AWS Academy Machine Learning Foundations

Module 3: Implementing a Machine Learning Pipeline with Amazon SageMaker

Module overview

Sections

1. Scenario introduction
2. Collecting and securing data
3. Evaluating your data
4. Preprocessing your data
5. Training
6. Evaluating the accuracy of the model
7. Hosting and using the model
8. Hyperparameter and model tuning

Demonstration

- Training a model using Amazon SageMaker
- Optimizing Amazon SageMaker Hyperparameters
- Running Amazon SageMaker Autopilot

Knowledge check

Module overview continued

Lab

- Guided lab: Exploring Amazon Sagemaker
- Guided lab: Visualizing data
- Guided lab: Encoding categorical data
- Guided lab: Splitting data and training a model with xgBoost
- Guided lab: Hosting and consuming a model on AWS
- Guided lab: Evaluating model accuracy
- Guided lab: Tuning with Amazon SageMaker
- Challenge Lab:

Module objectives

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

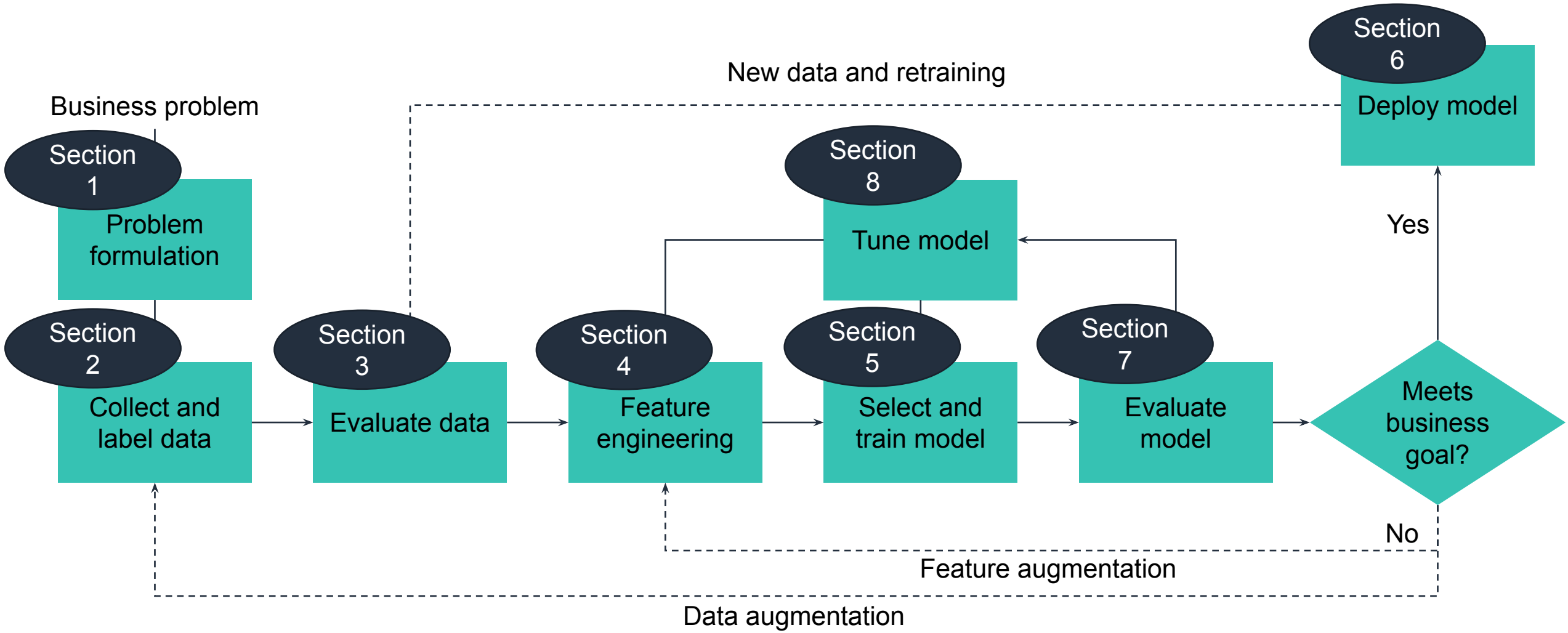
- Formulate a problem from a business request
- Obtain and secure data for machine learning (ML)
- Build a Jupyter Notebook using Amazon SageMaker
- Outline the process for evaluating data
- Explain why data needs to be preprocessed
- Use open source tools to examine and preprocess data
- Use Amazon SageMaker to train and host an ML model
- Use cross-validation to test the performance of an ML model
- Use a hosted model for inference
- Create an Amazon SageMaker hyperparameter tuning job to optimize a model's effectiveness

Module 3: Implementing a Machine Learning Pipeline with Amazon SageMaker

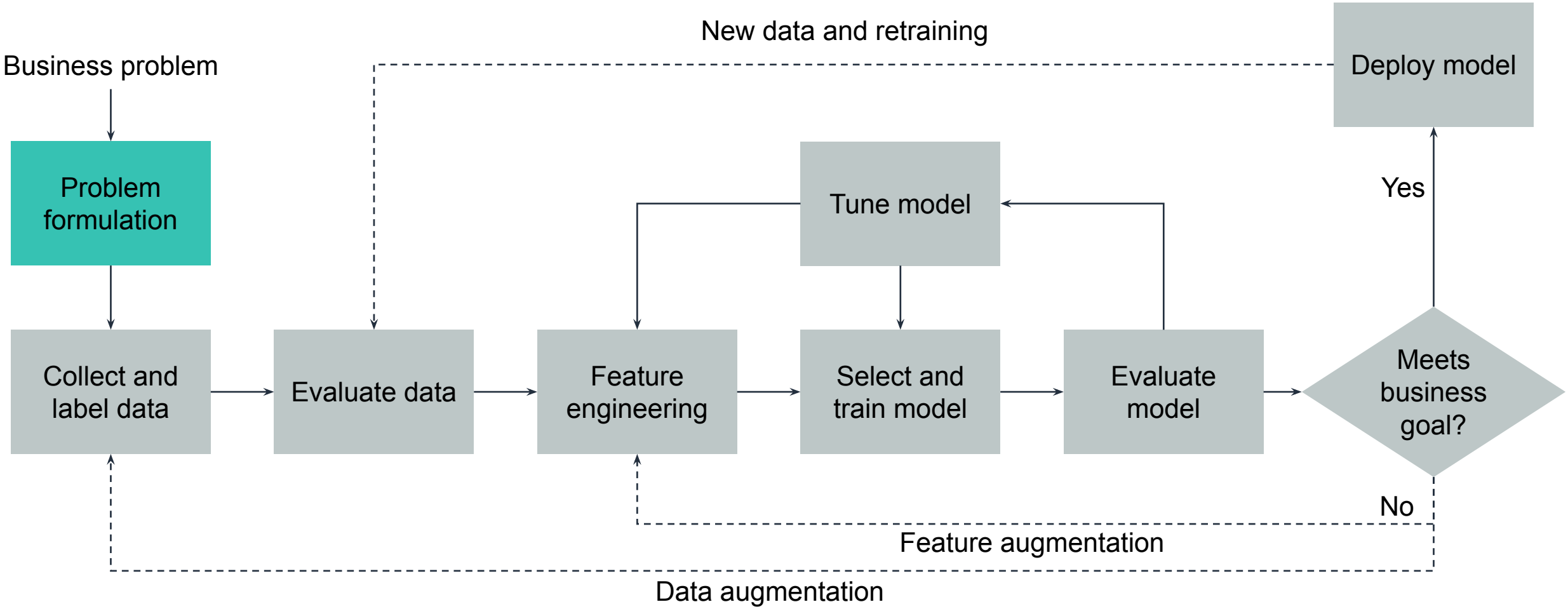
Section 1: Scenario introduction

aws academy

Machine learning pipeline



Machine learning pipeline



Define business objective

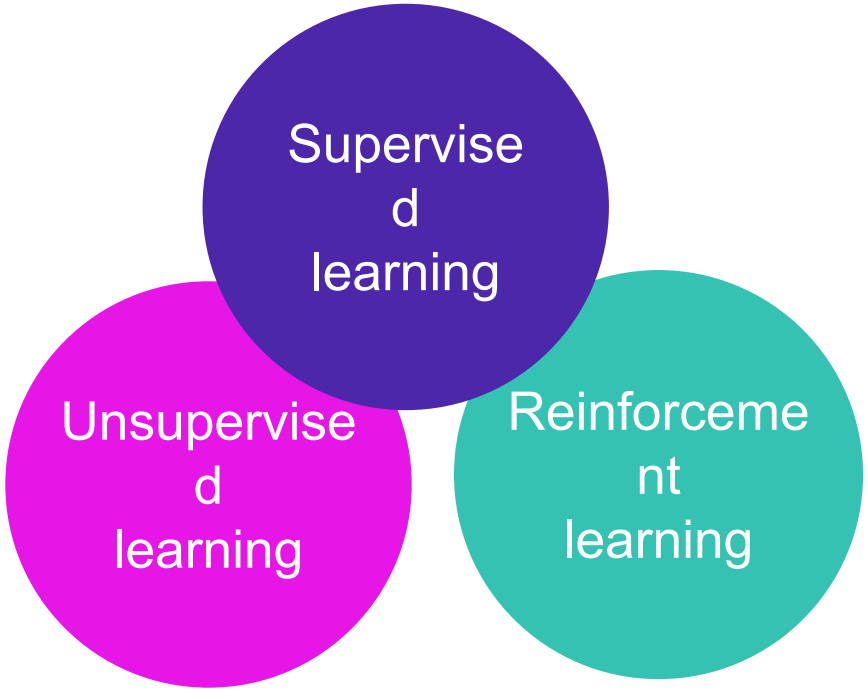
What is the business goal?

Questions to ask:

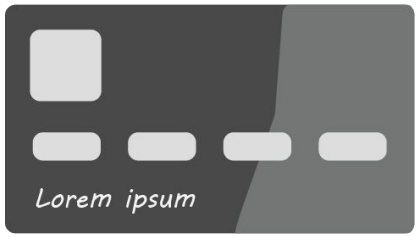
- How is this task done today?
- How will the business measure success?
- How will the solution be used?
- Do similar solutions exist, which you might learn from?
- What assumptions have been made?
- Who are the domain experts?

How should you frame this problem?

- Is the problem a machine learning problem?
- Is the problem supervised or unsupervised?
- What is the target to predict?
- Do you have access to the data?
- What is the minimum performance?
- How would you solve this problem manually?
- What's the simplest solution?



Example: Problem formulation



You want to identify fraudulent credit card transactions so that you can stop the transaction before it processes.

Why?

Reduce the number of customers who end their membership because of fraud.

10%

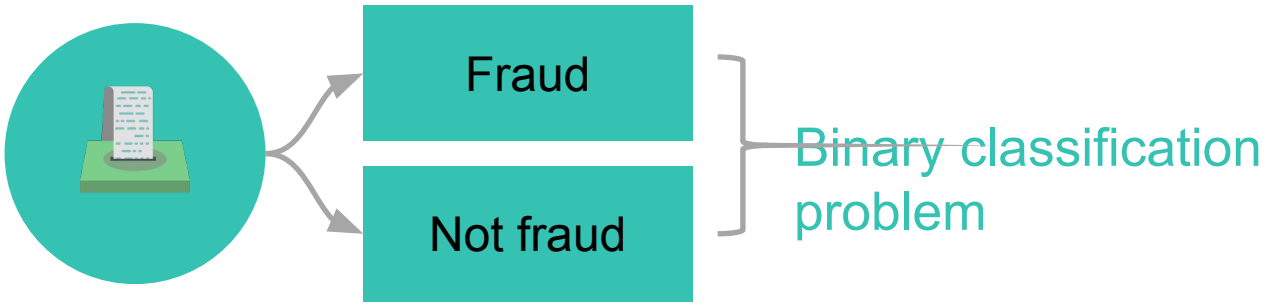
reduction in fraud claims in retail

Can you measure it?

Move from qualitative statements to quantitative statements that can be measured.

Make it an ML model

Credit card transaction is either ***fraudulent*** or ***not fraudulent***.



Use historical data of fraud reports to help define your model.

Wine quality dataset

Question: Based on the composition of the wine, can you predict the quality and therefore the price?

Why:

- View statistics
- Deal with outliers
- Scale numeric data

Citation

Source: UCI Wine quality dataset

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.
Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

Car evaluation dataset

Question: Can you use a car's attributes to predict whether the car will be purchased?

Why:

- View statistics
- Encode categorical data

Citation
Source: UCI Car evaluation dataset

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Vertebral column dataset

Question: Based on the biomechanical features, can you predict whether a patient has an abnormality (disk hernia or spondylolisthesis)?

Why:

- View statistics
- Encode categorical data
- Train and tune a model

Citation
Source: UCI Vertebral column dataset

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Section 1 key takeaways

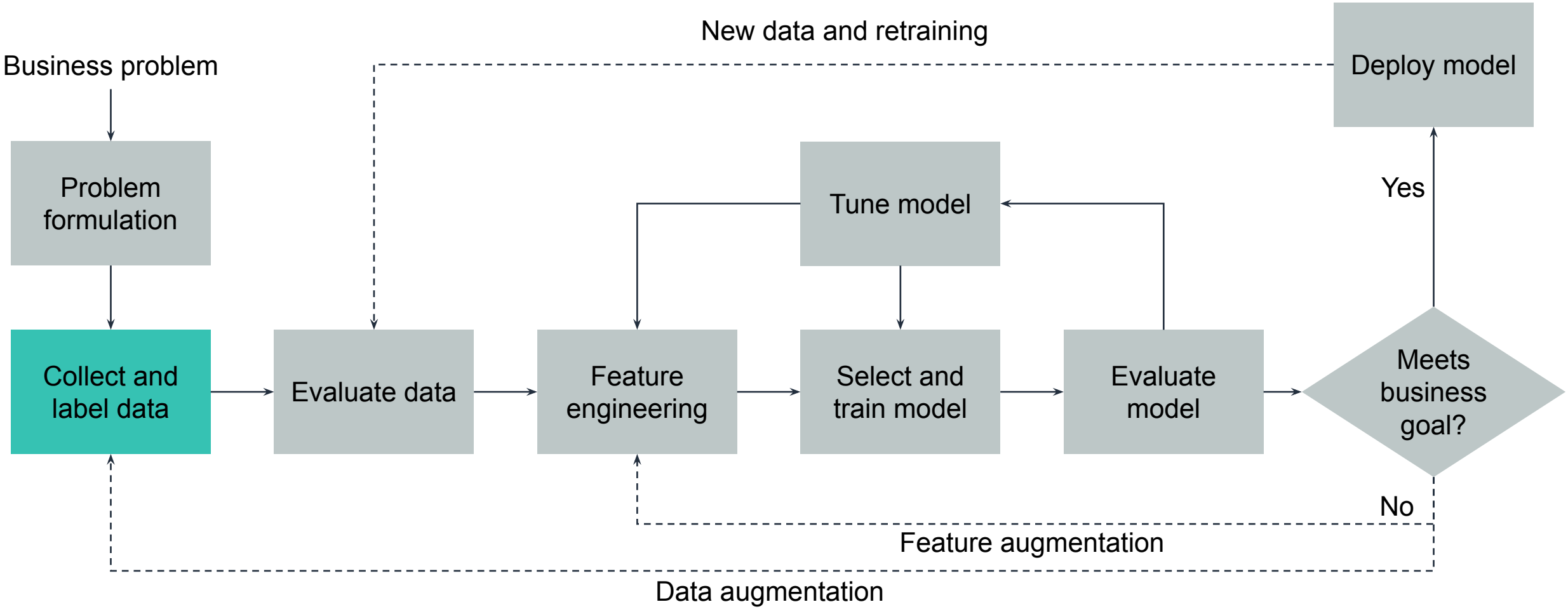
- Business problems must be converted to an ML problem
  - Why?
  - Can it be measured?
- What kind of ML problem is it?
  - Classification or regression?

Module 3: Implementing a Machine Learning Pipeline with Amazon SageMaker

Section 2: Collecting and securing data

aws academy

Machine learning pipeline



What data do you need?

- How much data do you have, and where is it?
- Do you have access to that data?
- What solution can you use to bring all of this data into one centralized repository?

Data sources

- Private data: Data that customers create
- Commercial data: AWS Data Exchange, AWS Marketplace, and other external providers
- Open-source data: Data that is publicly available (check for limits on usage)
  - Kaggle
  - World Health Organization
  - U.S. Census Bureau
  - National Oceanic and Atmospheric Administration (U.S.)
  - UC Irvine Machine Learning Repository
  - AWS

Observations

ML problems need a lot of data—also called observations—where the target answer or prediction is already known.

Customer	Date of transaction	Vendor	Charge amount	Was this fraud?
ABC	10/5	Store 1	10.99	No
DEF	10/5	Store 2	99.99	Yes
GHI	10/5	Store 2	15.00	No
JKL	10/6	Store 2	99.99	?
MNO	10/6	Store 1	99.99	Yes

Feature

Target

Get a domain expert

- Do you have the data that you need to try to address this problem?
- Is your data representative?

Storing data in AWS

Amazon Simple Storage Service (Amazon S3)

Amazon FSx

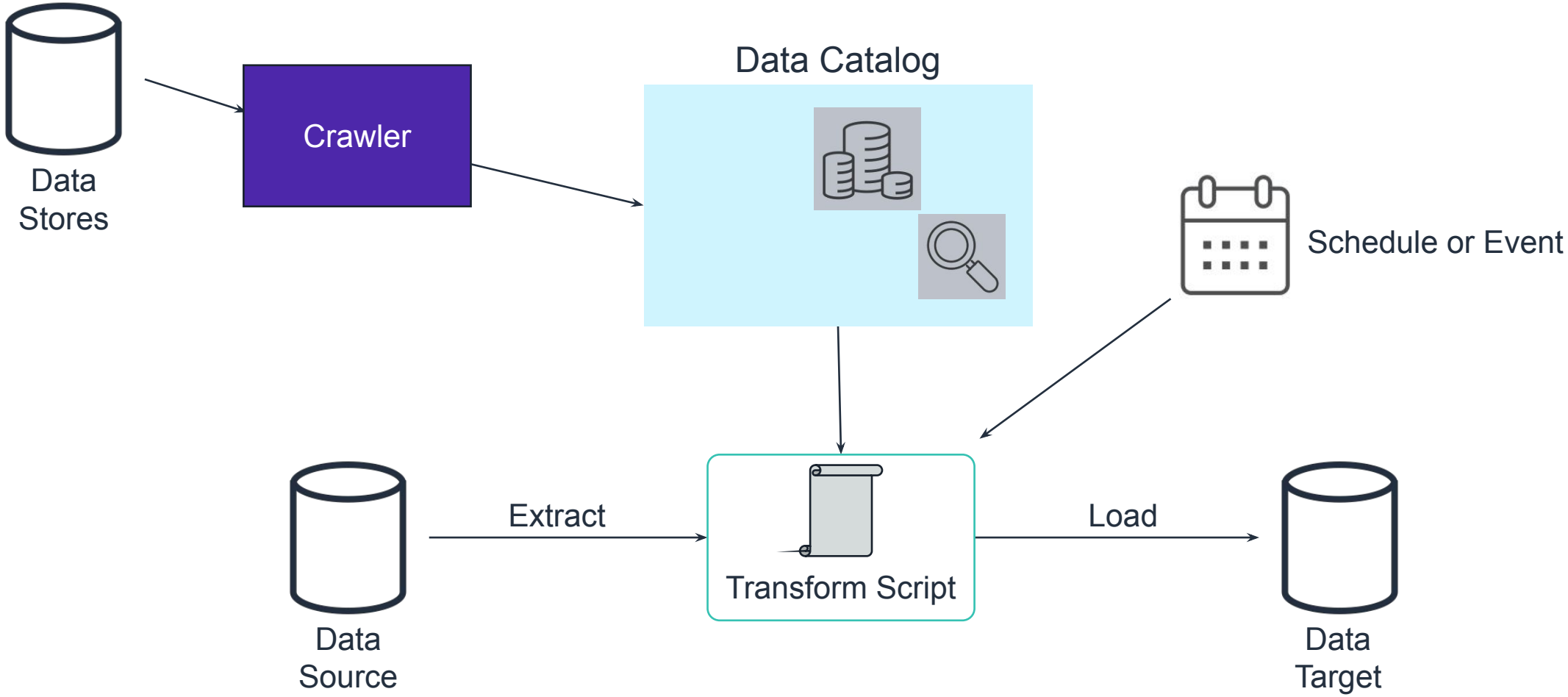
Amazon Elastic File System (Amazon EFS)

Amazon Relational Database Service (Amazon RDS)

Amazon Redshift

Amazon Timestream

Extract, transform, load (ETL)

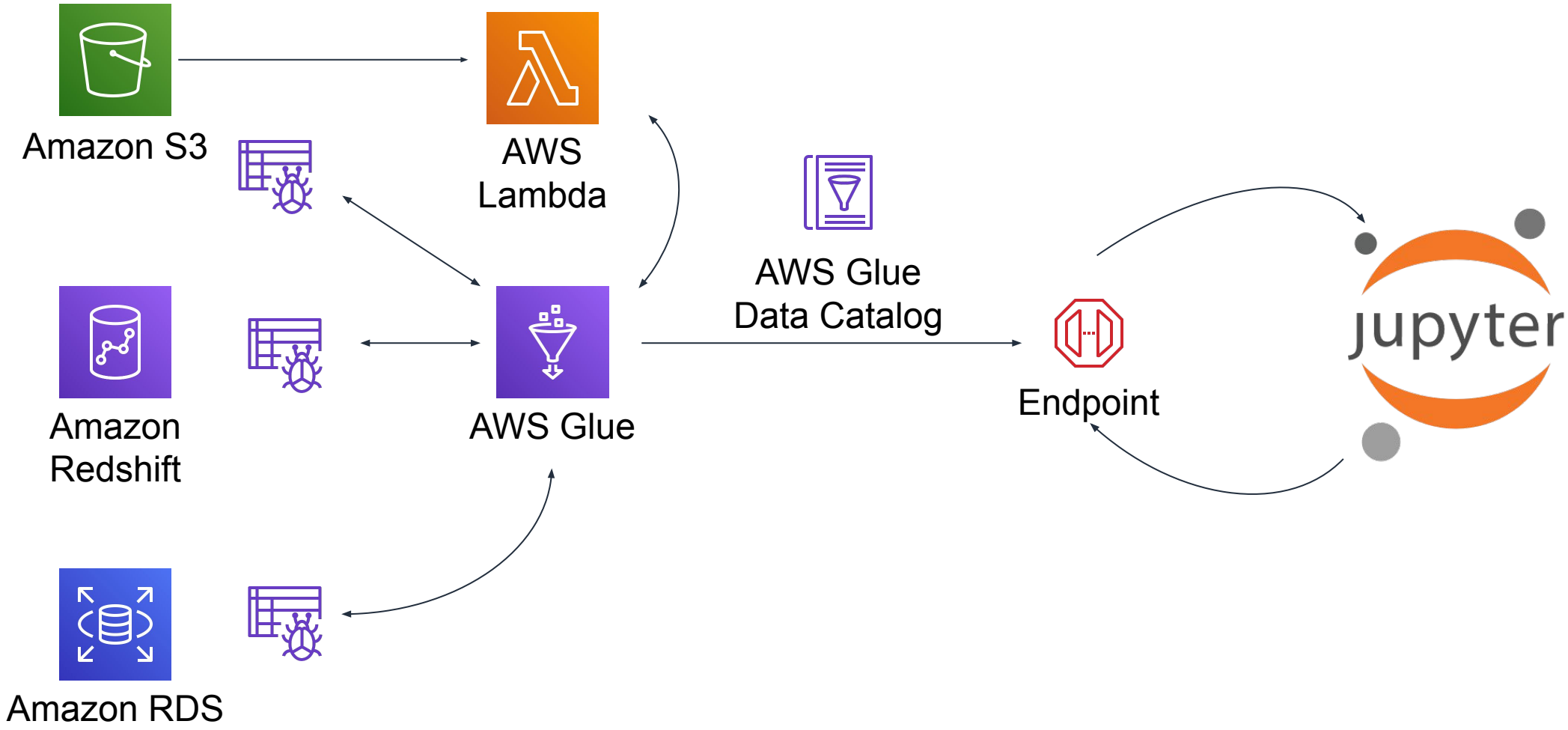


ETL with AWS Glue

- Runs the ETL process
- Crawls data sources to create catalogs that other systems can query
- ML functionality

AWS Glue can *glue together* different datasets and emit a single endpoint that can queried.

