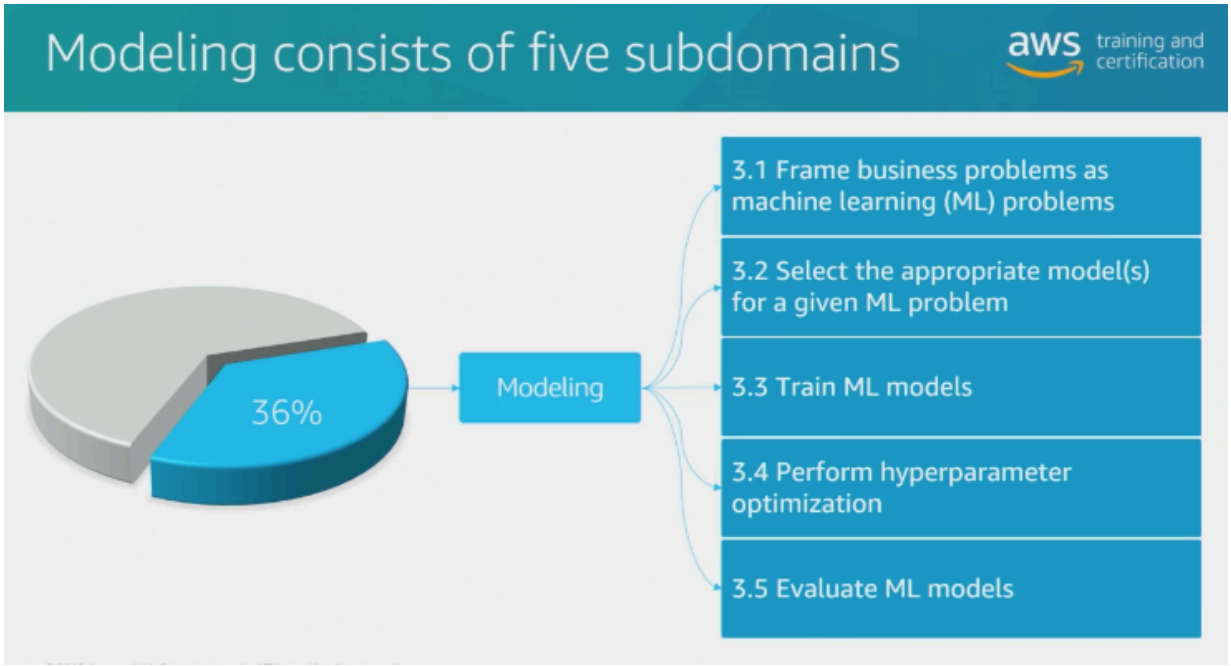
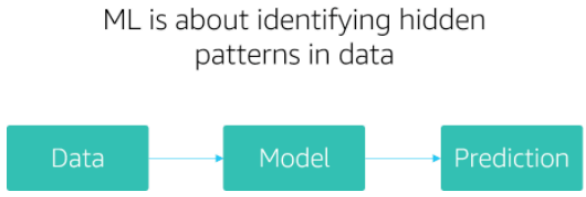
AWS Exam Readiness - Domain 3 - Modeling



Domain 3.1: Frame business problems as ML problems

Before framing your business problem, first consider whether ML is an appropriate solution to your problem

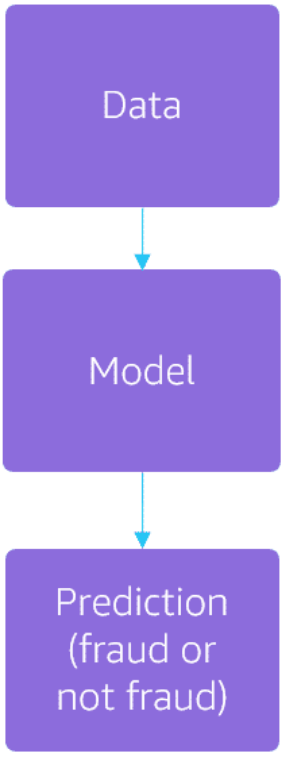


Let's start at the beginning. Is ML even an appropriate solution for your business case? First, remember that ML is about identifying hidden patterns in data. It has the potential to leverage large amounts of data to train an ML model on that data's patterns and structures to then make predictions. And the power of ML is that your model, in theory, gets progressively better at making these predictions as it's trained.

Think about a common and appropriate use case for ML, like predicting fraudulent credit card transactions. In this situation, the appropriate data would be mined to identify patterns among the card transactions, specifically looking for patterns that indicate a fraudulent transaction. With these patterns, you can train the ML model to predict future transactions as fraudulent or not fraudulent.

What about this use case makes ML an appropriate solution? Well, theoretically, you have millions of historical records in your credit card transactions data set. Given the nature of the business problem and this huge swath of data, there is no one single indicator that you can quickly use to identify fraudulent transactions. In other words, the patterns in the data are too complicated to identify on your own.

