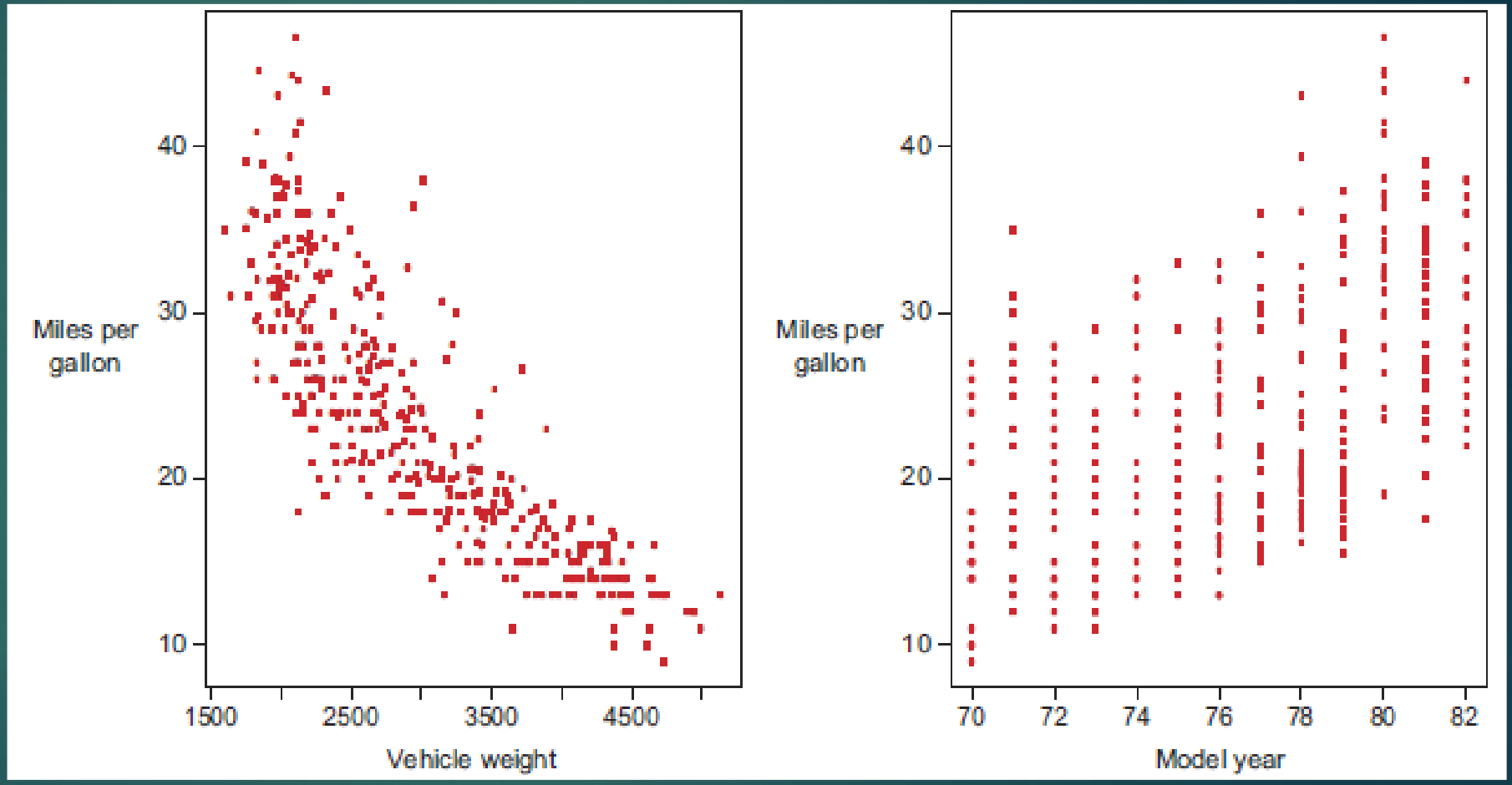
Not every machine-learning problem is about putting records into classes. Sometimes the target variable takes on *numerical* value



	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model/year	Origin
0	18	8	307	130	3504	12.0	70	1
1	15	8	350	165	3693	11.5	70	1
2	18	8	318	150	3436	11.0	70	1
3	16	8	304	150	3433	12.0	70	1
4	17	8	302	140	3449	10.5	70	1

Cont.

- Initial exploration



Cont.

