Chapter-5

Chapter-5
Implementing an ML Pipeline with JupyterLab – Part-3
(Selecting model, Training, Hosting, and using the Model)

Credit note:
Slides, content, web links, and chapter-end questions are prepared from the AWS Academy course and other Internet resources. Students are advised to use it for study purposes only.

Chapter overview

Topics

1. Predictive and classification methods
2. Hold out method for dataset
3. Train and deployment of the Model with XGBoot and K-NN
4. Train and deploy with AWS SageMaker – Automated way

<u>Labs</u> Included:

- Guided lab-4: Training the Model
- Guided Lab-5: Deploying a Model
- Extra Lab : Training and deploying with Amazon SageMaker

Methods: Predictive and Classification

- A variety of techniques for exploring data and building models have been around for a long time in the world of statistics: linear regression, logistic regression, discriminant analysis, and principal components analysis, for example.
- In comparison with statistics, machine learning deals with large datasets in an open-ended fashion, making it impossible to put strict limits around the question being addressed that classical statistical inference would require.
- As a result, the general approach to machine learning is vulnerable to the danger of overfitting, where a model fits so closely to the available sample of data that it describes not merely structural characteristics of the data but random peculiarities as well.

- We use the term machine learning algorithm to refer to methods that learn directly from data. Some of the selected ML methods are:
  1. Multiple Linear Regression
  2. K-Nearest Neighbors
  3. Naïve Bayes Classifier
  4. Classification tree and regression tree

Methods: Predictive and Classification

Multiple Linear Regression:

- The most popular model for making predictions is the multiple linear regression model encountered in most introductory statistics courses and textbooks.
- This model fits a relationship between a numerical target attribute Y and a set of predictors. $X_1, X_2, \ldots, X_p$
- Attributes are also called variables (particularly in statistics), and other terms for the target attribute include outcome, response, or dependent variable.
- The value of a target attribute is often called a label, meaning it is the target. Predictors also have alternative terms: independent variables, inputs, regressors, or covariates. The assumption is that the following function approximates the relationship between the predictors and target attribute:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon,$$

- where $\beta_0, \ldots, \beta_p$ are coefficients and ε is the noise or unexplained part. Data are then used to estimate the coefficients and to quantify the noise. In predictive modeling, the data are also used to evaluate model performance.

Methods: Predictive and Classification

- Regression modeling means not only estimating the coefficients but also choosing which predictors to include and in what form. For example, a numerical predictor can be included as is, or in logarithmic form [Log (X)], or in a binned form (e.g., age group). Choosing the right form depends on domain knowledge, data availability, and needed predictive power.

- Examples: (Multiple linear regression applies to numerous predictive modeling situations)
  - Predicting customer activity on credit cards from their demographics and historical activity patterns.
  - Predicting expenditures on vacation travel based on historical frequent flyer data.
  - Predicting staffing requirements at help desks based on historical data and product and sales information.
  - Predicting sales from cross-selling of products from historical information, and predicting the impact of discounts on sales in retail outlets.

Methods: Predictive and Classification

- Once we determine the predictors to include and their form, we estimate the coefficients of the regression formula from the data using a method called ordinary least squares (OLS). This method finds values $\widehat{\beta}_0, \widehat{\beta}_1, \widehat{\beta}_2, \ldots, \widehat{\beta}_p$ that minimizes the sum of squared deviations between the actual target values (Y) and their predicted values based on that model ($\hat{Y}$).
- To predict the value of the target for a record with predictor values, $x_1, x_2, \ldots, x_p$ we use the equation

$$\hat{Y} = \widehat{\beta}_0 + \widehat{\beta}_1 x_1 + \widehat{\beta}_2 x_2 + \cdots + \widehat{\beta}_p x_p.$$

- Predictions based on this equation are the best predictions possible in the sense that they will be unbiased (equal to the true values on average) and will have the smallest mean squared error compared with any unbiased estimates if we make the following assumptions:
    1. The noise ε (or equivalently, Y) follows a normal distribution.
    2. The choice of predictors and their form is correct (linearity).
    3. The records are independent of each other.
    4. The variability in the target values for a given set of predictors is the same regardless of the values of the predictors (homoskedasticity).

Methods: Predictive and Classification

- Example: Predicting the Price of Used Toyota Corolla Cars.

- The file ToyotaCorolla.csv contains data on used cars (Toyota Corolla) on sale during the late summer of 2004 in the Netherlands. It has 1436 records containing details on 38 attributes, including Price, Age, Kilometers, HP, and other specifications. The goal is to predict the price of a used Toyota Corolla based on its specifications. (The following is a sample subset of this dataset. Next slide)

- Tasks:
  - Split the data into training (50%), validation (30%), and holdout (20%) datasets.
  - Run a multiple linear regression with the target attribute Price and predictors Age_08_04, KM, Fuel_Type, HP, Automatic, Doors, Quarterly_Tax, Mfr_Guarantee, Guarantee_Period, Airco, Automatic_airco, CD_Player, Powered_Windows, Sport_Model, and Tow_Bar.
  - What appear to be the three or four most important car specifications for predicting the car's price?
  - Using metrics you consider useful, assess the performance of the model in predicting prices.

Methods: Predictive and Classification

- The following is a sample subset of this dataset.

	Age_		Fuel_		Met_	Auto-			Quarterly_	
Price	**08_04**	**KM**	**Type**	**HP**	**Color**	**matic**	**CC**	**Doors**	**tax**	**Weight**
13,500	23	46,986	*Diesel*	90	1	0	2000	3	210	1165
13,750	23	72,937	*Diesel*	90	1	0	2000	3	210	1165
13,950	24	41,711	*Diesel*	90	1	0	2000	3	210	1165
14,950	26	48,000	*Diesel*	90	0	0	2000	3	210	1165
13,750	30	38,500	*Diesel*	90	0	0	2000	3	210	1170
12,950	32	61,000	*Diesel*	90	0	0	2000	3	210	1170
16,900	27	94,612	*Diesel*	90	1	0	2000	3	210	1245
18,600	30	75,889	*Diesel*	90	1	0	2000	3	210	1245
21,500	27	19,700	*Petrol*	192	0	0	1800	3	100	1185
12,950	23	71,138	*Diesel*	69	0	0	1900	3	185	1105
20,950	25	31,461	*Petrol*	192	0	0	1800	3	100	1185
19,950	22	43,610	*Petrol*	192	0	0	1800	3	100	1185
19,600	25	32,189	*Petrol*	192	0	0	1800	3	100	1185
21,500	31	23,000	*Petrol*	192	1	0	1800	3	100	1185
22,500	32	34,131	*Petrol*	192	1	0	1800	3	100	1185

Methods: Predictive and Classification

k-Nearest Neighbors (k-NN):

- The k-nearest neighbors (k-NN) algorithm can be used for classification (of a categorical label) or prediction (of a numerical label). To classify or predict a new record, the method relies on finding “similar” records in the training data. These “neighbors” are then used to derive a classification or prediction for the new record by voting (for classification) or averaging (for prediction).

- The k-NN Classifier: The idea in k-NN methods is to identify k records in the training dataset that are similar to a new record that we wish to classify. We then use these similar (neighboring) records to classify the new record into a class, assigning the new record to the predominant class among these neighbors. Denote the values of the predictors for this new record by $x_1, x_2, \ldots, x_p$.
- We look for records in our training data that are similar to or “near” the record to be classified in the predictor space (i.e., records that have values close to $x_1, x_2, \ldots, x_p$).
- Then, based on the classes to which those proximate records belong, we assign a class to the record that we want to classify.

Methods: Predictive and Classification

- *Determining Neighbors*: The k-NN algorithm is a classification method that does not make assumptions about the form of the relationship between the class membership (Y) and the $x_1, x_2, \ldots, x_p$ predictors.
- This is a nonparametric method because it does not involve the estimation of parameters in an assumed function form, such as the linear form assumed in linear regression. Instead, this method draws information from similarities between the predictor values of the records in the dataset.
- A central question is how to measure the distance between records based on their predictor values. The most popular measure of distance is the Euclidean distance. The Euclidean distance between two records $(x_1, x_2, \ldots, x_p)$ and $(u_1, u_2, \ldots, u_p)$ is

$$\sqrt{(x_1 - u_1)^2 + (x_2 - u_2)^2 + \cdots + (x_p - u_p)^2}.$$

- However, the k-NN algorithm relies on many distance computations (between each record to be predicted and every record in the training set), and therefore the Euclidean distance, which is computationally cheap, is the most popular in k-NN.
- To equalize the scales that the various predictors may have, (in most cases), predictors should first be standardized before computing a Euclidean distance.

Methods: Predictive and Classification

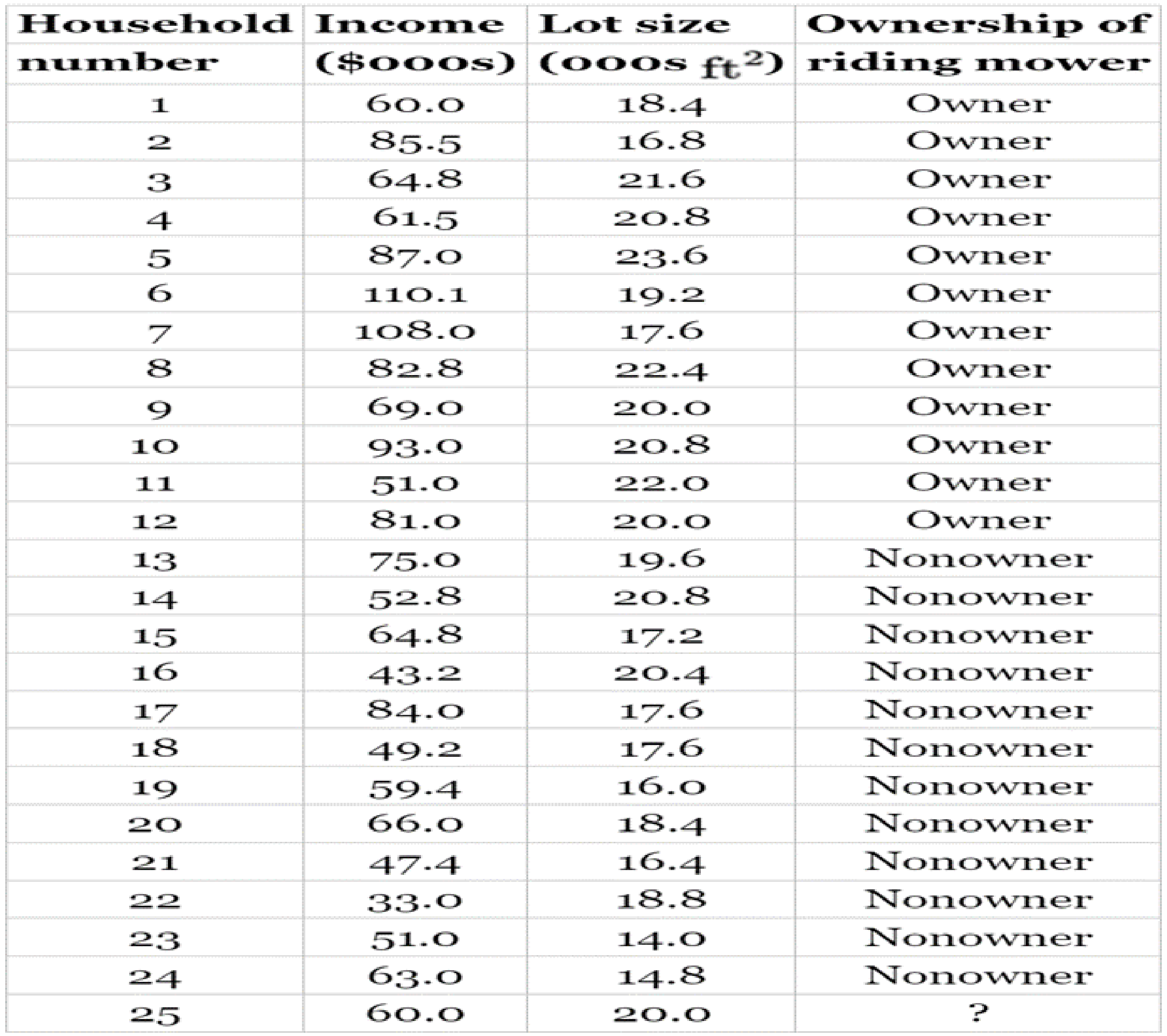
- *Classification rule:* After computing the distances between the record to be classified and existing records, we need a rule to assign a class to the record to be classified, based on the classes of its neighbors. The simplest case is, k = 1, where we look for the closest record (the nearest neighbor) and classify the new record as belonging to the same class as its closest neighbor.
- It is a remarkable fact that this simple, intuitive idea of using a single nearest neighbor to classify records can be very powerful when we have a large number of records in our training set. It turns out that the misclassification error of the 1-nearest neighbor scheme has a misclassification rate that is no more than twice the error when we know exactly the probability density functions for each class.

- The idea of the 1-nearest neighbor can be extended to k > 1 neighbors as follows:
  1. Find the nearest k neighbors to the record to be classified.
  2. Use a majority decision rule to classify the record, where the record is classified as a member of the majority class of the k neighbors.

Methods: Predictive and Classification

- Example: Classifying families those likely to purchase a "riding-mower".

- A riding mower manufacturer would like to find a way of classifying families in a city into those likely to purchase a riding mower and those not likely to buy one. A pilot random sample is undertaken of 12 owners and 12 nonowners in the city. The data are shown in the below Table.



Household number	Income ($000s)	Lot size (000s ft^2)	Ownership of riding mower
1	60.0	18.4	Owner
2	85.5	16.8	Owner
3	64.8	21.6	Owner
4	61.5	20.8	Owner
5	87.0	23.6	Owner
6	110.1	19.2	Owner
7	108.0	17.6	Owner
8	82.8	22.4	Owner
9	69.0	20.0	Owner
10	93.0	20.8	Owner
11	51.0	22.0	Owner
12	81.0	20.0	Owner
13	75.0	19.6	Nonowner
14	52.8	20.8	Nonowner
15	64.8	17.2	Nonowner
16	43.2	20.4	Nonowner
17	84.0	17.6	Nonowner
18	49.2	17.6	Nonowner
19	59.4	16.0	Nonowner
20	66.0	18.4	Nonowner
21	47.4	16.4	Nonowner
22	33.0	18.8	Nonowner
23	51.0	14.0	Nonowner
24	63.0	14.8	Nonowner
25	60.0	20.0	?

Methods: Predictive and Classification

Tasks:

- We first partition the data into training data (14 households) and validation data (10 households). This dataset is too small for partitioning, which can result in unstable results, but we will continue with this partitioning for illustration purposes.

- Now consider a new household with $60,000 income and lot size 20,000 ft2 (also shown in table). Among the households in the training set, the one closest to the new household (in Euclidean distance after normalizing income and lot size) is household 4, with $61,500 income and 20,800 ft2 lot size. If we use a 1-NN classifier, we would classify the new household as an owner, like household 4.
- If we use, k = 3, the three nearest households are 4, 9, and 14. Two of these neighbors are owners of riding mowers, and one is a nonowner. The majority vote is therefore Owner, and the new household would be classified as an owner.

Methods: Predictive and Classification

Tasks:

- *Choosing the value of k*: The advantage of choosing k > 1 is that higher values of k provide smoothing that reduces the risk of overfitting due to noise in the training data. In the extreme, k = n, the number of records in the training dataset, then this is a case of over-smoothing in the absence of useful information in the predictors. We would choose k = 3, which is the lowest k value that maximizes our accuracy in the validation set.
- *Setting the Threshold value*: k-NN uses a majority decision rule to classify a new record, where the record is classified as a member of the majority class of the k neighbors. with k = 5, it turns out that the five nearest neighbors to the new household (with income = $60,000 and lot size = 20,000 ft2) are households 4, 9, 14, 1, and 3. Since four of these are owners and one is a nonowner, we can estimate for the new household a probability of 0.8 of being an owner (and 0.2 for being a nonowner). Using a simple majority rule is equivalent to setting the threshold value to 0.5.

Methods: Predictive and Classification

Naive Bayes Classifier:

- The naive Bayes classifier can be applied to data with categorical predictors. The naive Bayes method (entire branch of statistics) is named after the Reverend Thomas Bayes (1702–1761). To understand the naive Bayes classifier, we first look at the concept of conditional probabilities and then present the complete, or exact, Bayesian classifier.
- The basic principle is simple, which is more generally applicable. For each record to be classified:
  1. Find all the other records with the same predictor profile (i.e., where the predictor values are the same).
  2. Determine what classes the records belong to and which class is most prevalent.
  3. Assign that class to the new record.

- Alternatively (or in addition), it may be desirable to tweak the method so that it answers the question: "What is the propensity of belonging to the class of interest?" instead of "Which class is the most probable?" Obtaining class probabilities allows using a sliding threshold to classify a record as belonging to class C_i even if C_i is not the most probable class for that record.

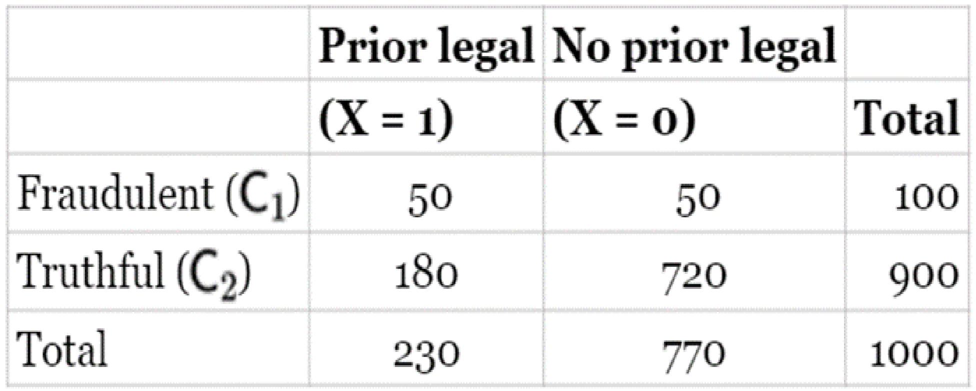
Methods: Predictive and Classification

- *Threshold Probability Method*
  1. Establish a threshold probability for the class of interest above which we consider that a record belongs to that class.
  2. Find all the training records with the same predictor profile as the new record (i.e., where the predictor values are the same).
  3. Determine the probability that those records belong to the class of interest.
  4. If that probability is above the threshold probability, assign the new record to the class of interest.
- *Conditional Probability*
- Both procedures incorporate the concept of conditional probability, or the probability of event A given that event B has occurred [denoted $P(A|B)$]. In this case, we will be looking at the probability of the record belonging to class C_i given that its predictor values are $x_1, x_2, \ldots, x_p$.
  In general, for a response with m classes $C_1, C_2, \ldots, C_m$ the predictor values $x_1, x_2, \ldots, x_p$, we want to compute $P(C_i|x_1, \ldots, x_p)$.
- To classify a record, we compute its probability of belonging to each of the classes in this way and then classify the record to the class that has the highest probability or use the threshold probability to decide whether it should be assigned to the class of interest.

Methods: Predictive and Classification

- Example: Predicting Fraudulent Financial Reporting.
- An accounting firm has many large companies as customers. Each customer submits an annual financial report to the firm, which is then audited by the accounting firm. For simplicity, we will designate the outcome of the audit as “fraudulent” or “truthful,” referring to the accounting firm's assessment of the customer's financial report. (The accounting firm has a strong incentive to be accurate in identifying fraudulent reports—if it passes a fraudulent report as truthful, it would be in legal trouble.)
- In this case, each customer is a record, and the class label of interest, Y = {fraudulent, truthful}, has two classes into which a company can be classified: $C_1 = fraudulent$ and $C_2 = truthful$ The predictor attribute—“prior legal trouble”—has two values: 0 (no prior legal trouble) and 1 (prior legal trouble).

- The accounting firm has data on 1500 companies that it has investigated in the past. For each company, it has information on whether the financial report was judged fraudulent or truthful and whether the company had prior legal trouble. The data were partitioned into a training set (1000 firms) and a holdout set (500 firms). Counts in the training set are shown in PIVOT Table:



	Prior legal	**No prior legal**	
	(X = 1)	**(X = 0)**	**Total**
Fraudulent (C_1)	50	50	100
Truthful (C_2)	180	720	900
Total	230	770	1000

Methods: Predictive and Classification

- Now consider the financial report from a new company, which we wish to classify as either fraudulent or truthful by using these data. To do this, we compute the probabilities, as above, of belonging to each of the two classes. If the new company had prior legal trouble, the probability of belonging to the fraudulent class would be P(fraudulent | prior legal) = 50/230 (of the 230 companies with prior legal trouble in the training set, 50 had fraudulent financial reports). The probability of belonging to the other class, “truthful,” is, of course, the remainder = 180/230.
- In this example, we are more interested in identifying the fraudulent reports. To identify the fraudulent reports, our approach is to establish a threshold value for the probability of being fraudulent and classify all records above that value as fraudulent. The Bayesian formula for the calculation of this probability that a record belongs to a class C_i is as follows:

$$P(C_i|x_1,\ldots,x_p) = \frac{P(x_1,\ldots,x_p|C_i)P(C_i)}{P(x_1,\ldots,x_p|C_1)P(C_1)+\cdots+P(x_1,\ldots,x_p|C_m)P(C_m)}.$$

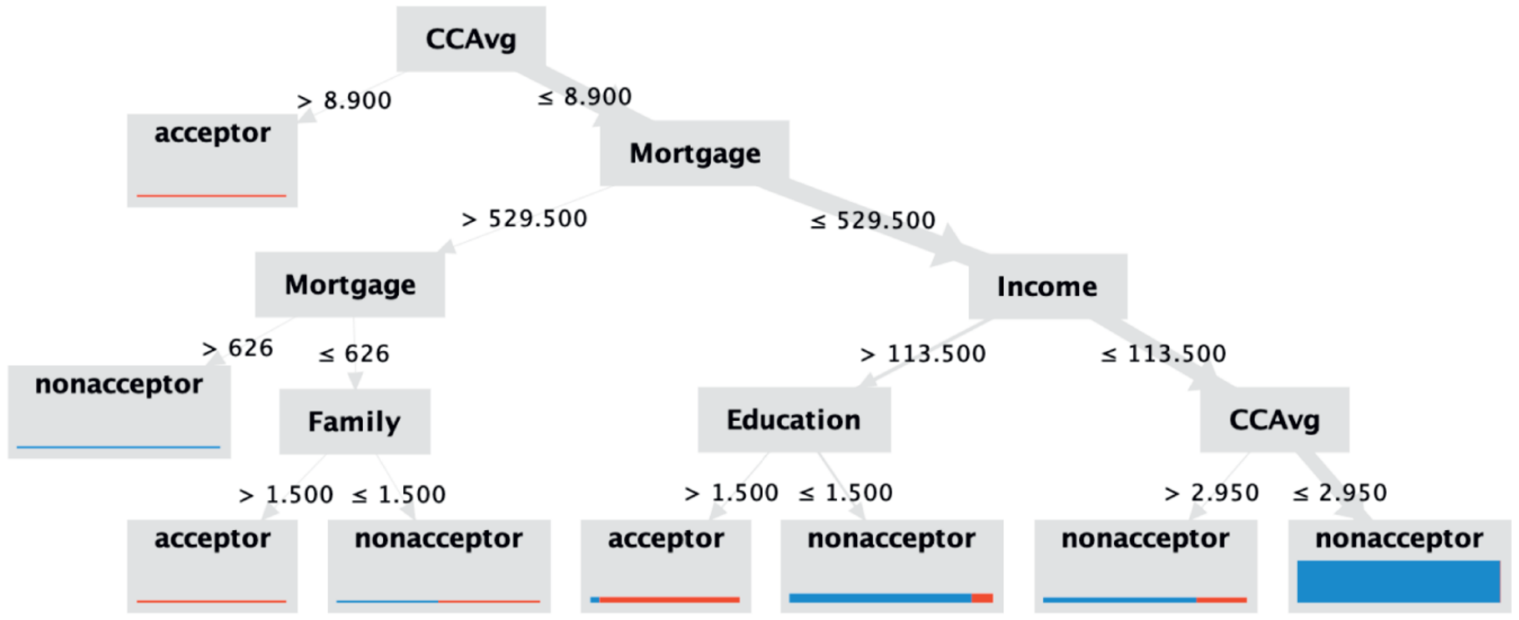
- In this example (where frauds are rarer), if the threshold were established at 0.20, we would classify a prior legal trouble record as fraudulent because P(fraudulent | prior legal) = 50/230 = 0.22.
- The user can treat this threshold as a “slider” to be adjusted to optimize performance, like other parameters in any classification model.

Methods: Predictive and Classification

Classification and Regression Trees:

- This flexible data-driven method can be used for both classification (called classification tree) and prediction (called regression tree). Among the data-driven methods, trees (also known as decision trees) are the most transparent and easy to interpret. Trees are based on separating records into subgroups by creating splits on predictors. These splits create logical rules that are transparent and easily understandable, for example, "IF Age < 55 AND Education > 12 THEN class = 1." The resulting subgroups should be more homogeneous in terms of the outcome variable, thereby creating useful prediction or classification rules.
- The two key ideas underlying trees are 1) recursive partitioning (for constructing the tree) and 2) pruning (for cutting the tree back). In the context of tree construction, we also describe a few metrics of homogeneity that are popular in tree algorithms, for determining the homogeneity of the resulting subgroups of records. (Note: The limiting tree size is a useful strategy for avoiding overfitting and showing how it is done, and alternative strategies for avoiding overfitting.)
- The trees require large amounts of data. However, once constructed, they are computationally cheap to deploy on large samples. They also have other advantages such as being highly automated, robust to outliers, and able to handle missing values. We need to see how trees can be used for dimension reduction. The important other concepts that improve predictive power are random forests and boosted trees, which need to be discussed and understood as well.

Methods: Predictive and Classification



Tree

```
CCAvg > 8.900: acceptor {nonacceptor=0, acceptor=6}
CCAvg ≤ 8.900
|   Mortgage > 529.500
|   |   Mortgage > 626: nonacceptor {nonacceptor=1, acceptor=0}
|   |   Mortgage ≤ 626
|   |   |   Family > 1.500: acceptor {nonacceptor=0, acceptor=8}
|   |   |   Family ≤ 1.500: nonacceptor {nonacceptor=1, acceptor=1}
|   Mortgage ≤ 529.500
|   |   Income > 113.500
|   |   |   Education > 1.500: acceptor {nonacceptor=12, acceptor=184}
|   |   |   Education ≤ 1.500: nonacceptor {nonacceptor=349, acceptor=41}
|   |   Income ≤ 113.500
|   |   |   CCAvg > 2.950: nonacceptor {nonacceptor=128, acceptor=42}
|   |   |   CCAvg ≤ 2.950: nonacceptor {nonacceptor=2221, acceptor=6}
```

- The tree methodology developed by Breiman et al. (1984), also called decision trees, can be used for both classification and prediction.
- The program that Breiman et al. created to implement these procedures was called CART (Classification And Regression Trees).
- *What is a classification tree*? Figure shows a tree for classifying bank customers who receive a loan offer as either acceptors or nonacceptors, based on information such as their income, education level, and average credit card expenditure.

Methods: Predictive and Classification

- *Tree Structure*
- We have two types of nodes in a tree: decision (= splitting) nodes and leaf nodes. Nodes that have successors are called decision nodes because if we were to use a tree to classify a new record for which we knew only the values of the predictor attributes, we would “drop” the record down the tree so that at each decision node, the appropriate branch is taken until we get to a node that has no successors. Such nodes are called the leaf nodes (or terminal nodes or leaves of the tree) and represent the partitioning of the data by predictors.

- The Decision Trees, numerical predictors create binary splits whereas categorical predictors can create as many splits as the predictor has categories. If the tree has only binary splits at each decision node, then such a tree has the property that the number of leaf nodes is exactly one more than the number of decision nodes.
-

Methods: Predictive and Classification

- *Decision Rules*
- Consider the tree in Figure, the leaf nodes are marked acceptor or nonacceptor based on the tree's classification for records in that node. The splitting node lists the predictor name and the left and right arrows from the splitting node give the condition for the split (e.g., CCAvg <= 8.9 from the top node to the right child node.) The colored bar within each leaf node gives a visual indication of the proportion of the two classes in that node. The text description available for each tree (see bottom of Figure) provides the counts of the two classes in each node. This tree can easily be translated into a set of rules for classifying a bank customer. For example, the bottom-left node under the "Family" decision node in this tree gives us the following rule:

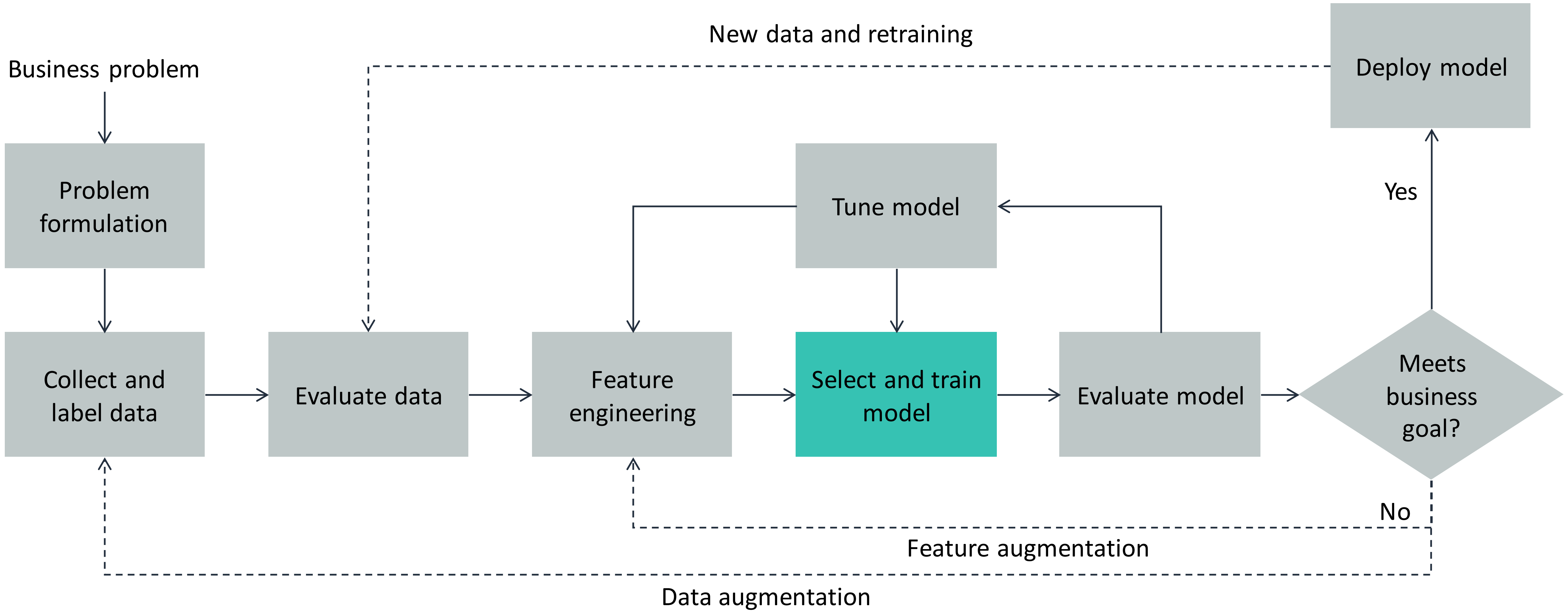
$$\text{IF}(CCAvg \leq 8.9) \text{ AND } (Mortgage > 29.5)$$
$$\text{AND } (Mortgage \leq 626) \text{ AND } (Family > 1.5)$$
$$\text{THEN } Class = \text{acceptor}.$$

- *Classifying a New Record*
- To classify a new record, it is "dropped" down the tree. When it has dropped down to a leaf node, we can assign its class simply by taking a "vote" (or average, if the outcome is numerical) of all the training data that belonged to the leaf node when the tree was grown. The class with the highest vote is assigned to the new record. For instance, a new record reaching the rightmost leaf node in Figure, which has a majority of records that belong to the nonacceptor class, would be classified as "nonacceptor."