AWS Glue overview



ETL with Python

```
import boto3, requests, zipfile, os, io
url = 'http://url.com/somezipfile.zip‘
folder='./extracts‘
```

Imports and variables

```
r = requests.get(url, stream=True)
thezip = zipfile.ZipFile(io.BytesIO(r.content))
thezip.extractall(folder)
```

Download and extract

```
s3 = boto3.client('s3')
bucket = 'bucketname'

with os.scandir(folder) as dir:
  for f in dir:
    if f.is_file():
      s3.upload_file(
        Filename=os.path.join(folder,f.name),
        Key=f.name, Bucket=bucket)
```

Upload to Amazon S3

Securing data: AWS Identity and Access Management (IAM) policy

IAM policies to control access:

```
{
  "Id": "Policy1583974368597",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1583974365198",
      "Action": ["s3:GetObject"],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::awsmachinelearningrepo/*",
      "Principal": {"AWS": ["DataReaderRole"]}
    }
  ]
}
```

GetObject limits access to read only…

…allowing the action…

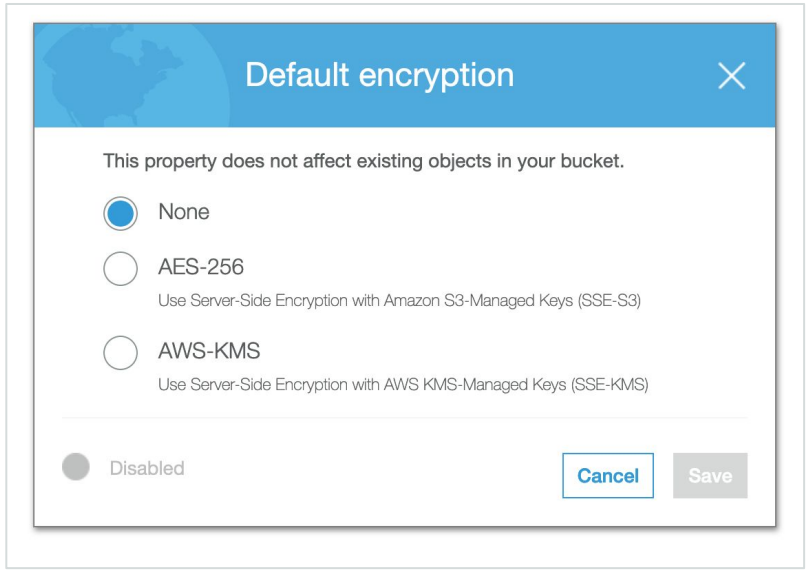
…for only this bucket…

…by only this IAM role.

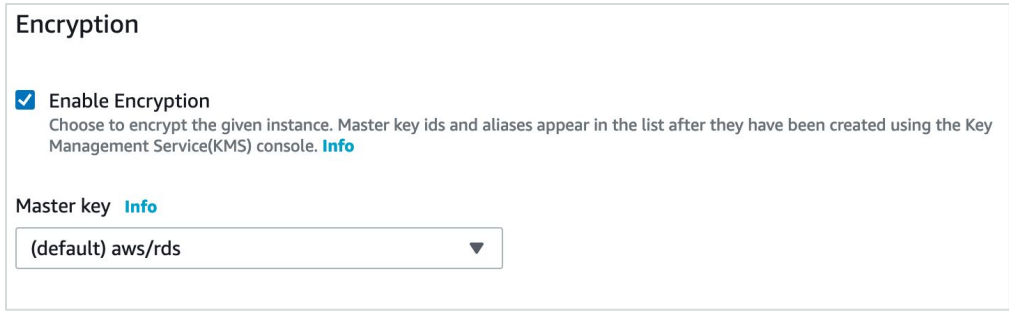
Securing data: Data encryption

The contents of many data repositories in AWS can be quickly and easily encrypted.

Amazon S3 default encryption

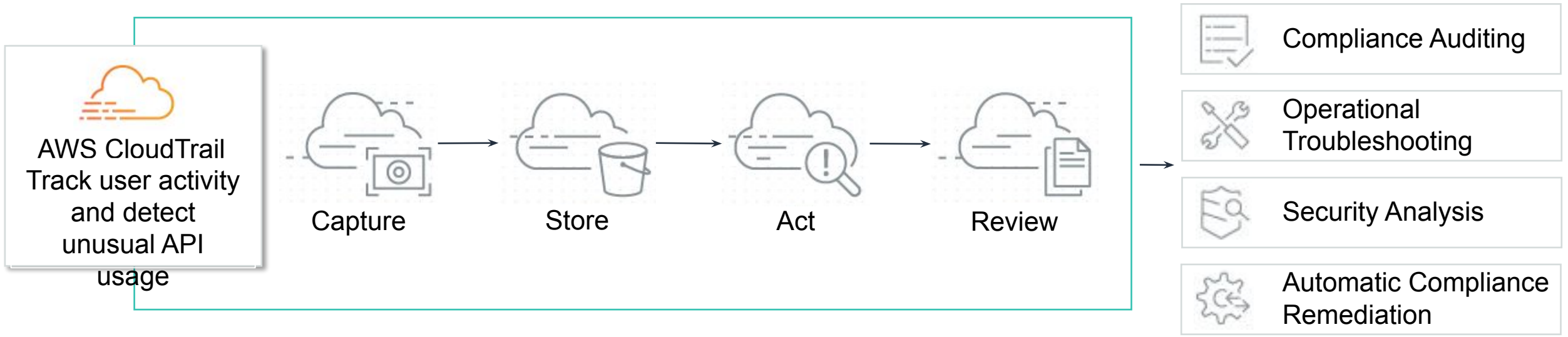


Amazon RDS encryption



Securing data: AWS CloudTrail for audit

AWS CloudTrail tracks user activity and application programming interface (API) usage.



Module 3 – Guided Lab 1: Exploring Amazon SageMaker

Section 2 key takeaways

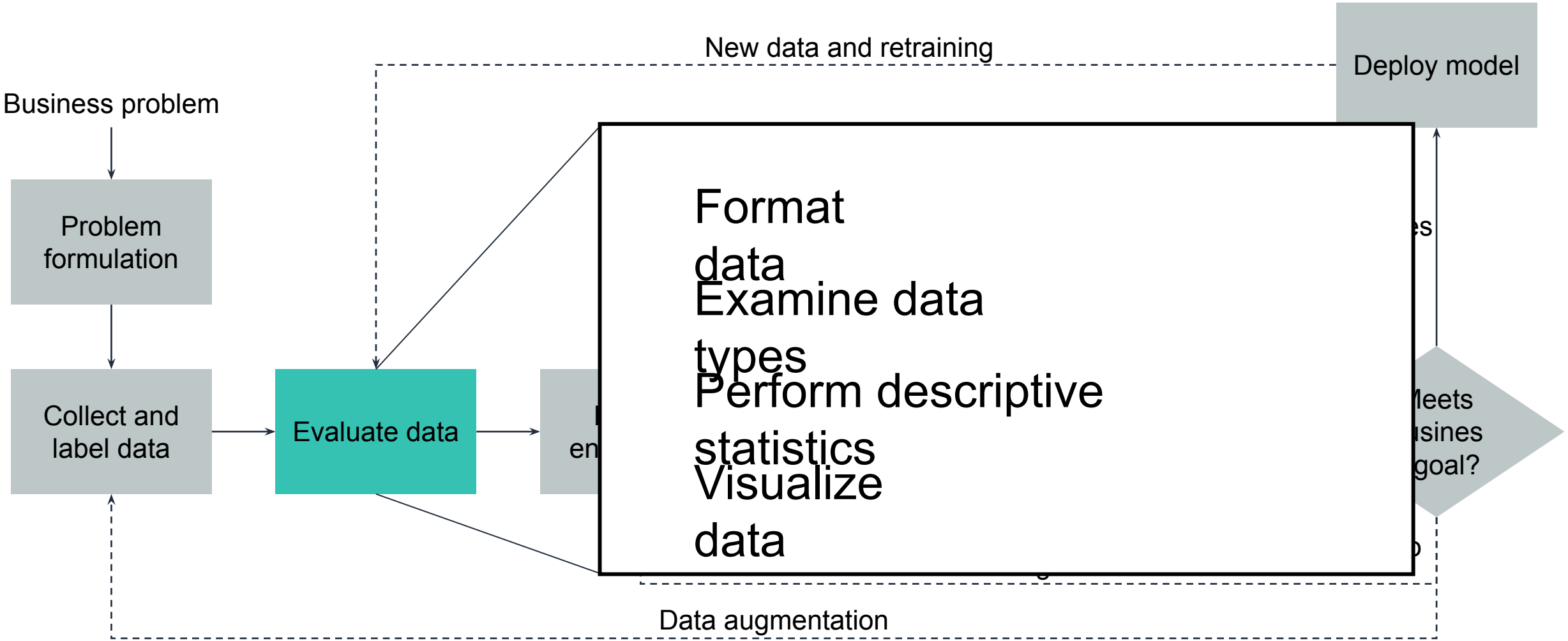
- The first step in the machine learning pipeline is to obtain data
- Extract, transform, and load (ETL) is a common term for obtaining data for machine learning
- AWS Glue makes it easy to run ETL jobs from various data stores
- Securing your data includes both controlling access and encrypting data

Module 3: Implementing a Machine Learning Pipeline with Amazon SageMaker

Section 3: Evaluating your data

aws academy

Machine learning pipeline



You must understand your data

Before you can run statistics on your data, you must ensure that it's in the right format for analysis.

"Customer:ABC,DateOfTransation:10/5,Vendor:Store1,ChargeAmount:10.99,WasThisFruad:No…"

Customer	Date of Transaction	Vendor	Charge Amount	Was This Fraud?
ABC	10/5	Store 1	10.99	No
DEF	10/5	Store 2	99.99	Yes
GHI	10/5	Store 2	15.00	No
JKL	10/6	Store 2	99.99	?
MNO	10/6	Store 1	99.99	Yes

Loading data into pandas

- Reformats data into tabular representation (DataFrame)
- Converts common formats like comma-separated values (CSV), JavaScript Object Notation (JSON), Excel, Pickle, and others

```
import pandas as pd
url = "https://somewhere.com/winequality-red.csv"
df_wine = pd.read_csv(url,';')
```

pandas DataFrame

```
df_wine.shape
```

Number of instances

```
(1599, 12)
```

Number of attributes

```
df_wine.head(5)
```

Columns/Attributes

Rows/Instances

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

Index and column names

```
df_wine.columns
```

```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

```
df_wine.index
```

```
RangeIndex(start=0, stop=1599, step=1)
```

DataFrame schema

```
df_wine.dtypes
```

```
quality                int64
fixed acidity          float64
volatile acidity       float64
citric acid            float64
residual sugar         float64
chlorides              float64
free sulfur dioxide    float64
total sulfur dioxide   float64
density                float64
pH             float64
sulphates                  float64
alcohol                float64
dtype: object
```

```
df_wine.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1597 entries,
0 to 1598
Data columns (total 12 columns):
quality                1597 non-null int64
fixed acidity          1597 non-null float64
volatile acidity       1597 non-null float64
citric acid            1597 non-null float64
residual sugar         1597 non-null float64
chlorides              1597 non-null float64
free sulfur dioxide    1597 non-null float64
total sulfur dioxide   1597 non-null float64
density                1597 non-null float64
pH             1597 non-null float64
sulphates                  1597 non-null float64
alcohol                1597 non-null float64
dtypes: float64(11), int64(1)
memory usage: 162.2 KB
```

Descriptive statistics

Use descriptive statistics to gain insights into your data before you clean the data:

Overall statistics

Multivariate statistics

Attribute statistics

Statistical characteristics

```
df_wine.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	pH	sulphates	alcohol	quality
count	1599.00	1599.00	1599.00	1599.00	1599.00	1599.00	1599.00	1599.00	1599.00	1599.00	1599.00
mean	8.32	0.53	0.27	2.54	0.09	15.87	46.47	3.31	0.66	10.42	5.64
std	1.74	0.18	0.19	1.41	0.05	10.46	32.90	0.15	0.17	1.07	0.81
min	4.60	0.12	0.00	0.90	0.01	1.00	6.00	2.74	0.33	8.40	3.00
25%	7.10	0.39	0.09	1.90	0.07	7.00	22.00	3.21	0.55	9.50	5.00
50%	7.90	0.52	0.26	2.20	0.08	14.00	38.00	3.31	0.62	10.20	6.00
75%	9.20	0.64	0.42	2.60	0.09	21.00	62.00	3.40	0.73	11.10	6.00
max	15.90	1.58	1.00	15.50	0.61	72.00	289.00	4.01	2.00	14.90	8.00

Categorical statistics identify frequency of values and class imbalances

```
df_car.head(5)
```

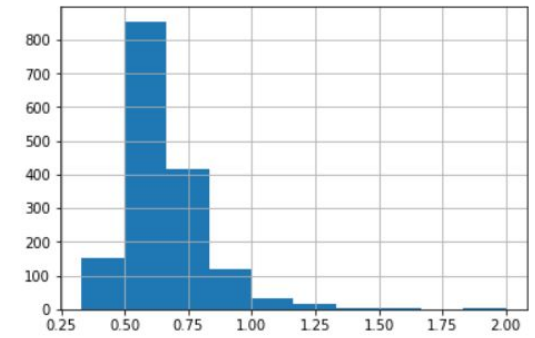
	buying	maint	doors	persons	lug_boot	safety	class
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc

```
df_car.describe()
```

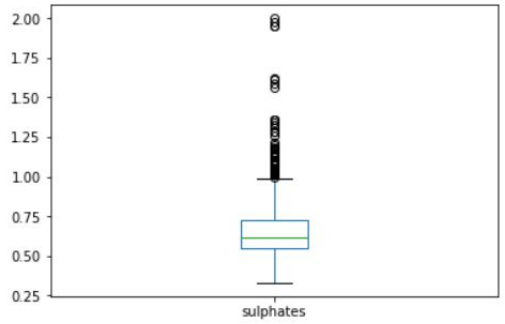
	buying	maint	doors	persons	lug_boot	safety	class
count	1728	1728	1728	1728	1728	1728	1728
unique	4	4	4	3	3	3	4
top	low	low	2	2	big	low	unacc
freq	432	432	432	576	576	576	1210

Plotting attribute statistics

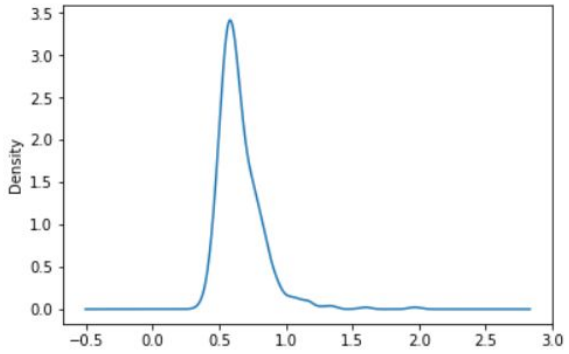
```
df_wine['sulphates'].hist(bins=10)
```




```
df_wine['sulphates'].plot.box()
```

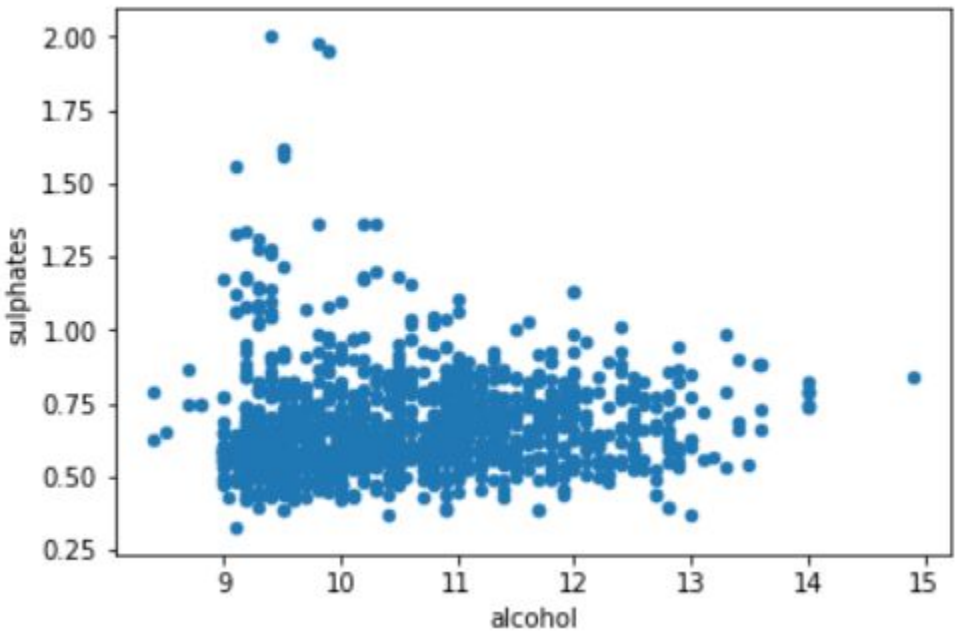



```
df_wine['sulphates'].plot.kde()
```

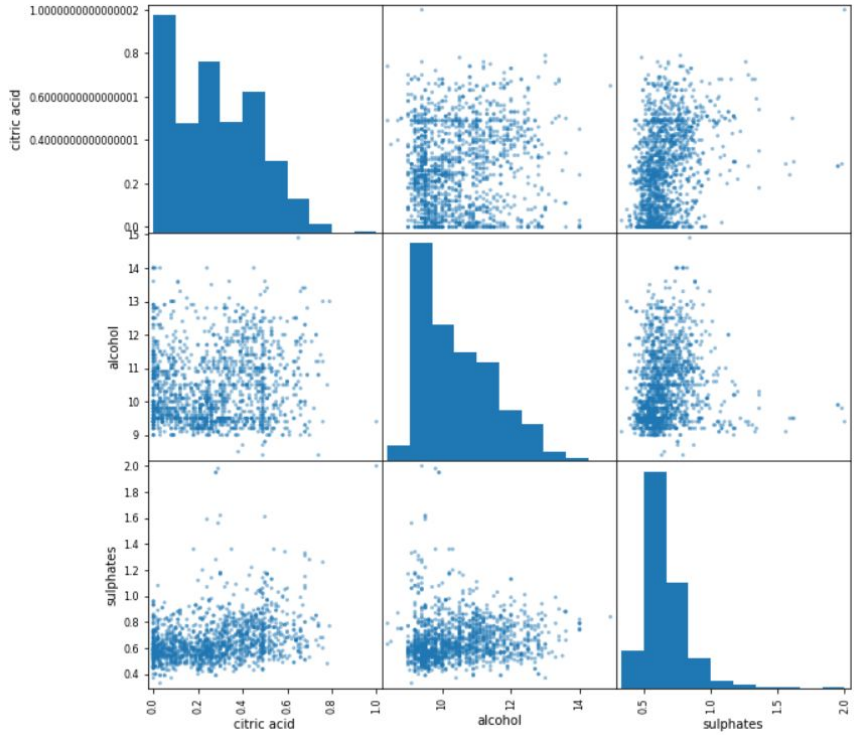


Plotting multivariate statistics

```
df_wine.plot.scatter(
    x='alcohol',
    y='sulphates')
```

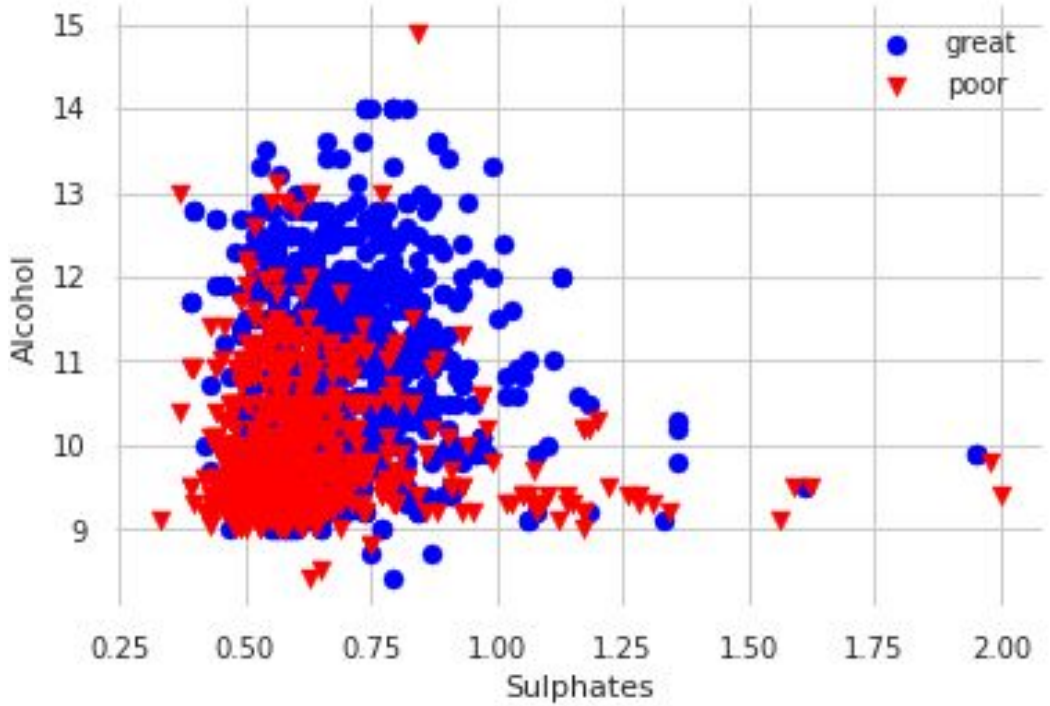



```
pd.plotting.scatter_matrix(
    df_wine[['citric acid',
        'alcohol',
        'sulphates']])
```