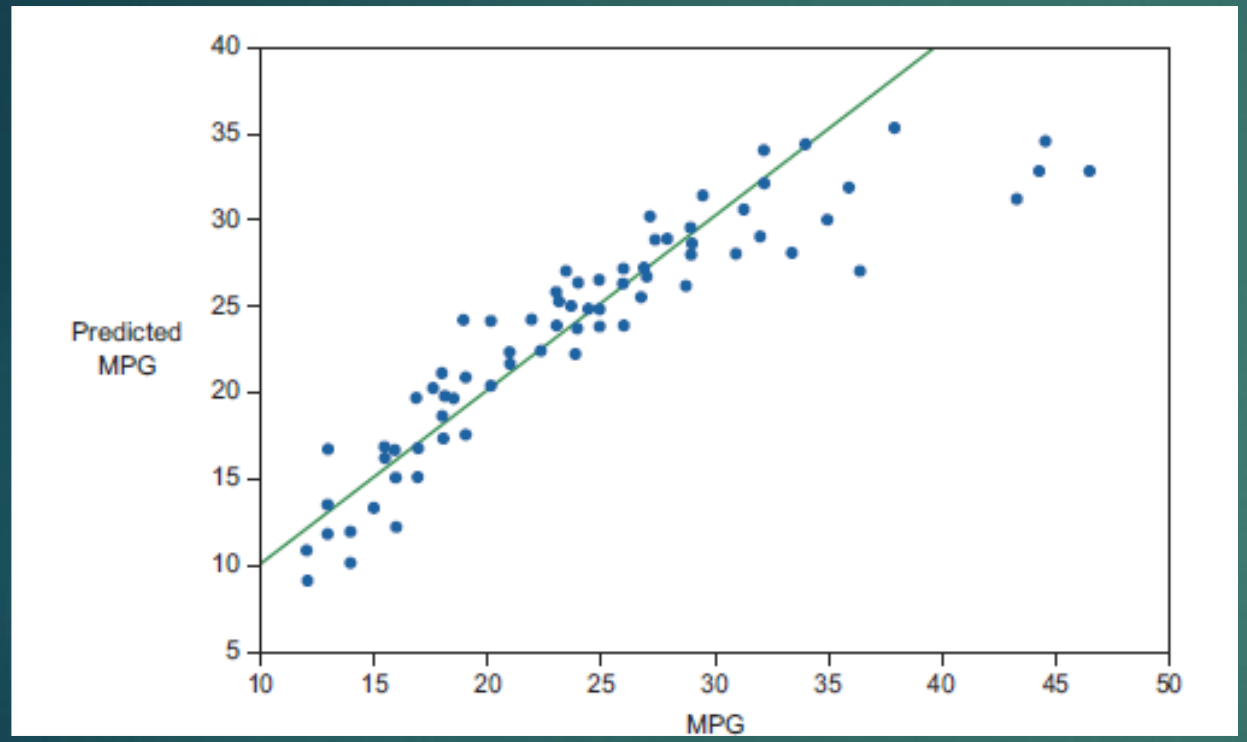
- 3.3.1 Building a regressor and making predictions

You need to (1) impute missing values (none) and (2) expand categorical features (done)

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model/year	Origin = 1	Origin = 2	Origin = 3
387	27	4	140	86	2790	15.6	82	1	0	0
388	44	4	97	52	2130	24.6	82	0	1	0
389	32	4	135	84	2295	11.6	82	1	0	0
390	28	4	120	79	2625	18.6	82	1	0	0
391	31	4	119	82	2720	19.4	82	1	0	0

Cont.

In this example, you train the model on 80% of the data and use the remaining 20% for testing.



A scatter plot of the actual versus predicted values on the held-out test set. The diagonal line shows the perfect linear regressor. The closer all of the predictions are to this line, the better the model.