Methods: Predictive and Classification

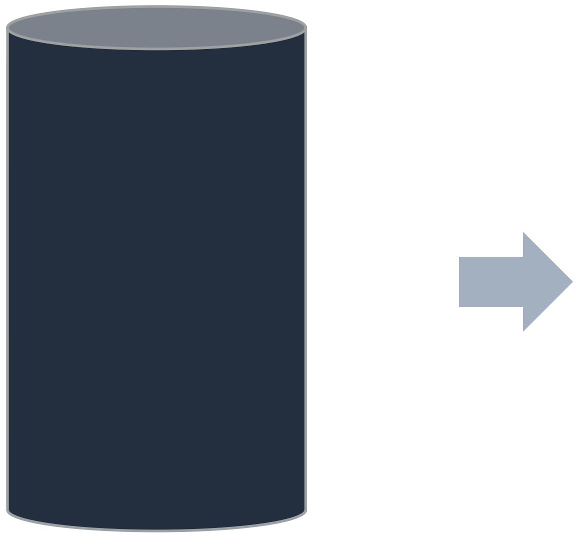
- *Recursive Partitioning:*
- The key idea underlying tree construction is recursive partitioning of the space of the predictor attributes. A second important issue is avoiding over-fitting.
- Let us denote the target attribute by Y and the predictors by $X_1, X_2, X_3, \ldots, X_p$. In classification, the target will be a categorical attribute (nominal type), sometimes called a class label. Specifically, in binary classification, the class label will be of binominal type. Recursive partitioning divides up the p-dimensional space of the X predictors into nonoverlapping multidimensional rectangles.
- The predictor attributes here are considered to be continuous (numeric type) or categorical (nominal type). This division is accomplished recursively (i.e., operating on the results of prior divisions). First, one of the predictor attributes is selected, say X_i , and a value of X_i , say s_i , is chosen to split the p-dimensional space into two parts: one part that contains all the points with $X_i < s_i$ and the other with all the points with $X_i \geq s_i$. Then, one of these two parts is divided similarly by again choosing a predictor attribute (it could be X_i or another attribute) and a split value for that attribute.

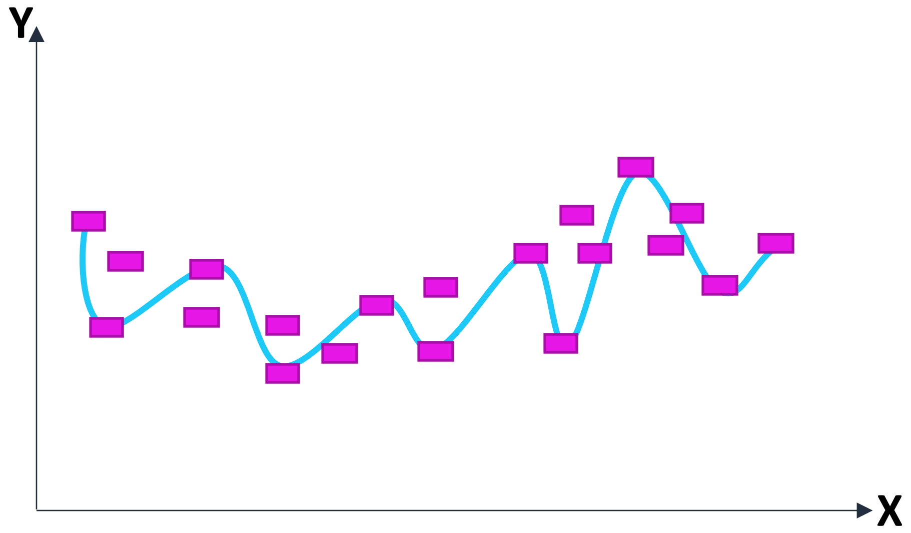
Machine learning pipeline



Why split the data?

- All data that is used for training and evaluation



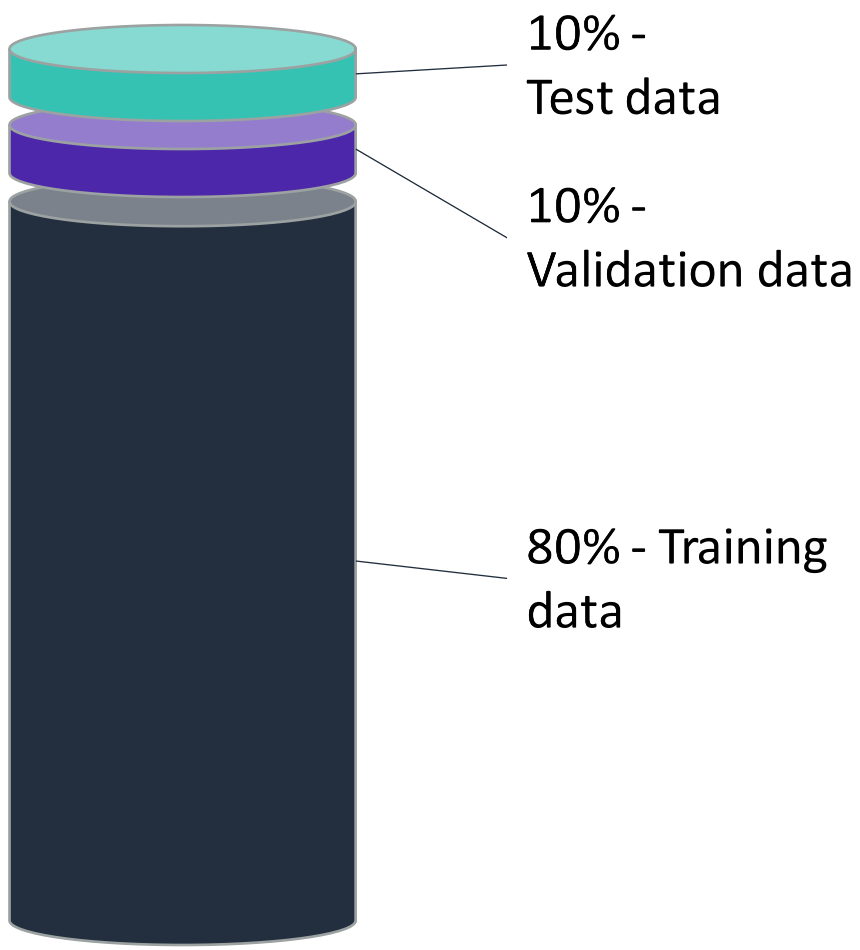


Overfitting

- Evaluating a model with the same data that it trained on can lead to overfitting.
- Overfitting is where your model learns the particulars of a dataset too well.
- It essentially memorizes the training data, instead of learning the relationships between features and labels.

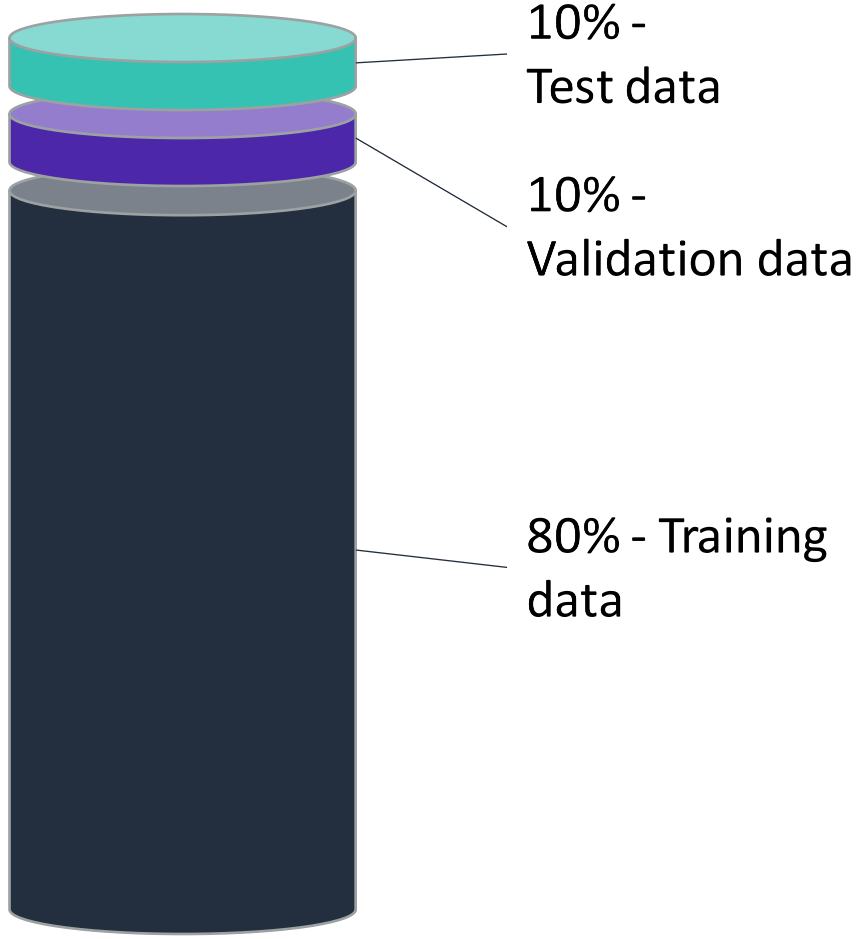
Thus, the model can’t use those relationships and patterns to apply to new data in the future.

Holdout method



- Hold-out is when you split your data into multiple sets.
- These sets are usually training data, validation data, and testing data.
- Training data, which includes both features and labels, feed into the algorithm that you selected to produce your model.
- The model is then used to make predictions over the validation dataset. The validation dataset is where you might notice things that you will want to tweak, tune, and change.
- Then you can use the test dataset.
- The test dataset, which includes only features, because you want the labels to be predicted.
- The performance you get here with the test dataset is what you can reasonably expect to see in a production environment.
- feeds

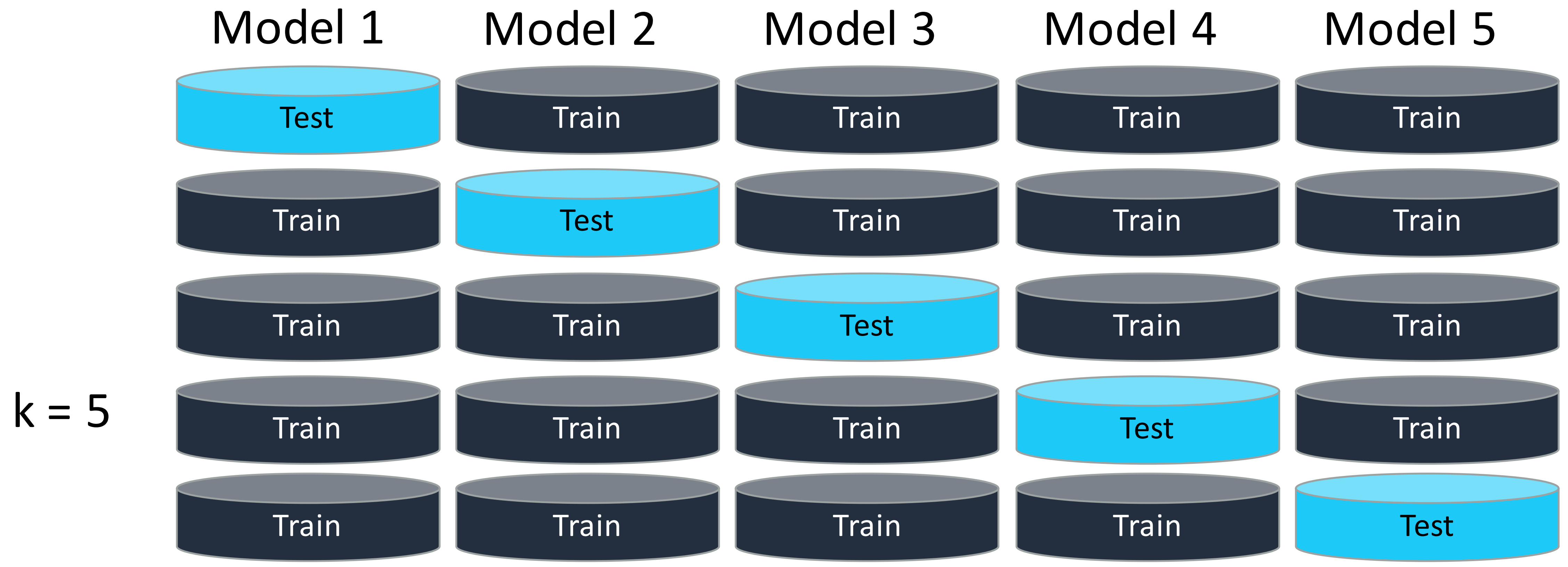
Holdout method



- A common split when you use the hold-out method is as follows:
  - use 80 percent of the data for a training set,
  - 10 percent for validation,
  - and 10 percent for the test.

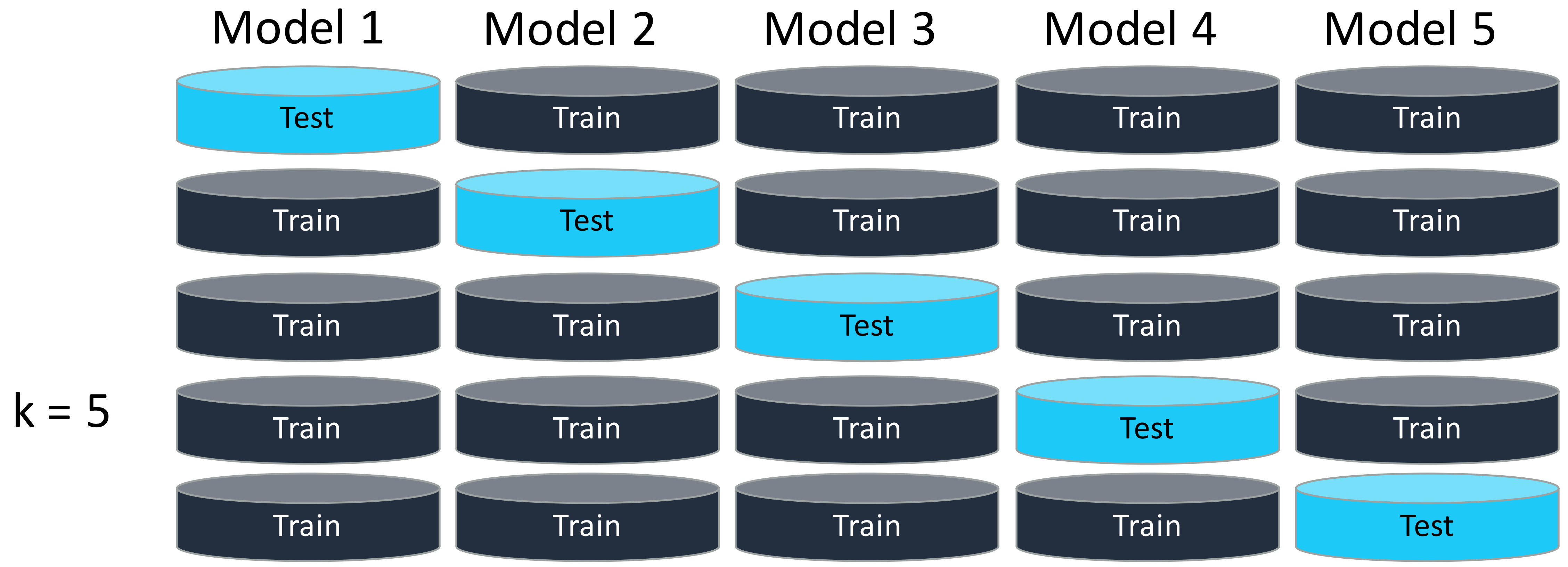
- Or,
- if you have a large amount of data, you can split it into 60-70 percent training, 15 percent validation, and 15 percent test.

K-fold cross validation



- For a small dataset, you can use k-fold cross-validation to use as much data as possible.
- This technique provides good metrics to choose which model is better.
- K-fold cross-validation randomly partitions the data into K different segments.

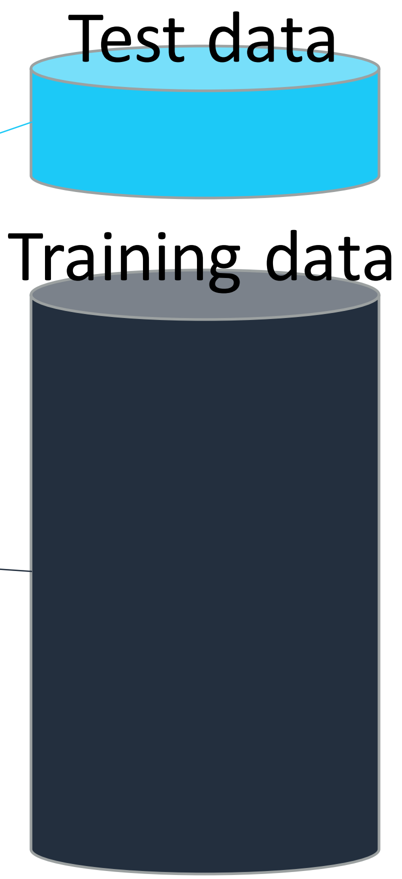
K-fold cross validation



- You apply different models to different pieces of the validation dataset to have some idea about how well the models perform.
- As a result, you use all the data for training. The cross-validation results are averaged to give you an idea about the performance of the model.

Shuffle your data

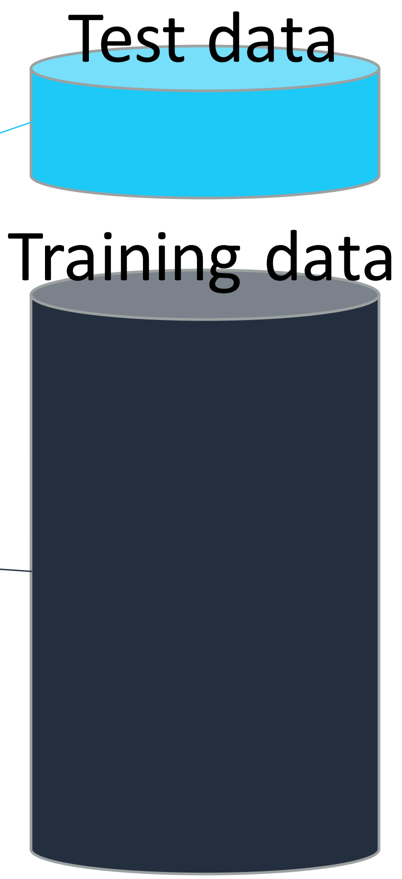
quality	Fixed acidity	sulphates
3	7.2	0.56
3	7.2	0.68
4	7.8	0.65
4	7.8	0.65
4	…	…
5	…	…
5	…	…
5	…	…
6	…	…
6	…	…



- Other potential problems with splitting your data lead to biases in your models
- Especially true when you work with structured data.
- For example, by the quality column, in the case of wine quality use case.
- when you run your model against your test data, this ordered pattern is applied, which biases the model.
- It might also mean that some targets are missing from the training data.

Shuffle your data

quality	Fixed acidity	sulphates
3	7.2	0.56
3	7.2	0.68
4	7.8	0.65
4	7.8	0.65
4	…	…
5	…	…
5	…	…
5	…	…
6	…	…
6	…	…



- With smaller sets, it is sometimes useful to use stratified sampling.
- Stratified sampling ensures that the training and test sets have approximately the same percentage of samples of each target class as the complete set.
- One of the easiest ways is to use the train_test_split function from sklearn.

```
from sklearn.model_selection import train_test_split
training_data, test_data =
train_test_split(all_the_data, test_size=0.2, random_state=
42, stratify=all_the_data[target'])
```

- Thus, The use of stratify ensures that the data—when it's split—maintains the ratio in the target column.

Guided Lab-4

- Guided Lab-4: Train the model
- DEMO of Lab-4
    - It is part of your lab_homework_4. (see at myCourse)
- DEMO of additional Lab training with *K*-NN Model
    - Comparison of model training with XGBoost and Knn models (not part of grade)

Training models with Amazon SageMaker

Amazon SageMaker provides ML algorithms that are optimized for speed, scale, and accuracy.

Amazon SageMaker built-in algorithms

Amazon SageMaker built-in algorithms: You can train and deploy these algorithms from Amazon SageMaker. Containers are used in the background when you use one of the Amazon SageMaker built-in algorithms.

Amazon SageMaker supported frameworks

Amazon SageMaker supported frameworks: Amazon SageMaker provides prebuilt containers to support deep learning frameworks such as Apache MXNet, TensorFlow, PyTorch, and Chainer. It also supports ML libraries, such as scikit-learn and SparkML by providing prebuilt Docker images.

Training models with Amazon SageMaker

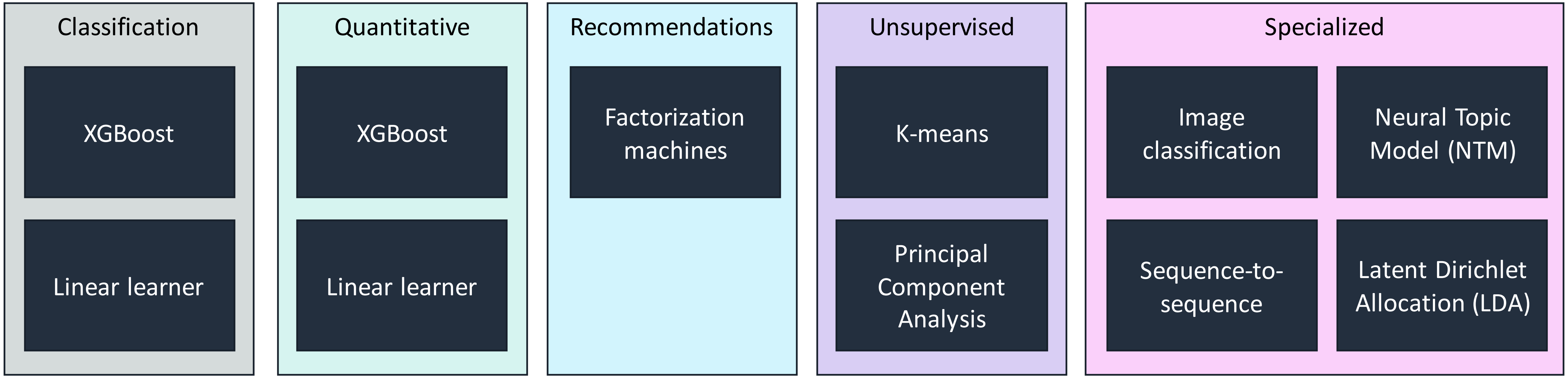
Amazon SageMaker custom frameworks

AWS Marketplace algorithms

<u>Amazon SageMaker custom frameworks</u>:
You can package your script or algorithm if you have no prebuilt Amazon SageMaker container image to use or modify for an advanced scenario. You can then use this script or algorithm with Amazon SageMaker. You can use any programming language or framework to develop your container. (For example, your team might work and build ML models in R. If so, you can build your own containers to train and host an algorithm in R.)

<u>AWS Marketplace algorithms:</u>
A third-party organization might have developed and tuned a model. It's worth looking in the AWS Marketplace to see what ready-to-use algorithms and models are available.
https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-mkt-find-subscribe.html

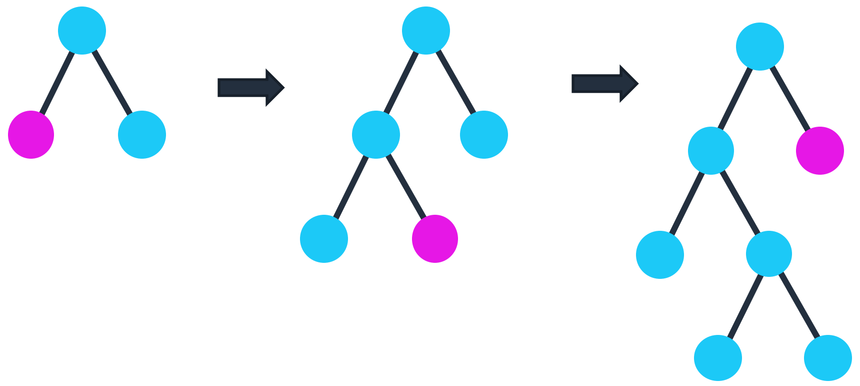
Amazon SageMaker built-in algorithms



Choose an algorithm -> https://docs.aws.amazon.com/sagemaker/latest/dg/algorithms-choose.html

XGBoost

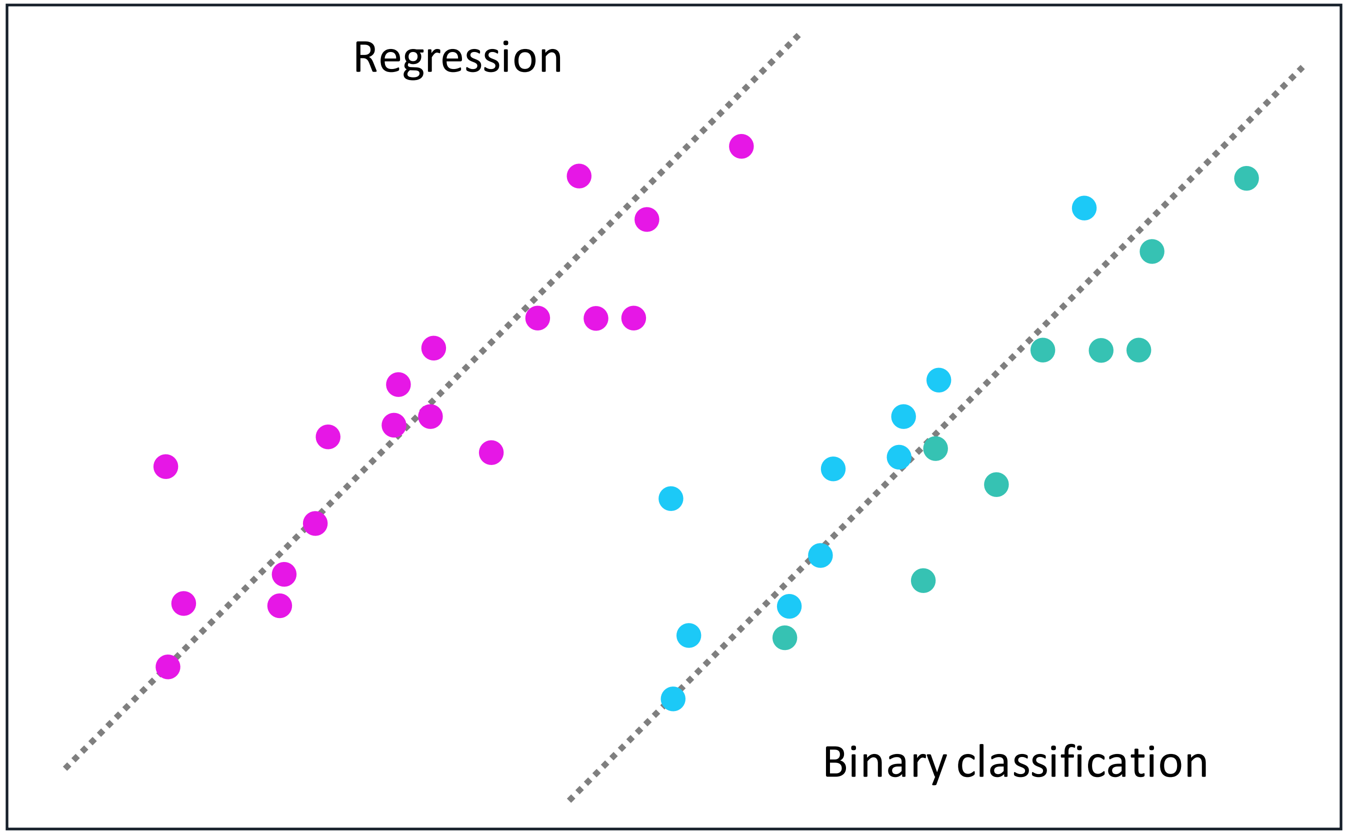
- The XGBoost (eXtreme Gradient Boosting) is a popular and efficient open-source implementation of the gradient-boosted trees algorithm.
- Has performed well in machine learning competitions
- Robustly handles various data types, relationships, and distributions, and a large number of hyperparameters



https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost.html

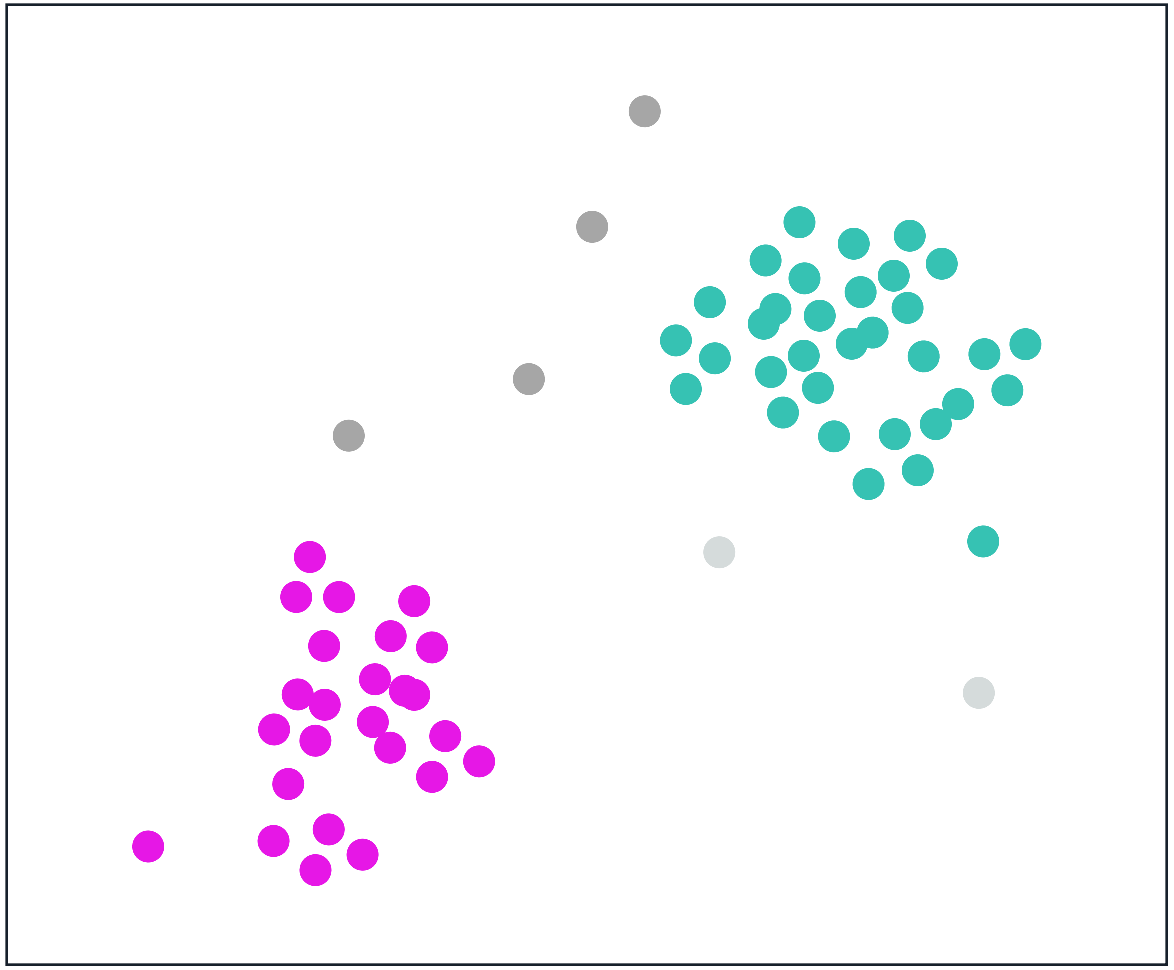
Linear learner

- Provides a solution for both classification and regression problems
- Enables you to simultaneously explore different training objectives and choose the best solution from your validation set.
- It provides a significant increase in speed over naive hyperparameter optimization techniques.