Scatter plot with identification

```
high = df_wine[['sulphates','alcohol']][df_wine['quality']>5]
low = df_wine[['sulphates','alcohol']][df_wine['quality']<=5]

plt.scatter(high['sulphates'],high['alcohol'],s=50,c='blue',marker='o',label='great')
plt.scatter(x=low['sulphates'],y=low['alcohol'],s=50,c='red',marker='v',label='poor')
```



Correlation matrix

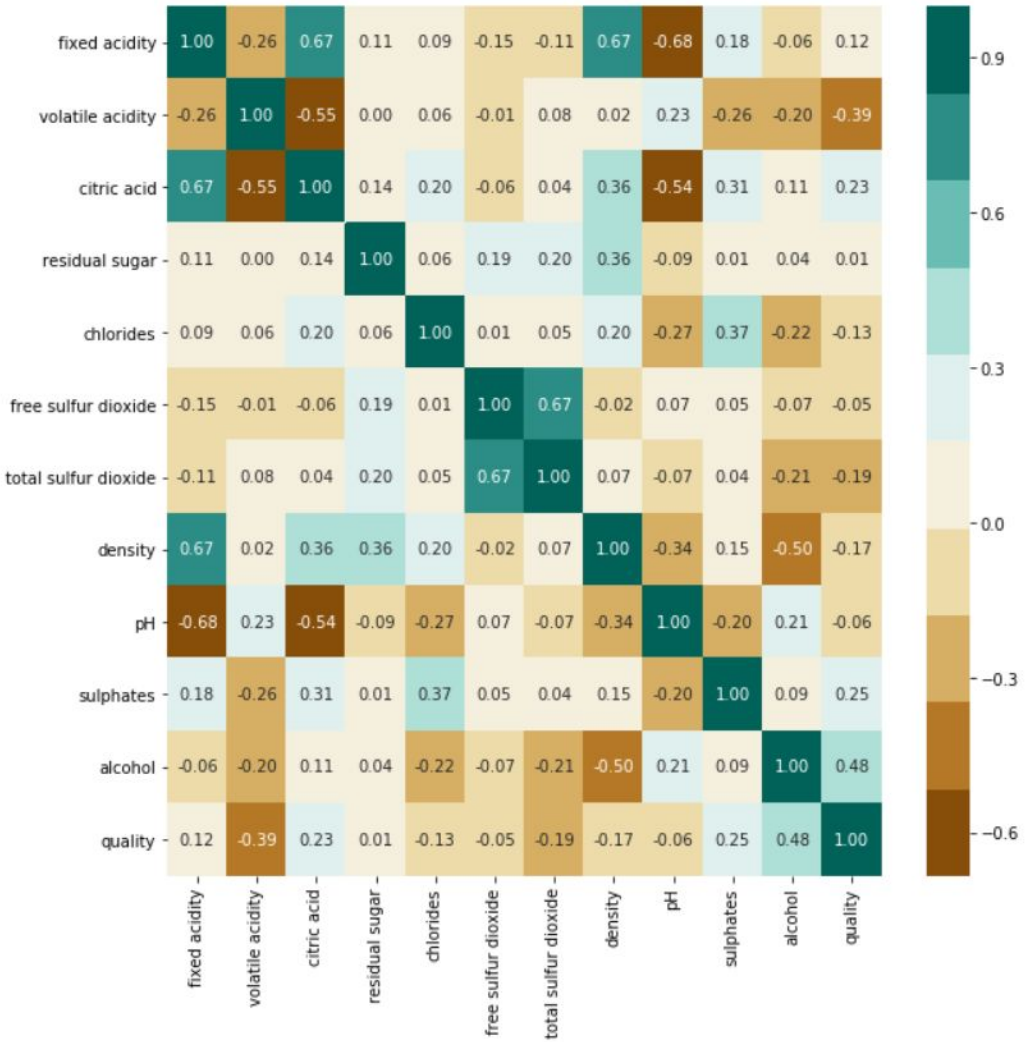
```
corr_matrix = df_wine.corr()
corr_matrix["quality"].sort_values(ascending=False)
```

```
quality                1.000000
alcohol                0.476166
sulphates              0.251397
citric acid            0.226373
fixed acidity          0.124052
residual sugar         0.013732
free sulfur dioxide   -0.050656
pH                    -0.057731
chlorides             -0.128907
density               -0.174919
total sulfur dioxide  -0.185100
volatile acidity      -0.390558
Name: quality, dtype: float64
```

Do alcohol and sulphates correlate to wine quality?

Correlation matrix heat map

```
import seaborn as sns

correlations = df_wine.corr()
fig, ax = plt.subplots(figsize=(10, 10))

colormap =
    sns.color_palette("BrBG", 10)

sns.heatmap(correlations,
    cmap=colormap,
    annot=True,
    fmt=".2f")

ax.set_yticklabels(colum_names);

plt.show()
```



Module 3 – Guided Lab 2: Visualizing Data

Section 3 key takeaways

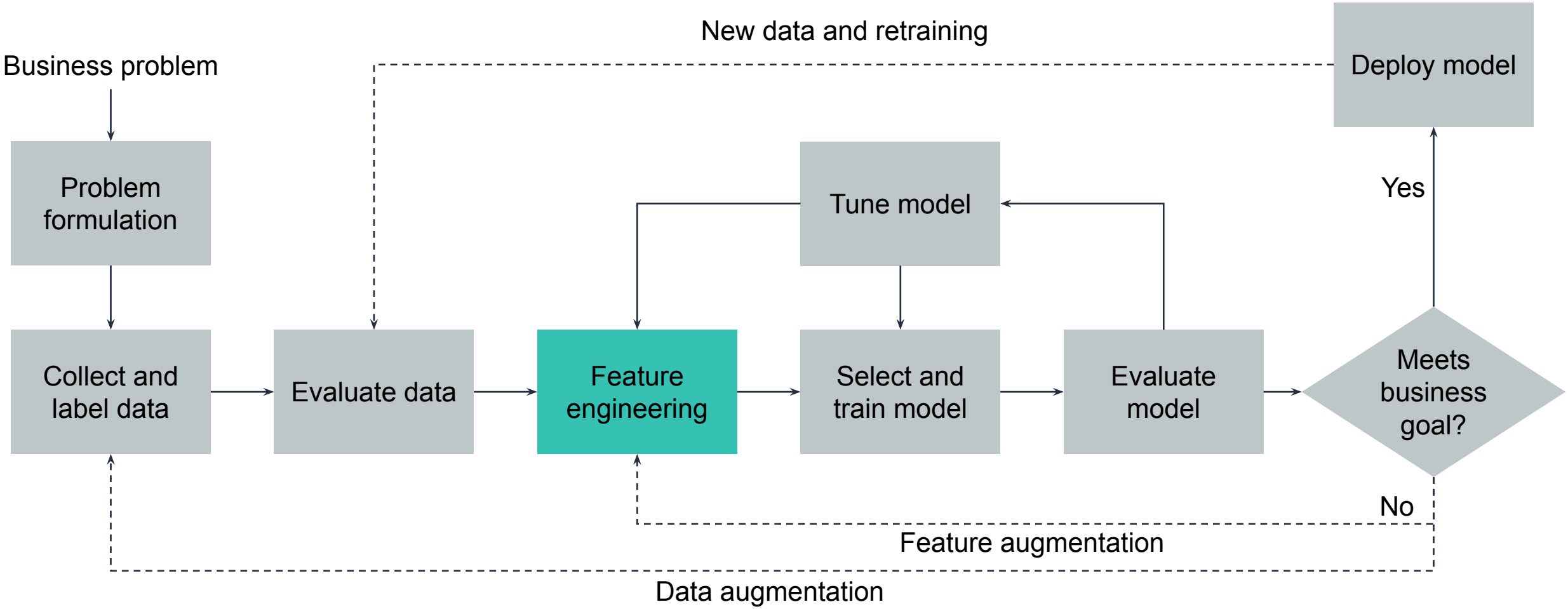
- The first step in evaluating data is to make sure that it's in the right format
- pandas is a popular Python library for working with data
- Use descriptive statistics to learn about the dataset
- Create visualizations with pandas to examine the dataset in more detail

Module 3: Implementing a Machine Learning Pipeline with Amazon SageMaker

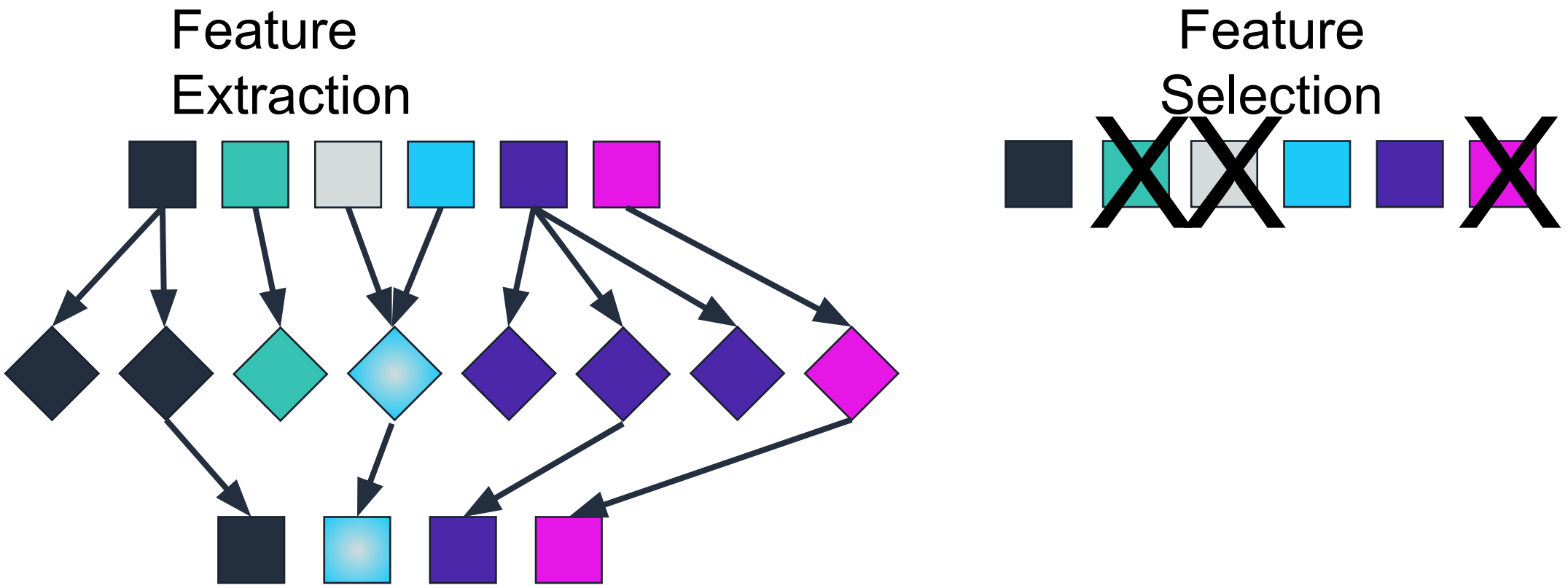
Section 4: Feature engineering

aws academy

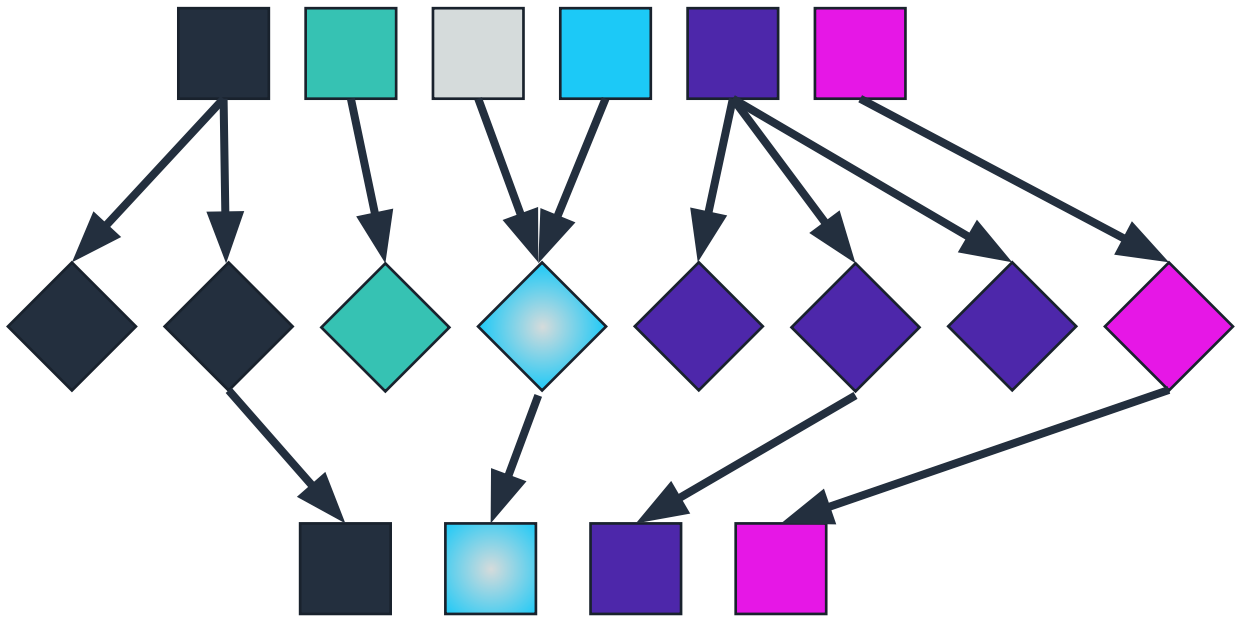
Machine learning pipeline



Feature selection and extraction



Feature extraction



- Invalid values
- Wrong formats
- Misspelling
- Duplicates
- Consistency
- Rescale
- Encode categories
- Remove outliers
- Reassign outliers
- Bucketing
- Decomposition
- Aggregation
- Combination
- Transformation
- Normalization
- Dimensionality reduction

Encoding ordinal data

Categorical data is non-numeric data.

Categorical data must be converted (encoded) to a numeric scale.

Maintenance Costs	Encoding
Low	1
Medium	2
High	3
Very High	4

Encoding non-ordinal data

If data is non-ordinal, the encoded values also must be non-ordinal.

Non-ordinal data might need to be broken into multiple categories.

…	Color
…	Red
…	Blue
…	Green
…	Blue
	Green

…	Red	Blue	Green
…	1	0	0
…	0	1	0
…	0	0	1
…	0	1	0
…	0	0	1

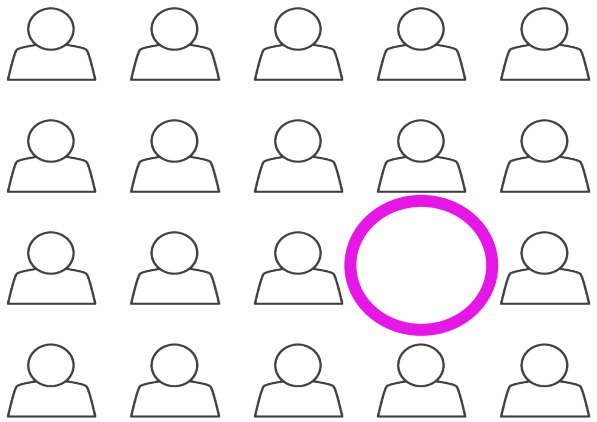
Cleaning data

Types of data to clean:

Type	Example	Action
Variations in strings	Med. vs. Medium	Convert to standard text
Variations in scale	Number of doors vs. number purchased	Normalize to a common scale
Columns with multiple data items	Safe high maintenance	Parse into multiple columns
Missing data	Missing columns of data	Delete rows or impute data
Outliers	Various	

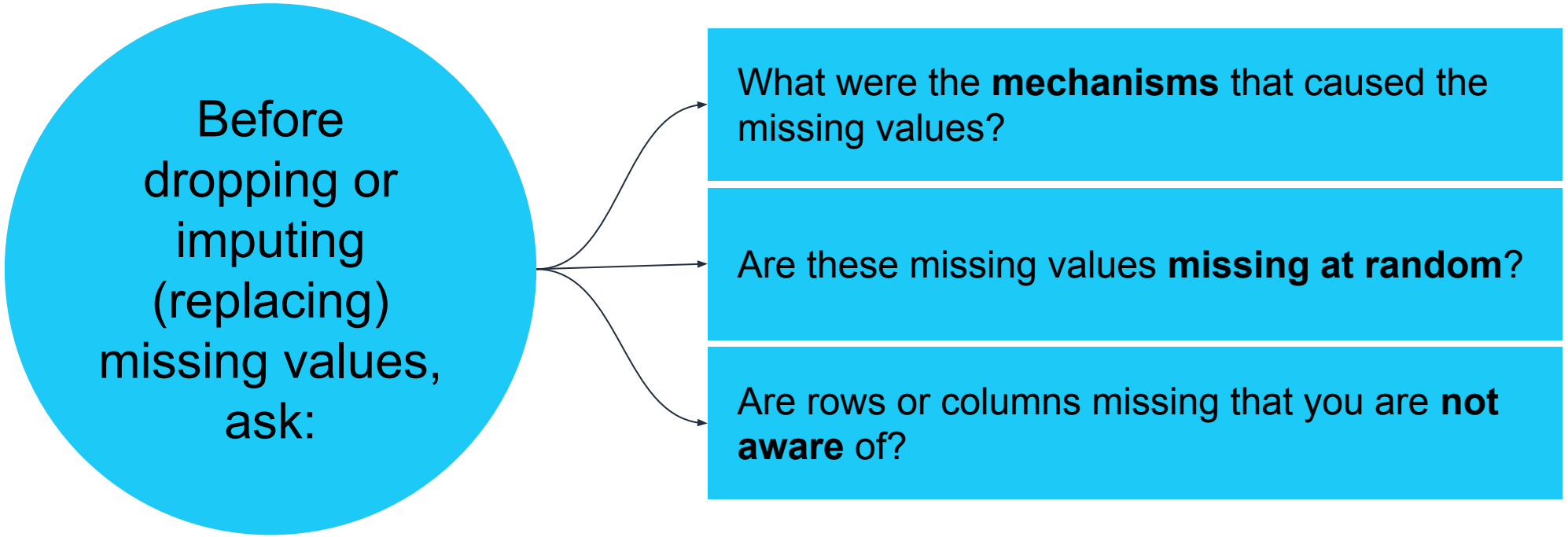
Finding missing data

- Missing data makes it difficult to interpret relationships
- Causes of missing data –
  - Undefined values
  - Data collection errors
  - Data cleaning errors
- Example pandas code to find missing data –



```
df.isnull().sum() #count missing values for each column
df.isnull().sum(axis=1) #count missing values for each row
```

Plan for missing values



Dropping missing values

Drop missing data with pandas

- dropna function to drop rows
  - `df.dropna()`
- dropna function to drop columns with null values
  - `df.dropna (axis=1)`
- dropna function to drop a subset
  - `df.dropna(subset=["buying"])`

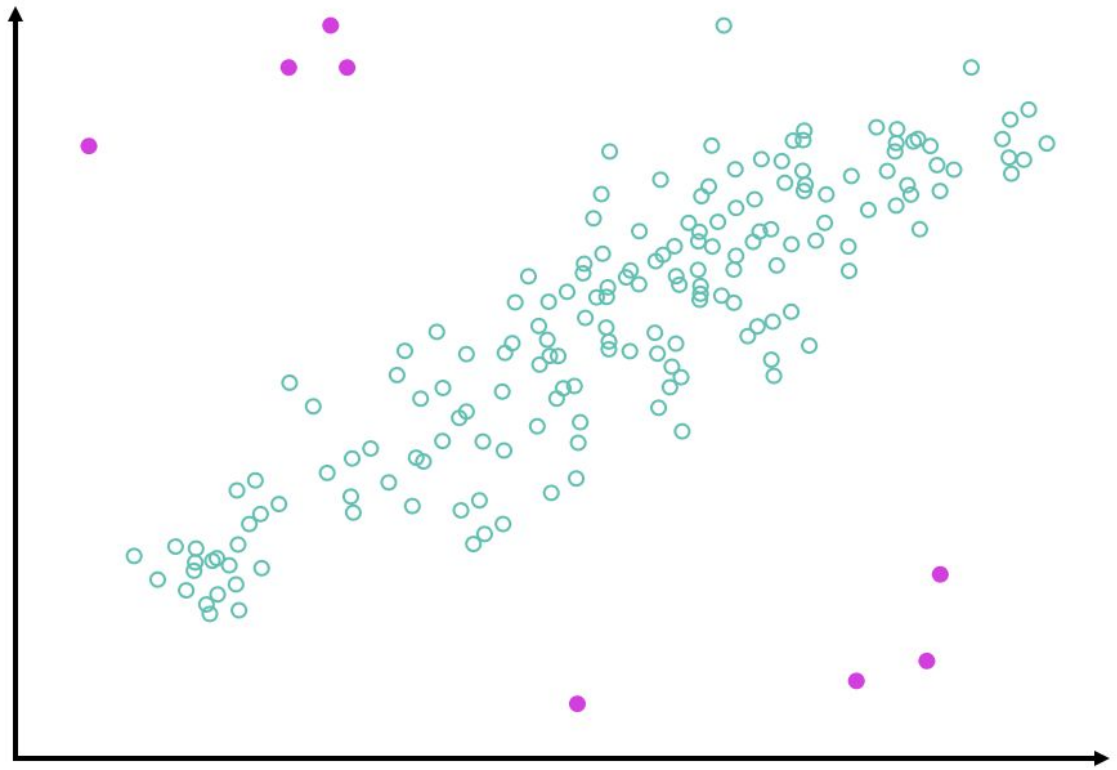
Imputing missing data

- First, determine why the data is missing
- Two ways to impute missing data –
  - Univariate: Adding data for a single row of missing data
  - Multivariate: Adding data for multiple rows of missing data

```
from sklearn.preprocessing import Imputer
import numpy as np
Arr = np.array([[5,3,2,2],[3,None,1,9],[5,2,7,None]])
imputer = Imputer(strategy='mean’)
imp = imputer.fit(arr)
imputer.transform(arr)
```