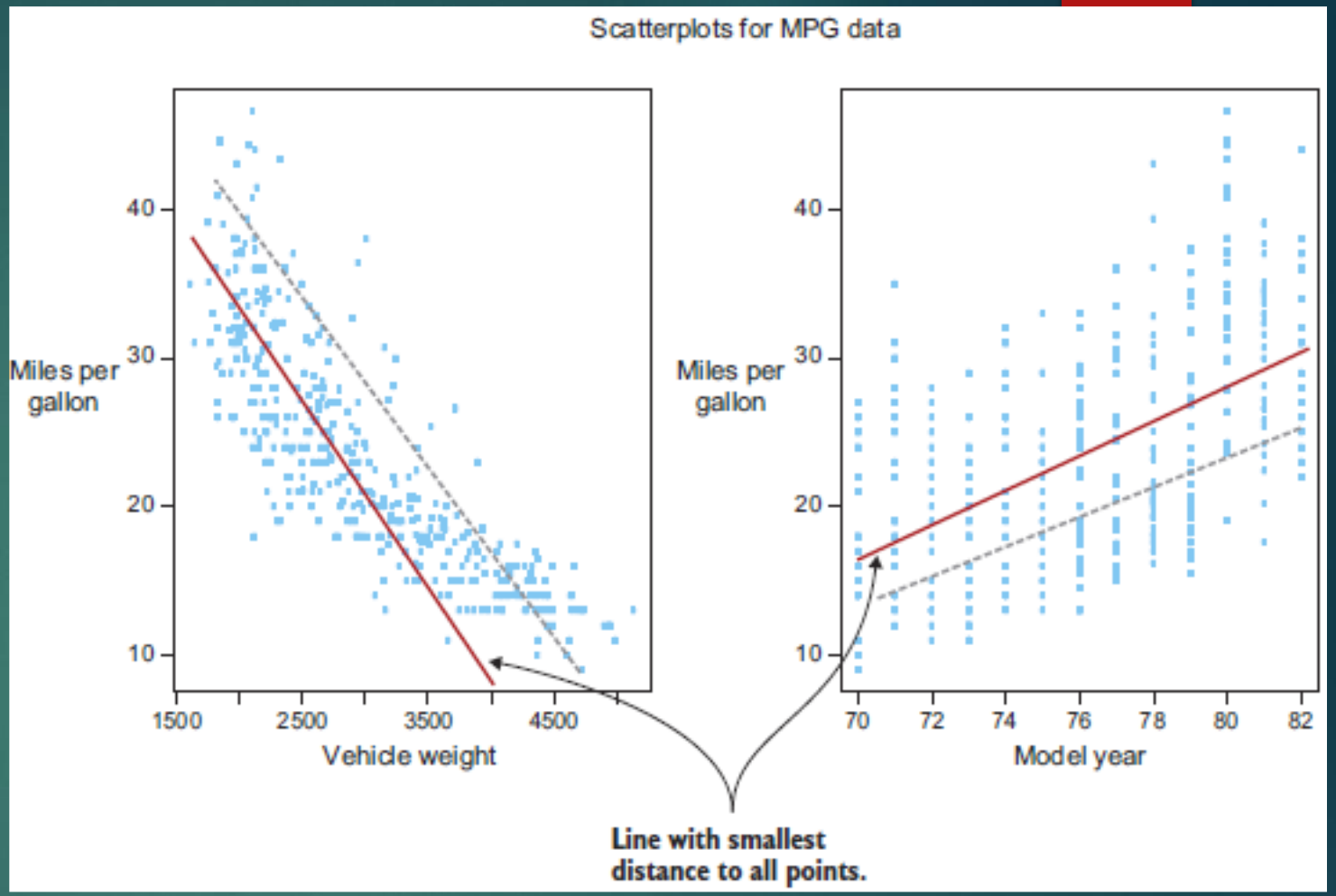
Origin = 1	Origin = 3	Origin = 2	MPG	Predicted MPG
0	0	1	26.0	27.172795
1	0	0	23.8	24.985776
1	0	0	13.0	13.601050
1	0	0	17.0	15.181120
1	0	0	16.9	16.809079

Comparing MPG predictions on a held-out testing set to actual values

Cont.

- Linear Regression

Like logistic regression for classification, linear regression is arguably the simplest and most widely used algorithm for building regression models. The main strengths are linear scalability and a high level of interpretability.