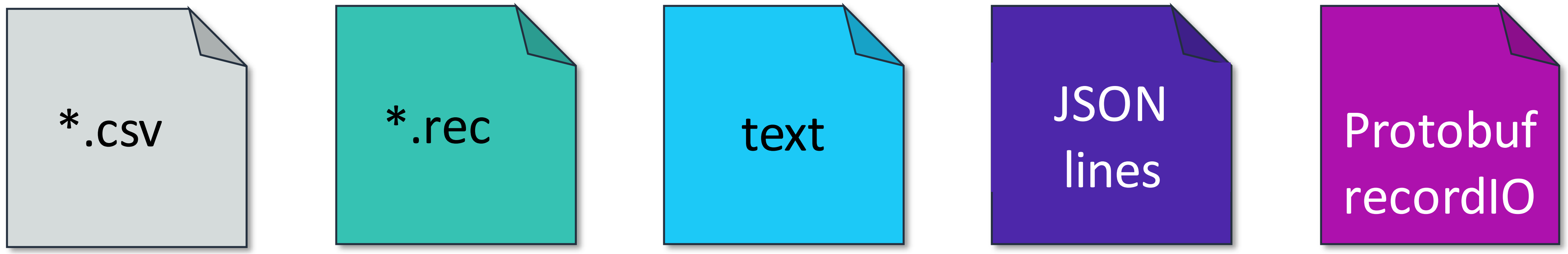


K-means

- K-means is an unsupervised learning algorithm.
- It attempts to find discrete groupings within data, where members of a group are as similar as possible.
- The *means* in *k-means* is the averaging of the data. This averaging helps to find the center of the grouping.
- You define the attributes that you want the algorithm to use to determine similarity.



File formats for machine learning



- Some algorithms require your data to be in a specific format.
- You should be aware of the formatting steps as a way to prepare for model training.
- Some algorithms might not be able to work with training data in a DataFrame format.
- The data that you use for ML training can be stored in various formats, depending on your use case and algorithm.

File formats for SageMaker

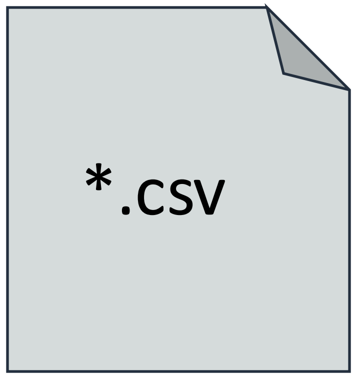
Protobuf recordIO

- Amazon SageMaker algorithms work best when you use the optimized protobuf recordIO format for the training data.
- In the protobuf recordIO format, Amazon SageMaker converts each instance in the dataset into a binary representation as a set of 4-byte floats, then loads it in the protobuf values field.

- This format enables you to take advantage of Pipe mode when you train the algorithms that support it.
- In Pipe mode, your training job streams data directly from Amazon S3. In contrast, File mode loads all your data from Amazon S3 to the training instance volumes. Streaming with Pipe mode can provide faster start times for training jobs and better throughput.
- You can also reduce the size of the Amazon Elastic Block Store (Amazon EBS) volumes for your training instances because pipe mode needs only enough disk space to store your final model artifacts.
- In contrast, File mode needs disk space to store both your final model artifacts and your full training dataset.

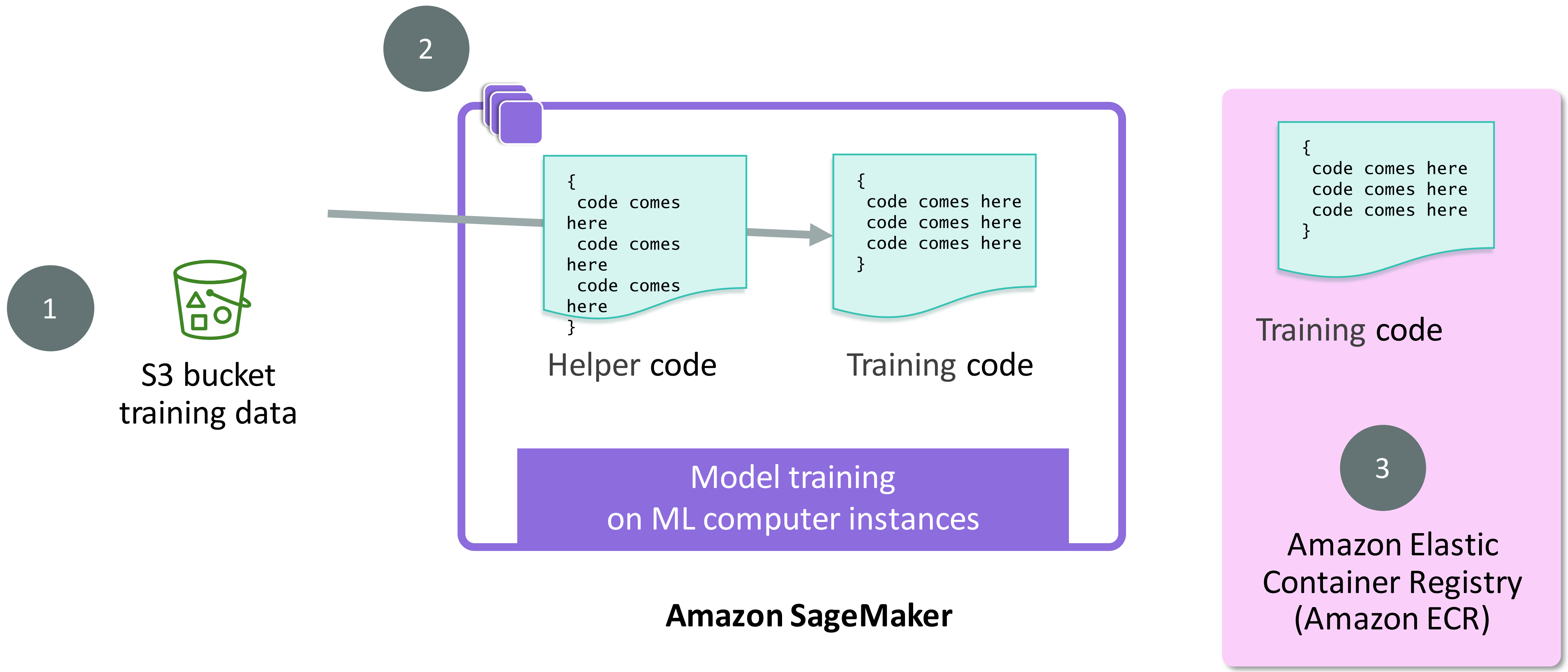
File formats for machine learning

- In general, most common file format used is CSV.
- Many Amazon SageMaker algorithms support training with data in CSV format.
- For use with Amazon SageMaker, CSV files can’t have a header record, and the target variable must be in the first column.



Target	Feature1	Feature 2	Feature 3	Feature 4	Feature 5	...
2	8.39	92.3	1	0	5	...
3	7.82	201.39	0	1	3	...
3	2.39	245.21	0	0	10	...
2	8.48	183.92	1	1	1	...
1	5.80	28.2	0	1	1	...

Creating a training job in Amazon SageMaker


1
S3 bucket training data
2
{ code comes here code comes here code comes here }
Helper code
{ code comes here code comes here code comes here }
Training code
Model training on ML computer instances
Amazon SageMaker
{ code comes here code comes here code comes here }
Training code
3
Amazon Elastic Container Registry (Amazon ECR)

Creating a training job in Amazon SageMaker

To train a model in Amazon SageMaker, you create a training job. The training job includes the following information:

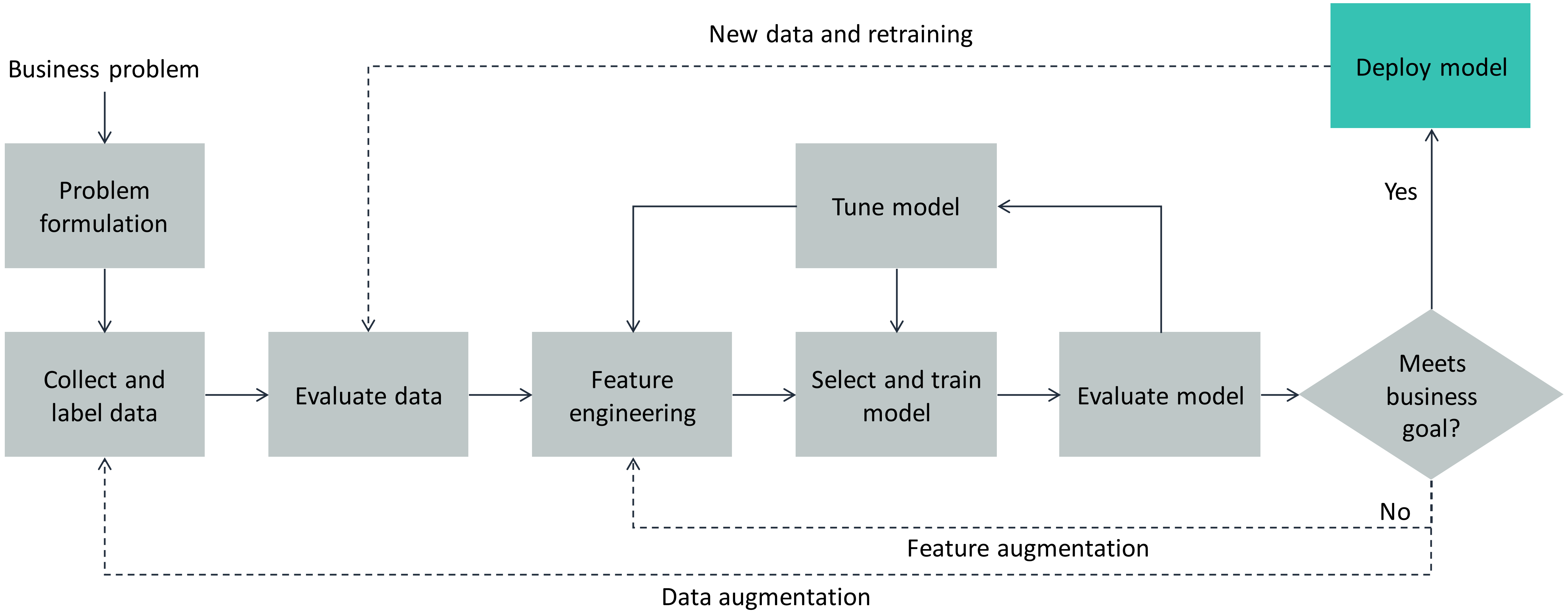
- The URL of the Amazon S3 bucket where you stored the training data.
- The compute resources that you want Amazon SageMaker to use for model training. Compute resources are ML compute instances that Amazon SageMaker manages.
  - Amazon SageMaker provides a selection of instance types that are optimized to fit different ML use cases. Instance types comprise varying combinations of CPU, GPU, memory, and networking capacity. You have the flexibility to choose the appropriate mix of resources for building, training, and deploying your ML models. Each instance type includes one or more instance sizes, and this feature enables you to scale your resources to the requirements of your target workload.
- The URL of the S3 bucket where you want to store the output of the job.
- The Amazon Elastic Container Registry (Amazon ECR) path where the training code is stored.

Lab-4 Using Amazon SageMaker

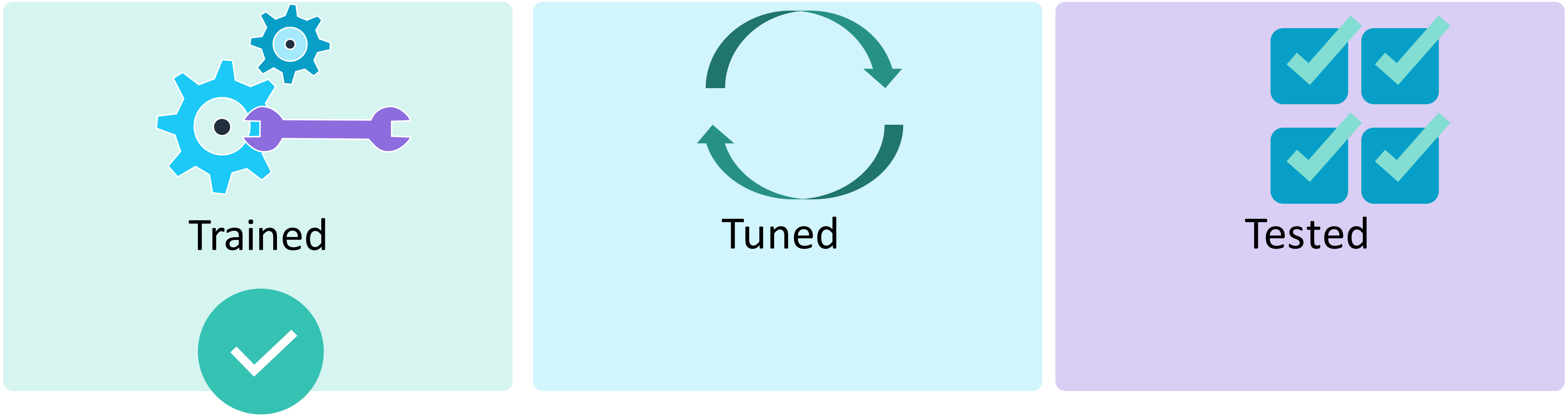
Extra DEMO:

- Train the model Using Amazon SageMaker (not part of grade)
- DEMO of Lab-4 using Amazon SageMaker

Model Deployment



Is your model ready to deploy?



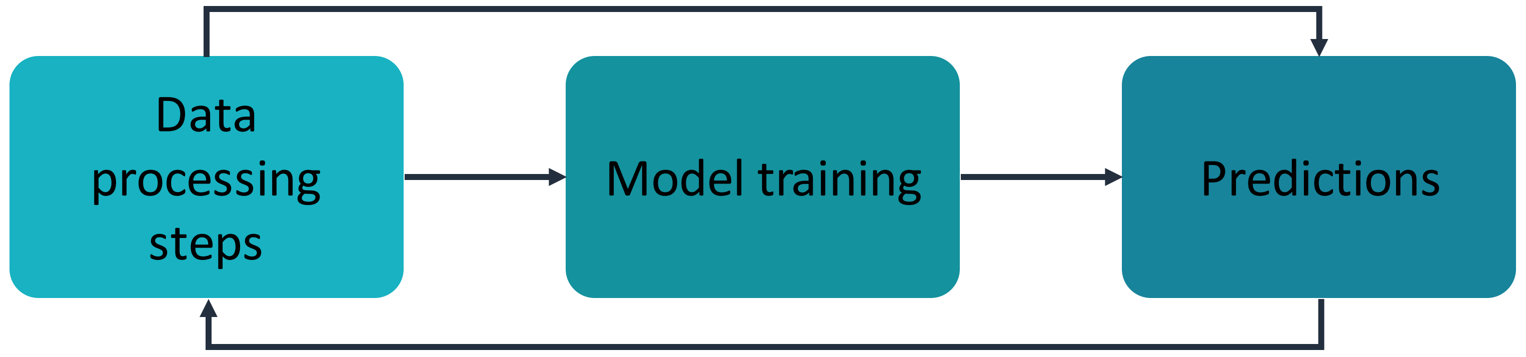
- After you have trained, tuned, and tested your model, you are ready to deploy it.
- <u>As per the sequence of phases in the ML pipeline it looks out of order.</u>
- To test and get performance metrics from your model, you must make inferences or predictions from the model—which typically requires deployment.

Is your model ready to deploy?

Trained	Tuned	Tested
✓		✓

- Deployment for testing is different from production, although the mechanics are the same.
- This is the step for hosting your model for simple testing and evaluation.
- These assessments can involve anything from a few requests to deployments that handle tens of thousands of requests.

Obtaining predictions



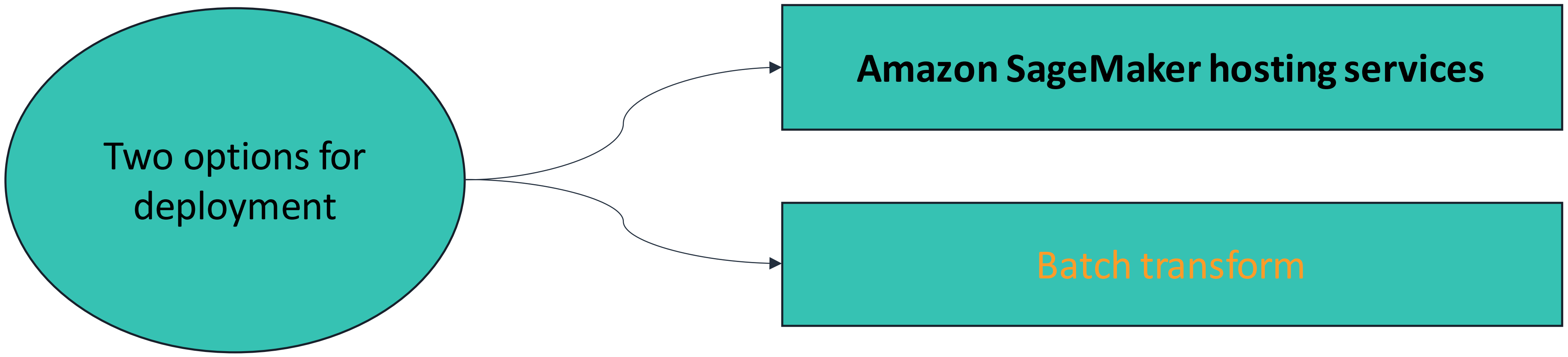
- When you deploy machine learning models into production, consistency is important, so you can make expected predictions on new data. You must ensure that the same data processing steps that were used in training are also applied to each inference request. Otherwise, you might get incorrect prediction results.
- By using inference pipelines, you can reuse the data processing steps that were applied in model training to inference.
- It isn't necessary to maintain two separate copies of the same code. This reuse of code helps with the accuracy of your predictions and reduces development costs.

Guided Lab-5

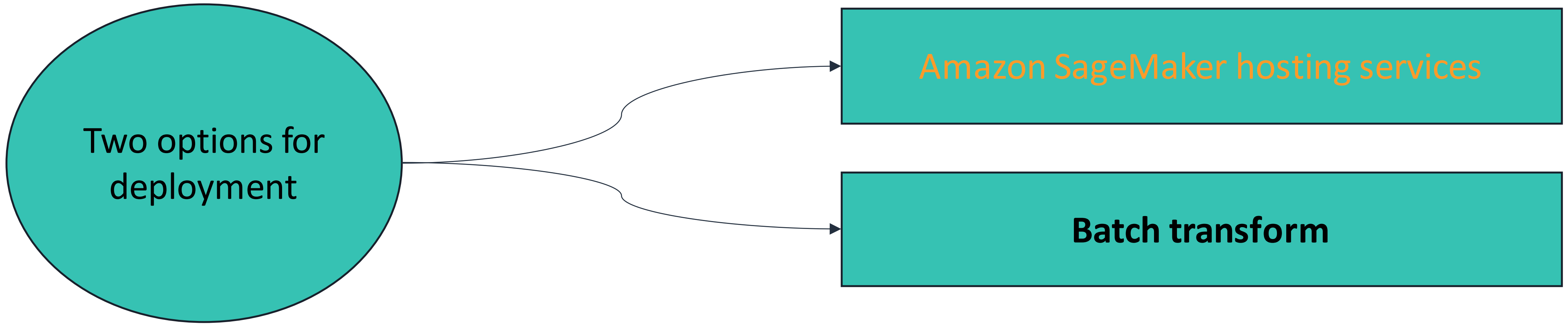
- Guided Lab-5: Hosting (Deployment) the model
- DEMO of Lab-5
  - It is part of your lab_homework_5. (see at myCourse)
- DEMO of Lab-5 hosting with KNN Model
  - Comparison of model hosting with XGBoost and K-NN model (not part of grade)
  - Comparison of single prediction and batch processing.

Deployment with Amazon SageMaker

- You can deploy your model in two ways.
- For single predictions, deploy your model with Amazon SageMaker hosting services.
- Applications can call the API at the endpoint to make predictions.
- With this method, you can scale the number of compute instances up or down, based on demand. (Amazon SageMaker deploys multiple compute instances that run your model behind a load-balanced endpoint.)

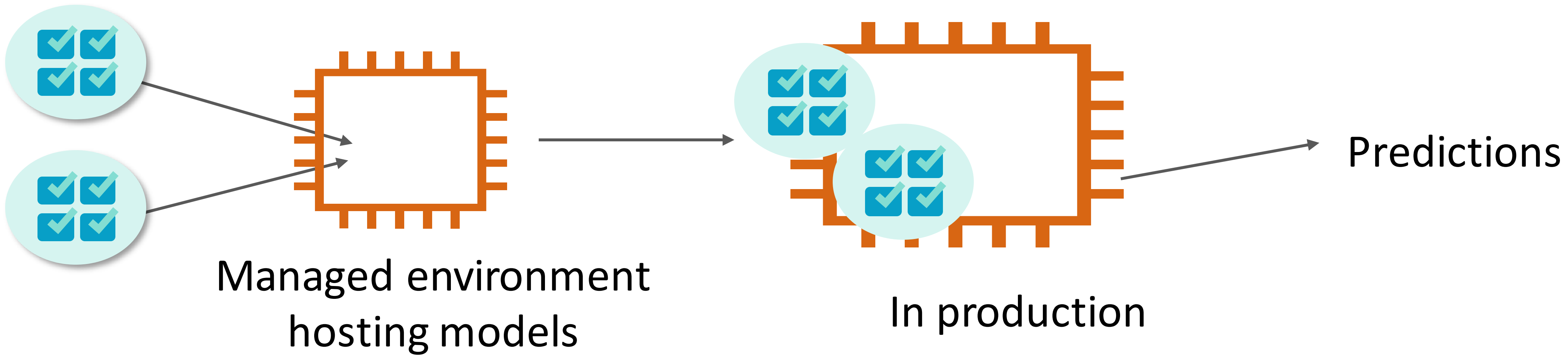


Deployment with Amazon SageMaker



- To get predictions for an entire dataset, use Amazon SageMaker batch transform.
- Instead of deploying and maintaining a permanent endpoint, Amazon SageMaker spins up your model and performs the predictions for the entire dataset that you provide.
- Performing batch predictions when you test the model is useful because you can quickly run your entire validation set against the model.
- (It stores the results in Amazon S3 before it shuts down and terminates the compute instances.)

The goal of deployment

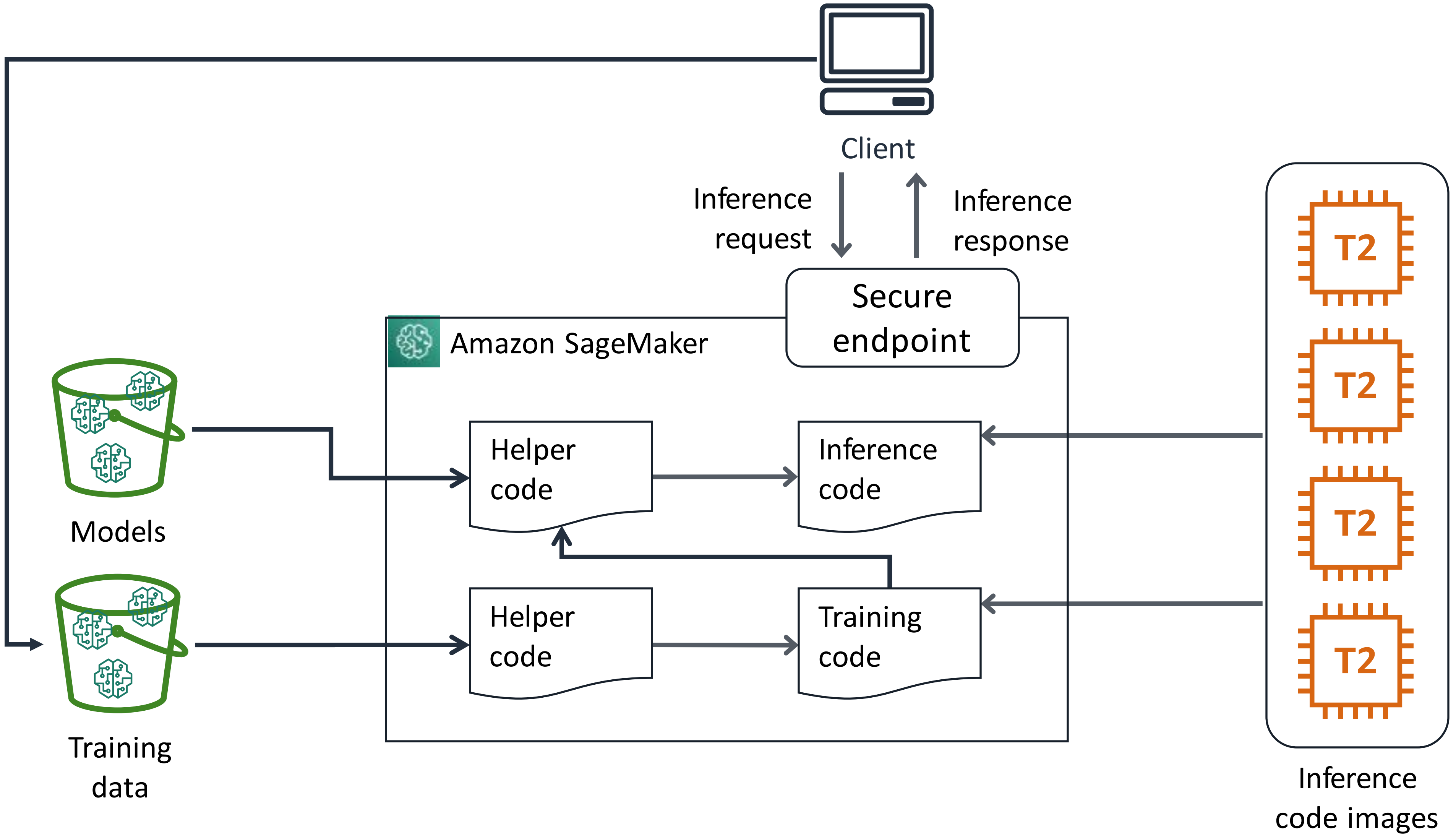


- The goal of the deployment phase is to provide a managed environment to host models for providing inference, both securely and with low latency.
- After deployment into production, you need to monitor your production data.
- If necessary, you must retrain your model, because newly deployed models must reflect the current production data. New data is accumulated over time, and it can potentially identify alternative or new outcomes.
- Thus, deploying a model is a continuous process, not a one-time exercise.

Deploying with Amazon SageMaker

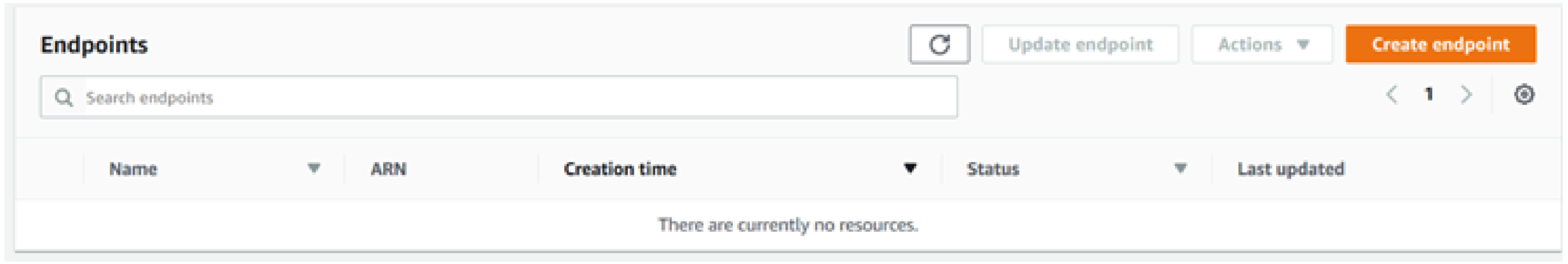
- Some of the detail of how the AWS environment works with SageMaker
  - For high availability and redundancy, you can easily deploy your model to automatically scale Amazon ML instances across multiple Availability Zones.
  - Specify the type of instance, and the maximum and minimum number that you want—Amazon SageMaker takes care of the rest. It launches the instances, deploys your model, and sets up the secure HTTPS endpoint for your application.
  - Your application must include an API call to this endpoint to achieve low latency and high-throughput inference.
  - This architecture enables you to integrate your new models into your application in minutes because model changes no longer require changes to application code.
  - Amazon SageMaker manages your production compute infrastructure on your behalf. It performs health checks, applies security patches, and conducts other routine maintenance.