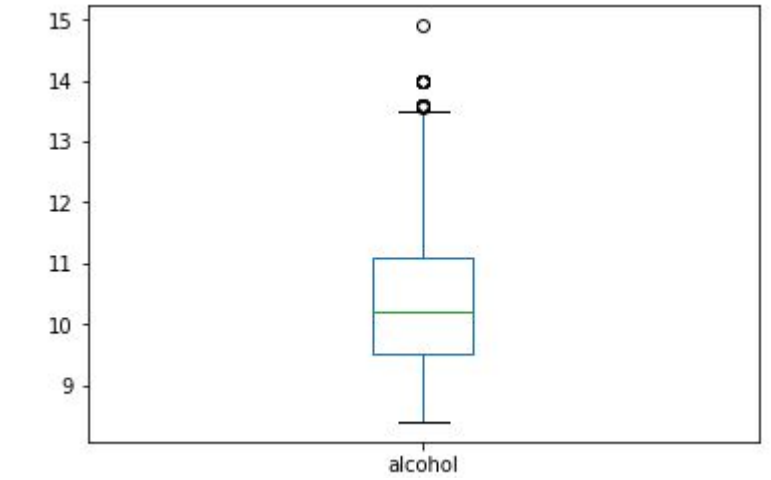
Outliers

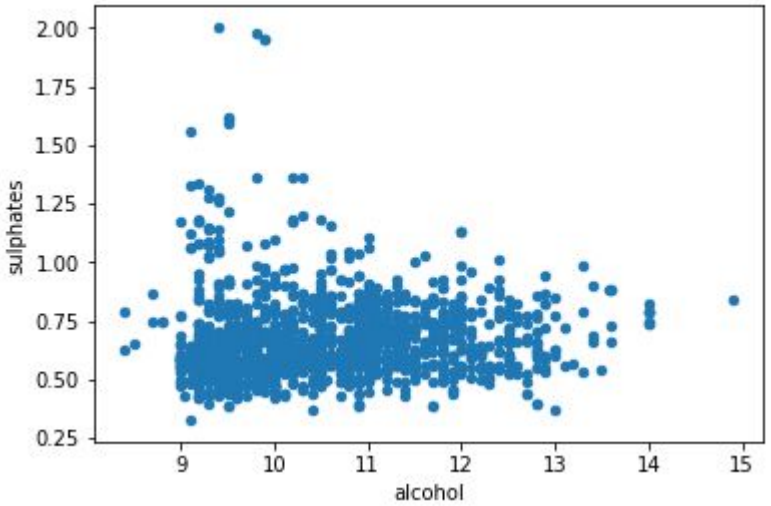
- Outliers can –
  - Provide a broader picture of the data
  - Make accurate predictions difficult
  - Indicate the need for more columns
- Types of outliers –
  - Univariate: Abnormal values for a single variable
  - Multivariate: Abnormal values for a combination of two or more variables



Finding outliers

- Box plots show variation and distance from the mean
  - Example to the right shows a box plot for the amount of alcohol in a collection of wines
- Scatter plots can also show outliers
  - Example to the right shows a scatter plot that shows the relationship between alcohol and sulphates in a collection of wines



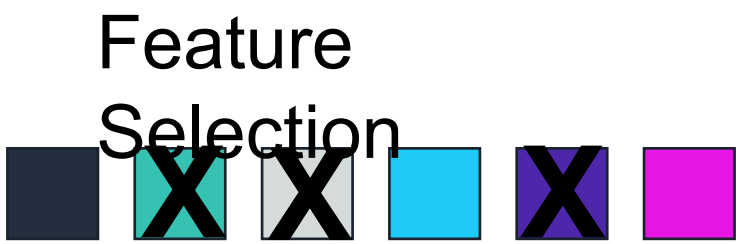


Dealing with outliers

Delete the outlier	Transform the outlier	Impute a new value for the outlier
Outlier is based on an artificial error.	Reduces the variation that the extreme outlier value causes and the outlier's influence on the dataset.	You might use the mean of the feature, for instance, and impute that value to replace the outlier value.

Feature selection

- Filter method
  - Use statistical methods
- Wrapper methods
  - Actually train the model
- Embedded methods
  - Integrated with model

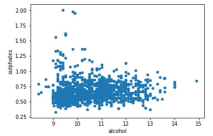


Feature selection: Filter methods

- Measures –
  - Pearson's correlation
  - Linear discriminant analysis (LDA)
  - Analysis of variance (ANOVA)
  - Chi-square

All features

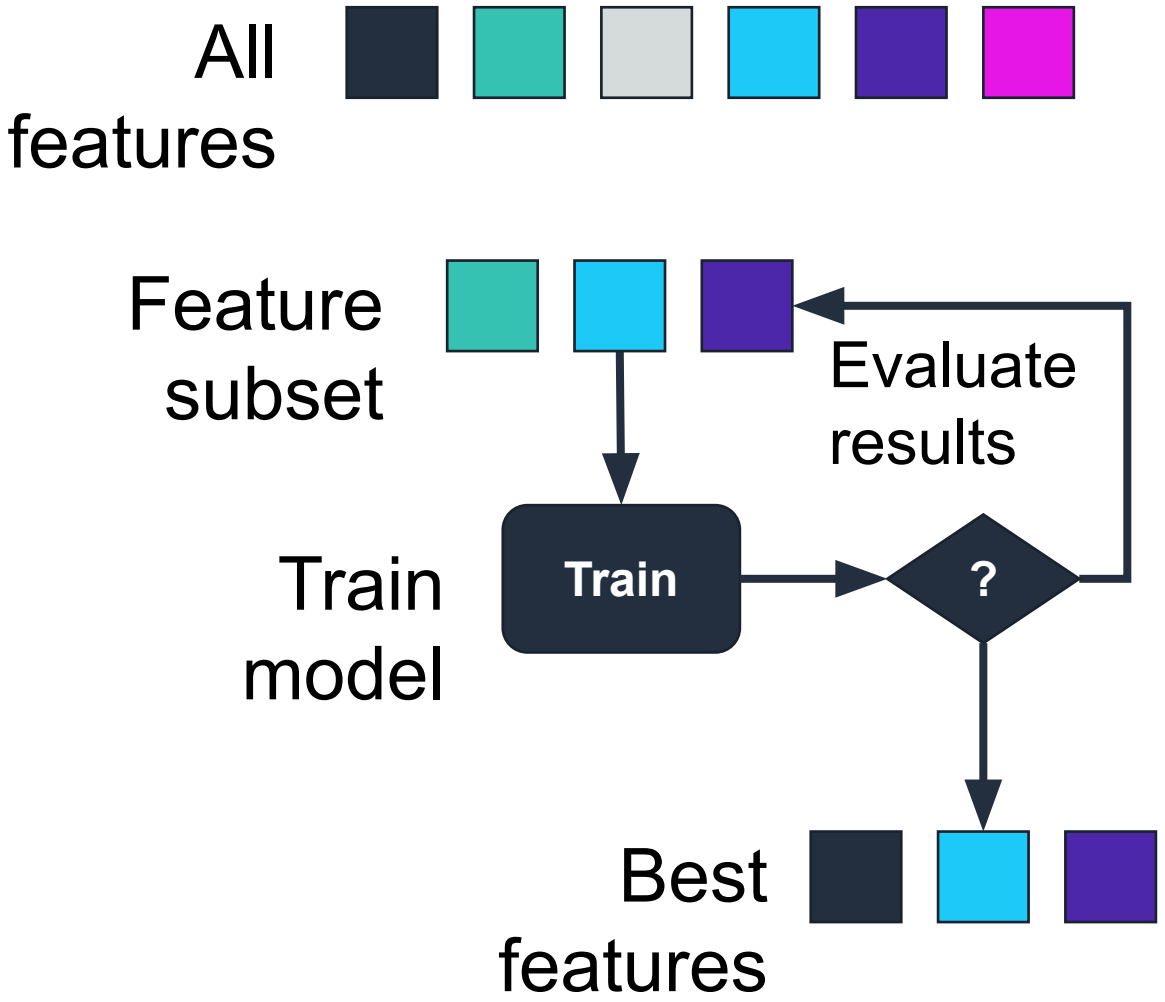
Statistics and correlation



Best features

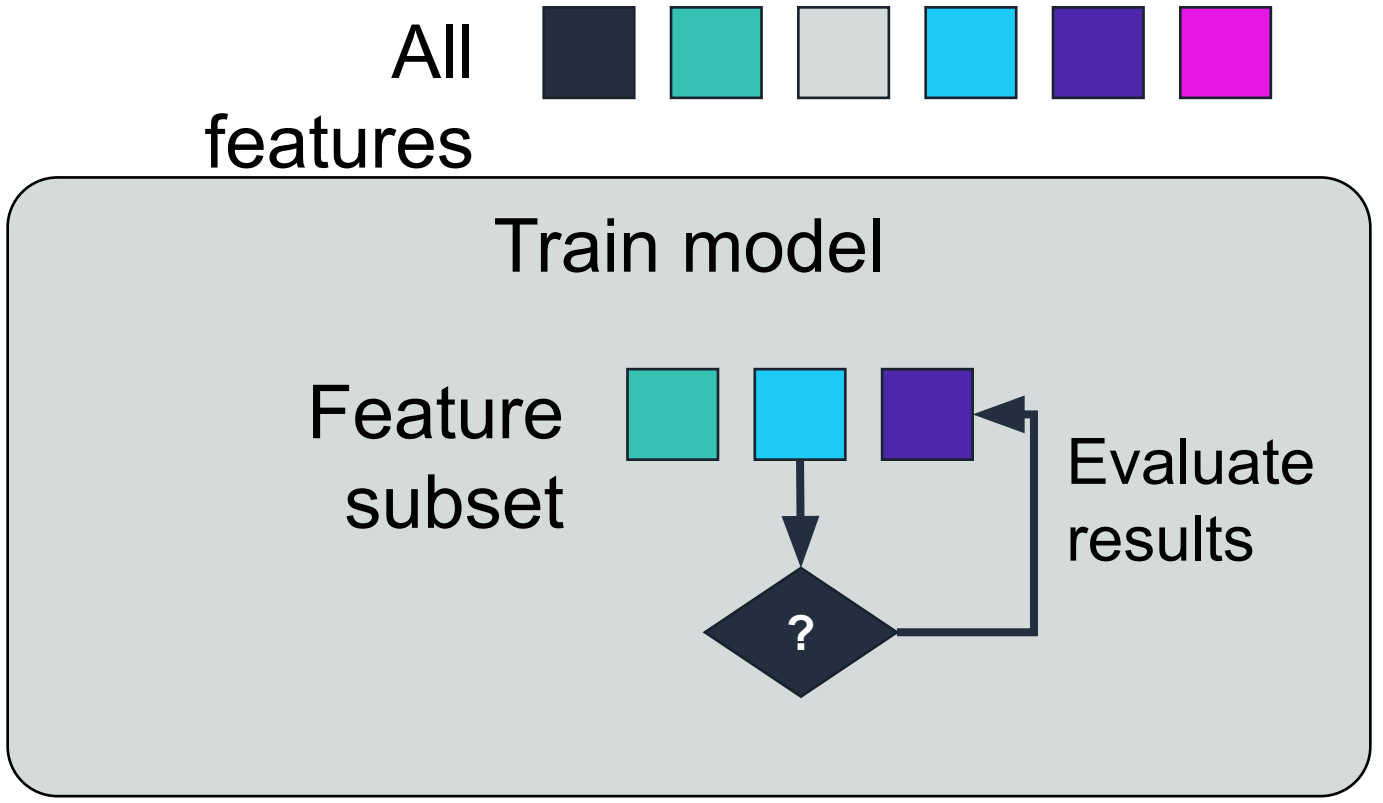
Feature selection: Wrapper

- Methods –
  - Forward selection
  - Backward selection



Feature selection: Embedded methods

- Methods –
  - Decision trees
  - LASSO and RIDGE



Module 3 – Guided Lab 3: Encoding Categorical Data

Section 4 key takeaways

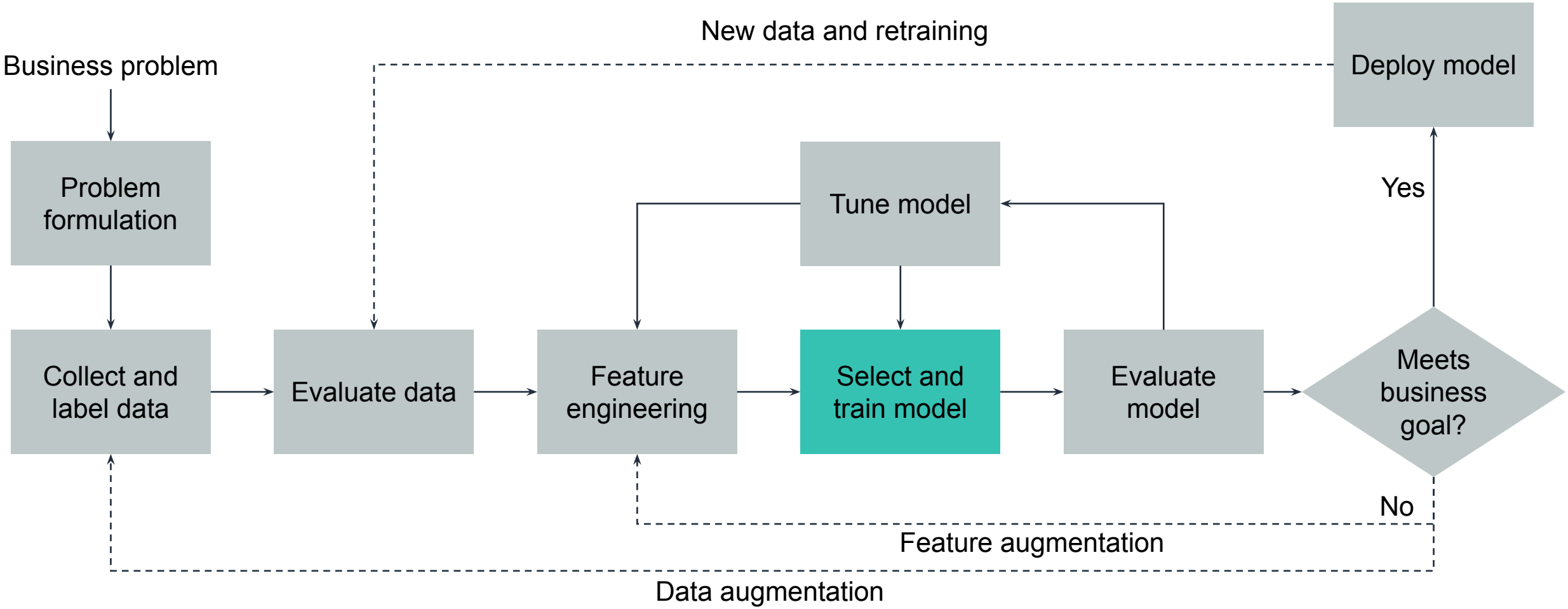
- Feature engineering involves –
  - Selection
  - Extraction
- Preprocessing gives you better data
- Two categories for preprocessing –
  - Converting categorical data
  - Cleaning up dirty data
- Use categorical encoding to convert categorical data
- Various types of dirty data –
  - Missing data
  - Outliers
- Develop a strategy for dirty data –
  - Replace or delete rows with missing data
  - Delete, transform, or impute new values for outliers

Module 3: Implementing a Machine Learning Pipeline with Amazon SageMaker

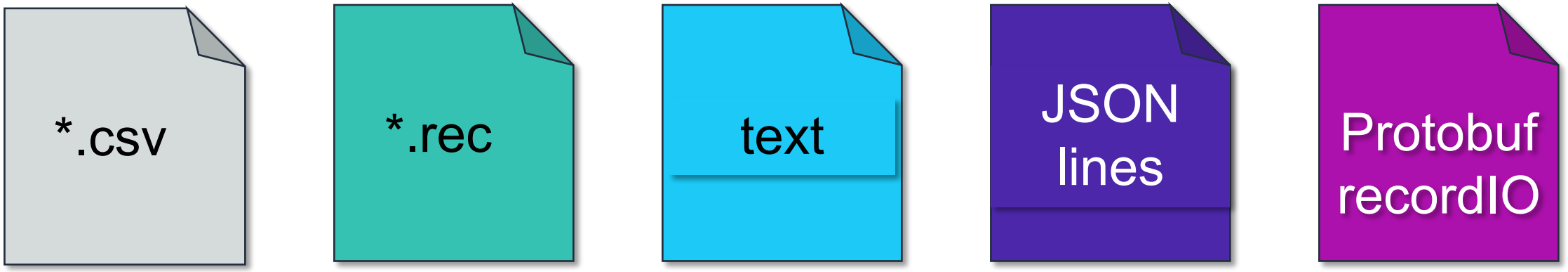
Section 5: Training

aws academy

Machine learning pipeline

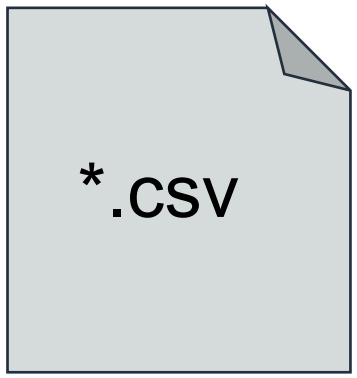


File formats for machine learning



Formatting the data for an algorithm

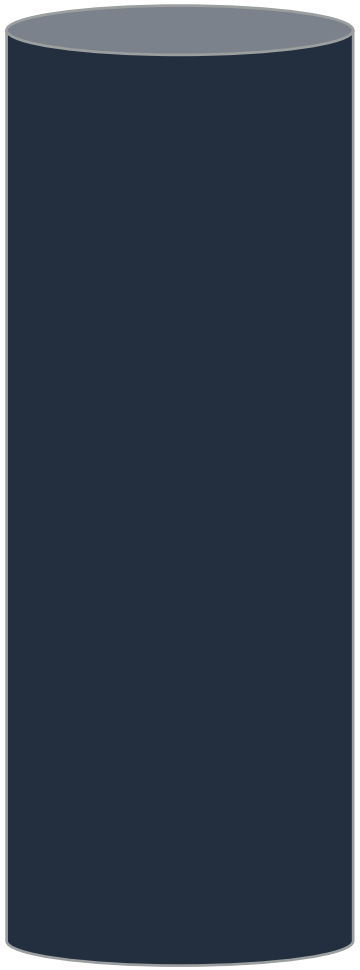
When you use CSV format, use the first column as the target variable.

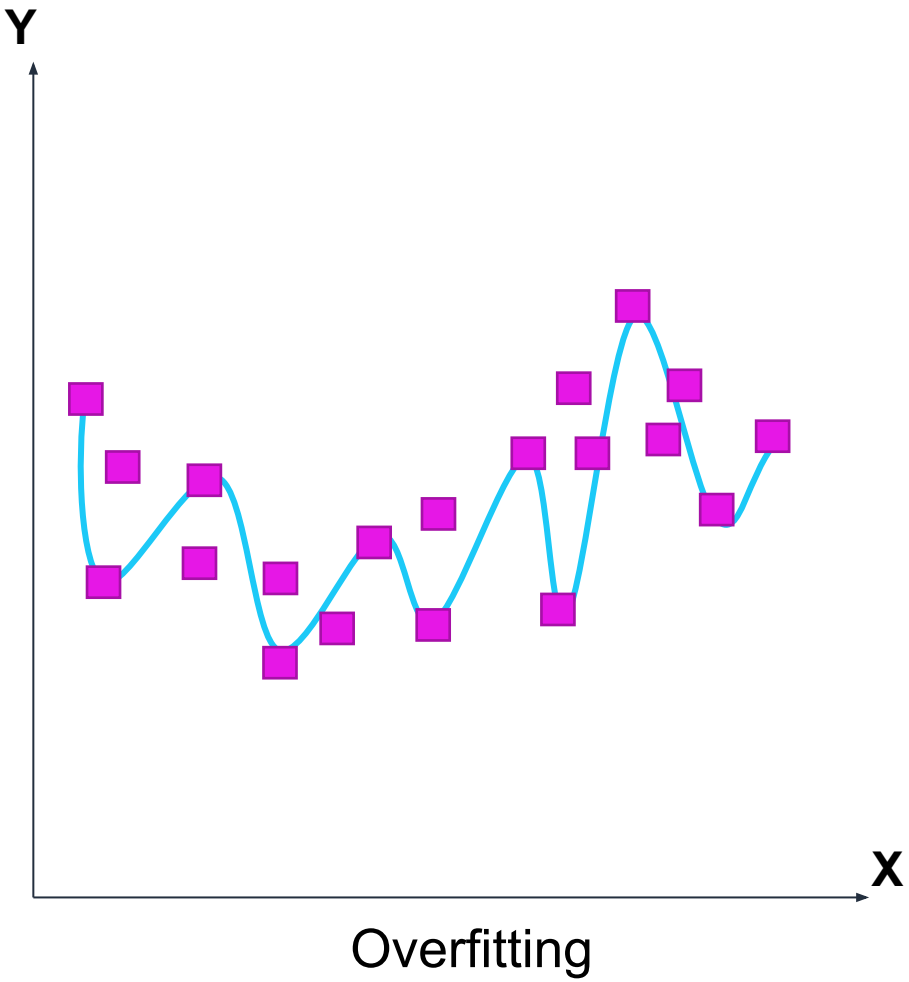


Target	Feature1	Feature 2	Feature 3	Feature 4	Feature 5
2	8.39	92.3	1	0	5
3	7.82	201.39	0	1	3
3	2.39	245.21	0	0	10
2	8.48	183.92	1	1	1
1	5.80	28.2	0	1	1

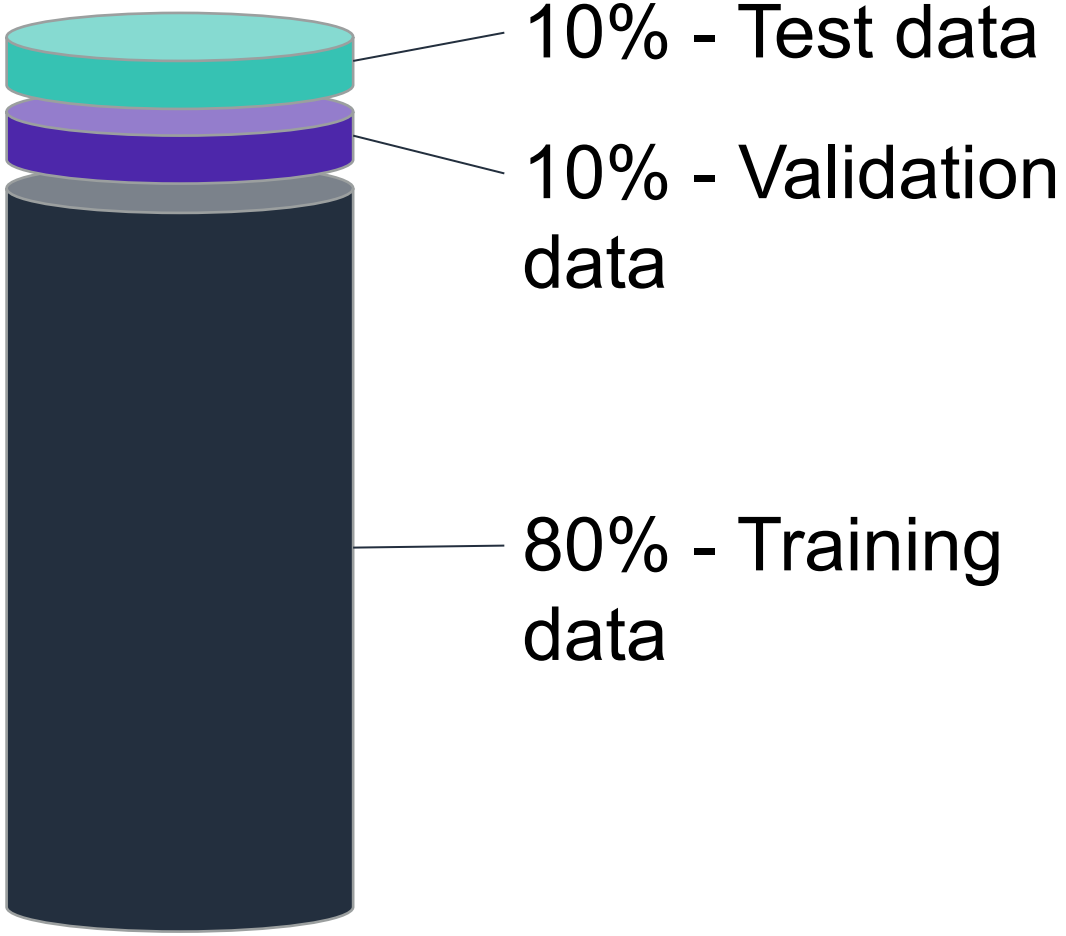
Why split the data?