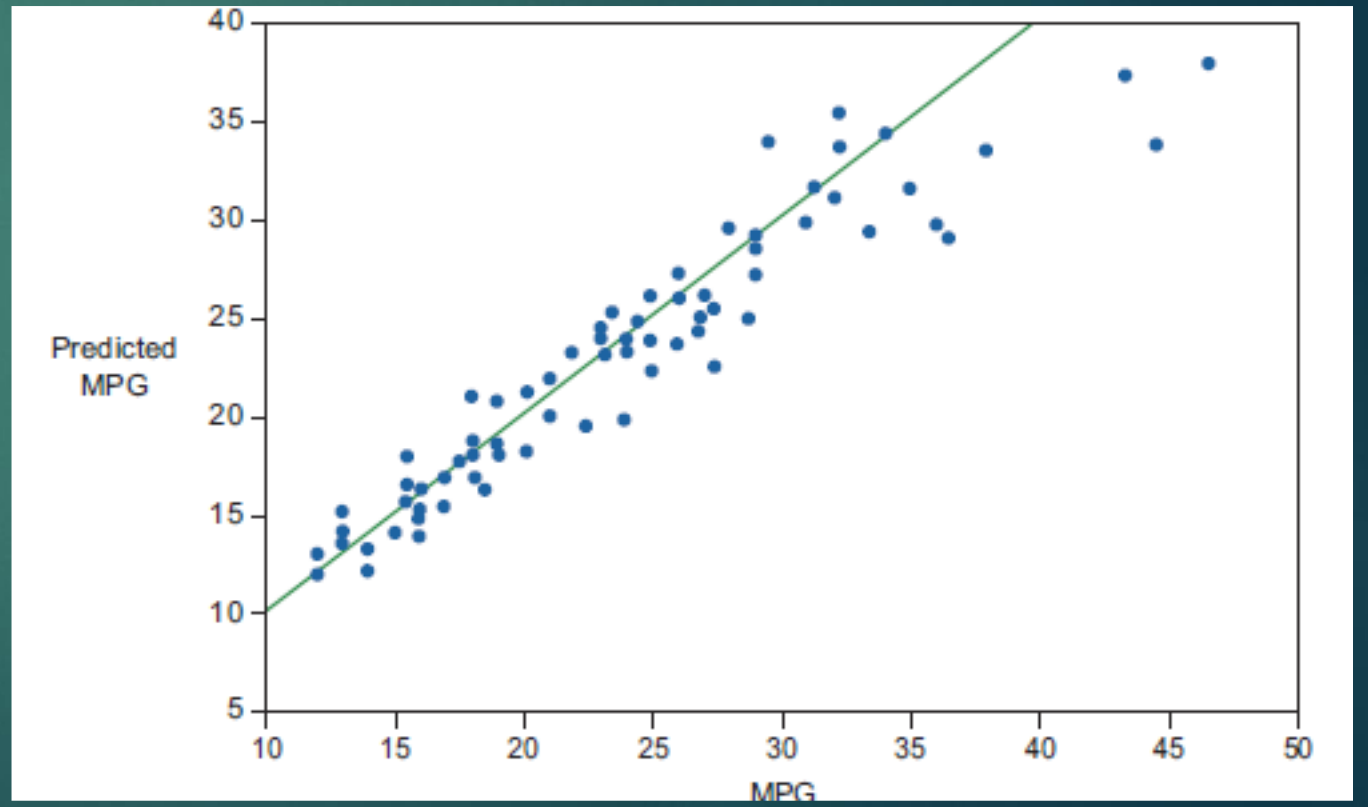
Here, the dark line is the optimal linear regression fitted line on this dataset, yielding a smaller mean-squared deviation from the data to any other possible line (such as the dashed line shown).

Cont.

- 3.3.2 Performing regression on complex, nonlinear data: random forests

- Random forest is a popular method for highly nonlinear problems for which accuracy is important.
- it's easy to use, as it requires minimal preprocessing of data
- Results shown

Origin = 1	Origin = 3	Origin = 2	MPG	Predicted MPG
0	0	1	26.0	27.1684
1	0	0	23.8	23.4603
1	0	0	13.0	13.6590
1	0	0	17.0	16.8940
1	0	0	16.9	15.5060



Cont. (random forests)

- A problem with decision trees is that the top levels of the tree have a huge impact on the answer, and if the new data doesn't follow exactly the same distribution as the training set, the ability to generalize might suffer. This is where the random forest method comes in.
- By building a collection of decision trees, you mitigate this risk. When making the answer, you pick the majority vote in the case of classification, or take the mean in case of regression.
- Because you use votes or means, you can also give back full probabilities in a natural way that not many algorithms share.

Summary

- The purpose of modeling is to describe the relationship between the input features and the target variable.
- There are hundreds of methods for ML modeling. Some are parametric/non-parametric, meaning that the form of the mathematical function relating the features to the target, are fixed in advance/not known.
- Machine-learning methods are further broken into supervised and unsupervised methods. Supervised methods require a training set with a known target
- The two most common problems in supervised learning are classification, in which the target is categorical, and regression, in which the target is numerical.
- You also dove more deeply into the problem of classification. Linear algorithms can define linear decision boundaries between classes, whereas nonlinear methods are required if the data can't be separated linearly.
- In contrast to classification (in which a categorical target is predicted), you predict a numerical target variable in regression models.