Deploying with Amazon SageMaker



Creating an endpoint

You can create an endpoint in the console:



Name	ARN	Creation time	Status	Last updated

There are currently no resources.

- After you train the model, you can create the endpoint either in code or by using the Amazon SageMaker console.
- If you plan to host only a single model, you can create an endpoint for that model. However, if you plan to host multiple models, you must create a multi-model endpoint.

Creating an endpoint

You can also create endpoint using the Amazon SageMaker SDK:

```
xgb_predictor = xgb_model.deploy(initial_instance_count=1,
                                 content_type='text/csv',
                                 instance_type='ml.t2.medium'
                                 )
```

- Multi-model endpoints provide a scalable and cost-effective solution for deploying large numbers of models. They use a shared serving container that is enabled to host multiple models.
- This process reduces hosting costs by improving endpoint utilization, compared to single-model endpoints. It also reduces deployment costs because Amazon SageMaker loads models in memory, and scales them according to traffic patterns.

Lab-5 with Amazon SageMaker

- Extra DEMO:
- DEMO of Lab-5 hosting with Amazon SageMaker
  - Deployment of the model with Amazon SageMaker. (not part of grade)

Key take away – Training the Model

Some key takeaways from this section include:

- Selected Predictive and Classification ML Methods.
- Split data into training, testing, and validation sets to help you validate the models' accuracy
- Can use K-fold cross-validation to help with smaller datasets
- Use k-means for unsupervised learning
- Example lab with use case with two key algorithms —XGBoost and k-NN

Key take away – Hosting the Model

Some key takeaways from this section of the chapter include:

- You can deploy your trained model by using Amazon SageMaker to handle API calls from applications or to perform predictions by using a batch transformation.
- The goal of your model is to generate predictions to answer the business problem.
- Be sure that your model can generate good results before you deploy it to production.
- Use Single-model endpoints for simple use cases and use multi-model endpoint support to save resources when you have multiple models to deploy.

Chapter end questions

1. Why splitting and holding out data is required, in the ML process? Explain with an example.
2. How K-fold cross-validation can help you? Explain with an example.
3. Describe the detail of the “Linear Regression” Method with the example use case.
4. Describe the detail of the “K – Nearest Neighbors” method with the example use case.
5. Describe the detail of the “Naive Bayes Classifier” Method with the example use case.
6. Describe the detail of the “Classification and Decision tree” Method with the example use case.

Chapter end questions

7. Describe the detail of any algorithm/model of ML of your choice. (you can prepare answers from the link https://docs.aws.amazon.com/sagemaker/latest/dg/algorithms-tabular.html
8. Explain the deployment options, with examples.

Chapter-6
Implementing an ML Pipeline with JupyterLab – Part-4

(Evaluating the model, and Hyperparameter Tuning)

Credit note:

Slides, content, web links, and chapter-end questions are prepared from the AWS Academy course and other Internet resources. Students are advised to use it for study purposes only.

Chapter overview

Topics

1. Evaluating the model
2. Tune the Model

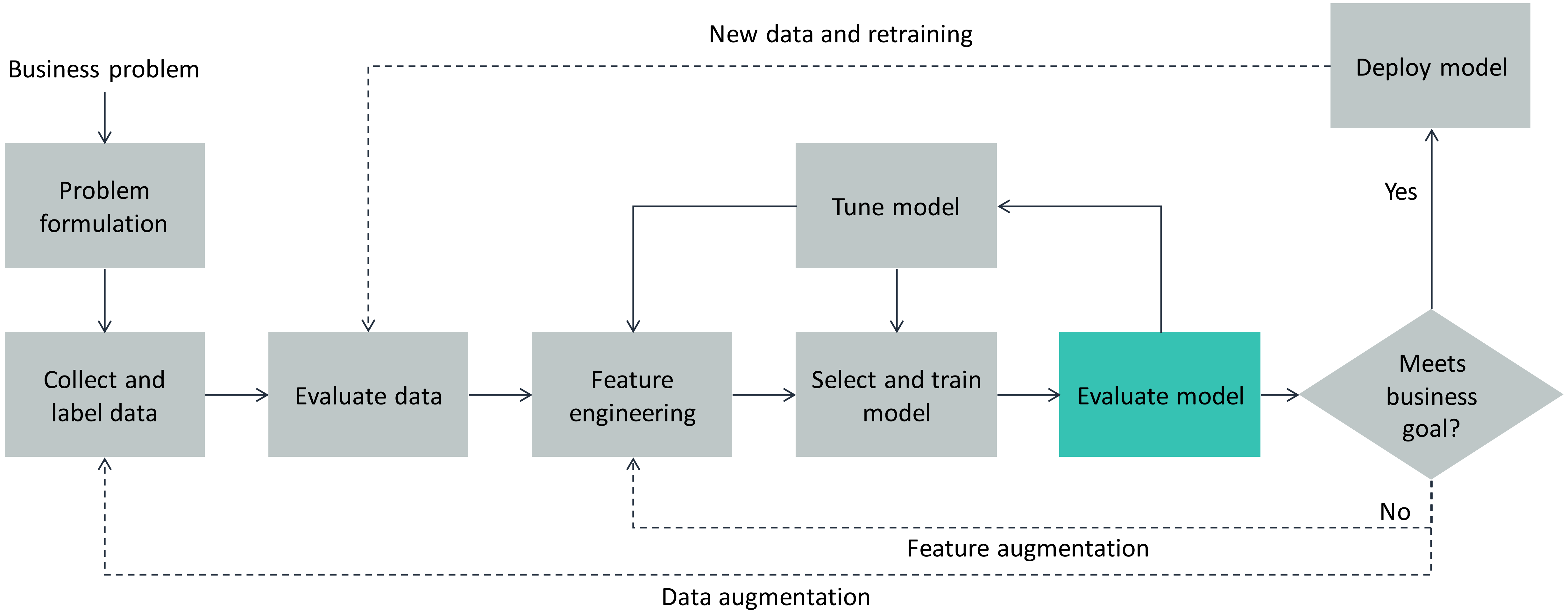
Labs Included:

- Guided Lab-6: Generating model performance metrics
- Guided Lab-7: Hyperparameter tuning

At the end of this Chapter, you should be able to:

- Use JupyterLab to evaluate model performance and tune ML model

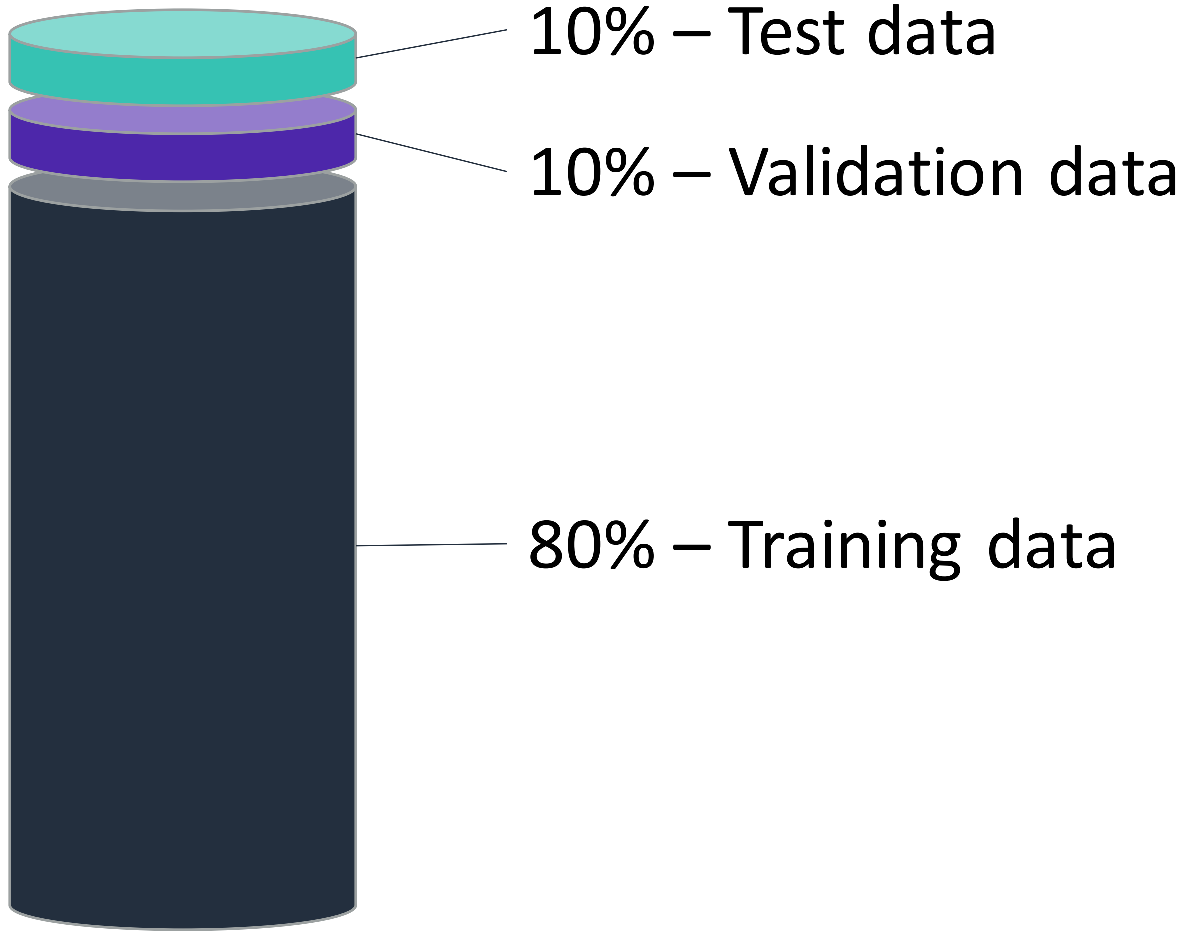
Machine learning pipeline



Evaluation determines how well your model predicts the target on future data

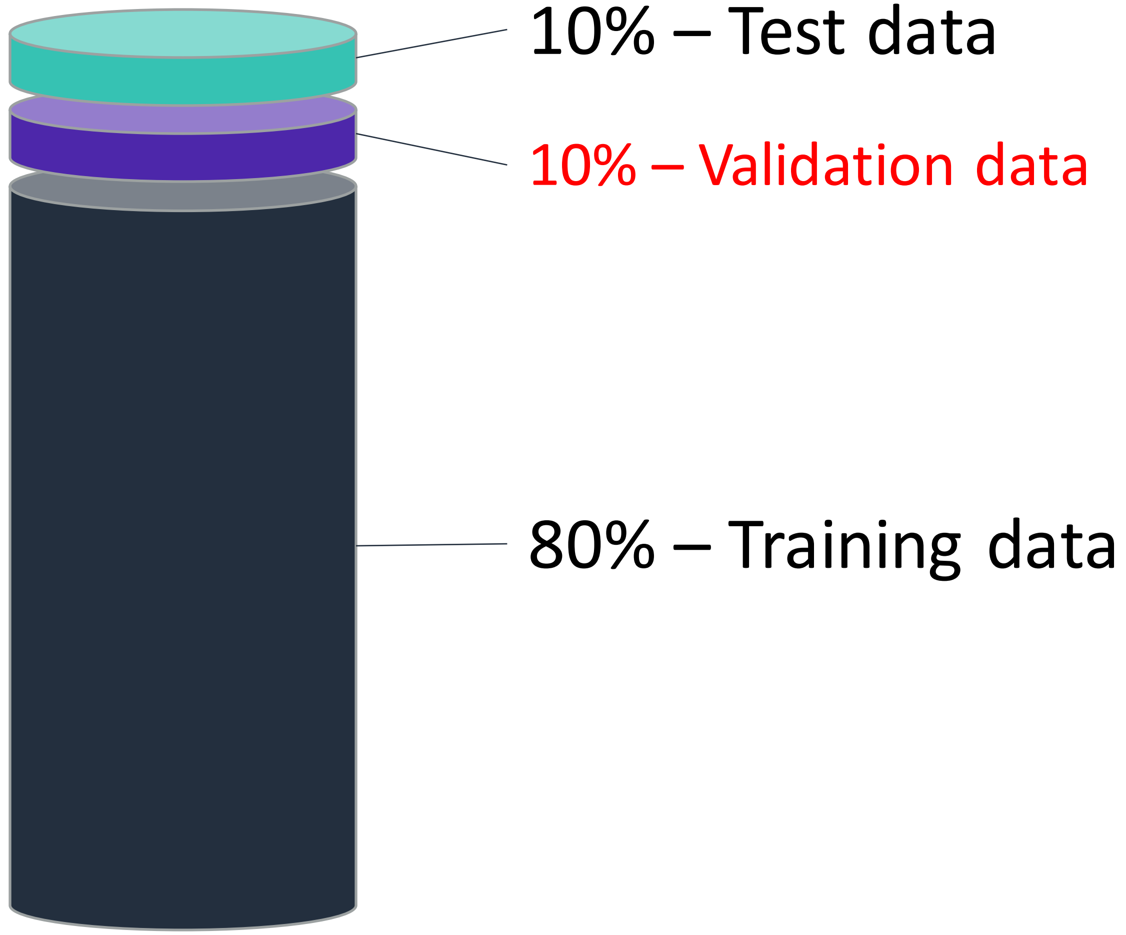
Testing and evaluation:

- To evaluate the predictive quality of the trained model.
- To determine whether it will perform well in predicting the target on new and future data.
- Because future instances have unknown target values, you must assess how the model will perform on data for which you know the target answer.

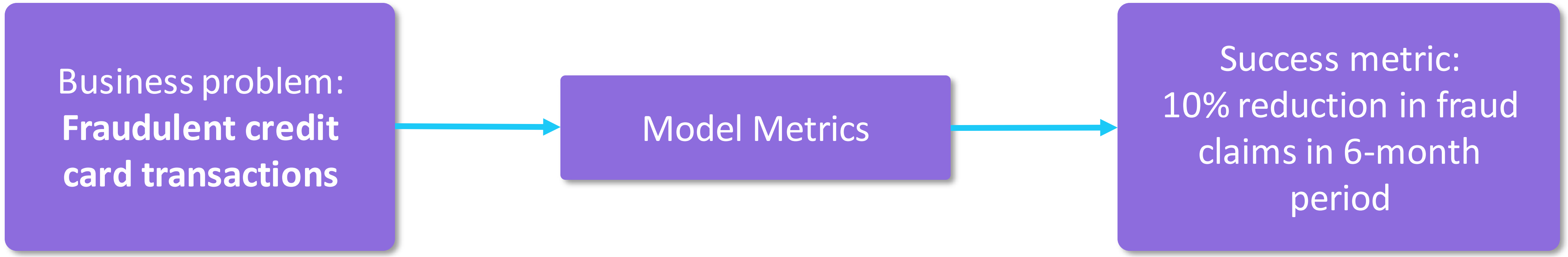


Evaluation determines how well your model predicts the target on future data

- Later on, use this assessment as a proxy for performance on future data.
- For this reason, you hold out a sample of your data for evaluation or testing purposes.

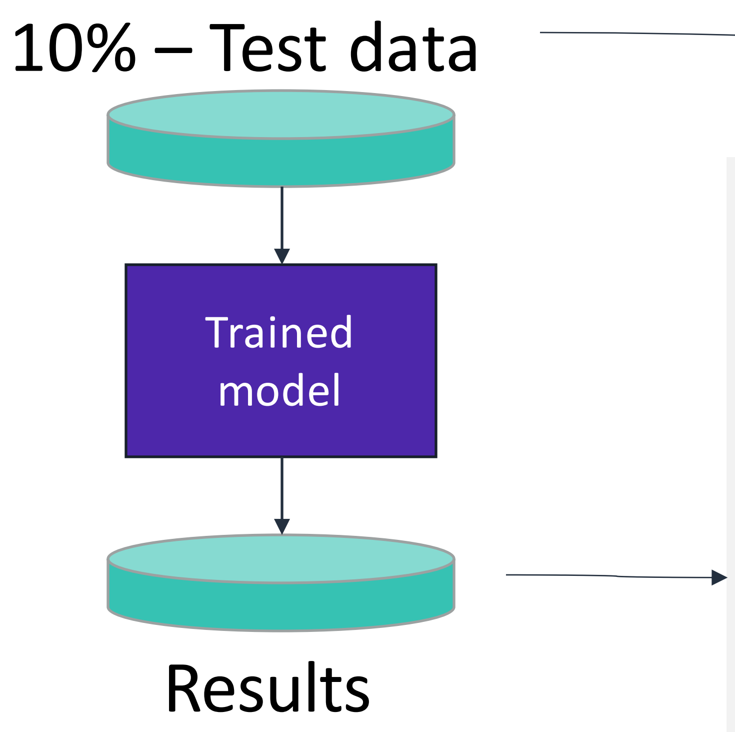


Success metric


Business problem:
Fraudulent credit card transactions
Model Metrics
Success metric:
10% reduction in fraud claims in 6-month period

- During the 1st phase, you defined your business problem and outcome, and you crafted a business metric to evaluate the success.
- During phase 6 you will evaluate the model metric.
- The model metric must align with both the business problem and the success metric. Frequently, the two metrics have a high correlation.

Confusion matrix



		Actual	
		Cat	Not cat
Predicted	Cat	107	23
	Not cat	69	42

- Suppose that you have a simple “image recognition model” that labels data as either cat or not cat. The total number of images is 241. *(176 cat images).*
- After the model is trained, you can use the test dataset that you held back to perform predictions.
- To help examine the performance of the model, you can compare the predicted values with the actual values.

Confusion matrix

		Actual	
		Cat	Not cat
Predicted	Cat	TP	FP
	Not cat	FN	TN

Two good outcomes for your model

True positive (TP)
Predicted a cat and it was a cat

True negative (TN)
Predicted *not* a cat and it was *not* a cat

Two other outcomes are possible, both of which are less than ideal

False positive (FP)
Predicted *not* a cat and it was a cat

False negative (FN)
Predicted a cat and it was *not* a cat

Sensitivity

Sensitivity: The first metric is sensitivity, which is sometimes referred to as hit rate or *true positive rate*. TPR metric is the percentage of positive identifications

- In this example -> What percentage of cats were correctly identified?
- To calculate sensitivity, you take the number of true positives (correct identifications of cats) and divide that number by the total number of actual cat images. (107+69 = 176)

		Actual	
		Cat	Not cat
Predicted	Cat	TP:107	FP:23
	Not cat	FN:69	TN:42

$$\text{sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{sensitivity} = \frac{107}{107+69} = 0.6079$$

60% of cats were identified as cats.

Specificity

Specificity: sometimes referred to as *selectivity* or true negative rate, is the percentage of negatives that were correctly identified

- What percentage of ***not*** cats were correctly identified?
- To calculate specificity, you take the number of true negatives and divide that by the total number of actual non-cat images. (23 +42 = 65)

		Actual	
		Cat	Not cat
Predicted	Cat	TP:107	FP:23
	Not cat	FN:69	TN:42

$$\text{specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

$$\text{specificity} = \frac{42}{42+23} = 0.6461$$

64% of ***not*** cats were identified as ***not*** cat

Which model is better?

- The following diagram shows the confusion matrices from two models that use the same data.
- Which model is better? Is better to make sure that you find all the cats, even if it means many false positives? Or is better to make sure the model is the most accurate?
- To answer these, you should consider and calculate metrics.

Model-1		Actual	
		Cat	Not cat
Predicted	Cat	107	23
	Not cat	69	42

Model-2		Actual	
		Cat	Not cat
Predicted	Cat	148	53
	Not cat	28	12

Which model is better now?

Model A

		Actual	
		Cat	Not cat
Predicted	Cat	107	23
	Not cat	69	42

Sensitivity	60%
Specificity	64%

Model B

		Actual	
		Cat	Not cat
Predicted	Cat	148	53
	Not cat	28	12

Sensitivity	84%
Specificity	18%

Which model is better now?

Model A

Sensitivity	60%
Specificity	64%

Model B

Sensitivity	84%
Specificity	18%

- Which model do you choose if you want to identify *as many cats as possible*? Model B is a good answer if you are not concerned about many false positives.
- Which model do you choose if you want to make sure that *you identify animals that are not cats*? Model A might work, depending on how many false negatives you can handle.

- However, If you are classifying patients that have heart disease or not, which model is best? This scenario is where it gets interesting. If you are trying to diagnose patients, your focus might be different.
- It's important to understand the tradeoffs. You can use more metrics to help with your decision-making.

Classification: Other metrics

Two more evaluations should be considered. 1) Precision and 2) F1 Score.

Precision, which removes the negative predictions from consideration.

- Precision is the *proportion of positive predictions* that are actually correct.
- You can calculate precision by taking the true positive and dividing it by the true positive plus the false positive.

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Proportion of positive results that were correctly classified

- When the cost of false positives is high in your particular business situation, precision might be a good, to consider for model adoption.

Classification: Other metrics

- Consider a model that must predict whether a patient has a terminal illness or not. In this use case, the use of precision as your evaluation metric doesn’t account for the false negatives in your model. Here, It is vitally important and essential to the success of the model to identify the illness in a patient who actually has that illness.
- In this situation, sensitivity is a better metric to use. However, it doesn’t always need to be one or the other.
- The **F1 score** combines precision and sensitivity to give you one number that quantifies the overall performance of a particular ML algorithm.
- You might want to use the F1 score when you have a class imbalance, but you want to preserve the equality between precision and sensitivity.

$$\text{F1 score} = 2 \cdot \frac{\text{precision} \cdot \text{sensitivity}}{\text{precision} + \text{sensitivity}}$$

Combines precision and sensitivity

Classification: Other metrics

Accuracy: You can calculate the model's accuracy (also known as its score). You can do this calculation by adding the correct predictions and then dividing that by the total number of predictions.

Model's score

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})}$$

- It has limitations. This metric is less effective when your dataset has many true negative cases. Like, in the example of cat/not cat, if the majority of your accuracy is based on true negatives, it says only that your model is good at predicting what isn't a cat.
- In the credit card fraud example, using accuracy as your main metric is probably not a good idea when you have many true negatives. Your high true-negative number might hide the fact that your model's ability to identify cases of fraud.
- Thus, It is important that the metric you choose for model evaluation aligns with your business goal.

Thresholds

Predicted
0.96
0.77
0.58
0.01
0.47

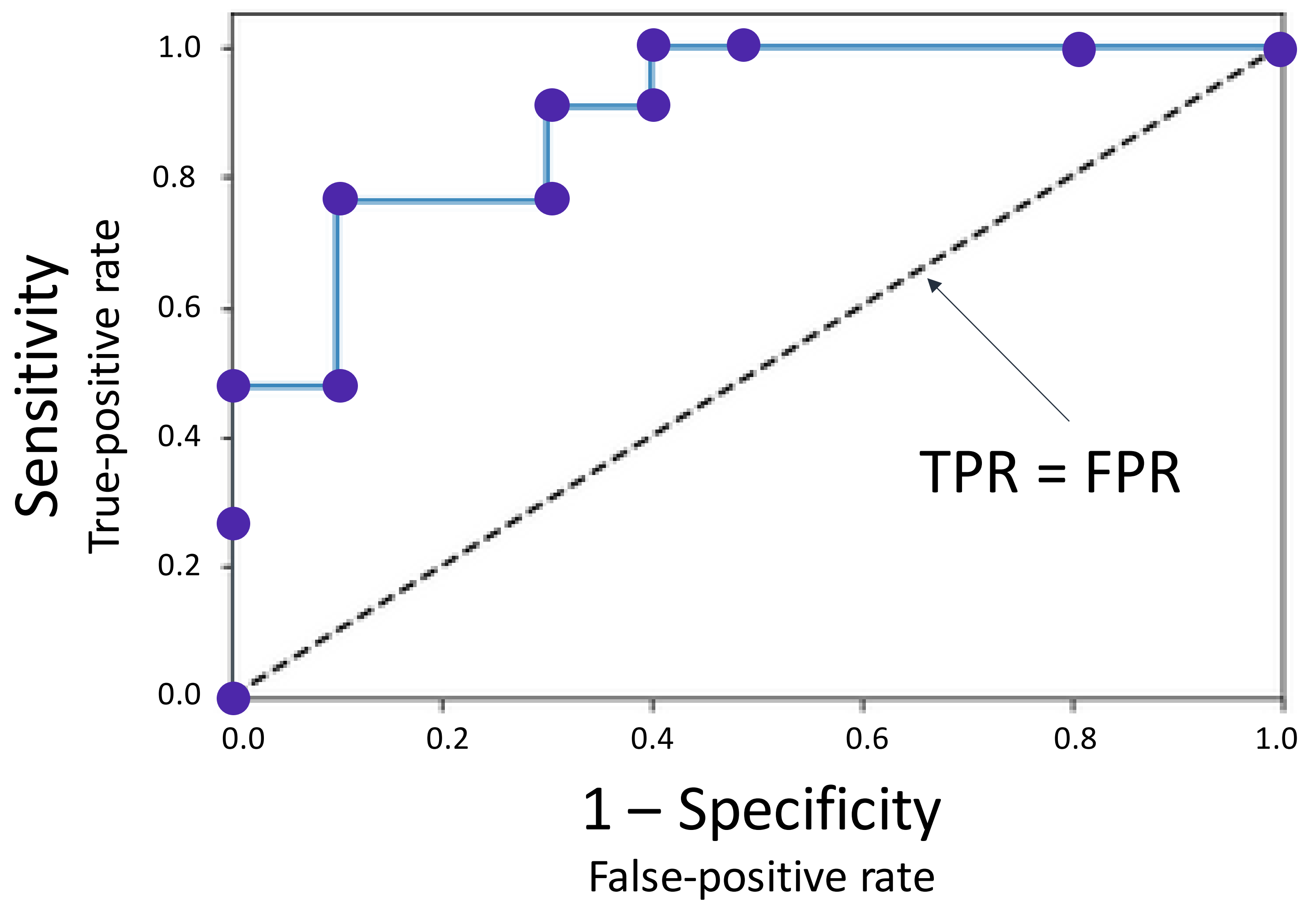
Classification Models return probabilities:

if [predicted] >= threshold   It's a cat!

if [predicted] < threshold   NOT cat!

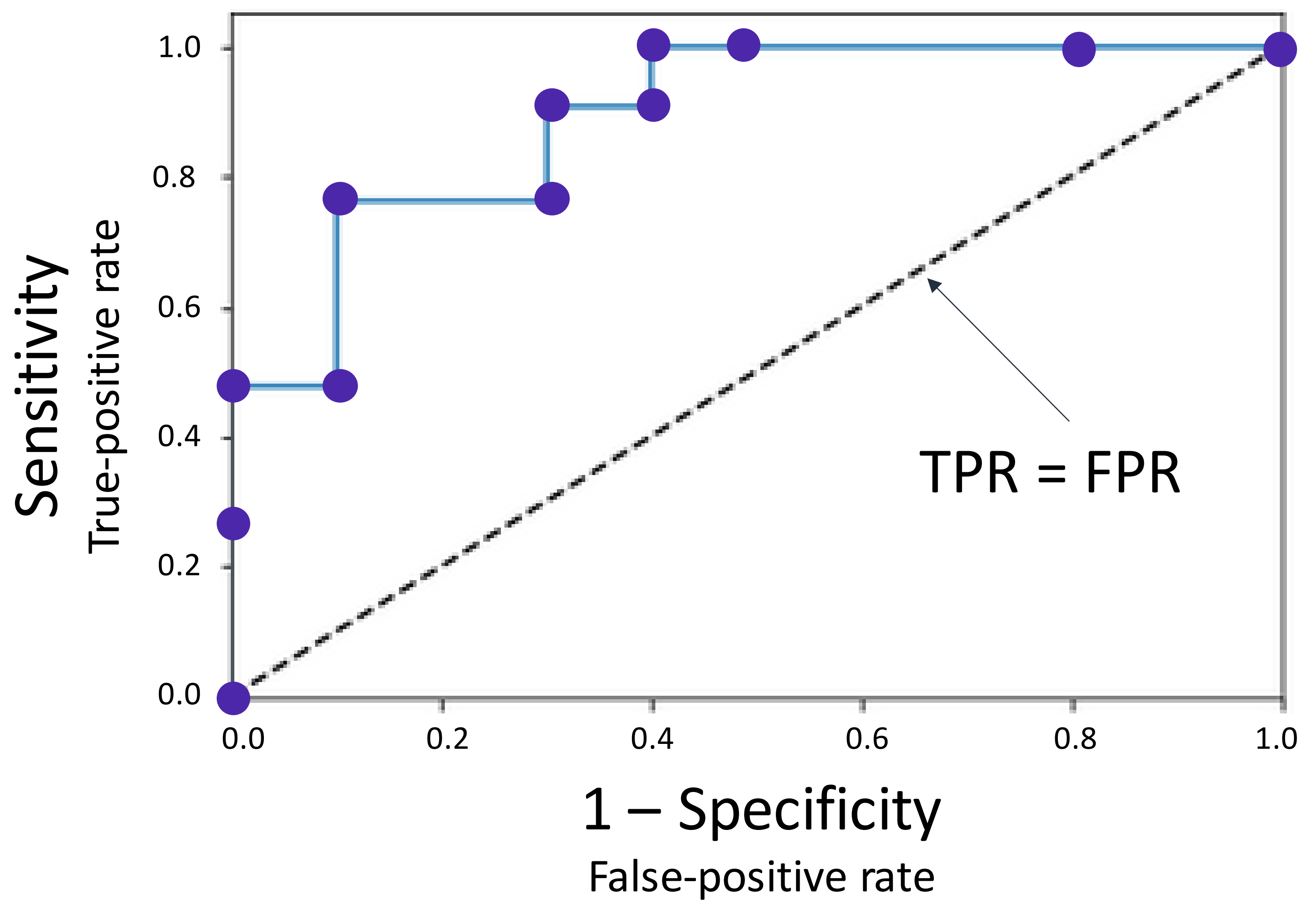
- The probability is a value between 0 and 1 of the input that belongs to the target class.
- To convert the value to a class, you must determine the threshold first.
- Like sensitivity and specificity, classification involves a tradeoff between correctly and incorrectly identifying classes.
- Changing the threshold can change the prediction.