All data that is used for training and evaluation

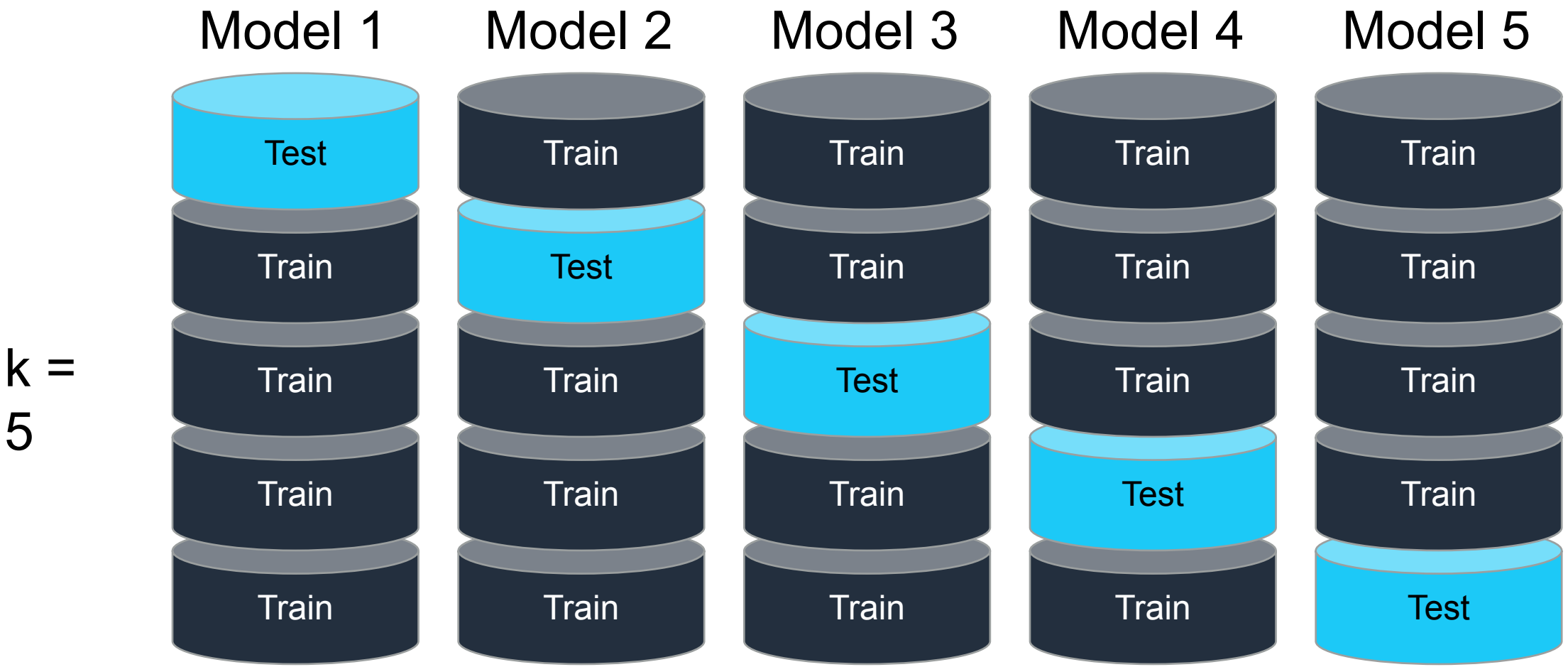




Holdout method

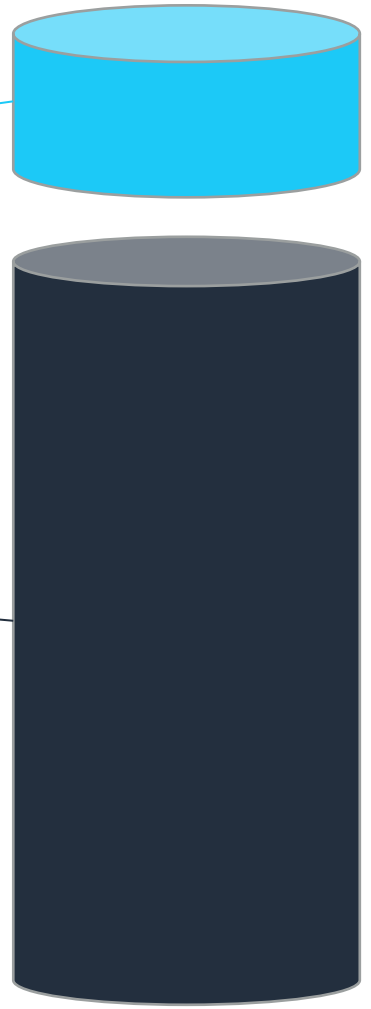


K-fold cross validation



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	…	…



Test data

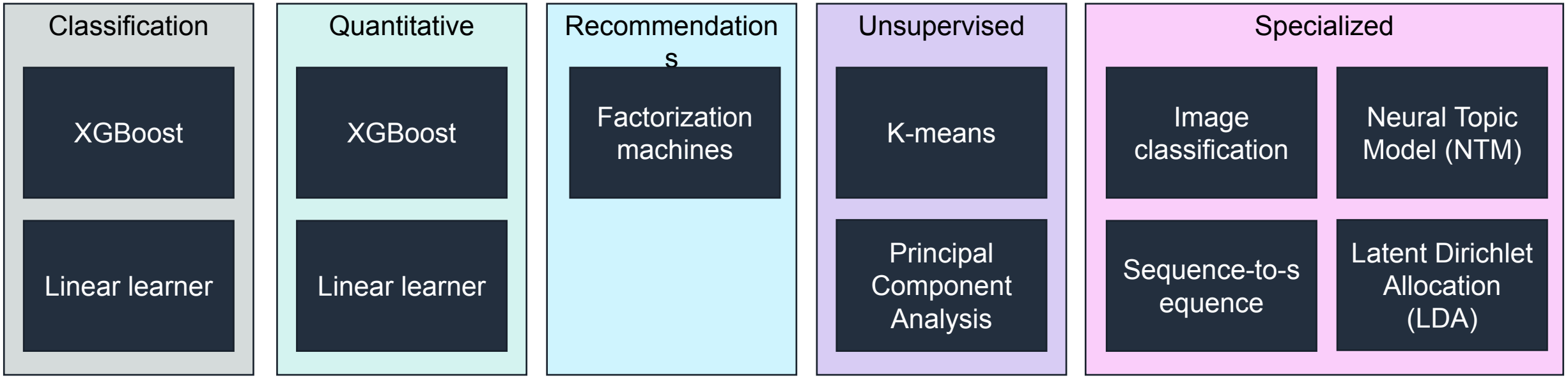
Training data

Training models with Amazon SageMaker

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

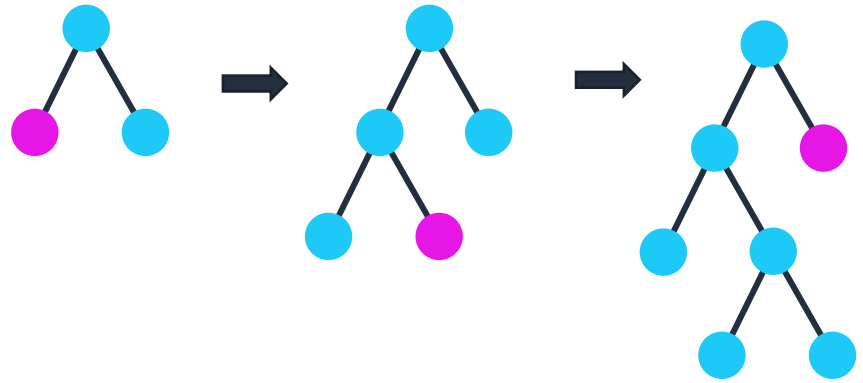
Amazon SageMaker built-in algorithms	Amazon SageMaker supported frameworks	Amazon SageMaker custom frameworks	AWS Marketplace algorithms

Amazon SageMaker built-in algorithms



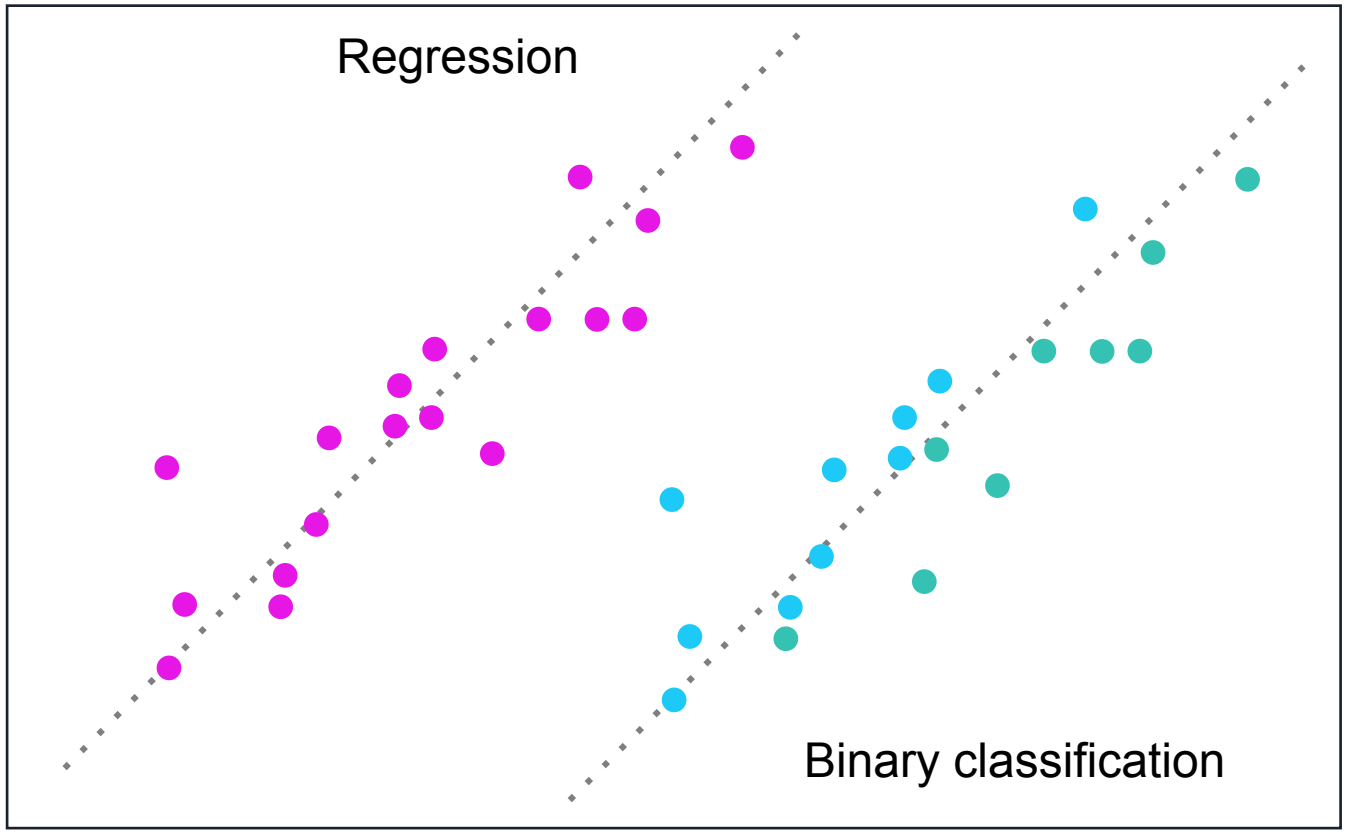
XGBoost

- Is an 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



Linear learner

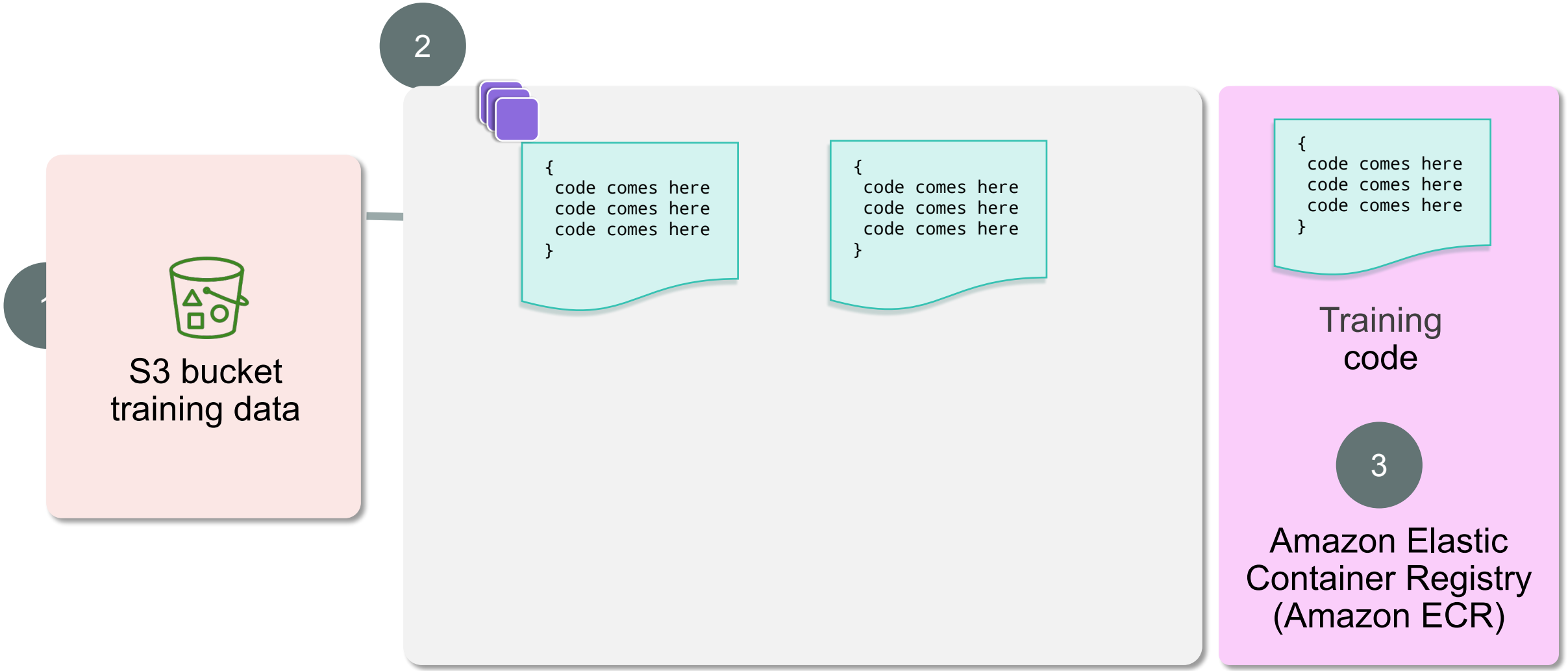
- Provides a solution for both classification and regression problems
- Enables you to simultaneously explore different training objectives and choose the best solution from your validation set



K-means

- 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.

Creating a training job in Amazon SageMaker



Demonstration: Training a Model Using Amazon SageMaker

Module 3 – Guided Lab 4: Splitting Data and Training a Model with XGBoost

Section 5 key takeaways

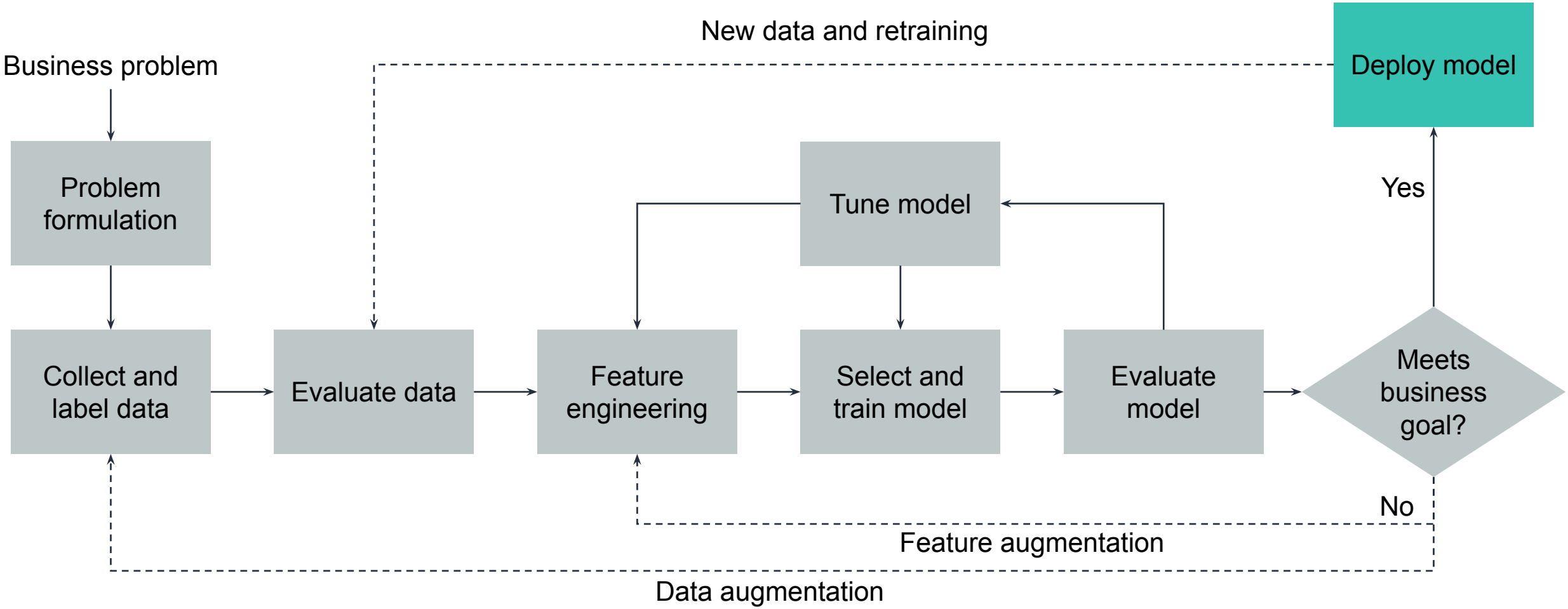
- Split data into training and testing sets –
  - Optionally, split into three sets, including validation set
- Can use k-fold cross validation to use all the non-test data for validation
- Can use two key algorithms for supervised learning –
  - XGBoost
  - Linear learner
- Use k-means for unsupervised learning
- Use Amazon SageMaker training jobs