Classification: ROC graph



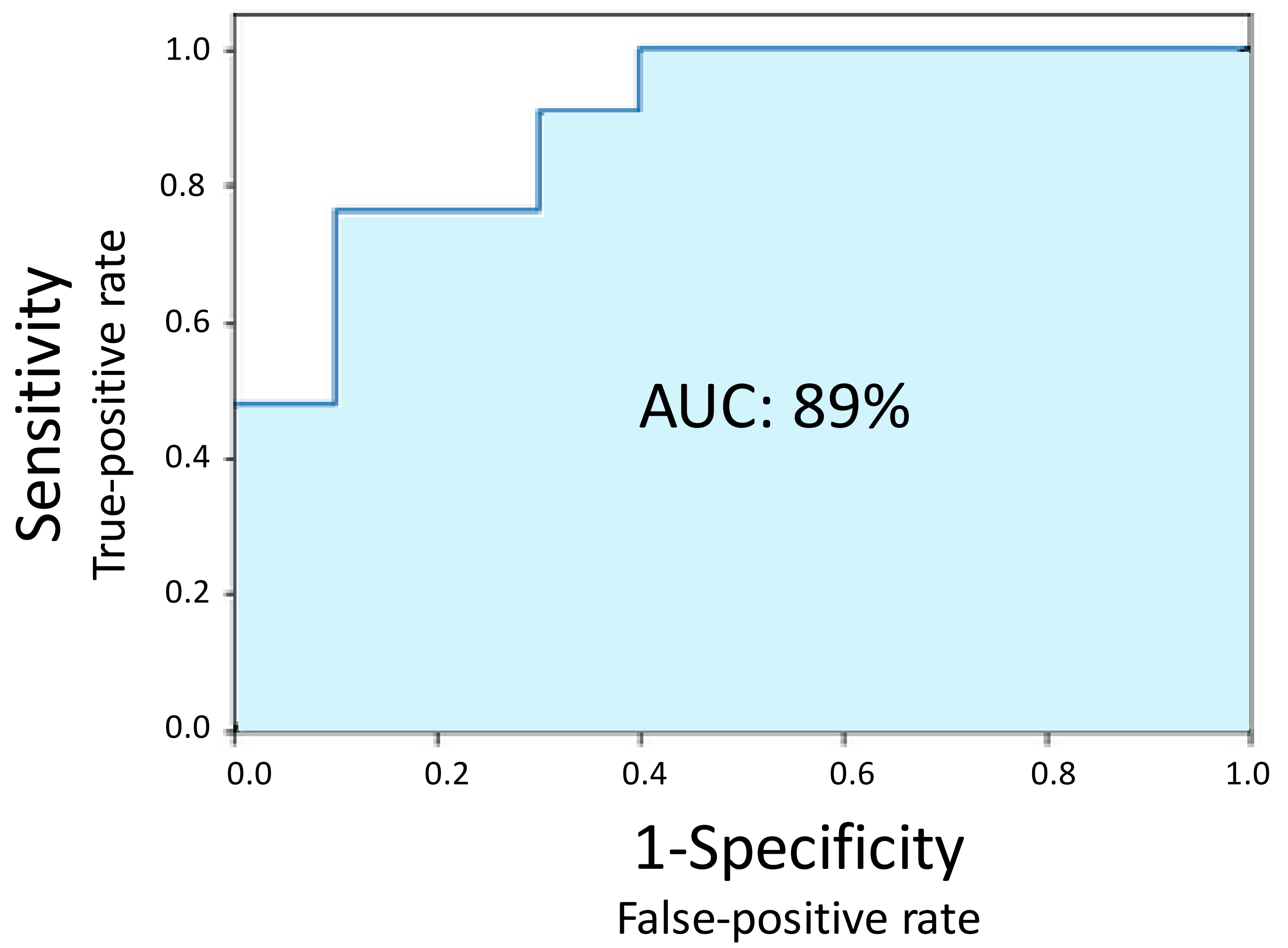
- A receiver operating characteristic (ROC) graph summarizes all the confusion matrices that each threshold produced.
- To build one, you calculate the sensitivity (or true-positive rate) against the false-positive rate for each threshold value. You then plot this value on a graph.
- You can calculate the false-positive rate by subtracting the specificity from 1.
- When those points are plotted, you can draw a line between them.

Classification: ROC graph



- The dotted black line from (0,0) to (1,1), which is the line that represents the sensitivity-to-TPR ratio, is equal to the false-positive rate.
- The point at (1,1) means that you would have correctly identified all the cats, but also incorrectly identified all the not cats. This result is not good.
- The point at (0,0) represents zero true positives and zero false positives.

Classification: AUC-ROC



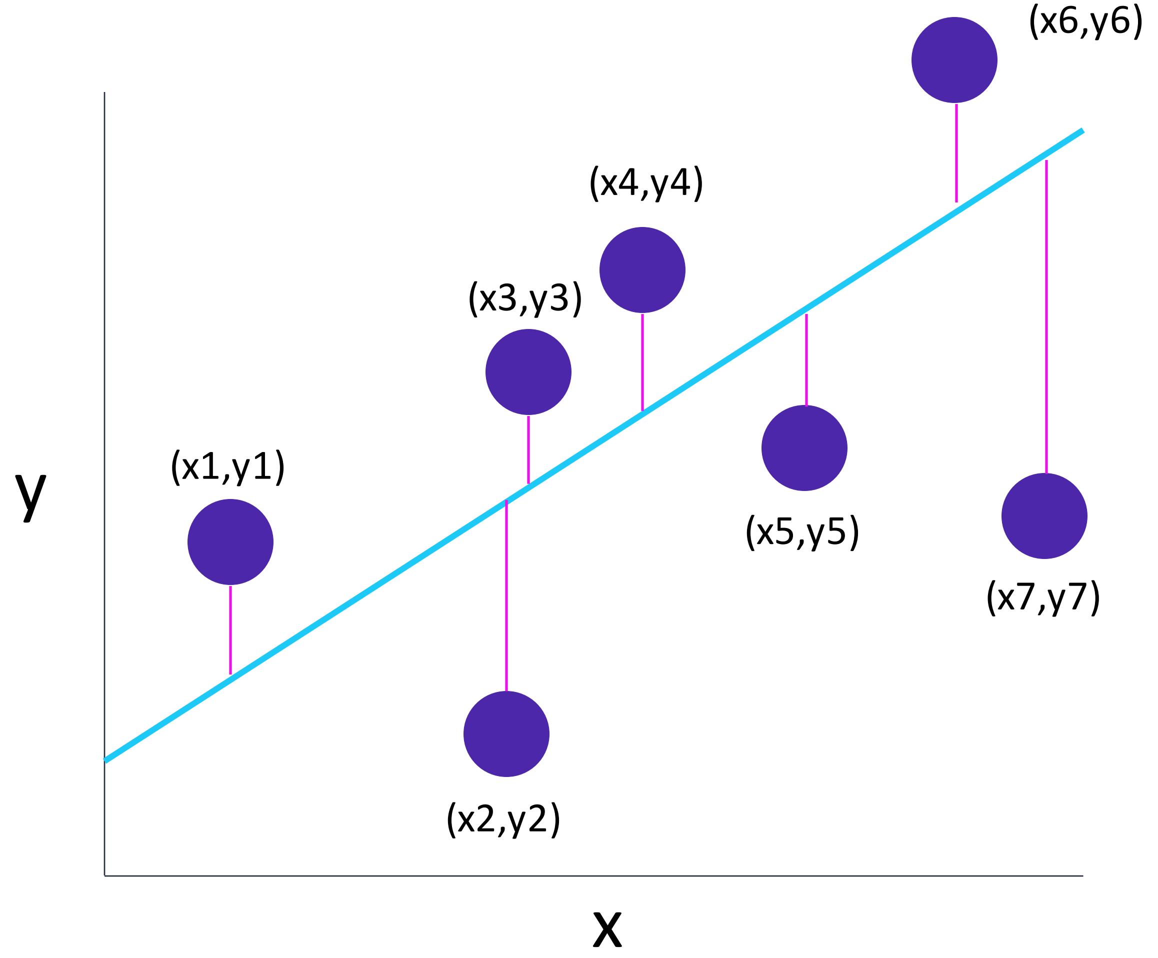
- A model that has high sensitivity and a low false-positive rate is usually your goal.
- It's better when the line between the threshold recordings is closer to the top-left corner.

- Therefore, the Area under the curve-receiver operator curve (AUC-ROC) is another evaluation metric. The AUC part is the area under the plotted line. The higher the AUC, the better the model is at predicting cats as cats and not cats as not cats.

Regression: Mean squared error

But what if you are dealing with a regression problem?
In case of a regression problem, you can use other common metrics to evaluate your model, including **mean squared error**.

$$\mathbf{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$



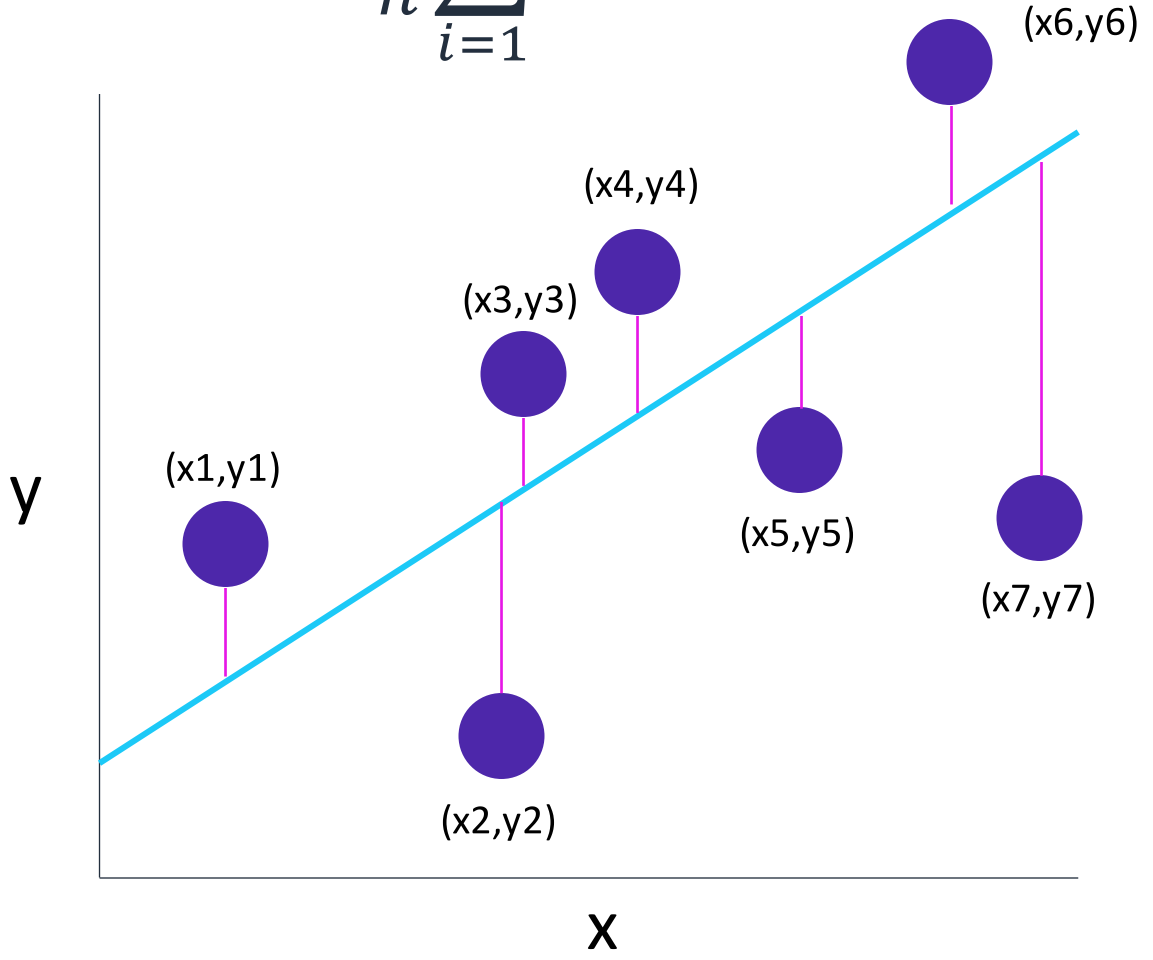
Regression: Mean squared error

- Mean squared error, you take the difference between the prediction and actual value, square that difference, and then add all the squared differences for all the observations.

- In scikit-learn, you can use the mean squared error function directly from the metrics library.

- You can use other metrics for linear models, including R squared.

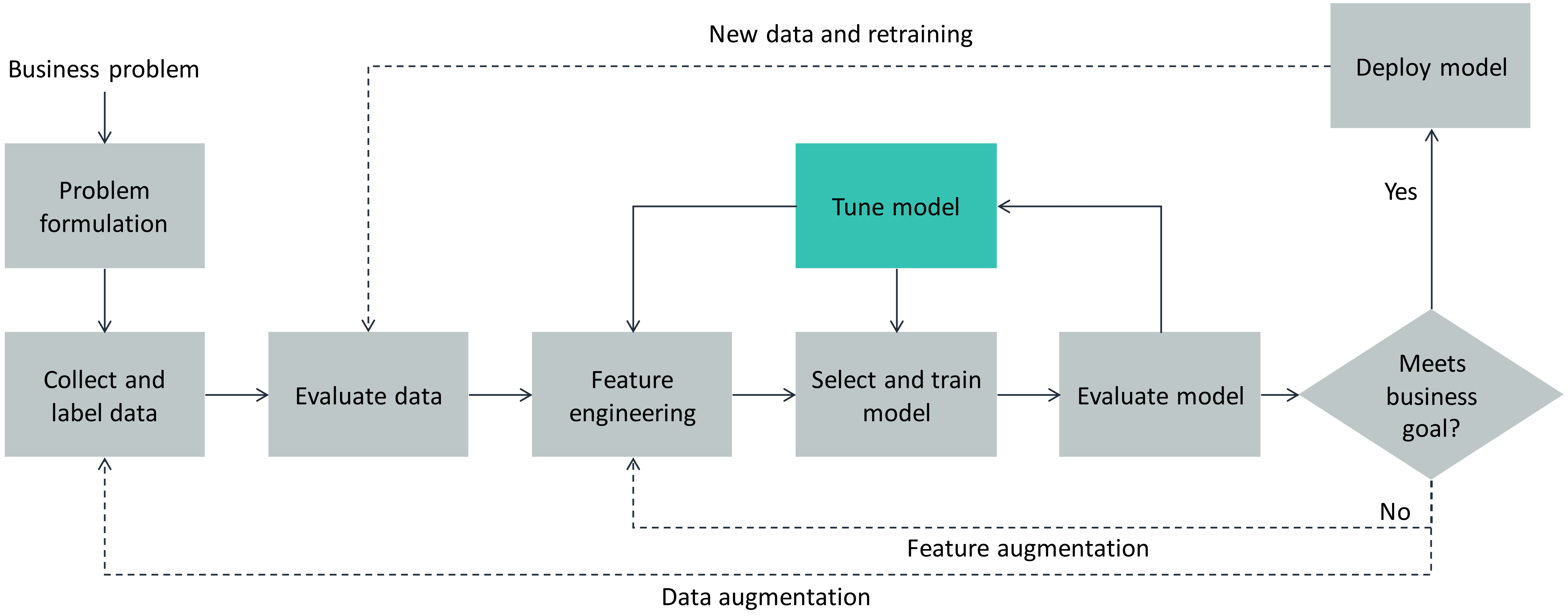
$$\mathbf{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$



Guided Lab-6

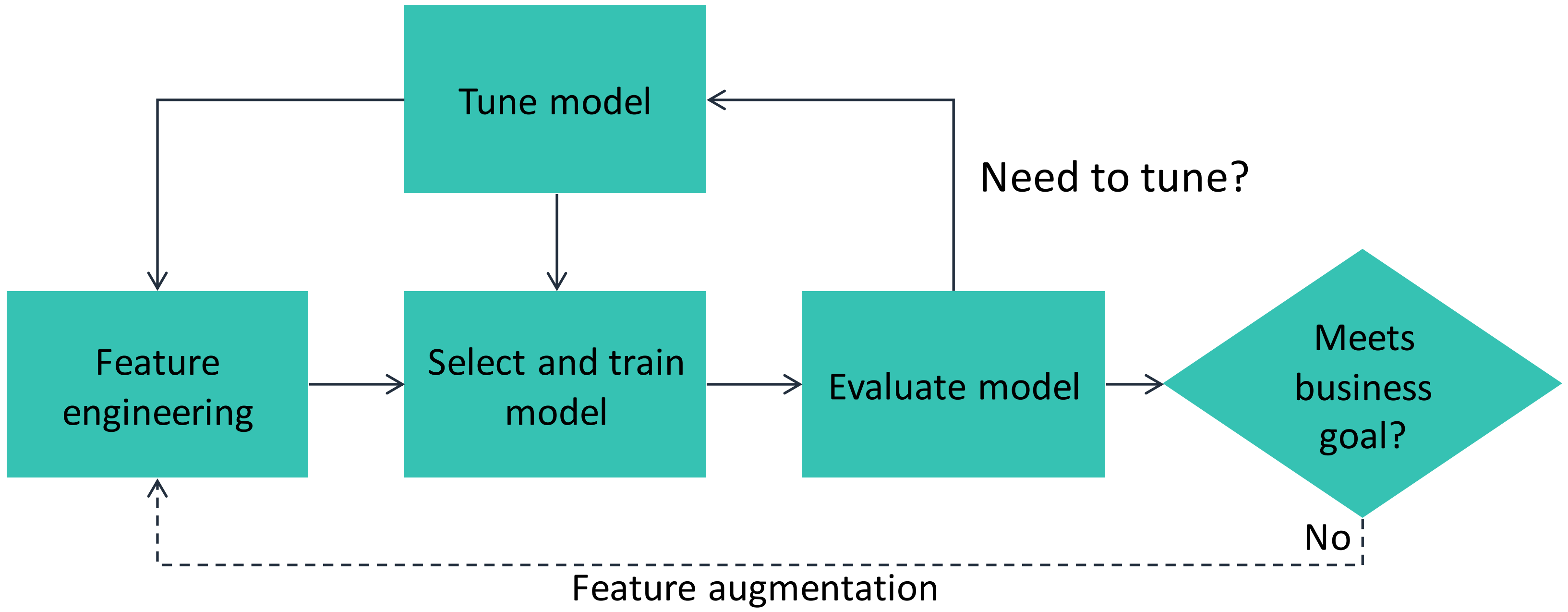
- Guided Lab-6: Evaluating the model performance
- DEMO of Lab-6
    - It is part of your lab_homework_6. (see at myCourse)

Machine learning pipeline



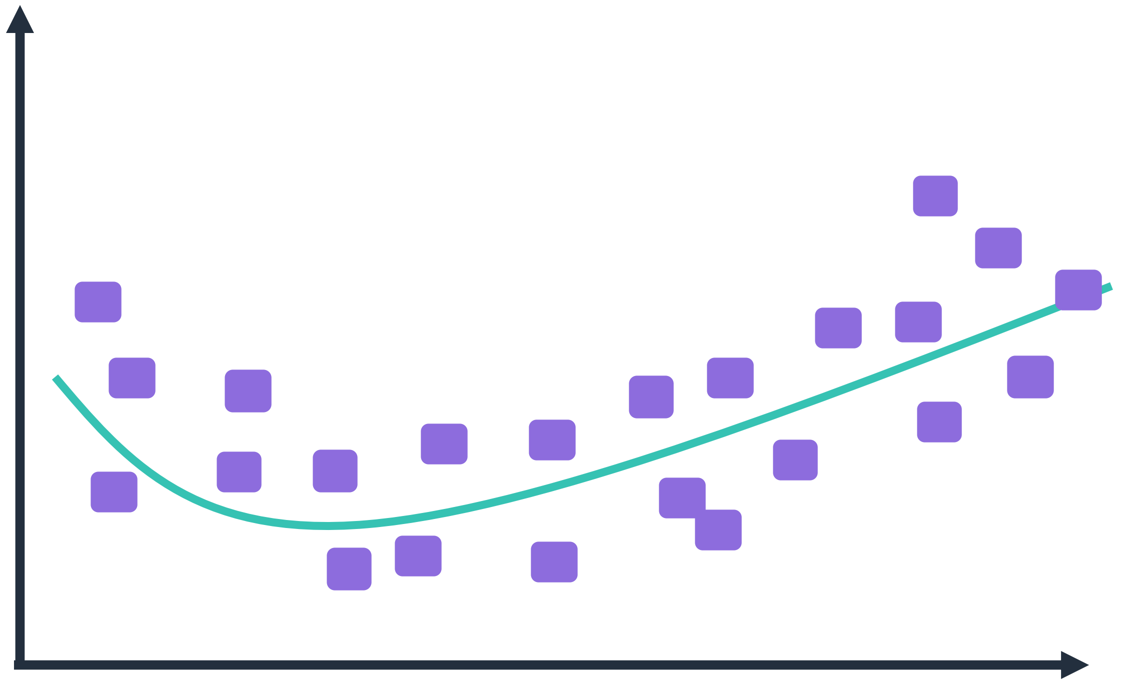
ML tuning process

- You have trained your model, performed a batch transformation on your test data, and calculated your metrics.
- Which was the better model? The metrics can help to inform you
- However, finally, you might tune the model's parameters, which is the next phase of the ML pipeline.



Recap: Goal for models

- The hyperparameters can be thought of as the knobs that tune the machine learning algorithm to improve its performance.
- Explore possible hyperparameters, and do optimization hyperparameters, so our final version of the tuned model should be "Balanced and generalized".


Balanced and generalizes well

Hyperparameter categories

- Hyperparameters have a few different categories. The first kind is **model hyperparameters**.
- For example, some neural-network-based models require you to define an architecture before you can start training them. The architecture includes a specific number of layers in the neural network and the activation functions that are used within.
- Suppose, it is a computer vision problem, in a neural network, additional attributes of the architecture must be defined. Such attributes include filter size, pooling, and stride or padding.

Model
Help define the model
Filter size, pooling, stride, padding

Hyperparameter categories

- The second kind is **optimizer hyperparameters**. These hyperparameters are related to how the model learns the patterns that are based on data. These types of hyperparameters include optimizers like gradient descent and stochastic gradient descent.
- They can also include optimizers that use momentum like Adam, or initialize the parameter weights by using methods such as Xavier initialization or He initialization.

Model	Optimizer
Help define the model	How the model learns patterns on data
Filter size, pooling, stride, padding	Gradient descent, stochastic gradient descent

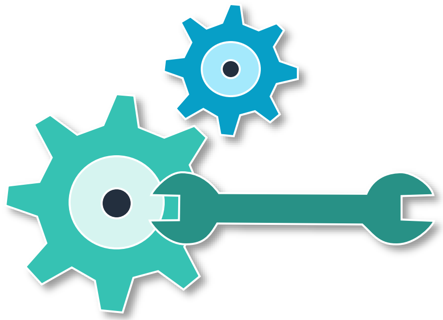
Hyperparameter categories

- The third kind is data hyperparameters, which relate to the attributes of the data itself.
- They include attributes that define different data augmentation techniques, such as cropping or resizing for image-related problems.
- They are often used when you don't have enough data or enough variation in your data.

Model	Optimizer	Data
Help define the model	How the model learns patterns on data	Define attributes of the data itself
Filter size, pooling, stride, padding	Gradient descent, stochastic gradient descent	Useful for small or homogenous datasets

Tuning hyperparameters

- Tuning hyperparameters can be labor-intensive.
- Traditionally, this kind of tuning was done manually. Someone—(who had domain experience that was related to that hyperparameter and use case) —would manually select the hyperparameters.



Manually select hyperparameters according to one's intuition or experience

However, this approach is often not as thorough or efficient as needed

- Then, they would train the model and score it on the validation data. This process would be repeated until satisfactory results were achieved.

XGBoost Hyperparameters

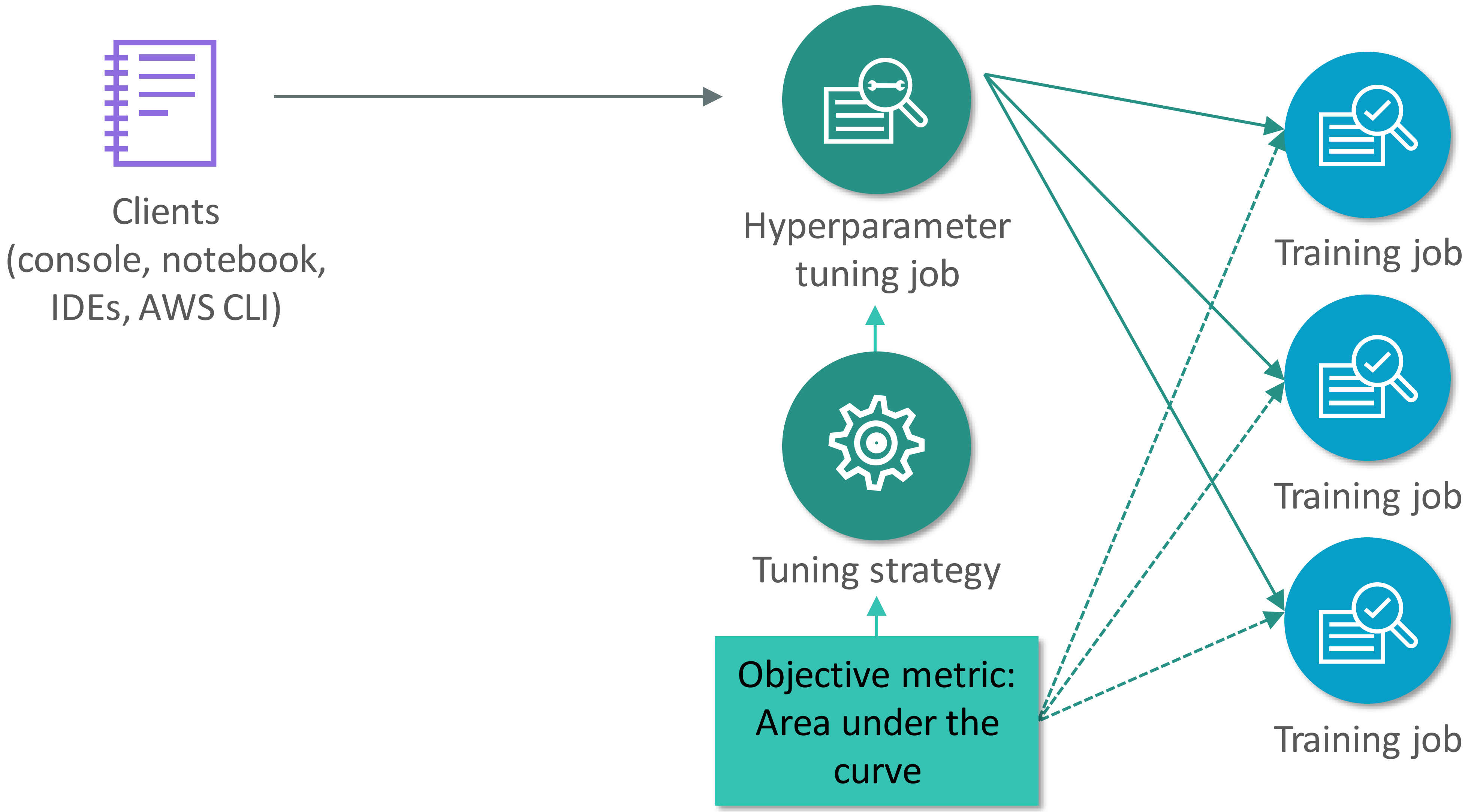
Guided Lab-7

- Guided Lab-7: Tuning model parameters.
- DEMO of Lab-7
  - It is part of your lab_homework_7. (see at myCourse)

Amazon SageMaker offers automated hyperparameter tuning

- Amazon SageMaker enables you to perform automated hyperparameter tuning.
- Amazon SageMaker automatic model tuning, finds the best version of a model by running many training jobs on your dataset.
  - It uses the algorithm and ranges of hyperparameters that you specify.
  - It then chooses the hyperparameter values that result in a model that performs the best, as measured by a metric that you choose.
  - It uses Gaussian Process regression to predict which hyperparameter values might be most effective at improving fit.
  - It also uses Bayesian optimization to balance exploring the hyperparameter space and exploiting specific hyperparameter values, when appropriate.
- The Automatic model tuning can be used with the Amazon SageMaker built-in algorithms, prebuilt deep learning frameworks, and bring-your-own-algorithm containers.

Amazon SageMaker offers automated hyperparameter tuning


Clients
(console, notebook, IDEs, AWS CLI)
Hyperparameter tuning job
Tuning strategy
Objective metric: Area under the curve
Training job
Training job
Training job

Automated hyperparameter tuning

Example: Suppose that you want to solve a binary classification problem on a fraud dataset. Your goal is to maximize the area under the curve (AUC) metric of the algorithm by training a linear learner algorithm model. You don't know which values of the learning rate, beta_1, beta_2, and epochs to use to train the best model.

- To find the best values for these hyperparameters, you can specify ranges of values for Amazon SageMaker hyperparameter tuning to search.
- It uses these ranges to run training jobs so it can find the combination of values that performs the best, as measured by the objective metric that you chose.
- It will then return to the training job with the highest AUC.

Model Name	Objective Metric	Eta	Max_depth	…
Model3	0.8	0.7	6	…
Model1	0.75	0.09	5	…
…	…	…	…	…

Tuning best practices

To improve optimization, use the following guidelines when you create hyperparameters.

- Don't adjust every hyperparameter
- Limit your range of values to what's most effective
- Run one training job at a time instead of multiple jobs in parallel
- In distributed training jobs, make sure that the objective metric that you want is the one that is reported back
- With Amazon SageMaker, convert log-scaled hyperparameters to linear-scaled when possible

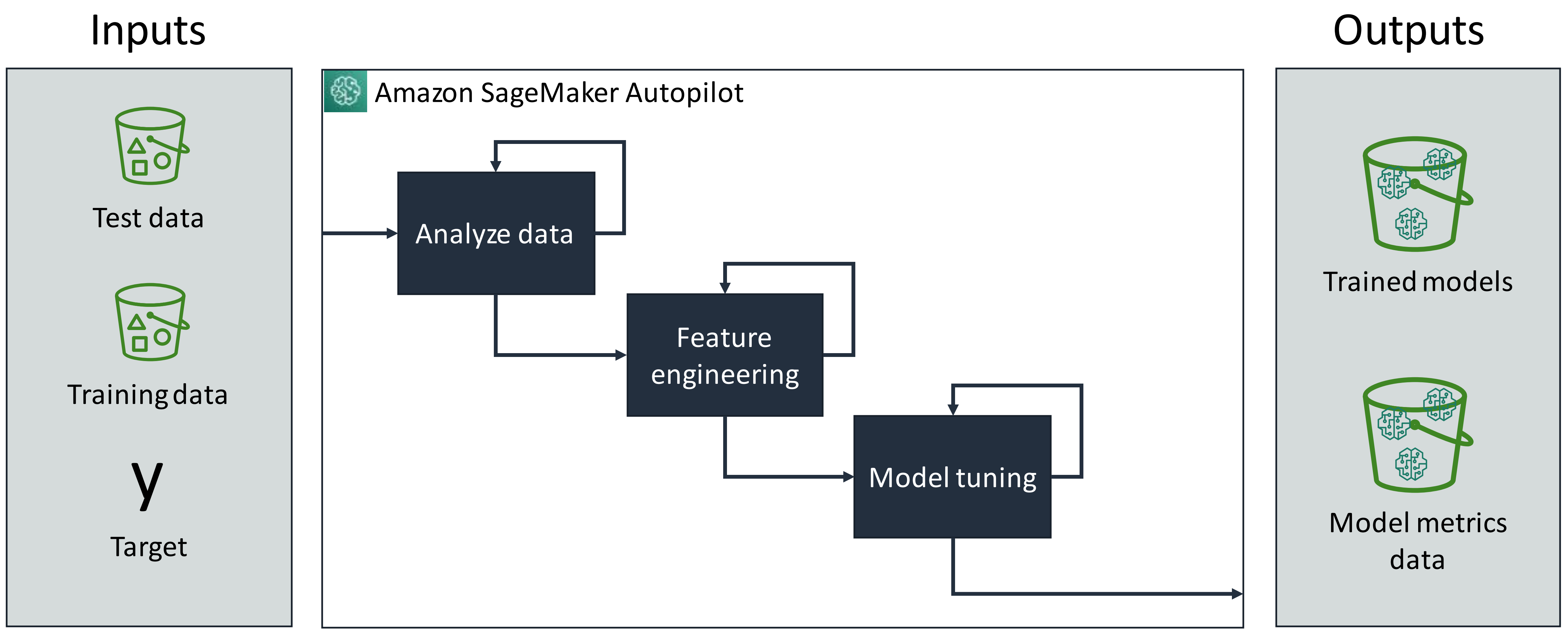
Note: Tuning doesn't always improve your model. It's an advanced tool for building machine solutions, and so it is part of the process of using the scientific method.

Amazon SageMaker Autopilot

- Now that you have walked through the end-to-end process of training and tuning an ML model, it's worth learning about Amazon SageMaker Autopilot.
- This service can help you find a good model, with little effort or input from you.
- With Autopilot, you create a job that supplies the test, training, and target.
- Autopilot analyzes the data, selects appropriate features, and then trains and tunes models. It documents the metrics and finds the best model that is based on the provided data.
- The results include the winning model, metrics, and a Jupyter notebook that you can use to investigate the results.

Note: This service doesn't remove your need to perform preprocessing of the data, but it can save time in feature selection and model tuning.

Amazon SageMaker Autopilot



Key Take Away

- Two types of model evaluation
  - Classification
    - Confusion matrix
    - AUC-ROC
  - Regression testing
    - Mean squared

- Model tuning helps find the best solution
- Hyperparameters
  - Model
  - Optimizer
  - Data
- Use Amazon SageMaker to help tune hyperparameters
- Use Autopilot for faster development

Chapter end questions

1. Explain the “Evaluate Model” phase of the ML pipeline with an example.
2. Explain the “Confusion matrix” with an example.
3. How the “Sensitivity” and “Specificity” can help with model tuning? with an example.
4. How “Area under curve-receiver operator curve (AUC-ROC)” can help to find out which model is better? Explain with an example.
5. What are the Precision and F1 Score? Explain with an example.
6. Explain the “Tune Model” phase of the ML pipeline with an example.
7. Describe categories of Hyperparameters.

Chapter-7
Introduction to Forecasting
(Using Amazon Forecast)

Credit note:
Slides, content, web links, and chapter-end questions are prepared from the AWS Academy course and other Internet resources. Students are advised to use it for study purposes only.

Chapter overview

Sections

1. Forecasting overview
2. Processing time series data
3. Using Amazon Forecast
4. Evaluating the forecast

Guided Lab

- Creating a Forecast with Amazon Forecast (non-graded lab)

Chapter objectives

At the end of this Chapter, you should be able to:

- Describe the business problems solved by using Amazon Forecast
- Describe the challenges of working with time series data
- List the steps that are required to create a forecast by using Amazon Forecast
- Use Amazon Forecast to make a prediction
- Evaluate your forecast

Overview of forecasting

- Forecasting is an important area of machine learning.
- Predict future values that are based on historical data.
- Many of these opportunities involve a time component
- You can think of time series data as falling into two broad categories.
  - The first type is univariate, which means that it has only one variable.
  - The second type is multivariate, which means that it has more than one variable.

Trending data

Seasonal data

Overview of forecasting

- In addition to these two categories, most time series datasets also follow one of the following patterns:
  - Trends: Patterns that increase, decrease, or are stagnant
  - Seasonal: Pattern that is based on seasons
  - Cyclical: Other repeating patterns
  - Irregular: Patterns that might appear to be random

Cyclical data

Irregular data

Forecasting use cases

Sales and demand forecasts

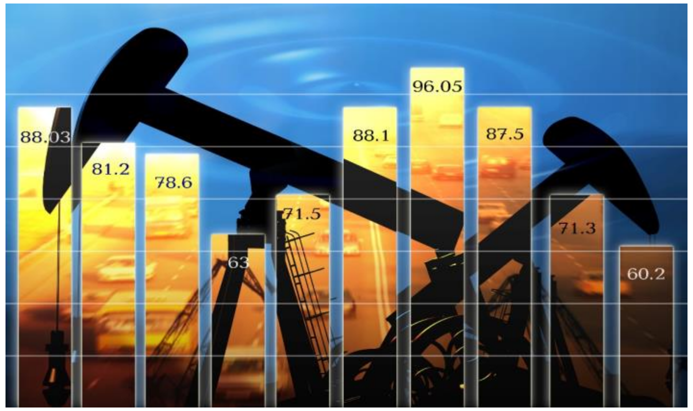
Inventory

Forecasting use cases

- Marketing applications, such as sales forecasting or demand projections.
- Inventory management systems to anticipate required inventory levels. Often, this type of forecast includes information about delivery times.

Forecasting use cases

- Energy consumption to determine when and where energy is needed.
- Weather forecasting systems for governments, and commercial applications such as agriculture.



Energy consumption

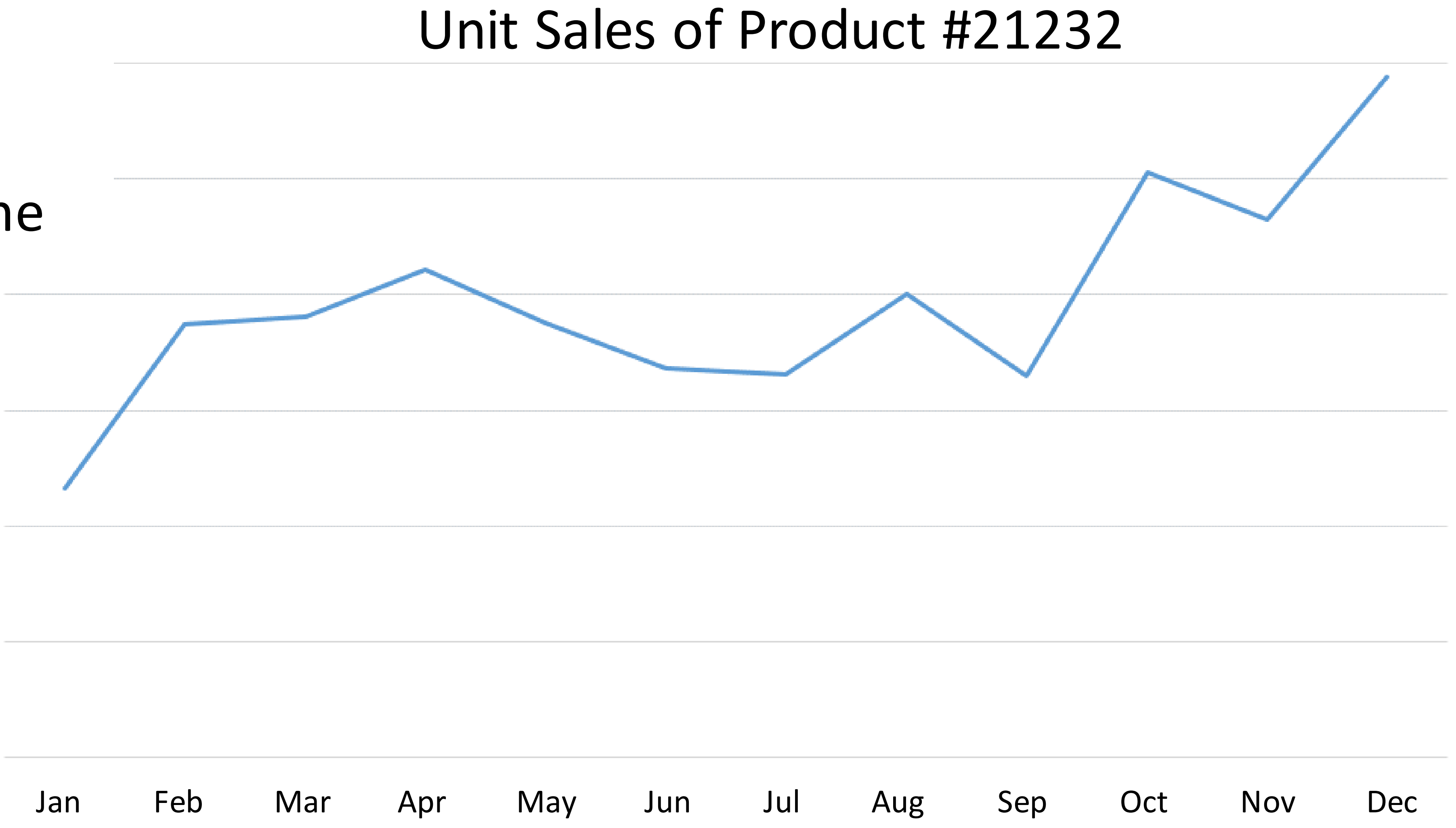
Weather forecasts

Time series data

Time series data is captured in sequence over time

Related data informs the time series data—for example, price or promotions

Metadata might also be needed to explain predictions—for example, brand name or category



Time series data

- Time series data is captured in chronological sequence over a defined period of time.
- Introducing time into a machine learning model has a positive impact because the model can derive meaning from changes in the data points over time.
- Time series data tends to be correlated, which means that a dependency exists between data points.

- In addition to the time series data, you can add related data to augment a forecasting model.
- For example, for a prediction about retail sales, you might include information about the product being sold, like sales price. This information is in addition to the number of units that are sold per time period.
- Another type of data is metadata about the dataset. For example, for a retail dataset, you might want to include metadata to group results, like a brand name.
- Another example of metadata could be including a genre for music or videos.

Time and date challenges

Incomplete and varying timestamps

- The more data that you have, the better.
- A challenge that you can expect with multiple data sources is the timestamp of the data.
  - For example, imagine that you have some data that contains both the month and the day, but no year.
  - Suppose that the data appears to sequence through the month numbers in the database, and repeats after 12. In that case, you can add the year if you know when the data started. You can infer future years, based on the order of the data.

yyyy-mm-dd HH:MM:SS	Includes time
yyyy-dd-mm	Year, day, month
yyyy-mm-dd	Year, month, day
yyyy-mm	No day
ss:s	Seconds
mm-dd	No year

Time and date challenges

UTC, local, and time zones: **Is the time in UTC format?**

2020-06-02T13:15:30Z

- Much data is stored in Universal Coordinated Time (UTC) format.
- Sometimes the timestamp doesn’t represent the time that you think it does.
  - For example, suppose you have a database of cars that were serviced at a garage. Does the timestamp indicate the time that the car arrived, was completed or was picked up? Or does it indicate when the final entry was entered into the system?
  - In another example, If you try to the model hourly caloric intake of patients, but you have only daily data, then you must adjust your target timescale.
  - In some cases, you might not have a timestamp present in your data. You might have other ways to extrapolate a time series, depending on the data and domain. For example, you might have wavelength measurements or vectors in an image.

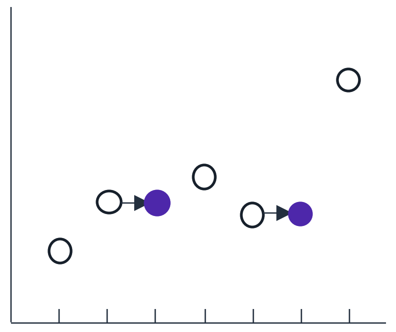
Time series handling: Missing data

- A common occurrence in real-world forecasting problems is missing values in the raw data.
- Missing values make it harder for a model to generate a forecast.
- The primary example in retail is an out-of-stock situation in demand forecasting. If an item goes out of stock, the sales for the day will be zero. If the forecast is generated based on those zero sales values, the forecast will be incorrect.
- Missing values can occur because of no transaction, or possibly because of measurement errors.
- Maybe a service/ device that monitored certain data was not working correctly, or the measurement could not occur correctly.

- The missing data can be calculated in several ways:
  - Forward fill – Uses the last known value for the missing value.
  - Moving average – Uses the average of the last known values to calculate the missing value.
  - Backward fill – Uses the next known value after the missing value. Be aware that it is a potential danger to use the future to calculate the past, which is bad for forecasting. This practice is known as lookahead, and it should be avoided.
  - Interpolation – Essentially uses an equation to calculate the missing value.

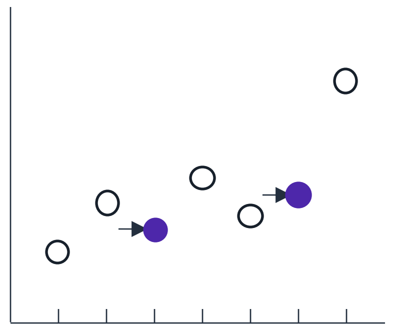
Time series handling: Missing data

Forward Fill



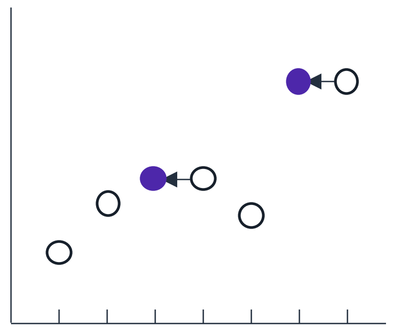
Last known == missing value

Moving Average



Avg(previous) == missing value

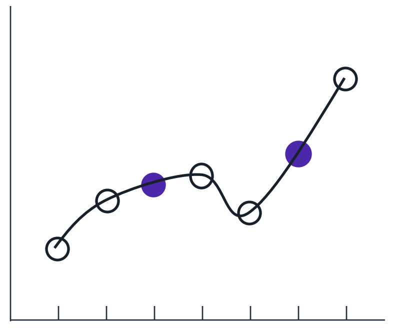
Backward Fill



Next value == missing value

Danger! Lookahead risk!

Interpolation

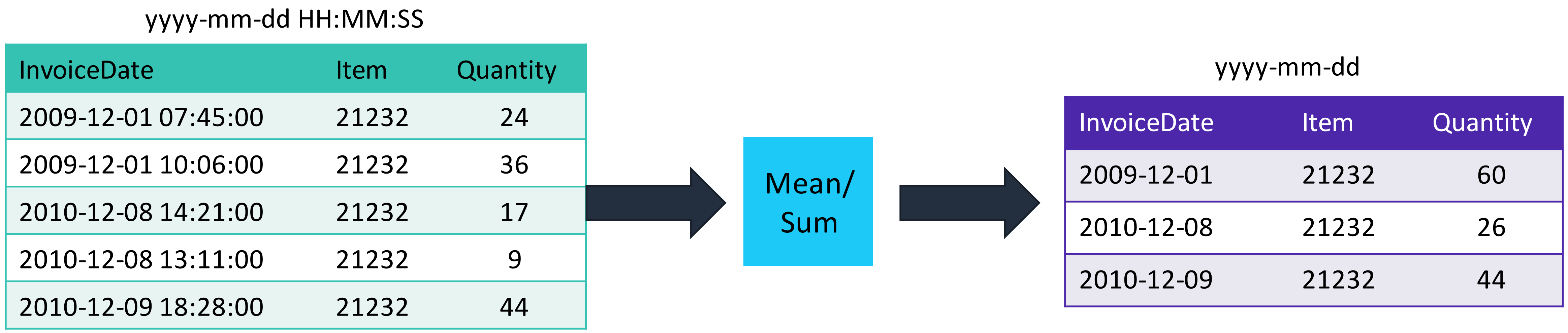


Linear
Spline
Polynomial

Note: Zero is sometimes the perfect fill value

Time series handling: Downsampling

- You might obtain data at different frequencies. For example, you might have sales data that includes the exact timestamp that the sale was recorded. However, the inventory data might contain only the year, month, and day of the inventory level. When you have data that is at a different frequency than other datasets or isn't compatible with your question, you might need to *downsample*. This example converts an hourly dataset to a daily dataset.



yyyy-mm-dd HH:MM:SS

InvoiceDate	Item	Quantity
2009-12-01 07:45:00	21232	24
2009-12-01 10:06:00	21232	36
2010-12-08 14:21:00	21232	17
2010-12-08 13:11:00	21232	9
2010-12-09 18:28:00	21232	44

yyyy-mm-dd

InvoiceDate	Item	Quantity
2009-12-01	21232	60
2010-12-08	21232	26
2010-12-09	21232	44

Downsample means moving from a more finely-grained time to a less finely-grained time.

Time series handling: Upsampling

yyyy-mm-dd

InvoiceDate	Item	Quantity
2009-12-01	21232	60
2010-12-08	21232	26
2010-12-09	21232	44

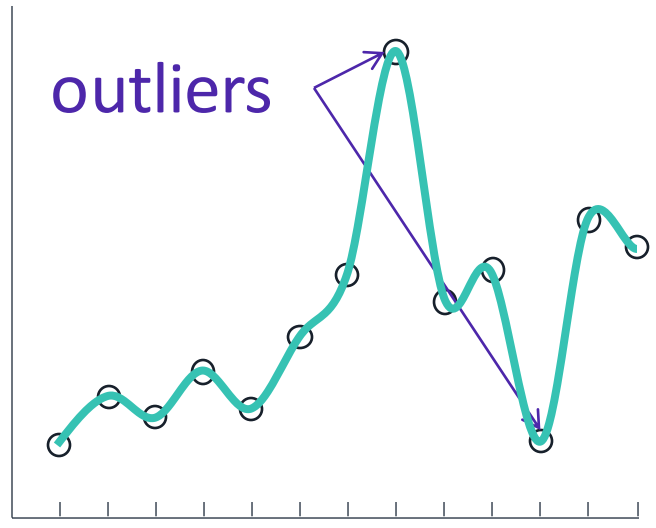
?

yyyy-mm-dd HH:MM:SS

InvoiceDate	Item	Quantity
2009-12-01 ??:??:??	21232	??
2009-12-01 ??:??:??	21232	??
2010-12-08 ??:??:??	21232	??
2010-12-08 ??:??:??	21232	??
2010-12-09 ??:??:??	21232	44

- Reasons to Upsample: Match different time series, Irregular time series, getting knowledge of the domain.
- The problem with upsampling is that it's difficult to achieve in most cases.
- Suppose that you wanted to upsample your sales data from daily sales to hourly sales.
- Unless you have some other data source to reference, you wouldn't be able to change from daily to hourly sales.

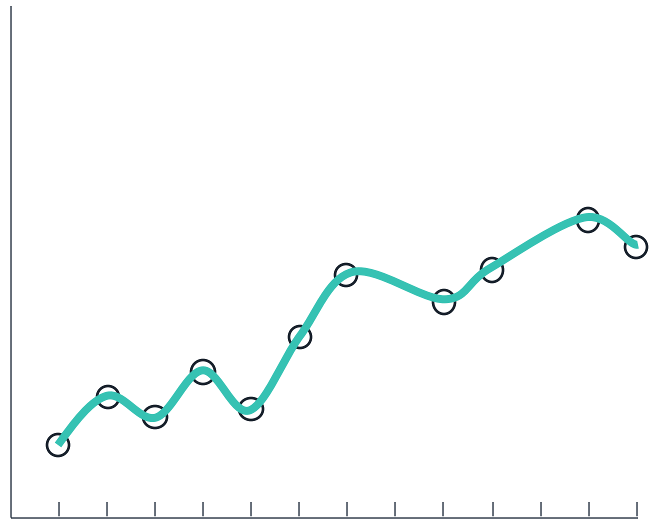
Time series handling: Smoothing data



Smoothing function

Why are you smoothing?

- Data Preparation - Removing error values and outliers
- Visualization - Reducing noise.



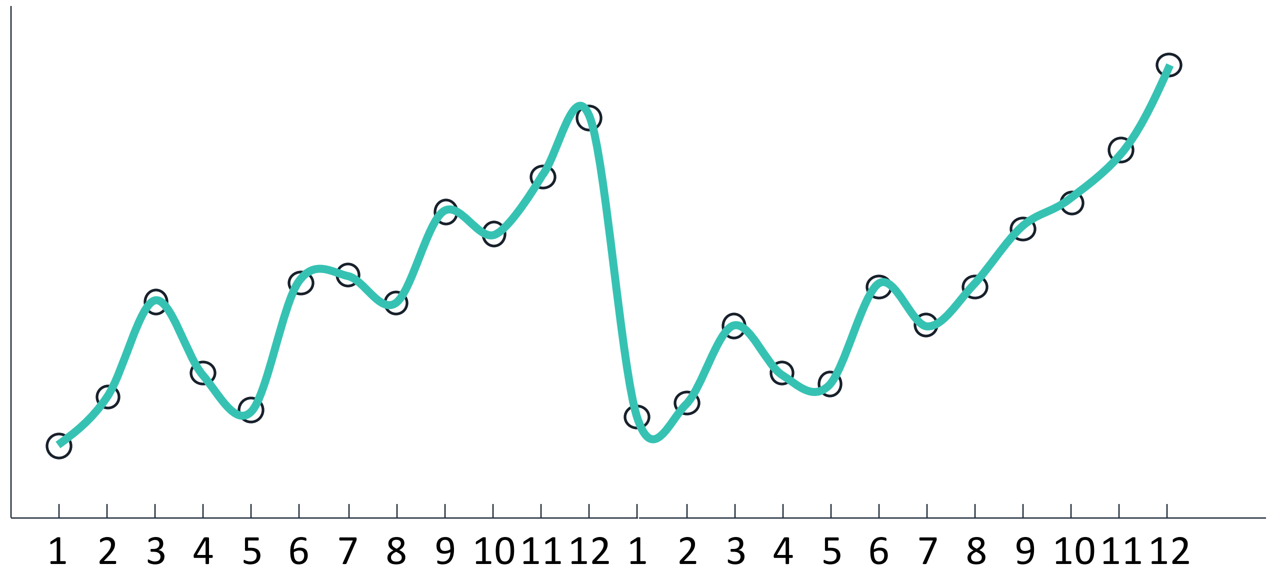
- If you examine sales data and you see an order with an unusually large quantity of items, you might not want to include that order in your forecast calculations.
- The order size might never be repeated. Removing these outliers and anomalies is known as smoothing.

How does smoothing affect your outcome?

- Cleaner data to model
- Model compatibility
- Production improvements

Seasonality

- Seasonality in data is any kind of repeating observation where the frequency of the observation is stable.
- Seasonality frequency:
  - Hourly, daily, quarterly, yearly
  - Spring, summer, fall, winter
  - Major holiday sales, winter holiday season



- Incorporating holidays: For example, in sales, you typically see higher sales at the end of a quarter, and into Q4.
- Consumer retail exhibits this pattern even more in Q4. Be aware that data can have multiple types of seasonality in the same dataset.

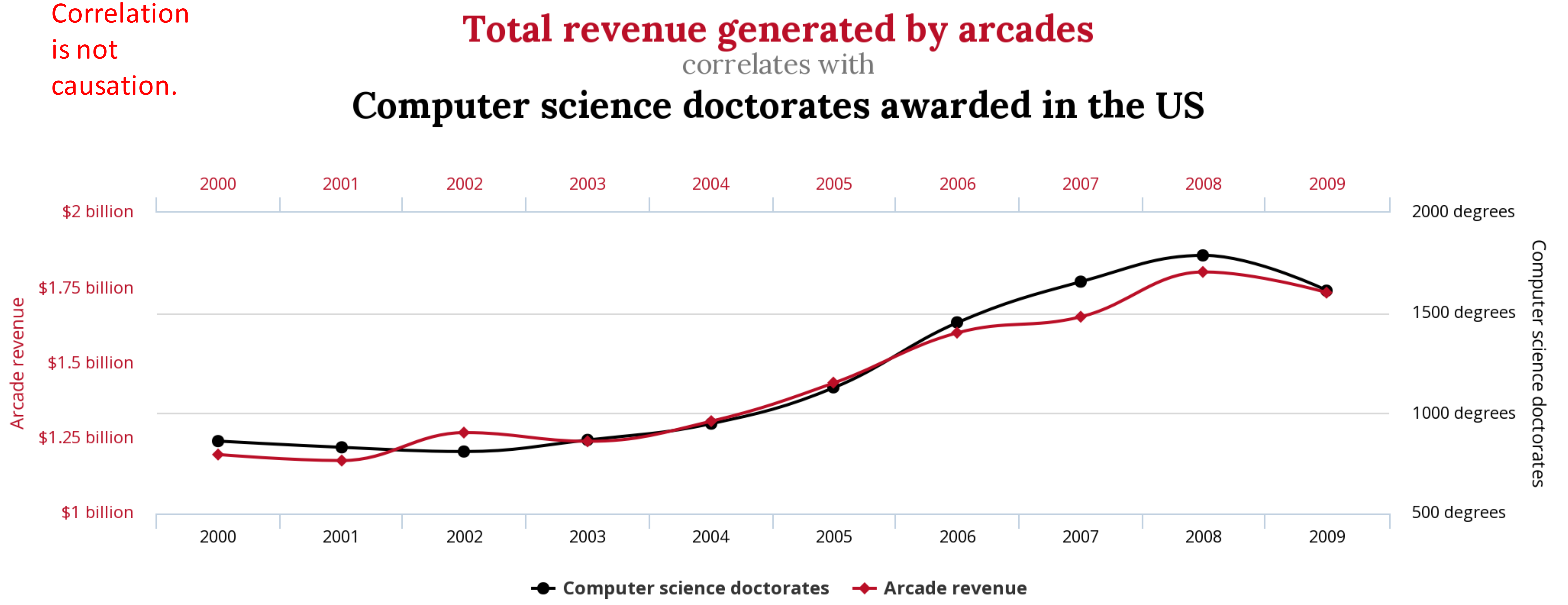
Stationarity, trends, and autocorrelation

- Stationarity: It is important to know how stable a system is. The level of stability, or stationarity, can tell you how much you should expect the system's past behavior to inform future behavior. A system with low stability is not good for predicting the future.
  - How stable is the system?
  - Does the past inform the future?

- Trends: You need to determine the trend for a time series. However, adjusting the series for the trend can make it difficult to compare the series with another series that was also adjusted for the trend. The trends might dominate the values in the series, which can lead you to overestimate the correlation between the two series.
- Note: For a detailed explanation of reasons for this occurrence, see "Avoiding Commons Mistakes with Time Series Analysis"

Time series correlations

Correlation is not causation.

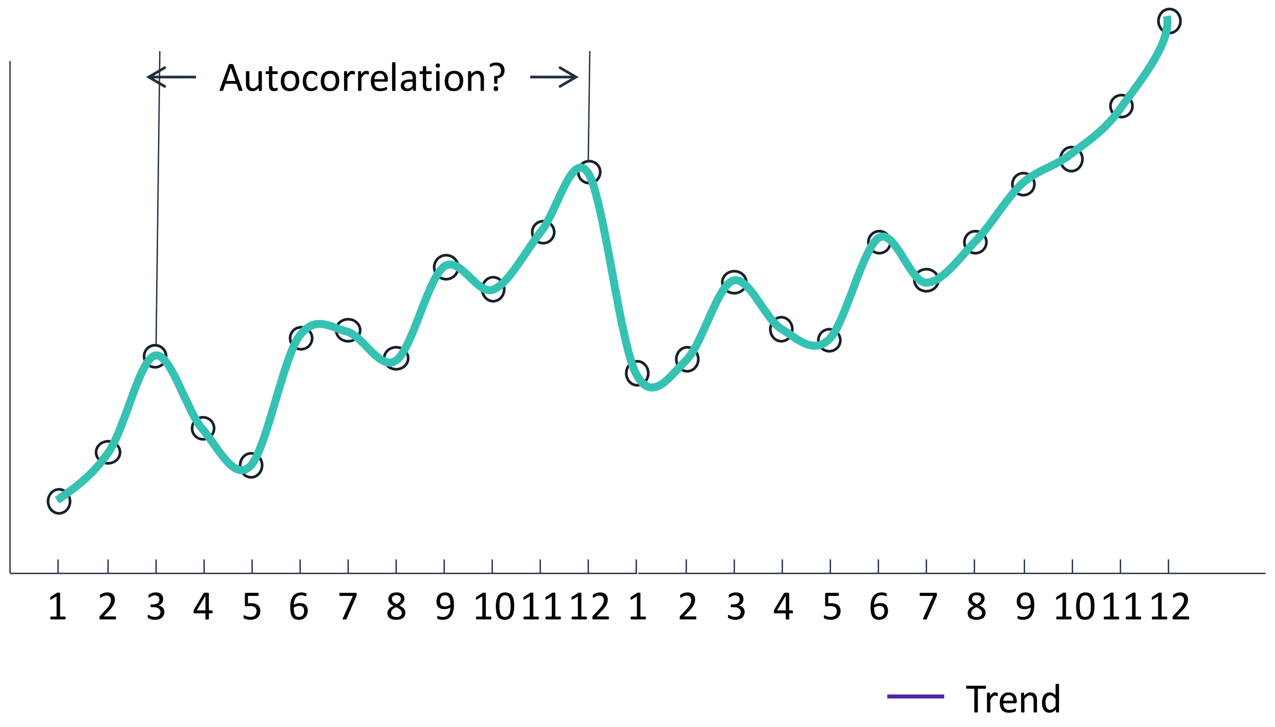


This chart is originally from Tyler Vigen: Spurious Correlations

tylervigen.com

Stationarity, trends, and autocorrelation

- Autocorrelation:
  - How points in time are linearly related
  - Influences algorithm choice.
- Autocorrelation is one of the special problems that you face with time series data. Autocorrelation is a form of noise because separate observations are not independent of each other.