Module 3: Implementing a Machine Learning Pipeline with Amazon SageMaker

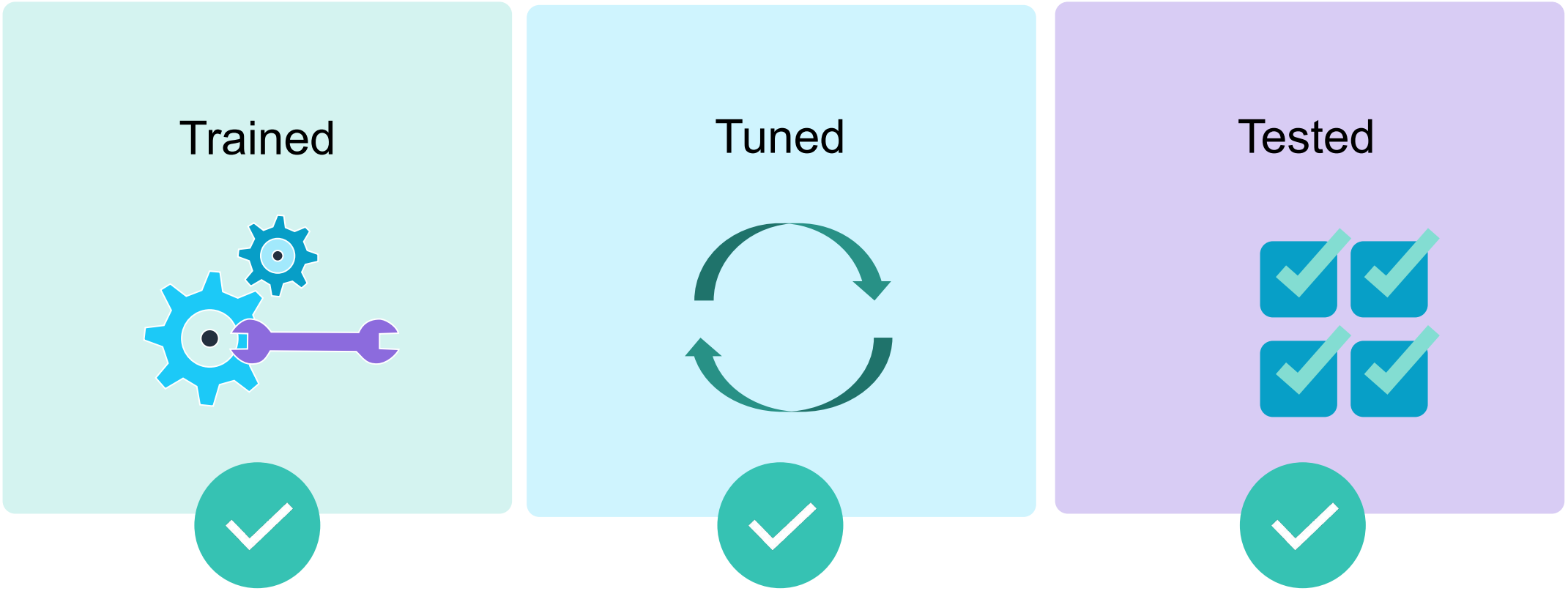
Section 6: Hosting and using the model

aws academy

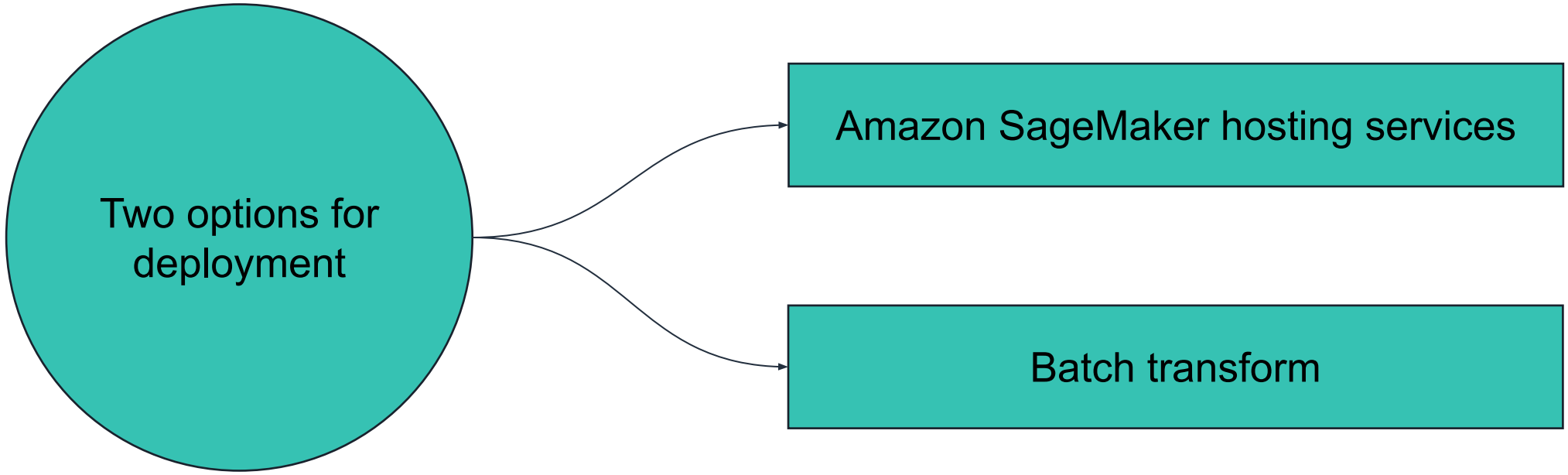
Machine learning pipeline



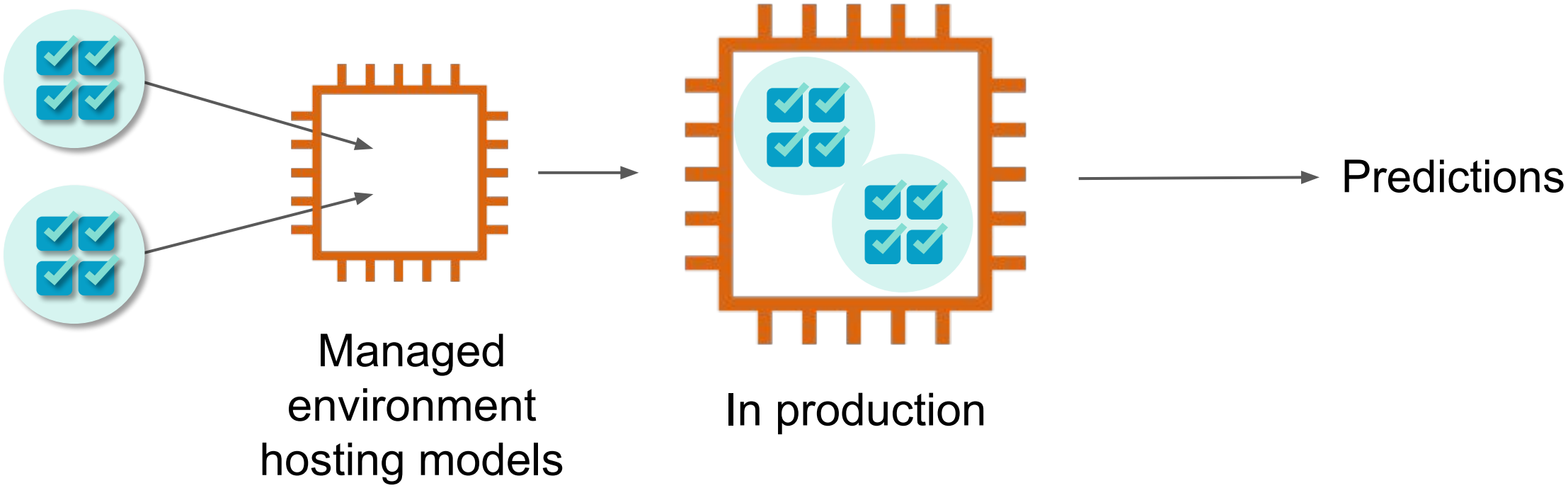
Is your model ready to deploy?



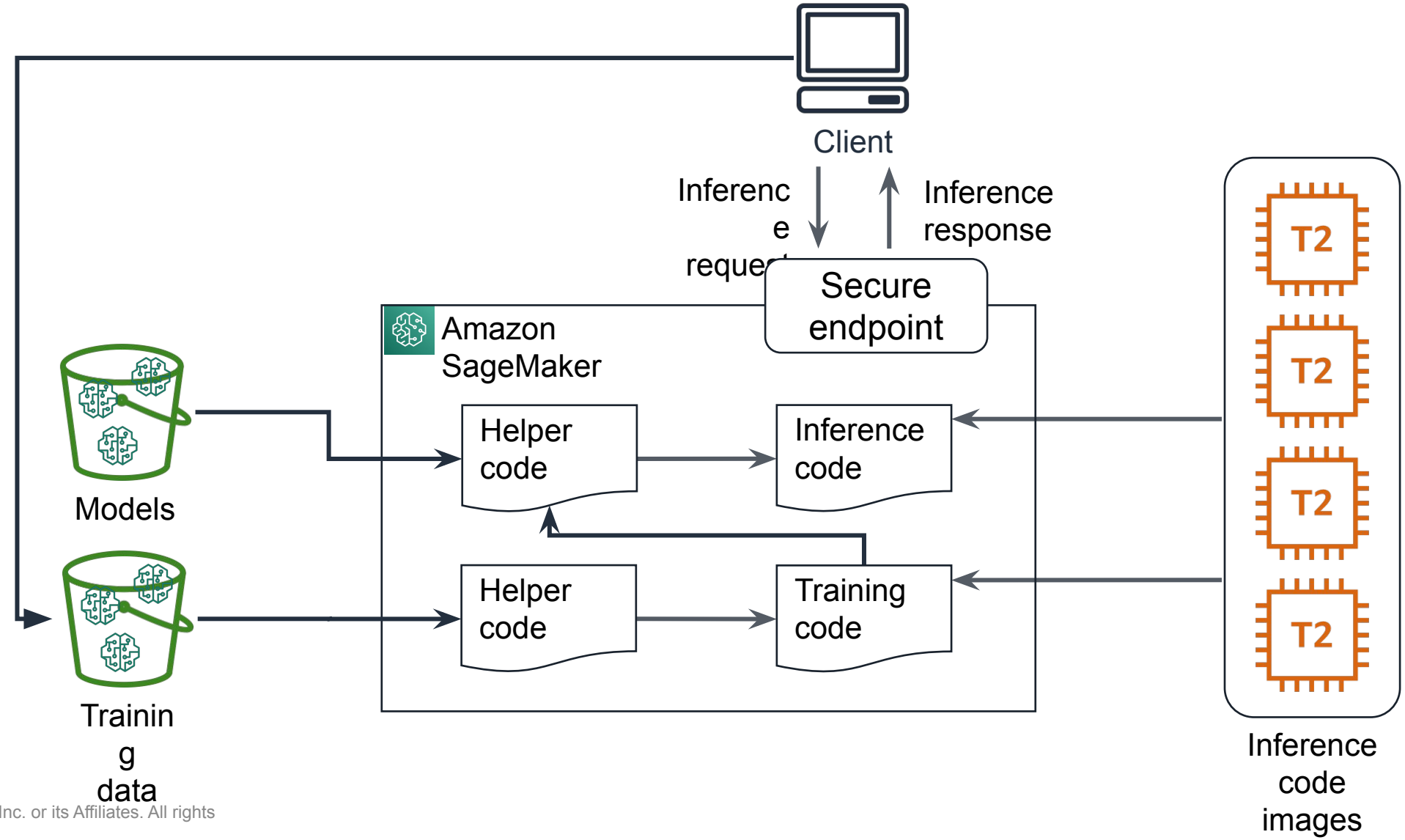
Deployment options



The goal of deployment

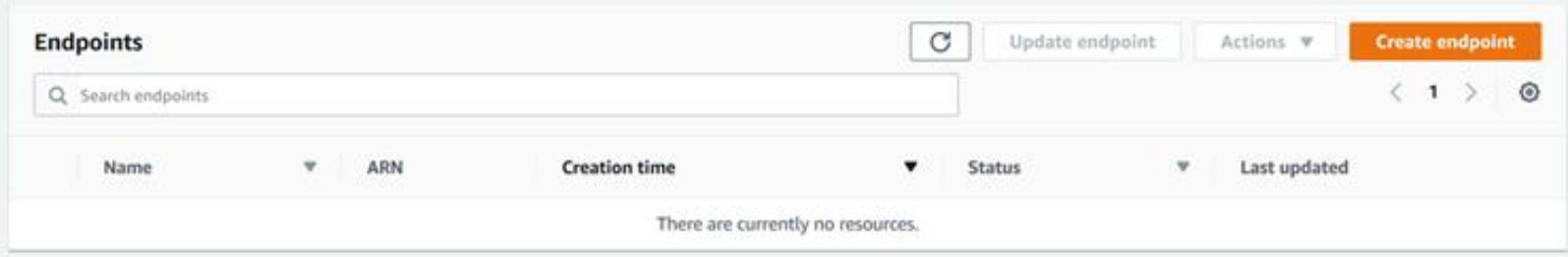


Deploying with Amazon SageMaker



Creating an endpoint

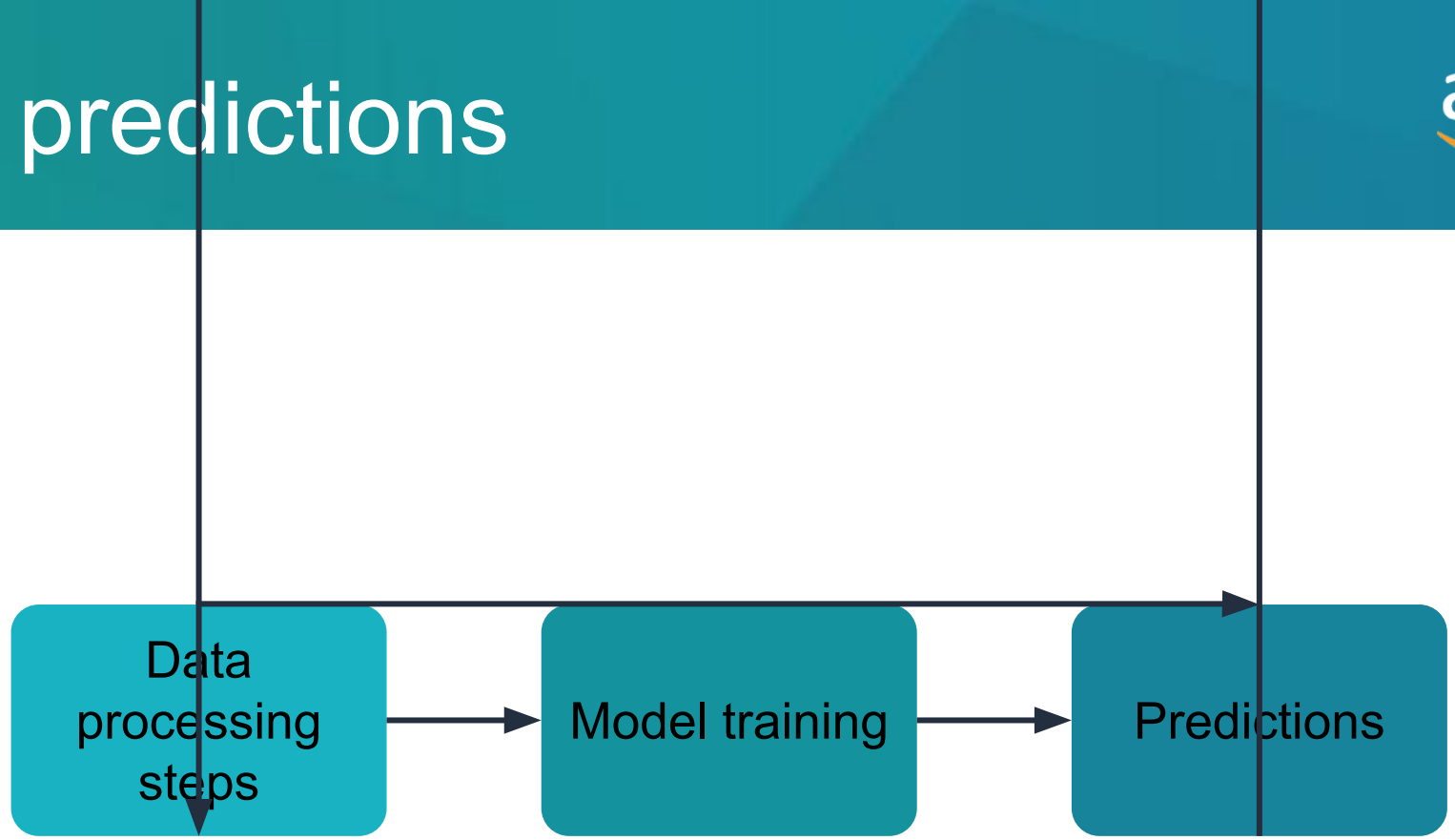
You can create an endpoint in the console:



Or, you can use the Amazon SageMaker SDK:

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

Obtaining predictions



Module 3 – Guided Lab 5: Hosting and Consuming a Model on AWS

Section 6 key takeaways

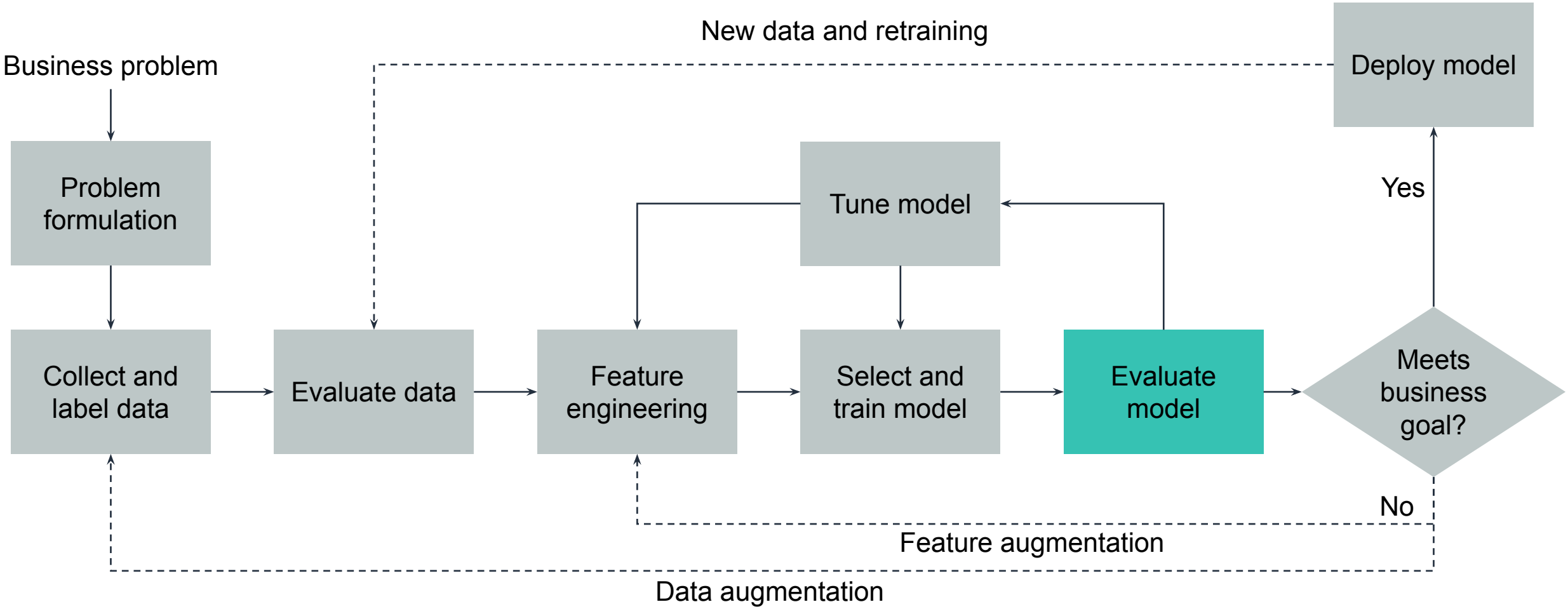
- Can use two options for deployment
  - Amazon SageMaker hosting
  - Batch transform
- Deploy only after you have tested your model
  - Goal is to generate predictions for client applications
- Create an endpoint
  - Single-model endpoint for simple use cases
  - Multi-model endpoint to support multiple use cases

Module 3: Implementing a Machine Learning Pipeline with Amazon SageMaker

Section 7: Evaluating the accuracy of the model

aws academy

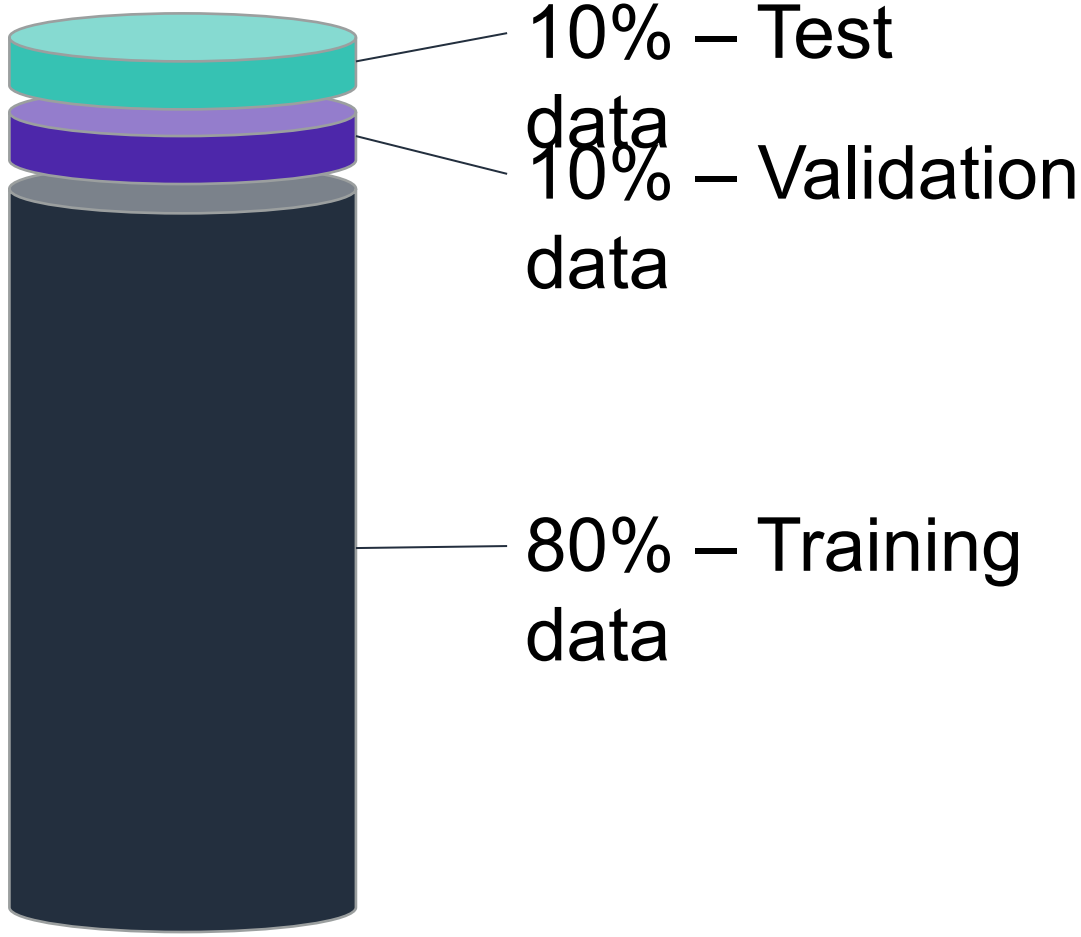
Machine learning pipeline



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

Testing and evaluation:

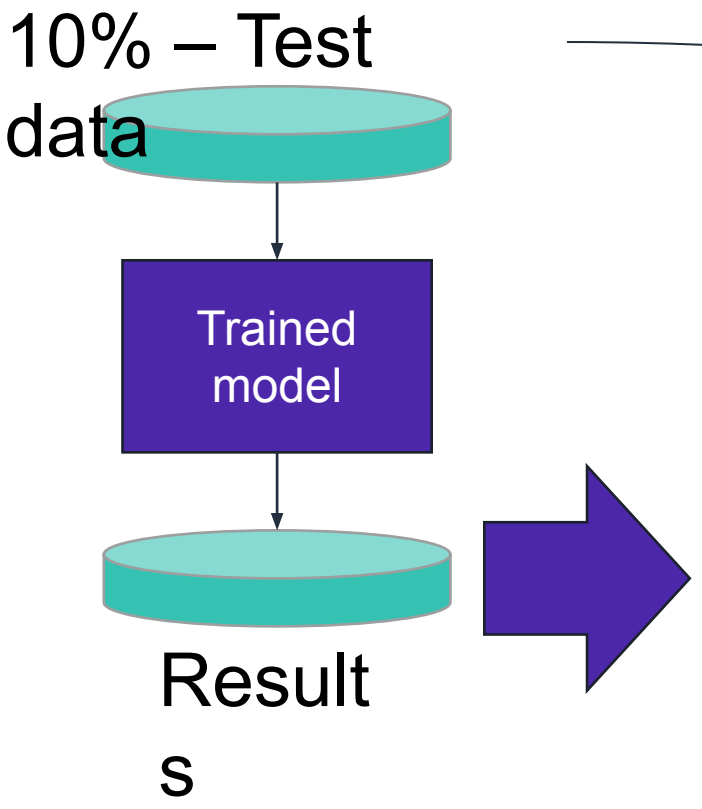
To evaluate the predictive quality of the trained model



Success metric

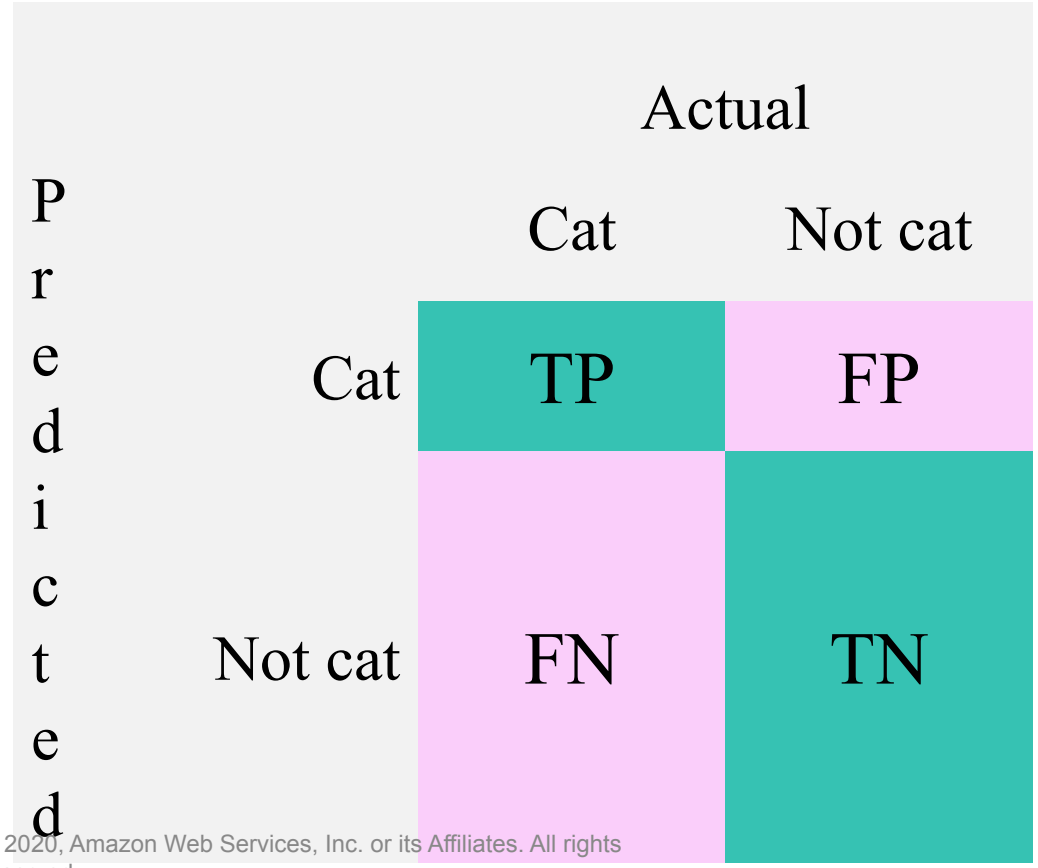
The model metric must align to both the business problem and the success metric.

Confusion matrix



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

Confusion matrix terminology



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

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

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

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

Which model is better?

Confusion matrices from two models that use the same data:

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

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

Sensitivity

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

Sensitivity:

What percentage of cats were correctly identified?

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

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

60% of cats were identified as cats.

Specificity

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

Specificity:

What percentage of ***not*** cats were correctly identified?

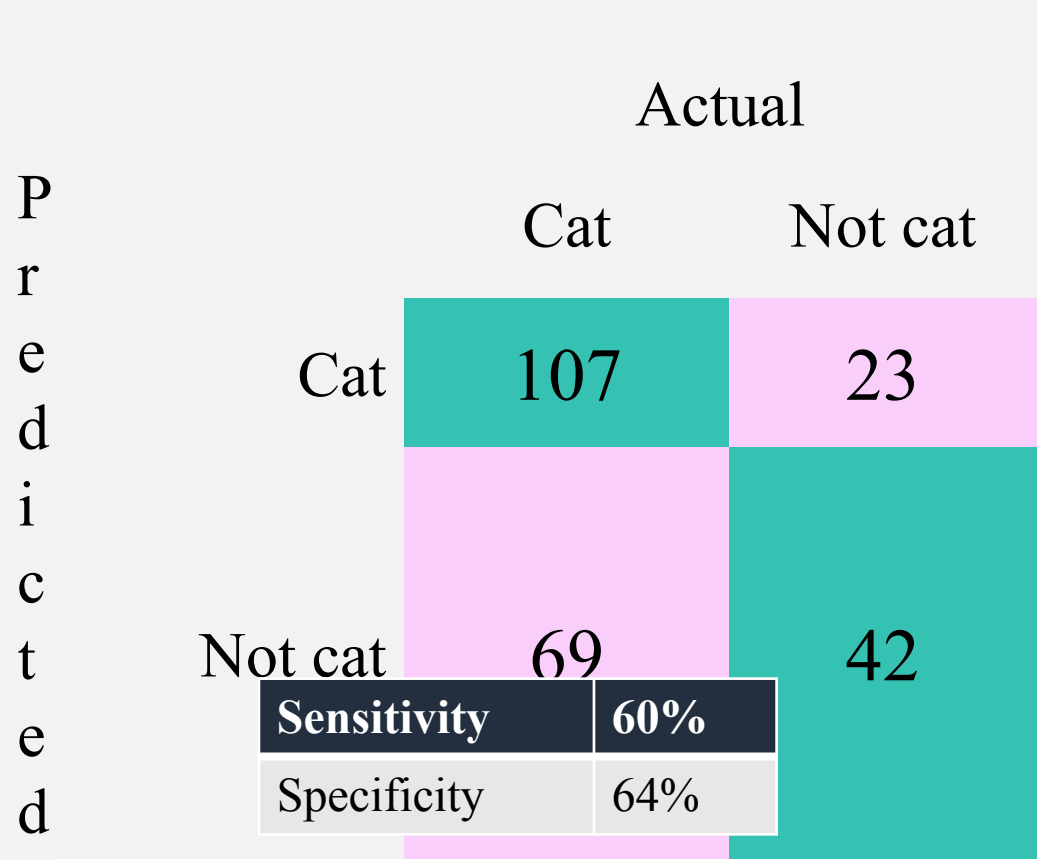
$$\text{specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

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

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

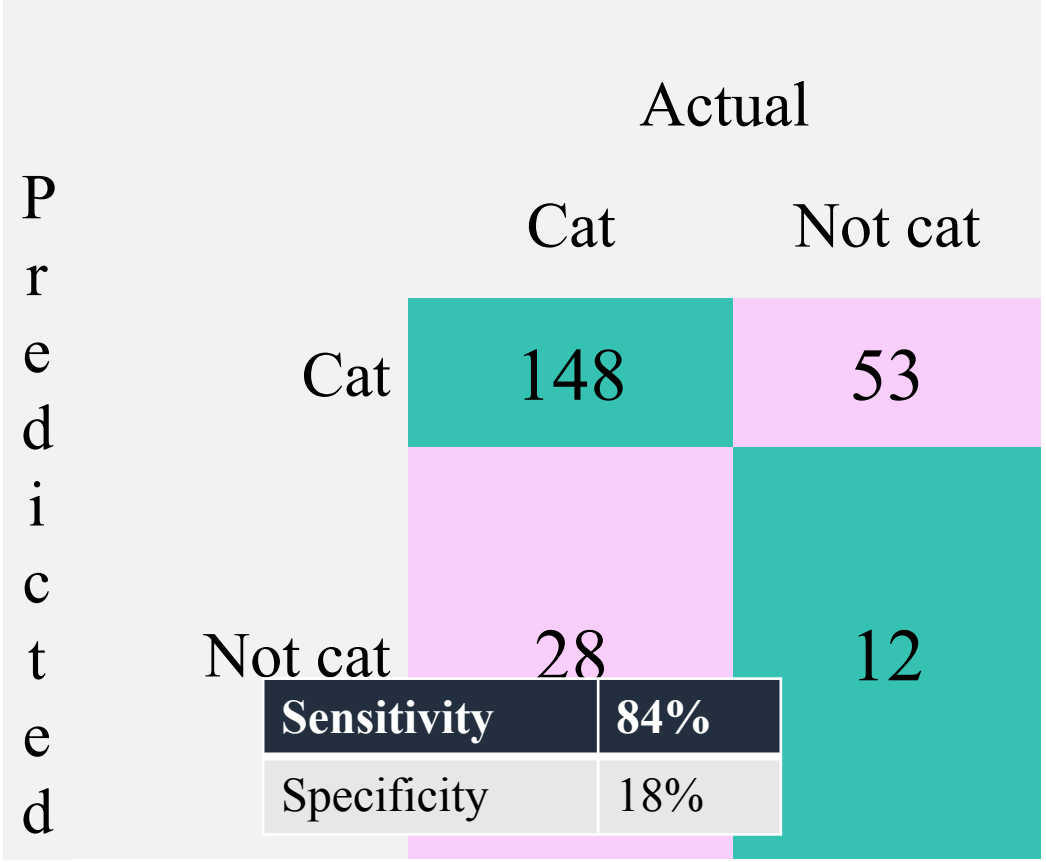
Which model is better now?

Model A



Sensitivity	60%
Specificity	64%

Model B



Sensitivity	84%
Specificity	18%

Thresholds

Models return probabilities:

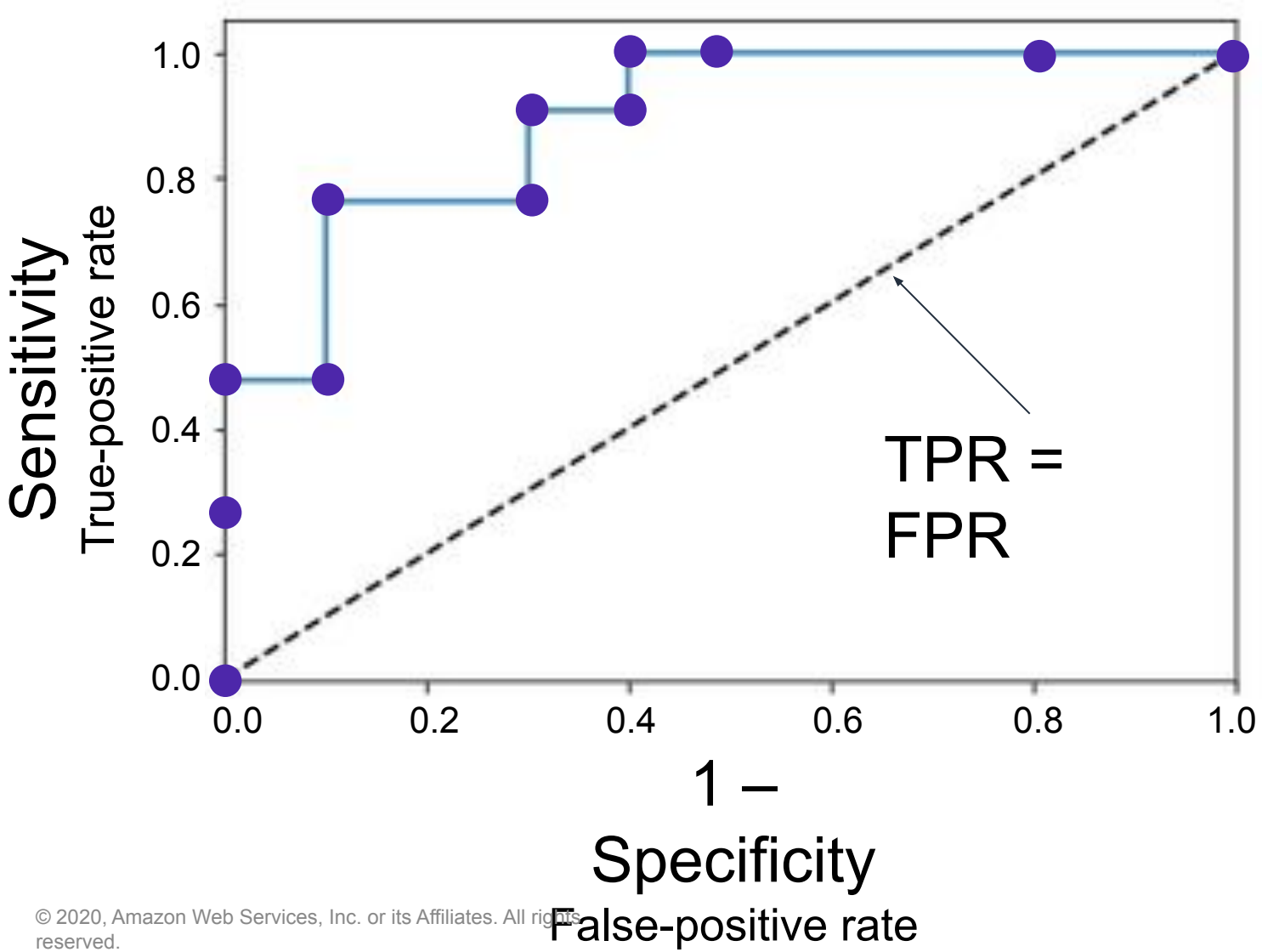
Predicted
0.96
0.77
0.58
0.01
0.47

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

if [predicted] < threshold — NOT cat!

Changing the threshold can change the prediction.

Classification: ROC graph

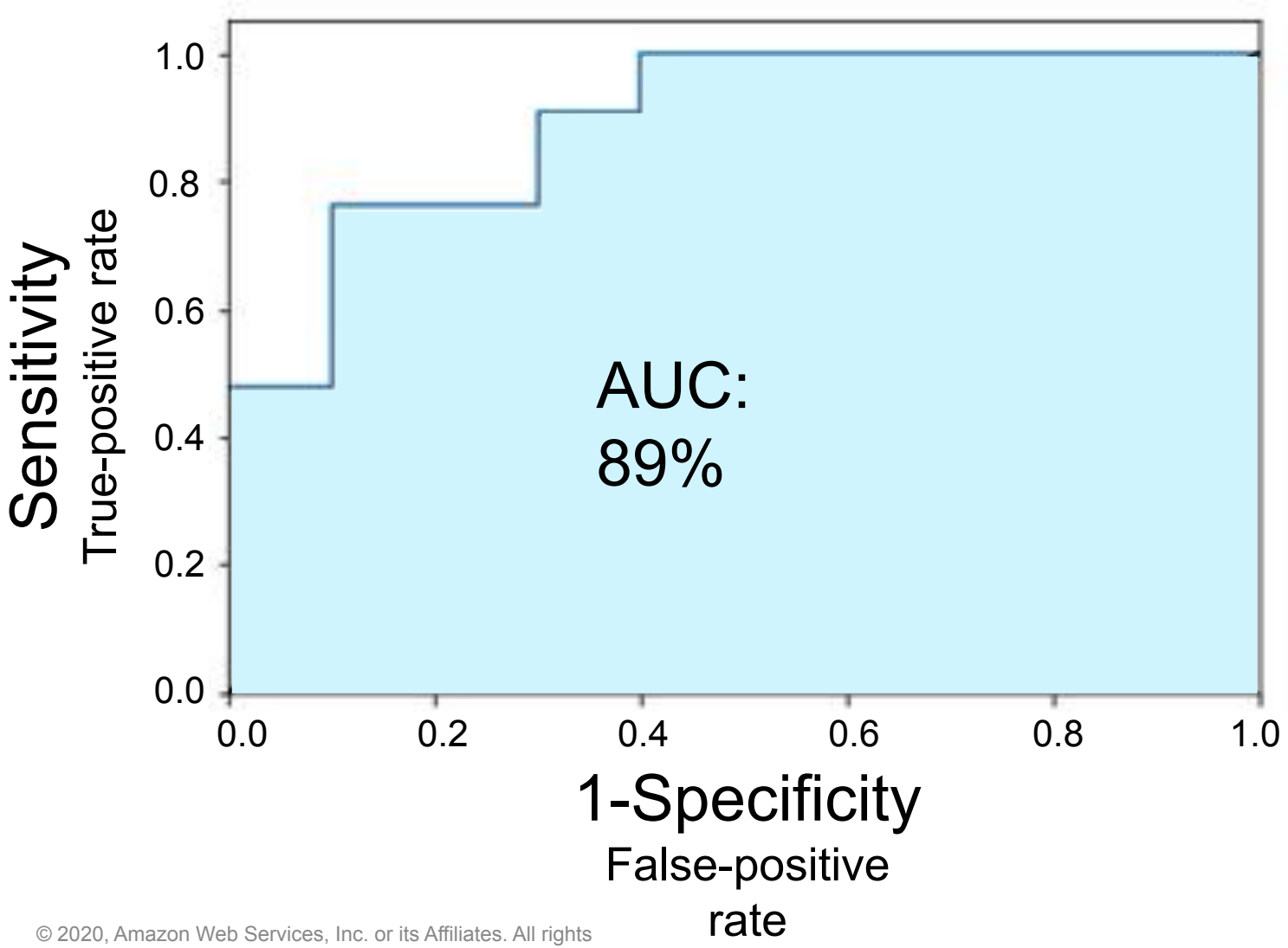


Receiver operating characteristic (ROC)

Plot points for each threshold value

Select threshold based on business case

Classification: AUC-ROC



Area
under the
curve

AUC-ROC is used to easily compare one model to another

Classification: Other metrics

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

Model's score

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

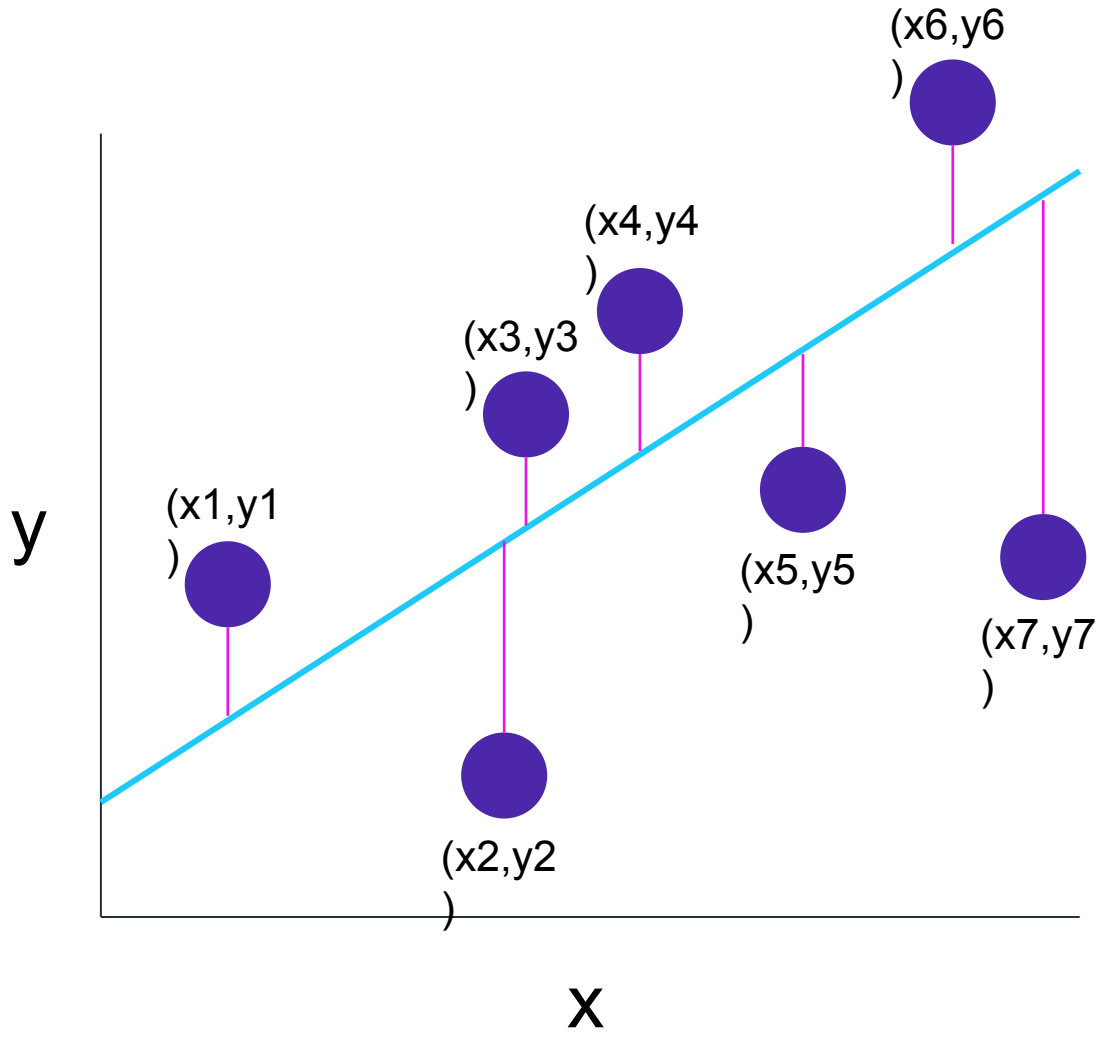
Proportion of positive results that were correctly classified