- A time series with autocorrelation might overstate the accuracy of the model that is produced. These factors, along with seasonality, can influence the model that you select to produce your forecast.
- Some algorithms handle seasonality and autocorrelation, but others do not.

Using pandas for time series data

- Time-aware index

```
dataframe['2010-01-04']
```

```
dataframe['2010-02':'2010-03']
```

```
dataframe['weekday_name'] = dataframe.index.weekday_name
```

- df_time_series = df_time_series.groupby('StockCode').resample('D').sum().reset_index()

- The pandas’ libraries were developed with financial data analysis in mind. As such, it is also good at handling time series data.
- You can set the index of your pandas DataFrame to be a datetime, which enables you to use the date and time to select your data. You can use ranges that contain partial dates.
- You can also extract date parts, such as year, month, weekday_name, and more.

Using pandas for time series data

- For grouping and resampling tasks, pandas has built-in functions to do both.
- GroupBy and resampling operations

```
dataframe.groupby('StockCode')
```

```
dataframe.groupby('StockCode').resample('D').sum()
```

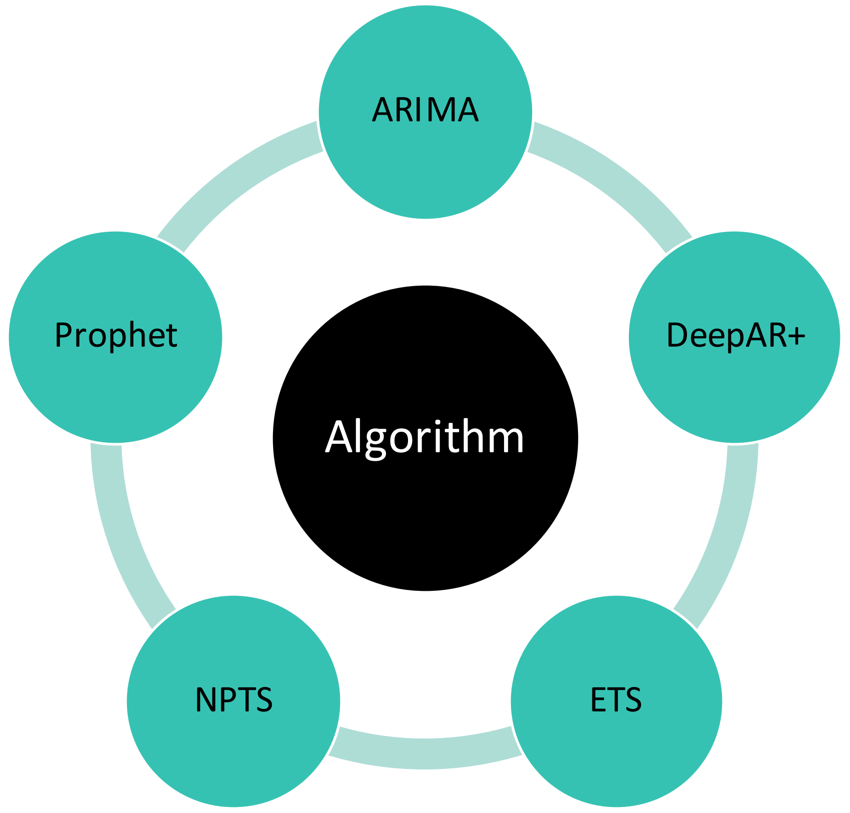
- pandas can give you insights into autocorrelation.

```
dataframe['Quantity'].autocorr()
```

- For more information about pandas and time series data, see Time series / date functionality., In the pandas' documentation

Time series algorithms

- One of the tasks in building a forecasting application is to choose an appropriate algorithm.
- The type of dataset that you are using and the features of that dataset should determine your choice of algorithm. Amazon Forecast supports these five algorithms.
  - Autoregressive Integrated Moving Average (ARIMA)
  - DeepAR+
  - Exponential Smoothing (ETS)
  - Non-Parametric Time Series (NPTS)
  - Prophet



Time series algorithms

- Autoregressive Integrated Moving Average (ARIMA): This algorithm removes autocorrelations, which might influence the pattern of observations.
- DeepAR+ : A supervised learning algorithm for forecasting one-dimensional time series. It uses a recurrent neural network to train a model over multiple time series.
- Exponential Smoothing (ETS): This algorithm is useful for datasets with seasonality. It uses a weighted average for all observations. The weights are decreased over time.
- Non-Parametric Time Series (NPTS) : Predictions are based on sampling from past observations. Specialized versions are available for seasonal and climatological datasets.
- Prophet: A Bayesian time series model. It's useful for datasets that span a long time, have missing data, or have large outliers.

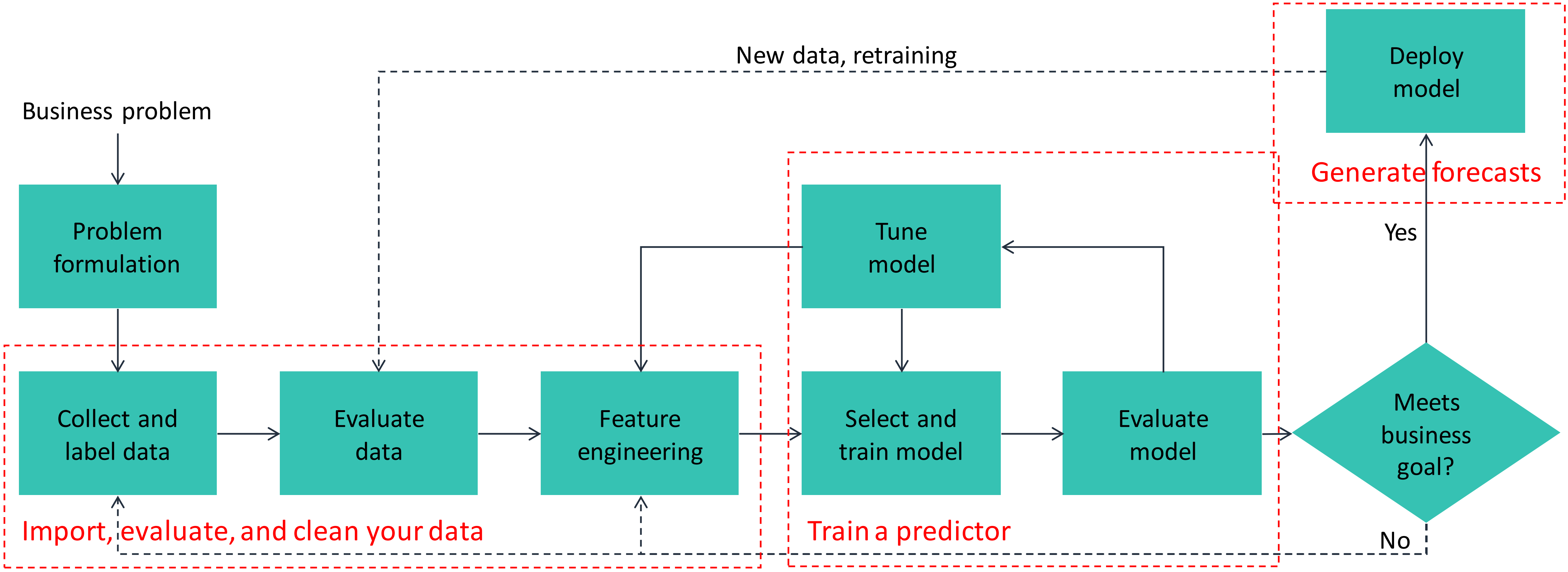
https://docs.aws.amazon.com/forecast/latest/dg/aws-forecast-choosing-recipes.html

Some key takeaways from this section

<ins>Some key takeaways from this section</ins>:

- Time series data is sequenced data that includes a time element, which makes it different from regular datasets
- Some of the time challenges include –
  - Handling different time formats
  - Handling missing data through down sampling, up sampling and smoothing
  - Handling seasonality, such as weekdays and yearly cycles
  - Avoiding bad correlations
- The pandas library offers support for time series data through functions that deal with time
- With Amazon Forecast, you can choose between five algorithms –
  - ARIMA, DeepAR+, ETS, NPTS, Prophet

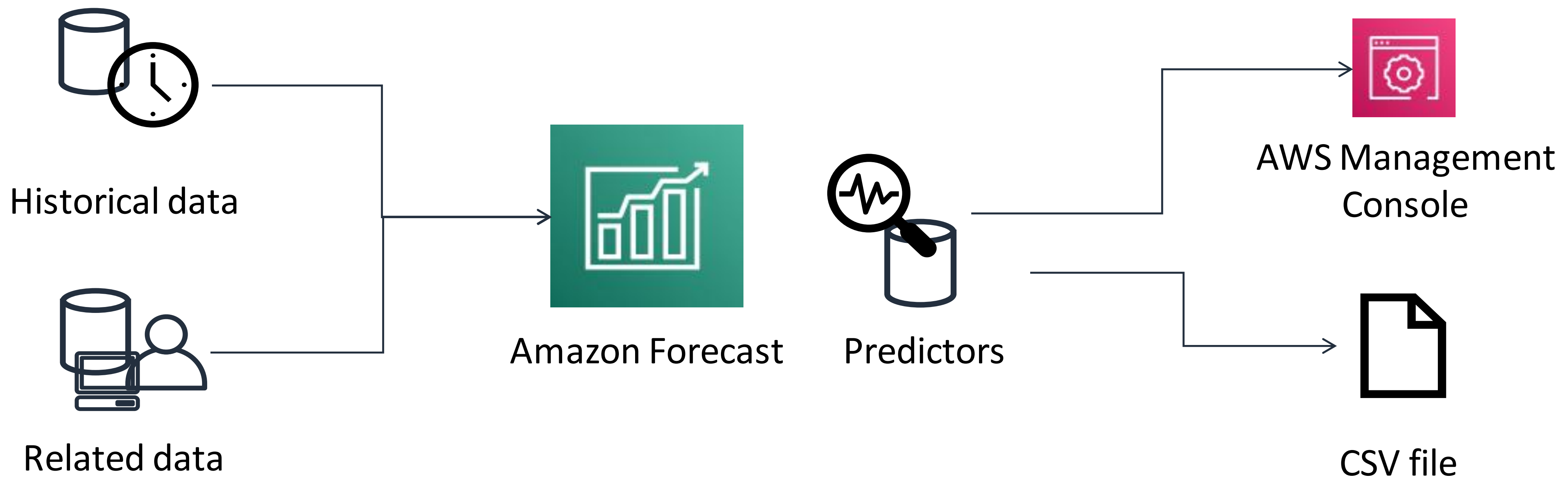
Amazon Forecast workflow



Amazon Forecast workflow

- When you generate forecasts, you can apply the ML development pipeline that you use throughout this course.
  1. Import your data – You must import as much data as you have—both historical data and related data. You should do some basic evaluation and feature engineering before you use the data to train a model.
  2. Train a predictor – To train a predictor, you must choose an algorithm. If you are not sure which algorithm is best for your data, you can let Amazon Forecast choose by selecting AutoML as your algorithm. You also must select a domain for your data, but if you're not sure which domain fits best, you can select a custom domain. Domains have specific types of data that they require.
  3. Generate forecasts – As soon as you have a trained model, you can use the model to make a forecast by using an input dataset group. After you generate a forecast, you can query the forecast, or you can export it to an Amazon S3 bucket. You also have the option to encrypt the data in the forecast before you export it.

Amazon Forecast overview

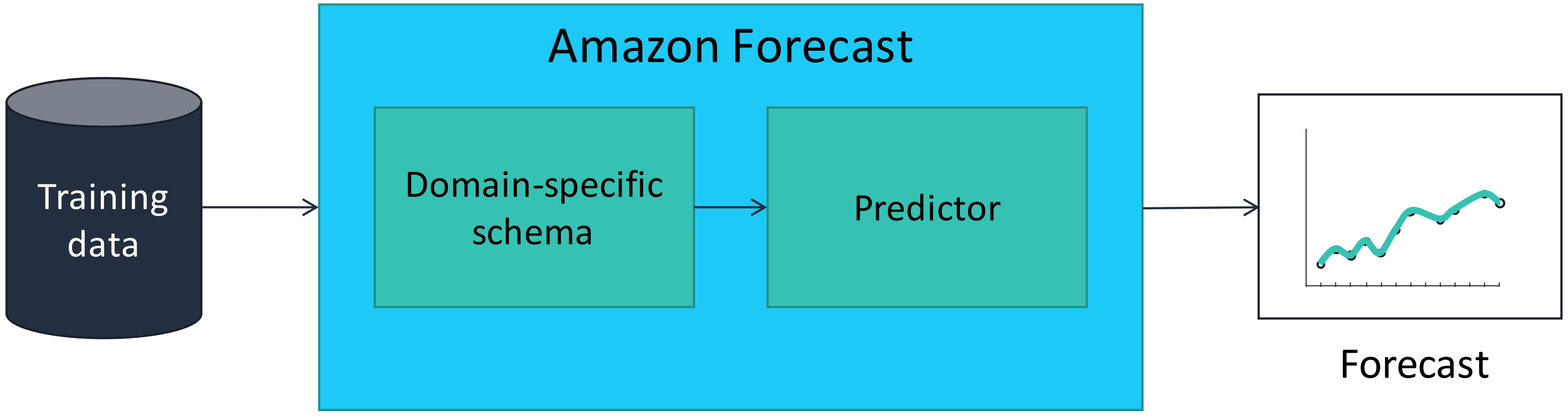


The overall process for working with Amazon Forecast is to 1) import historical and related data. 2) Amazon Forecast inspects the data, identifies key data, and selects an appropriate algorithm. 3) It uses the algorithm to train and optimize a custom model and produce a predictor. 4) You create forecasts by applying the predictor to your dataset. 5) Then, you can either retrieve these forecasts in the AWS console, or export the forecasts as comma-delimited files.

Supported domains

- Retail
- Inventory planning
- Work force
- Web traffic
- Metrics
- Custom

https://docs.aws.amazon.com/forecast/latest/dg/howitworks-domains-ds-types.html

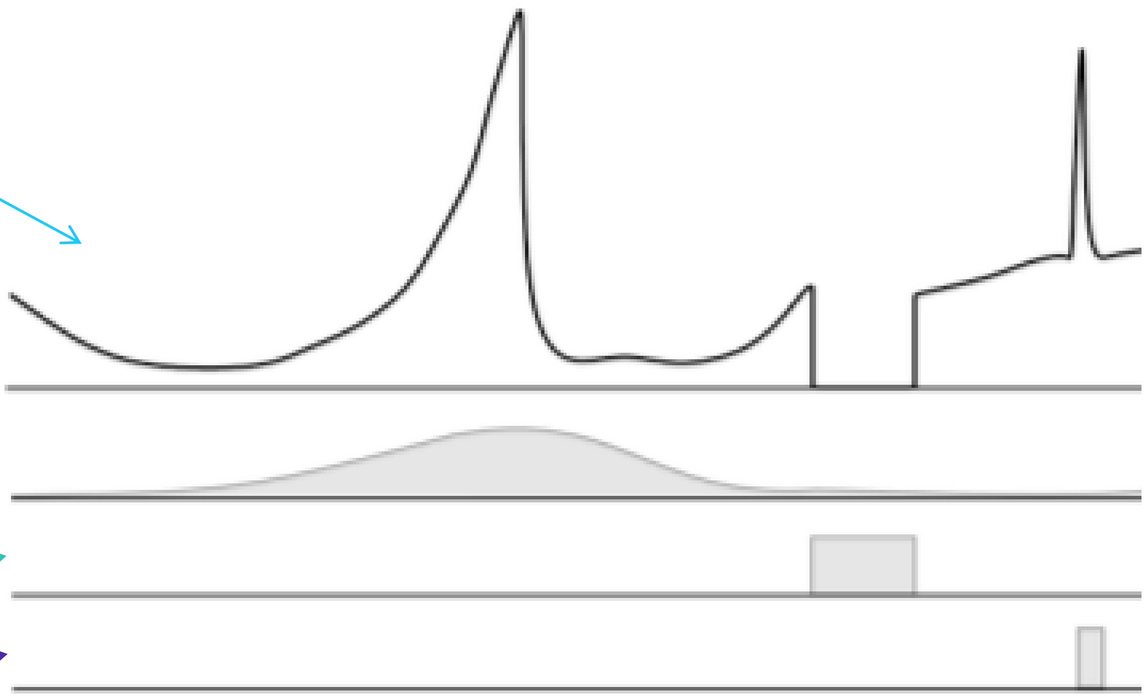


Web traffic forecast example

- Time series data
  - Webpage ID
  - Page views per month
  - Timestamp
- Related data
  - Page category (such as navigation, or content category)
  - Geographic identifier for web client
- Metadata
  - Region
  - Sales promotion information

Retail forecasting example

- Time series data
  - Transactional sales data
  - Timestamp, item, quantity
- Metadata
  - Category, item color, brand
  - Item, metadata
- Related data
  - Time series
  - In-stock data
  - Promotion data
  - Timestamp, item, price



https://docs.aws.amazon.com/forecast/latest/dg/retail-domain.html

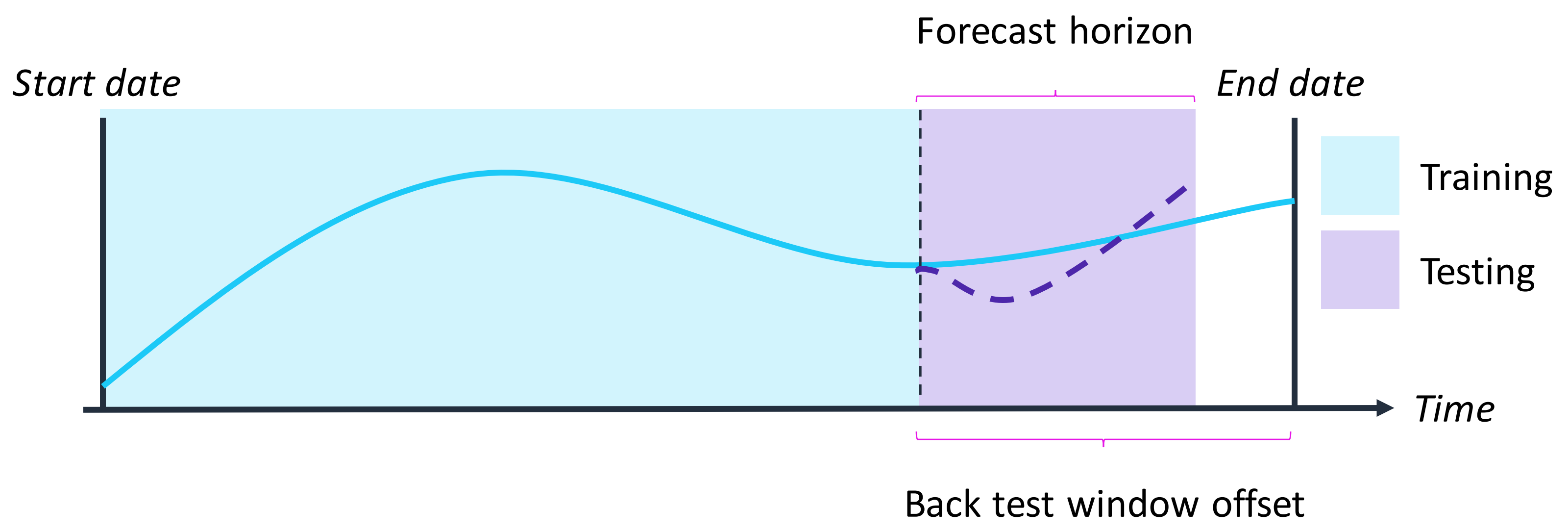
Guided Lab:

Guided Lab:

Creating a Forecast with Amazon Forecast

Evaluating your forecast: Back testing

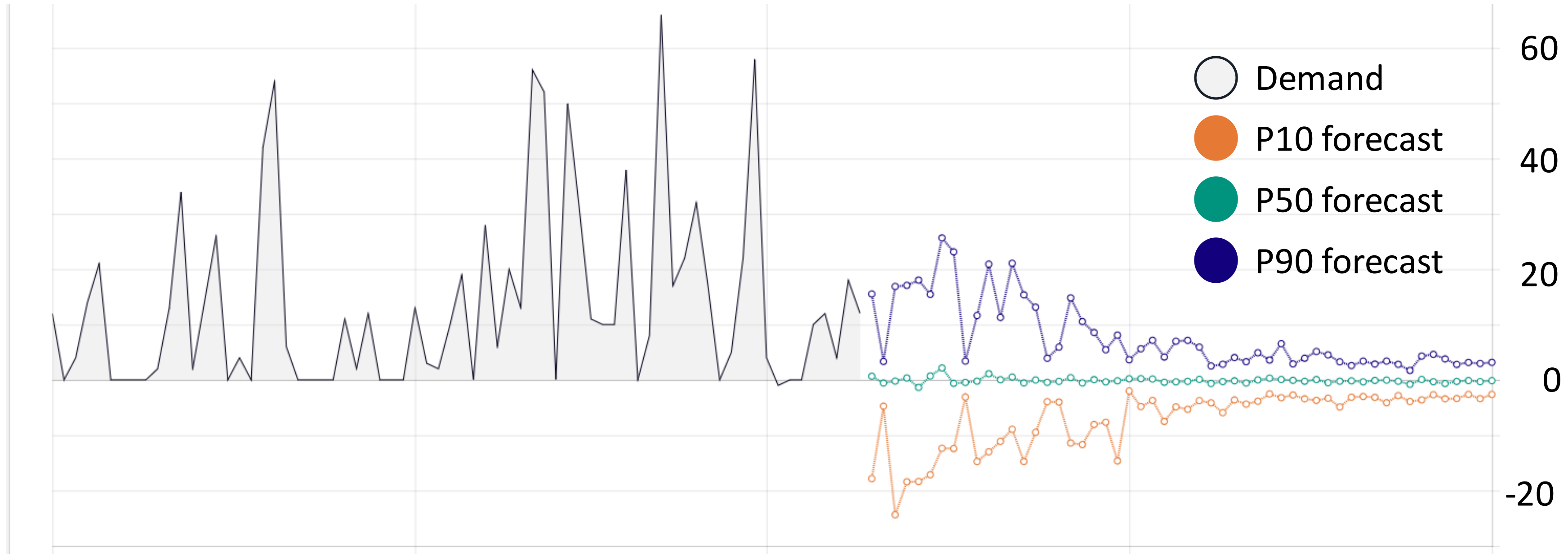
Predictor accuracy metrics are based on back testing.



Backtest window 1	Training	Testing
Backtest window 2	Training	Testing
Backtest window 3	Training	Testing

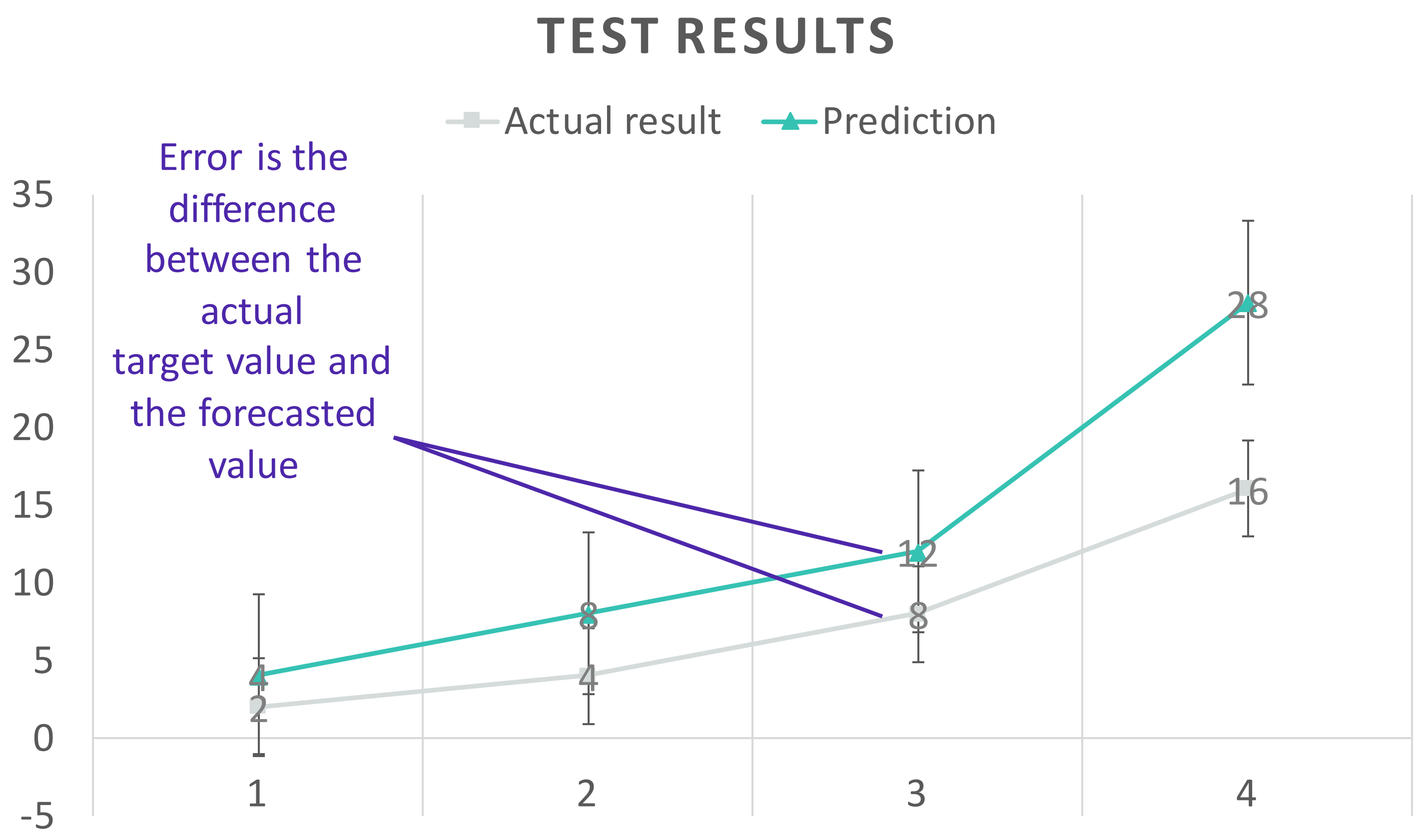
Evaluation metrics: wQuantileLoss

- Quantiles determined for 10%, 50%, and 90% quantiles
- wQuantileLoss is the average error for each quantile in a set –
  - Works best for models with greater variability in the errors



Root mean square error (RMSE)

RMSE is the square of the errors.



Test	Actual result	Prediction	Deltas
1	2	4	2
2	4	8	4
3	8	12	4
4	16	28	12
		RMSE	6.708204

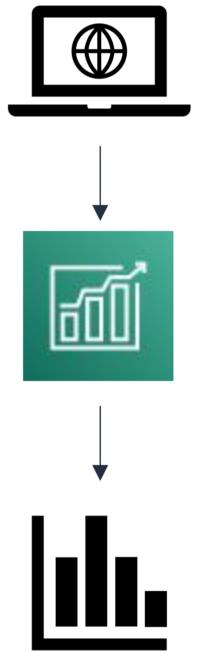
Works best for a model where most errors are of similar size

Model accuracy example

Web retailer of shoes wants to predict how often it will be unable to fill orders for AnyCompany brand shoes.

Amazon Forecast predicts demand of 1,000 pairs per month

- P10: 10% of the time, fewer than 880 pairs will be ordered
- P50: 50% of the time, fewer than 1,050 pairs will be ordered
- P90: 90% of the time, fewer than 1,200 pairs will be ordered



P10 = 880
P50 = 1050
P90 = 1200

Forecast = 1000

Some key takeaways from this section

Some key takeaways from this section:

- You can use Amazon Forecast to train and use a model for time series data
- There are specific schemas defined for domains such as retail and EC2 capacity planning, or you can use a custom schema
- You need to supply at least the time series data, but can also provide metadata and related data to add move information to the model
- As with most supervised machine learning problems, your data is split into training and testing data, but this split takes into account the time element
- Use RMSE and wQuantileLoss metrics to evaluate the efficiency of the model

Chapter summary

In summary, in this Chapter you learned how to:

- Describe the business problems solved by using Amazon Forecast
- Describe the challenges of working with time series data
- List the steps that are required to create a forecast by using Amazon Forecast
- Use Amazon Forecast to make a prediction

Additional resources

- Amazon Forecast documentation
- Amazon Forecast product page
- How to not use machine learning for time series forecasting
- Time series forecasting principles Amazon Forecast whitepaper

Chapter end questions

1. What is Forecasting? Describe related use cases.
2. How "Time series handling for Missing data" can be done in real-world forecasting problems?
3. Describe features of any one forecasting algorithm of your choice. (you can prepare an answer from the link https://docs.aws.amazon.com/forecast/latest/dg/aws-forecast-choosing-recipes.html
4. Write on any one "Amazon Forecast evaluation metric" with an example.
5. Describe the Amazon forecasting phases, using the diagram.