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

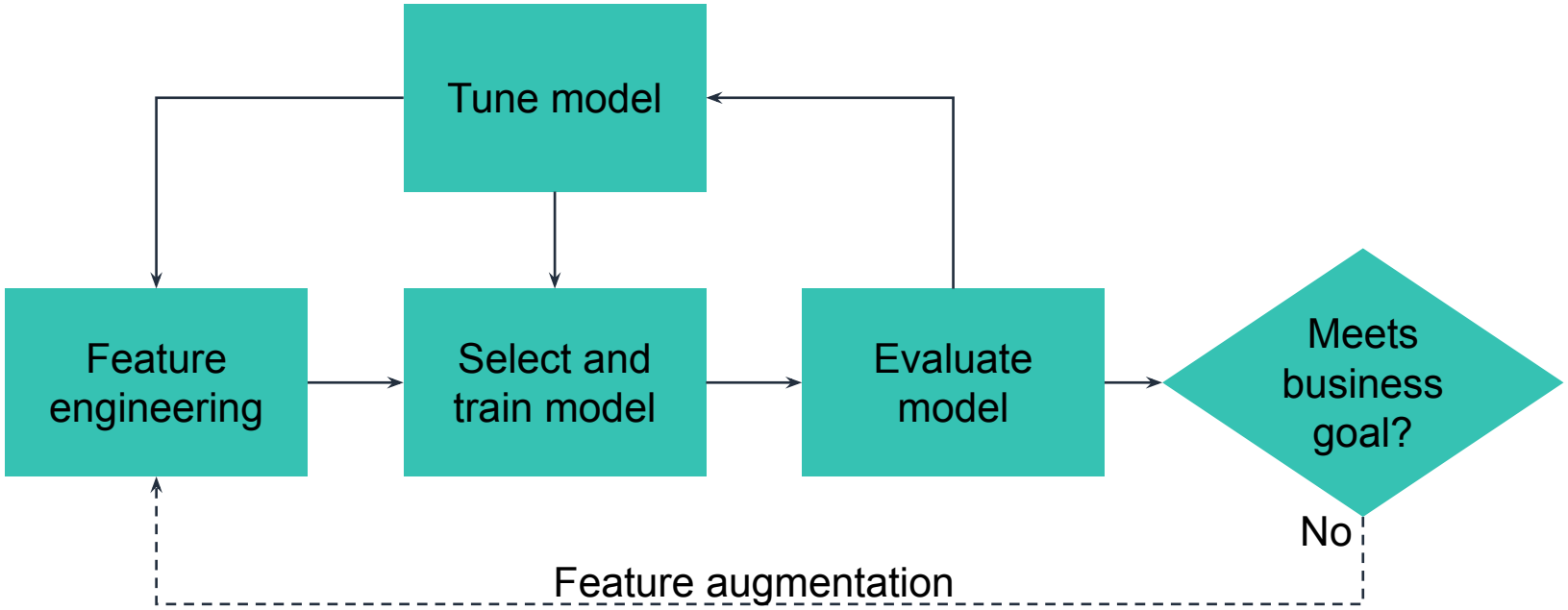
Combines precision and sensitivity

Regression: Mean squared error

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



ML tuning process



Module 3 – Guided Lab 6: Evaluating Model Accuracy

Section 7 key takeaways

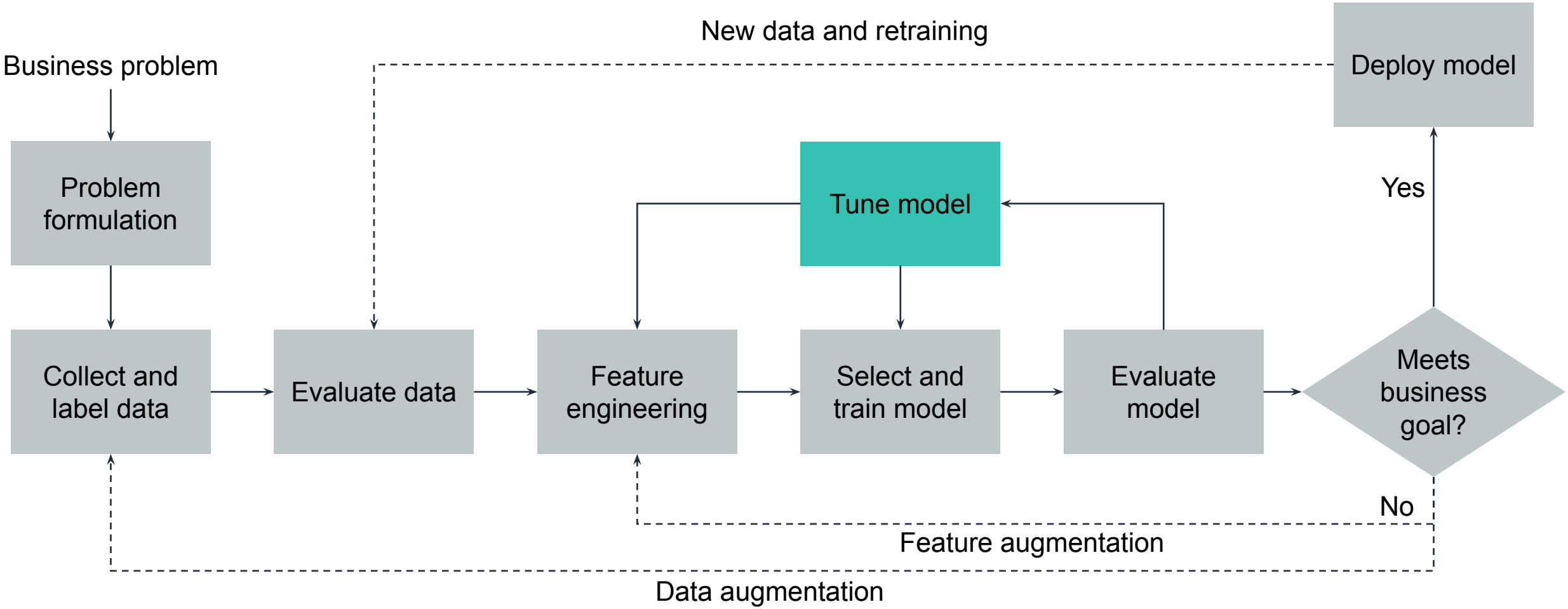
- Several ways to validate the model
  - Hold-out
  - K-fold cross validation
- Two types of model evaluation
  - Classification
    - Confusion matrix
    - AUC-ROC
  - Regression testing
    - Mean squared

Module 3: Implementing a Machine Learning Pipeline with Amazon SageMaker

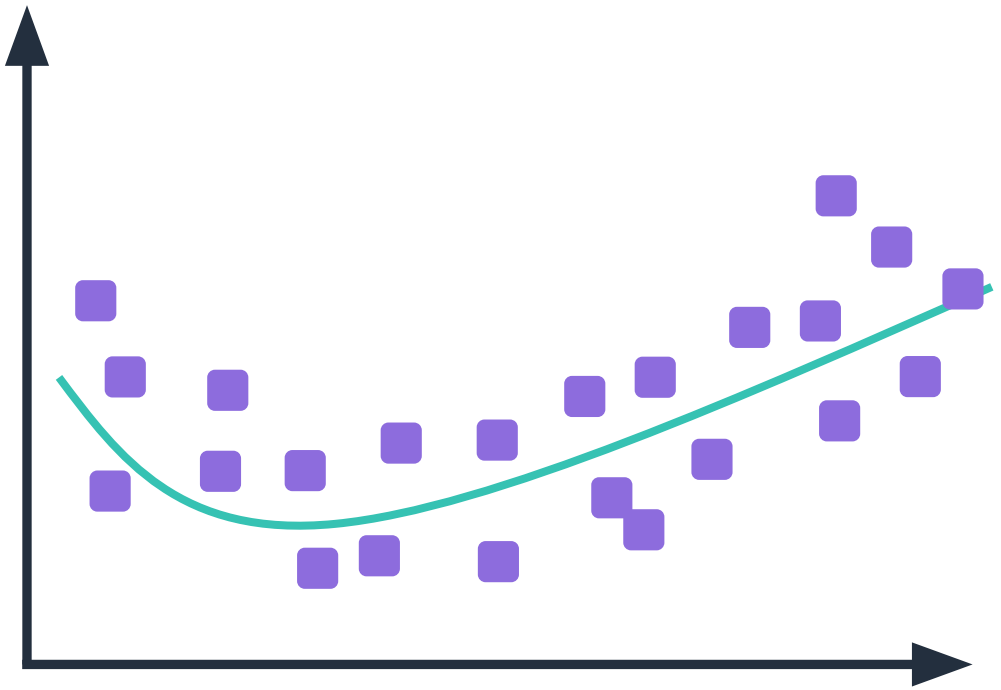
Section 8: Hyperparameter and model tuning

aws academy

Machine learning pipeline



Recap: Goal for models

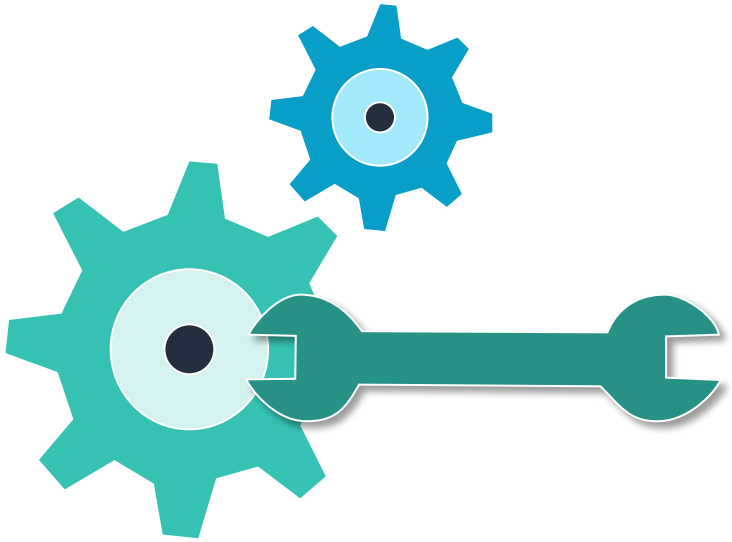


Balanced and generalizes well

Hyperparameter categories

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

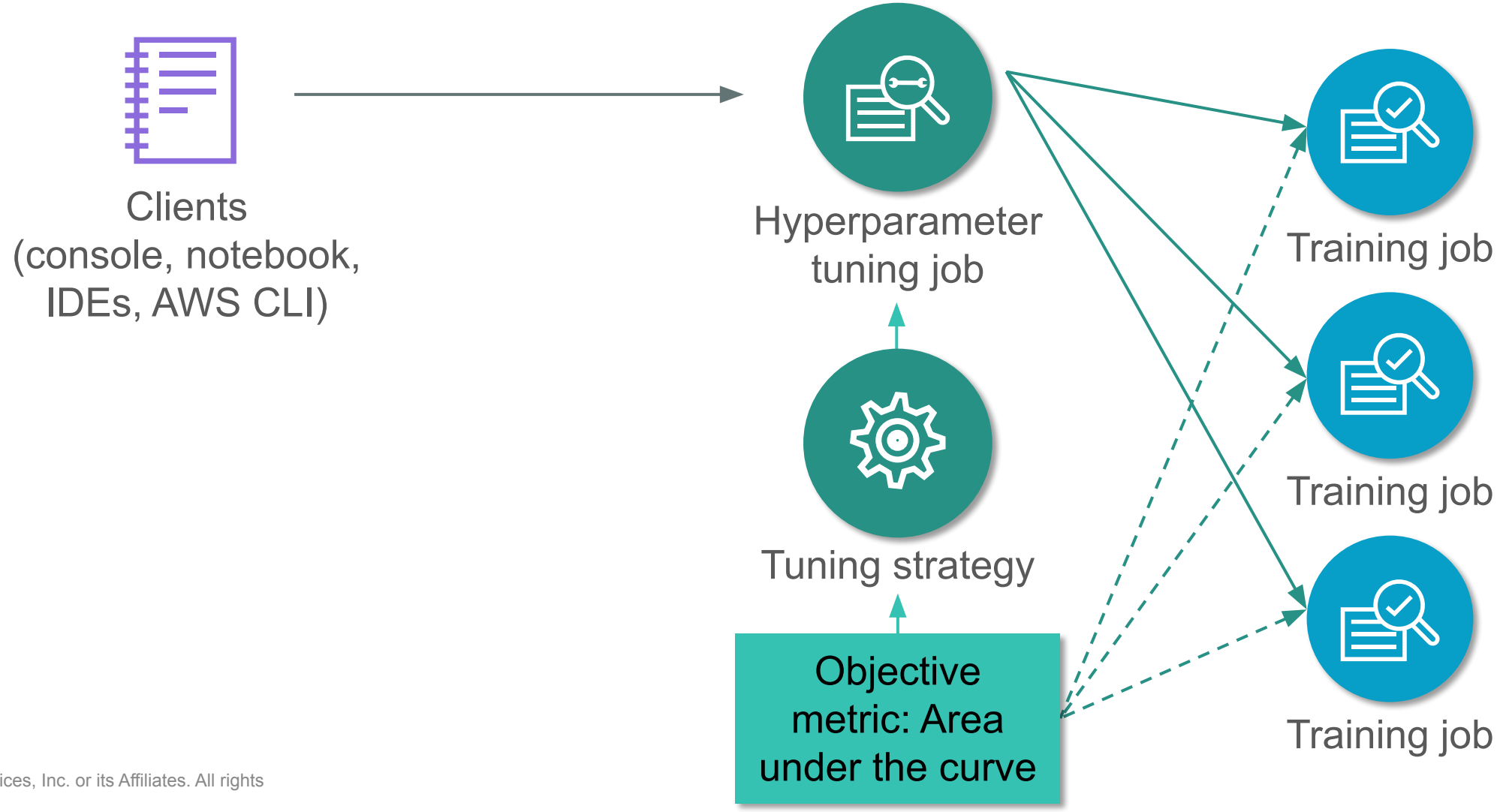
Tuning hyperparameters



Manually select hyperparameters according to one's intuition or experience

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

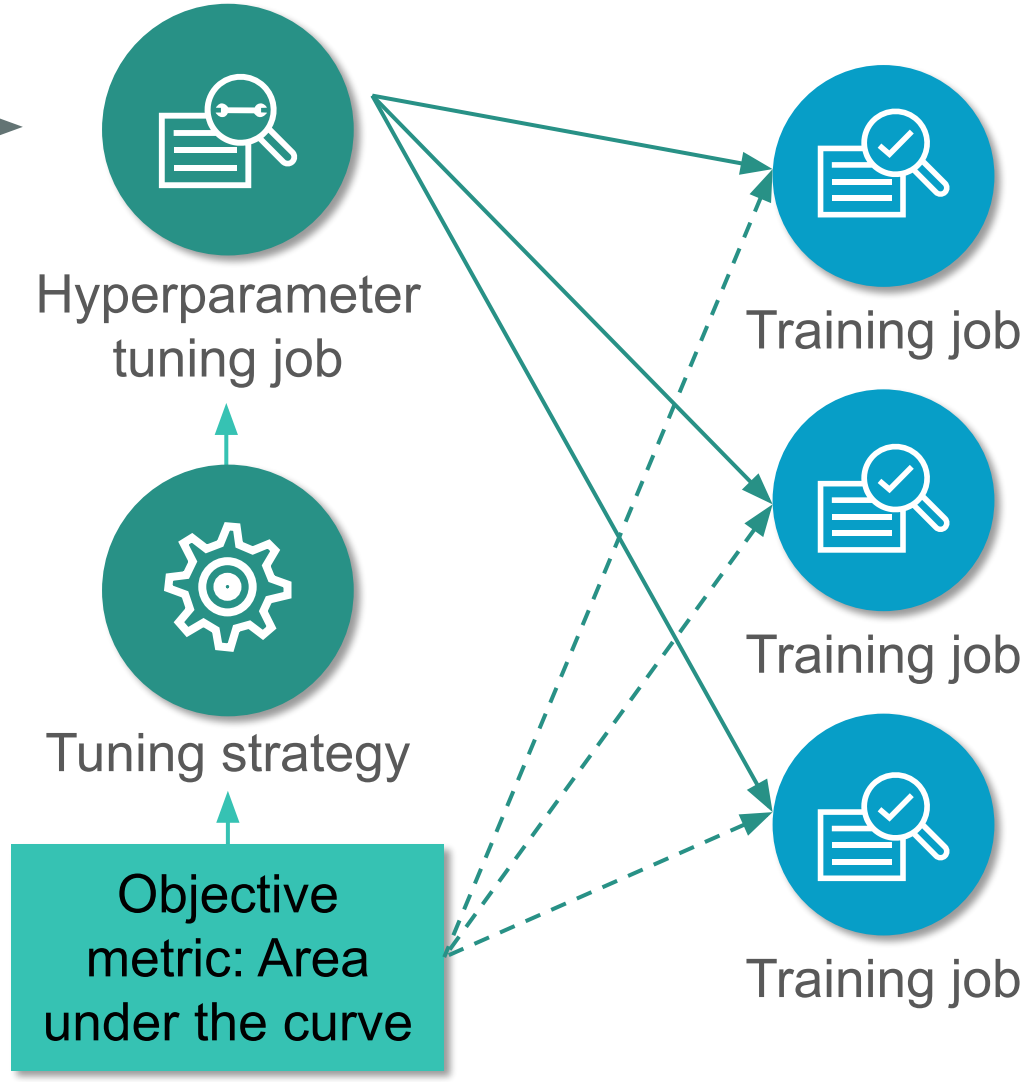
Amazon SageMaker offers automated hyperparameter tuning



Automated hyperparameter tuning

Clients
(console, notebook,
IDEs, AWS CLI)

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



Hyperparameter tuning

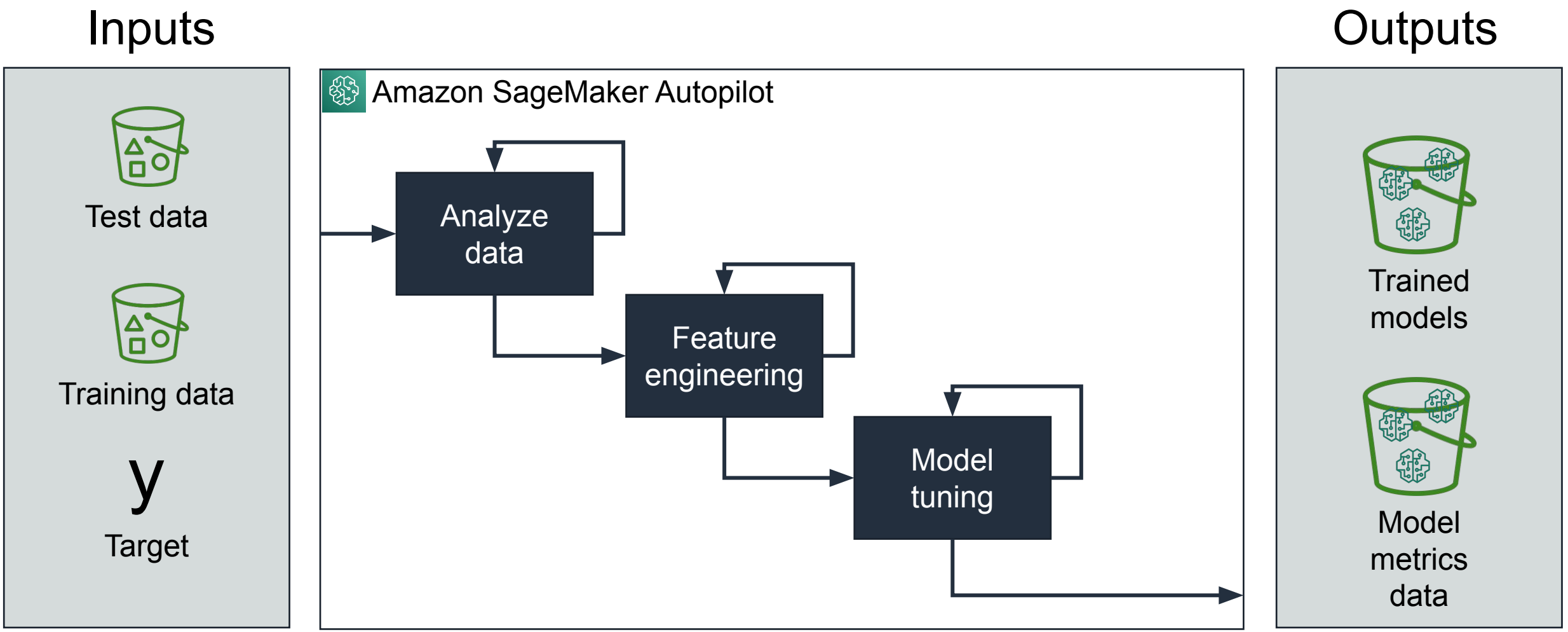
Note: Tuning doesn’t always improve your model.

Tuning best practices

- 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

Demonstration: Optimizing Amazon SageMaker Hyperparameters

Amazon SageMaker Autopilot



Demonstration: Running Amazon SageMaker Autopilot

Module 3 – Guided Lab 7: Tuning with Amazon SageMaker

Section 8 key takeaways

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

Module 3: Implementing a Machine Learning Pipeline with Amazon SageMaker

Module wrap-up

aws academy

Module summary

- In this module, you learned how to:
  - Formulate a problem from a business request
  - Obtain and secure data for machine learning (ML)
  - Build a Jupyter Notebook by using Amazon SageMaker
  - Outline the process for evaluating data
  - Explain why data must be preprocessed
  - Use open source tools to examine and preprocess data
  - Use Amazon SageMaker to train and host an ML model
  - Use cross-validation to test the performance of an ML model
  - Use a hosted model for inference
  - Create an Amazon SageMaker hyperparameter tuning job to optimize a model's effectiveness

Complete the knowledge check

Additional resources

- <Add URLs to resources that students might find helpful for further exploration on topics discussed in this module.>
- <Especially, link to whitepapers or other resources mentioned in the Exam Guide of the certification exam this course is intended to help students prepare for.>

Thank you

aws academy