Chapter-8
Introduction to Natural language processing

Chapter overview

1. Overview of natural language processing (NLP)
2. Natural language processing managed services from AWS:
   - Amazon Polly
   - Amazon Comprehend
   - Amazon Translate
   - Amazon Transcribe
   - Amazon Lex

Guided Lab:

- Creating a Bot for Schedule Appointments

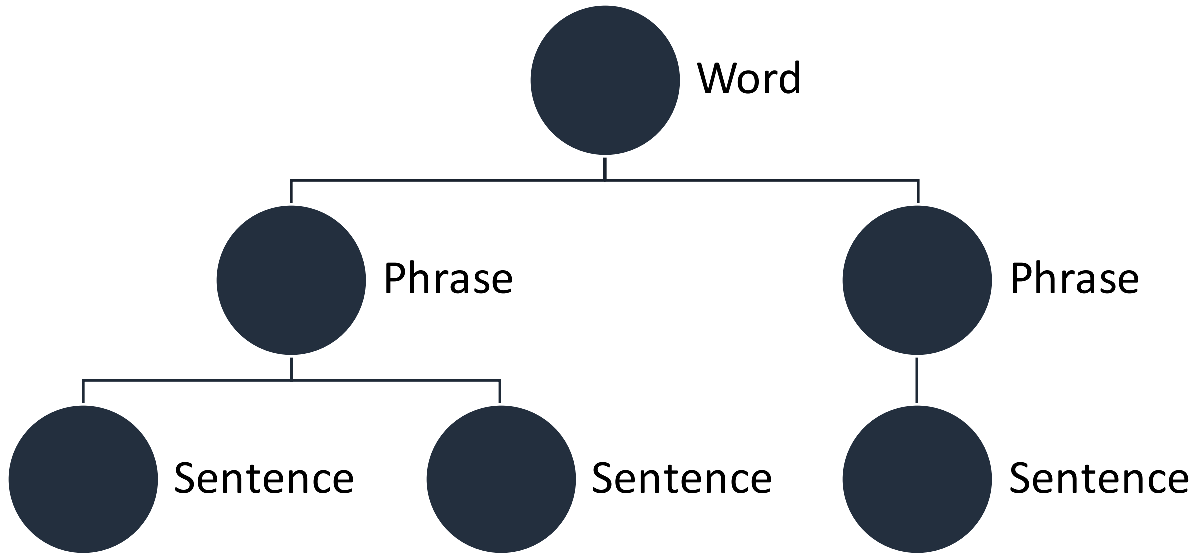
Chapter objectives

At the end of this Chapter, you should be able to:

- Describe the natural language processing (NLP) use cases that are solved by using managed Amazon ML services
- Describe the managed Amazon ML services available for NLP
- Use managed Amazon ML Services

What is NLP?

- NLP is a broad term for a general set of business or computational problems related to natural language processing that you can solve with machine learning.
- NLP systems predate ML. Two examples are speech-to-text on your old cell phone and screen readers.
- Many NLP systems now use some form of machine learning.



- NLP considers the hierarchical structure of language. Words are at the lowest layer of the hierarchy. A group of words makes a phrase. The next level up consists of phrases, which make a sentence, and ultimately, sentences convey ideas.

Natural language processing (NLP)

- A device, such as an Echo, records your words. The recording of your speech is sent to Amazon servers to be analyzed more efficiently.
- Its breaks down your phrase into individual sounds. Then, it connects to a database that contains the pronunciations of various words to find which words most closely correspond to the combination of individual sounds.
- It identifies important words to make sense of the tasks and to carry out corresponding functions.

"Hey, Alexa, what's it like outside?"

For instance, if Alexa notices words like outside or temperature, it opens the weather app. Then servers send the information back to your device, and Alexa might speak.

NLP challenges

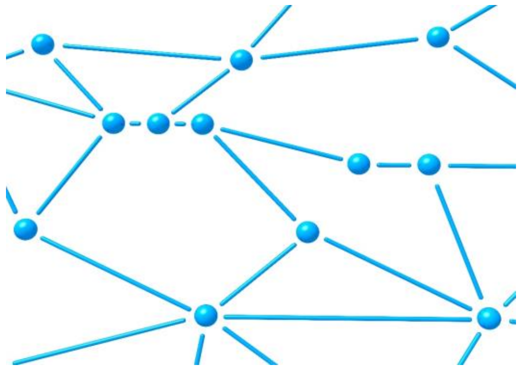
Lack of precision

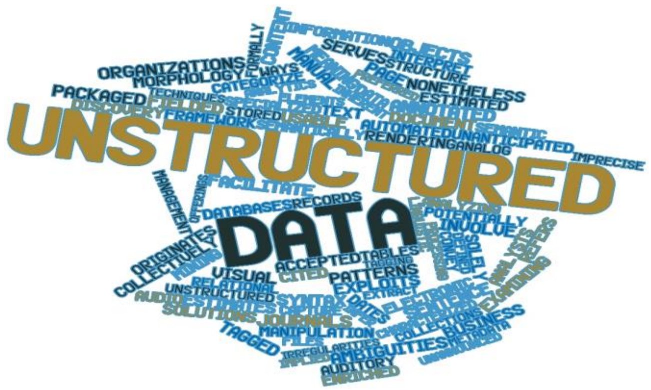
Meaning that is based on context

- Language is not precise.
- Words can have different meanings, which are based on the other words that surround them (context). Often, the same words or phrases can have multiple meanings.
- For example, consider the term weather. You might be under the weather, which means that you are sick. However, there is wonderful weather outside means that the weather conditions outside are good.
- In another example, The phrase Oh, really? might convey surprise, disagreement, or many other meanings, depending on context and inflection.

NLP challenges

- Some of the main challenges for NLP include:
- Discovering the structure of the text – One of the first tasks of any NLP application is to break the text into meaningful units, such as words, phrases, and sentences.
- Labeling data – After the system converts the text to data, the next challenge is to apply labels that represent the various parts of speech. Every language requires a different labeling scheme to match the language's grammar.
- Representing context – Because word meaning depends on context, any NLP system needs a way to represent the context. Because of the large number of contexts, converting context into a form that computers can understand is difficult.
- Applying grammar – Although grammar defines a structure for language, the application of grammar is nearly infinite. Dealing with the variation in how humans use language is a major challenge for NLP systems.


Many complex dependencies



Lack of structure

Natural language processing use cases

Search applications

Human machine interfaces

Some of the more common applications include:

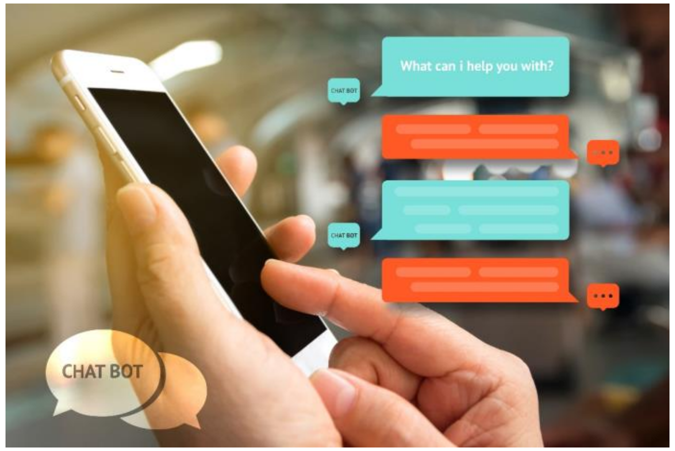
1. Search applications (such as Google and Bing)
2. Human-machine interfaces (such as Alexa)
3. Sentiment analysis for marketing or political campaigns

Natural language processing use cases

4. Social research that is based on media analysis

Market and social research

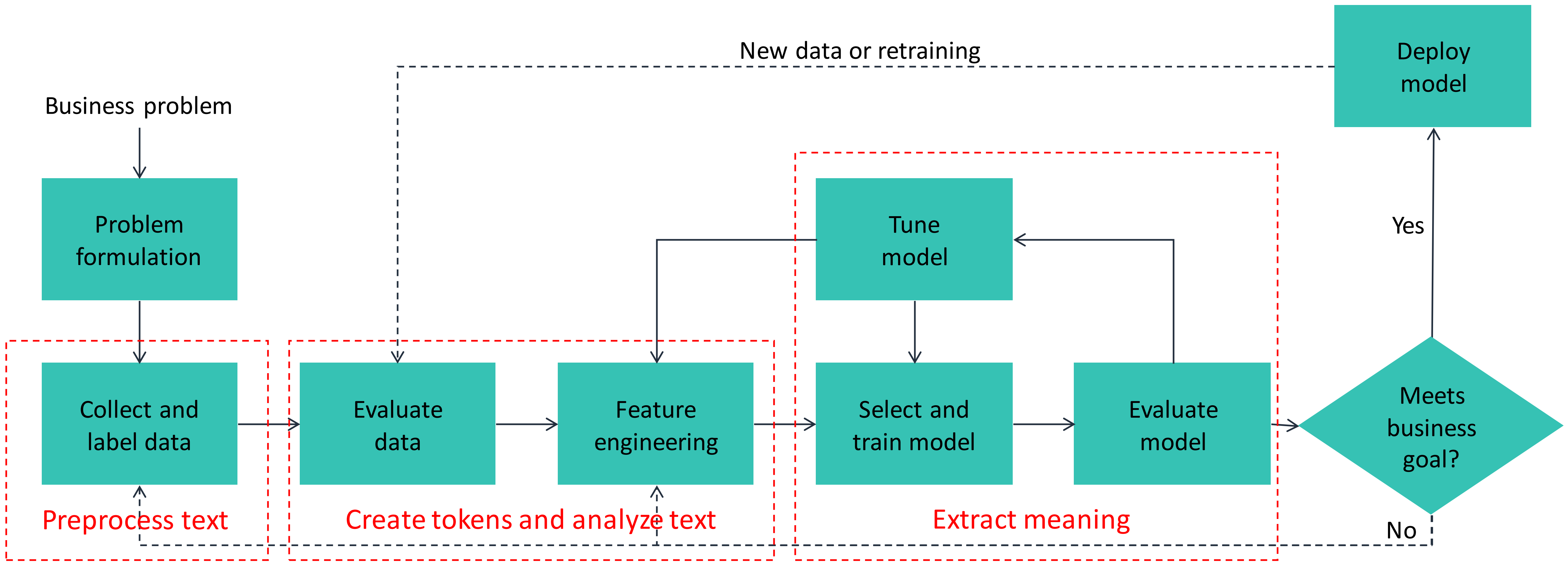
5. Chatbots to mimic human speech in applications



Chatbots

Natural language processing flow

- You can apply the ML pipeline that you have seen throughout this course when you develop an NLP solution.



Preprocessing text

- The Preprocessing task for an NLP application is to convert the text to data so that it can be analyzed.
  - You convert text by removing words that are not needed (known as a stop word) for the analysis from the input text.
  - After you remove stop words, you can normalize text by converting similar words into a common form. For example, the words run, runner, ran, and running are all different forms of the word run. You can normalize all instances of these words within a block of text by using processes of stemming and lemmatization.
  - After you normalize the text, you can standardize it by removing words that are not in the dictionary that you use for analysis. Examples include acronyms, slang, and special characters.
- The Natural Language Toolkit (NLTK) Python library provides functions that you can use to 1) remove stop words, 2)normalize, and 3)standardize text.

Preprocessing text

- Common preprocessing steps –
  1. Remove stop words
  2. Normalize similar text
  3. Standardize unrecognized text
- Other preprocessing steps –
  4. Encoding
  5. Spelling and grammar checks

Sample "This is sample text"

Stop words "This", "is"

Sample "He ran for the bus because he was running late. "

Words to normalize "ran", "running"

Sample "DM me ltr"

Standardize words:
"DM" = "direct message"
"ltr" = "later"

Sample Preprocessing

Creating tokens and feature engineering

- One of the first steps for creating an NLP system is to convert the text into a data collection, such as a DataFrame. (Most of the NLP libraries provide functions to assist with this type of conversion.)
- After you load it into a DataFrame, you can apply one of the NLP models to create features. Common models include:
    1. Bag of words – This simple model captures the frequency of words in a document. For each word in the document, a key is created, with a value that is the number of occurrences within that document.
    2. Term frequency and inverse document frequency (TF-IDF) – Term frequency is a count of how many times a word appears in a document. Inverse document frequency is the number of times a word occurs in a group of documents. These two values are used together to calculate a weight for the words. Words that frequently appear in many documents have a lower weight.

Creating tokens and feature engineering

- Load data by using tokens
- You can use tokens to convert words into items in a DataFrame

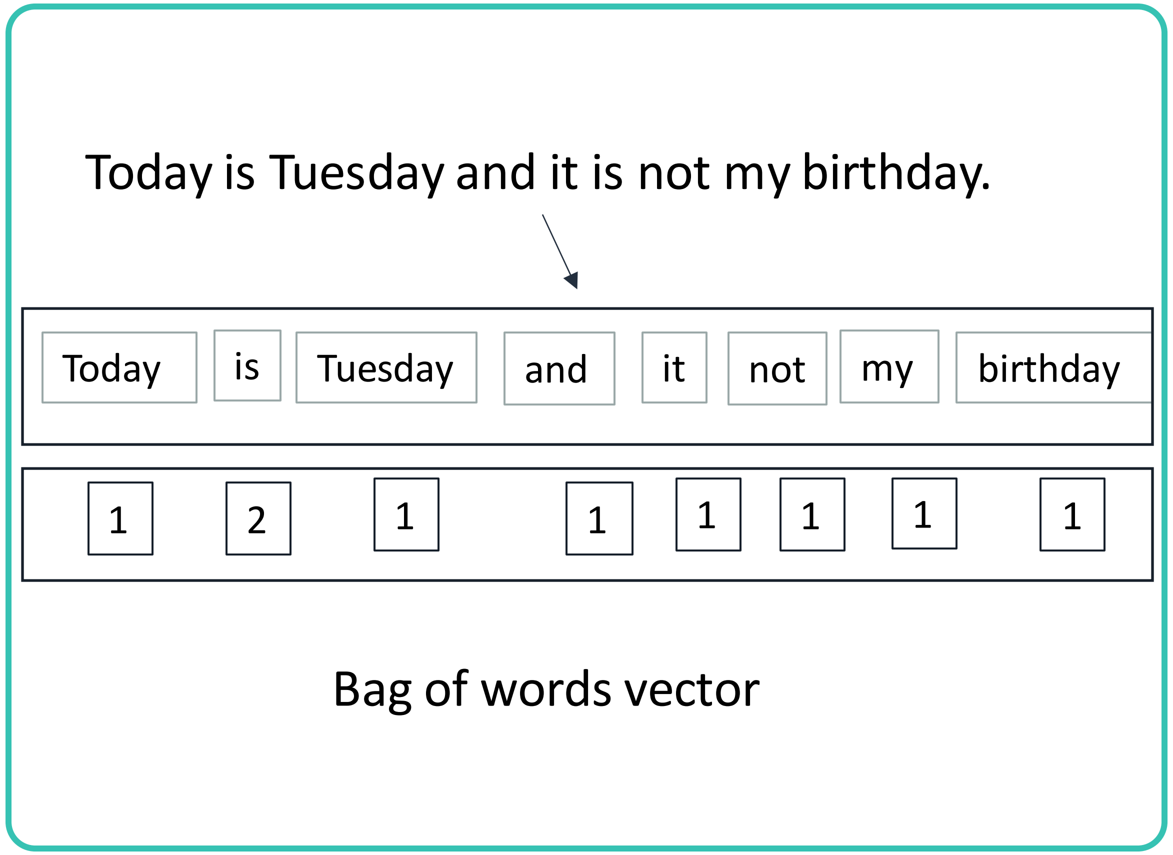
```
from nltk.tokenize import word_tokenize
text = “this is some sample text.”
Print(word_tokenize(text))

Output: [‘this’,‘is’,‘some’,‘sample’,‘text’ ‘.’]
```

Sample token code

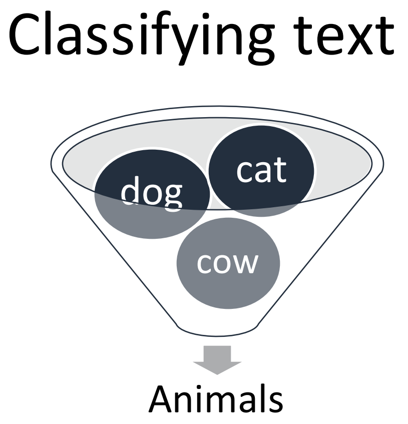
Example NLP model: Bag of words

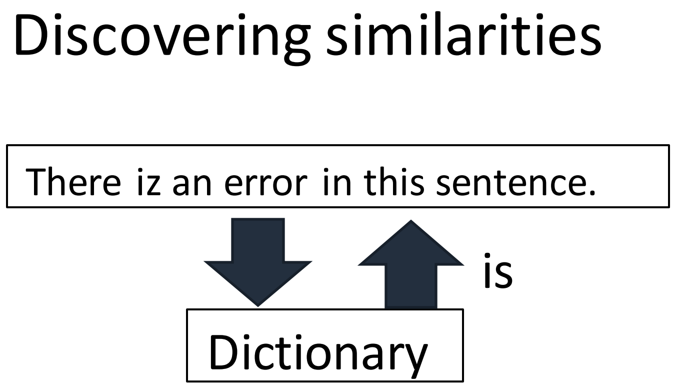
- *Bag of words* is a vector model. Vector models convert each sentence or phrase into a vector, which is a mathematical object that records both directionality and magnitude.
- In the example, a simple sentence is converted into a vector where each word is recorded in terms of frequency. The word "is" has a value of 2 because it appears twice in the sentence.
- It is often used to classify documents into different categories. It is also used to derive attributes that feed into NLP applications, such as sentiment analysis.

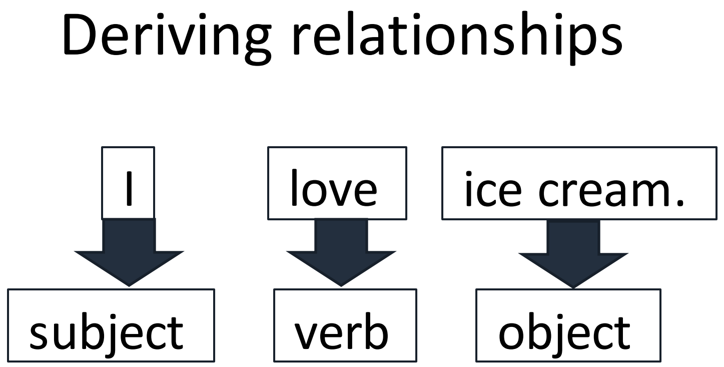


Example NLP Model

Text analysis categories





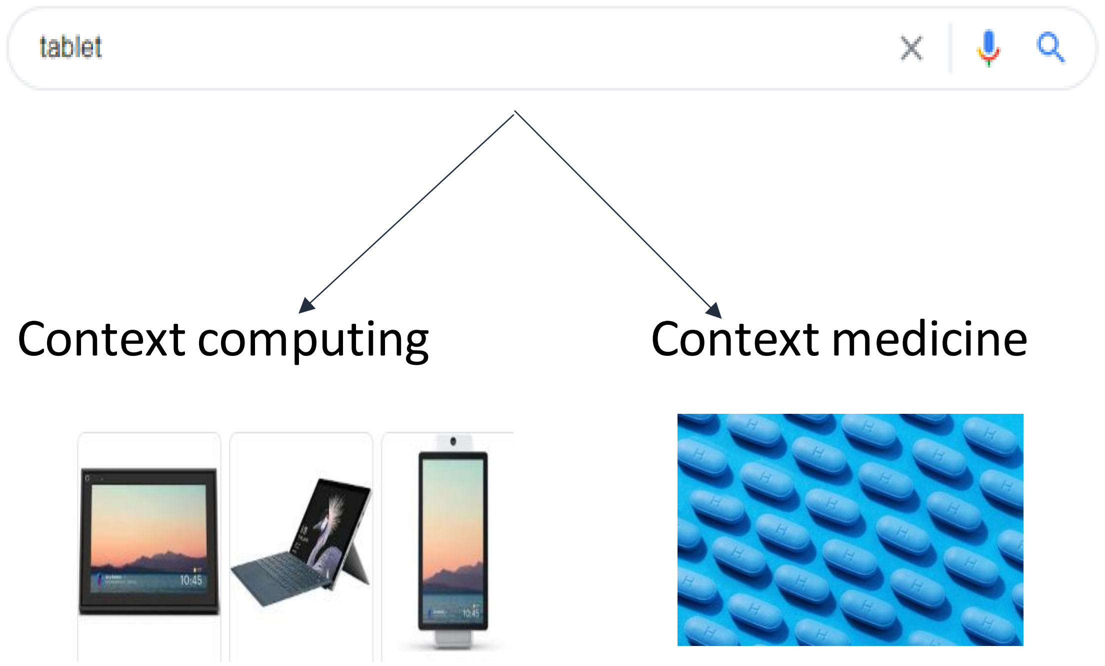


Text analysis has three broad categories:

- Classifying text – This category of analysis is similar to other classification systems. The text provides the input to a process that extracts features, which are then sent through an ML algorithm. This algorithm interacts with a classifier model to infer the classification.
- Discovering similarities – Text matching has many applications. For example, auto-correct, spell check, and grammar checks are all based on text matching. The edit distance algorithm (also known as the Levenshtein distance) is frequently used.
- Deriving relationships – You can derive relationships between different words or phrases in the text by using a process called coreference resolution.

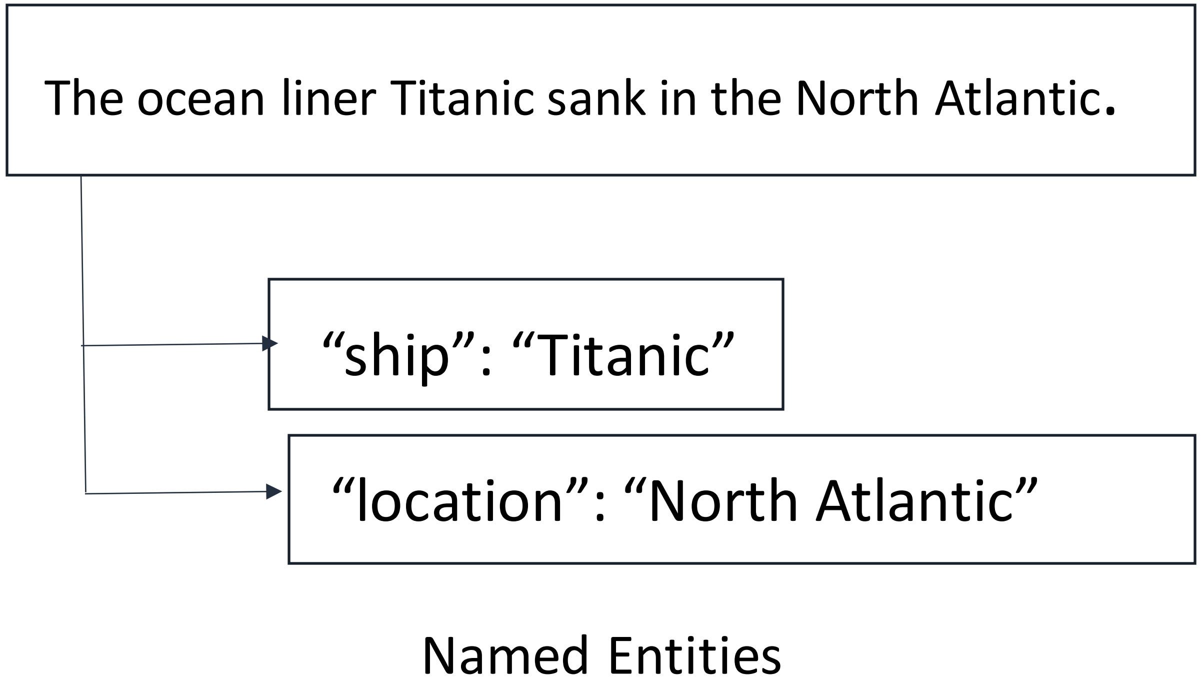
Capture context

- One of the largest challenges for NLP is how to describe the context of the text.
- Consider the example where a user is searching for the term *tablet*. Because the word tablet has at least two distinct meanings, the search engine must know which meaning the user has in mind.
- The term might be qualified further (for example, by adding another term like medicine or computing). If it's not qualified by other terms, then most search engines rely on the most commonly used context.



Derive meaning by entity extraction

- The process of extracting entities is known as named entity recognition (NER).
- A NER model has the following components:
  - Identify noun phrases by using dependency charts and part of speech tagging
  - Classify phrases by using a classification algorithm, such as Word2Vec
  - Disambiguate entities by using a knowledge graph


The ocean liner Titanic sank in the North Atlantic.
"ship": "Titanic"
"location": "North Atlantic"
Named Entities

- This example shows how to use NER to extract the entities Titanic and North Atlantic.
- After the named entities are extracted, you can use a knowledge graph to extract meaning. A knowledge graph combines subject matter expertise with ML to derive meaning.

NLP managed services

- In this section, you will review five managed AWS services that you can use for various NLP use cases. These services simplify the process of creating an NLP application.
- Natural language processing managed services:
  1. Amazon Transcribe
  2. Amazon Polly
  3. Amazon Translate
  4. Amazon Comprehend
  5. Amazon Lex

Amazon Transcribe

Amazon Transcribe

- Amazon Transcribe is a fully managed service that uses advanced machine learning technologies to "recognize speech in audio files and transcribe them into text".
- You can use Amazon Transcribe to convert audio to text and to create applications that incorporate the content of audio files.
  - Recognize recorded voices
  - Convert streaming audio to text
  - Customize specialized vocabulary
  - Integrate with applications by using WebSockets
    - (WebSockets provide an internet-facing interface that enables two-way communication between an application and Amazon Transcribe.)
  - Build subtitles for multiple languages in real-time

Amazon Transcribe use cases

Medical transcription

Streaming content labeling

- Medical transcription – Medical professionals can record their notes, and Amazon Transcribe can capture their spoken notes as text.
- Video subtitles – Video production organizations can generate subtitles automatically from video. It can also be done in real-time for a live feed to add closed captioning (CC).
- Streaming content labeling – Media companies can capture and label content, and then feed the content into Amazon Comprehend for further analysis.
- Customer call center monitoring – Companies can record customer service or sales calls, and then analyze the results for training or strategic opportunities.

Subtitles

Call center monitoring

Amazon Polly

Amazon Polly

- Amazon Polly is a managed service that converts text into lifelike speech. Amazon Polly supports multiple languages and includes various lifelike voices.
- Generate voice from plain text or Speech Synthesis Markup Language (SSML) format.
  - SSML is a markup language that you can use to provide special instructions for how speech should sound. For example, you might want to introduce a pause in the flow of speech. You can add an SSML tag to instruct Amazon Polly to pause between two words.
- Create output in multiple audio formats.
  - You can also output speech from Amazon Polly to MP3, Vorbis, and pulse-code modulation (PCM) audio stream formats.
- Amazon Polly is eligible for use with regulated workloads for the U.S. Health Insurance Portability and Accountability Act of 1996 (HIPAA), and the Payment Card Industry Data Security Standard (PCI DSS).

Amazon Polly use cases

News service production

Navigation systems

News service production – Major news companies use Amazon Polly to generate vocal content directly from their written stories.

Language training systems – Language training companies use Amazon Polly to create systems for learning a new language.

Navigation systems – Amazon Polly is embedded in mapping application programming interfaces (APIs) so that developers can add voice to their geo-based applications.

Animation production – Animators use Amazon Polly to add voices to their characters.

Language training

Animation production

Amazon Translate

Amazon Translate

- Amazon Translate is a fully managed text translation service that uses advanced machine learning technologies to provide high-quality translation on demand.
- Develop multilingual user experiences for your applications
- Translate documents to multiple languages
  - You can create systems for reading documents in one language, and then rendering or storing them in another language
- Analyze incoming text in multiple languages.
- Amazon Translate is fully integrated with other Amazon ML services, such as Amazon Comprehend, Amazon Transcribe, and Amazon Polly. With this integration, you can:
  - Extract named entities, sentiment, and key phrases by integrating with Amazon Comprehend
  - Create multilingual subtitles with Amazon Transcribe
  - Speak translated content with Amazon Polly

Amazon Translate use cases

International websites

Multilingual chatbots

- International websites – You can use Amazon Translate to quickly globalize your websites.
- Software localization – Localization is a major cost for all software that is aimed at a global audience. Amazon Translate can decrease software development time and significantly reduce costs for localizing software.
- Multilingual chatbots – Chatbots are used to create a more human-like interface to applications. With Amazon Translate, you can create a chatbot that speaks multiple languages.
- International media management – Companies that manage media for a global audience use Amazon Translate to reduce their costs for localization.

Software localization

International media

Amazon Comprehend

Amazon Comprehend

- Amazon Comprehend uses NLP to extract insights about the content of documents. It develops insights by recognizing the entities, key phrases, language, sentiments, and other common elements in a document.

- Extract key entities from a document, such as people or locations
- Identify the language that is used in a document
- Determine the sentiment—such as positive, negative, neutral, or mixed—that is expressed in a document
- Identify the part of speech for individual words in a document

Amazon Comprehend use cases

Document analysis

Mobile app analysis

- Analysis of legal and medical documents – Legal, insurance, and medical organizations have used Amazon Comprehend to perform many of the NLP functions.
- Financial fraud detection – Banking, financial, and other institutions have used Amazon Comprehend to examine very large datasets of financial transactions to uncover fraud and look for patterns of illegal transactions.
- Large-scale mobile app analysis – Developers of mobile apps can use Amazon Comprehend to look for patterns in how their apps are used so they can design improvements.
- Content management – Media and other content companies can use Amazon Comprehend to tag content for analysis and management purposes.

Fraud detection

Content management

Amazon Lex

Amazon Lex

- Amazon Lex is an AWS service for building conversational interfaces for applications by using voice and text.
- Build a chatbot that can interact with voice and text to ask questions, get answers, or complete tasks.
- With Amazon Lex, you can add a human language frontend to your applications. The same conversational engine that powers Amazon Alexa is now available to any developer.
- Automatically scale your chatbot with AWS Lambda
- Store log files of conversations for analysis

Amazon Lex use cases

Inventory and sales

Customer service interfaces

- Building frontend interfaces for inventory management and sales – Voice interfaces are becoming more common. Companies add chatbots to their inventory & sales applications.
- Developing interactive assistants – By combining Amazon Lex with other ML services, customers create more sophisticated assistants for many different industries.
- Creating customer service interfaces – Human-like voice applications are quickly becoming the norm for customer service applications. That reduces the time and increases the quality.
- Query databases with a human-like language – Amazon Lex is combined with other AWS database services to create data analysis applications with a human-like language interface.

Interactive assistants

Database queries

Guided Lab:

Guided Lab: (Not part of Grade)

Creating a Bot to Schedule Appointments.

Chapter summary

In summary, in this Chapter you learned how to:

- Describe the natural language processing (NLP) use cases that are solved by using managed Amazon ML services
- Describe the managed Amazon ML services available for NLP
- Use managed Amazon ML Services

- What is Amazon Comprehend?
- What is Amazon Polly?
- What is Amazon Lex?
- What is Amazon Transcribe?
- What is Amazon Translate?

Chapter end questions

1. What is NLP? Describe any two use cases of NLP.
2. Describe the main challenges of NLP.
3. Explain the NLP model "Bag of words" with an example scenario.
4. Explain the NLP model "Term frequency and inverse document frequency (TF-IDF)" with an example scenario.
5. Describe any one AWS NLP managed services. (Lex, Polly, Transcribe, Translate)