So, you have tons of data that need to be analyzed for patterns that can be used to make predictions. This is a perfect use case for ML.



Now, think of a business case that is not appropriate for ML

ML is not appropriate when you can determine a target value by using simple rules or computations that can be programmed without needing any data-driven learning.

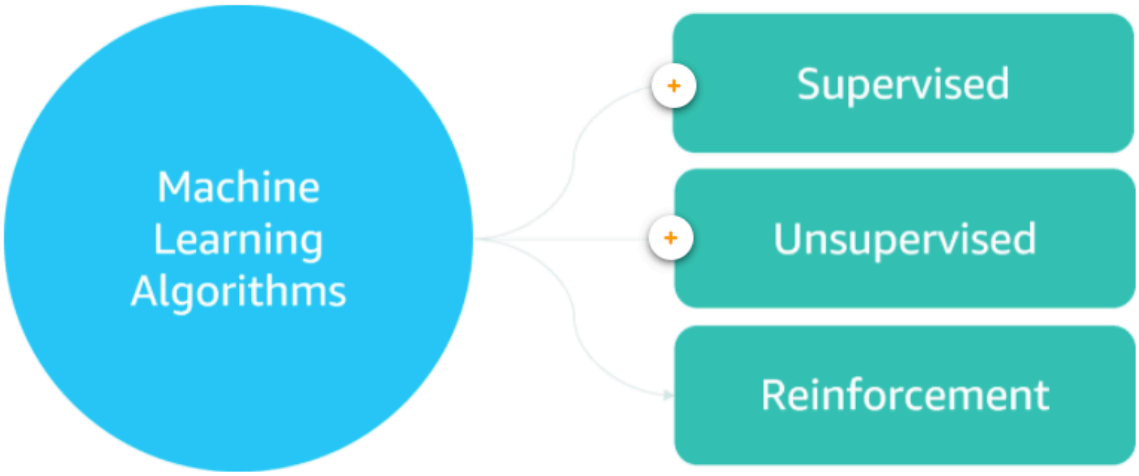
Alternatively, you should use machine learning when:

- You cannot code the rules. Many human tasks (such as recognizing whether an email is spam or not spam) cannot be adequately solved using a simple (deterministic), rule-based solution. Many factors could influence the answer. When rules depend on too many factors and many of these rules overlap or need to be tuned very finely, it soon becomes difficult for a human to accurately code the rules. You can use ML to effectively solve this problem.

- You cannot scale. You might be able to manually recognize a few hundred emails and decide whether they are spam or not. However, this task becomes tedious for millions of emails. ML solutions are effective at handling large-scale problems.

Different types of ML algorithms have their own particular use cases

Select the supervised and unsupervised marker below to learn more.



Within supervised learning, you have different types of problems. These can be broadly categorized into two categories: classification and regression.

Learn more about the different types of classification problems by expanding the sections below.

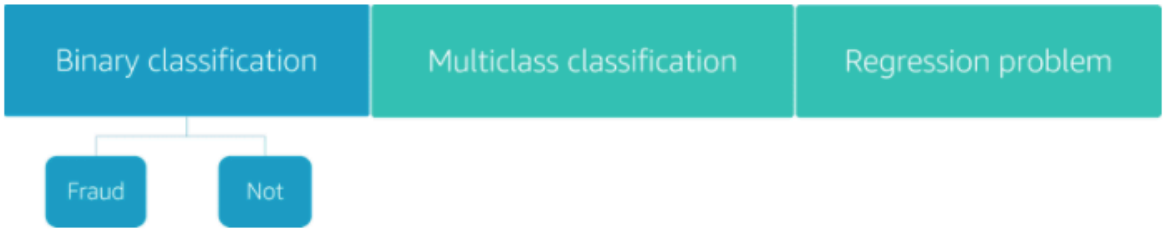
Binary classification +

Multiclass classification +

Regression problems +

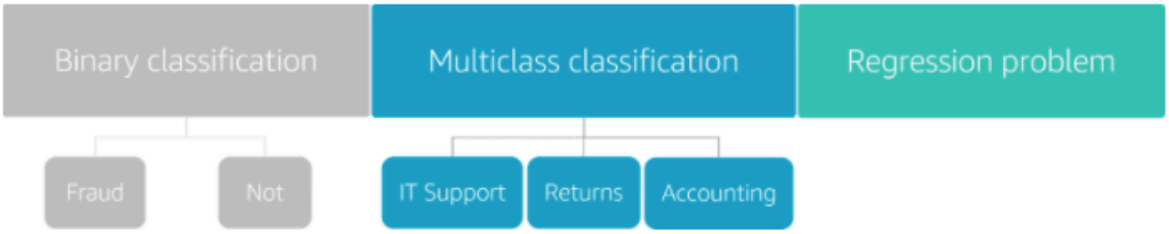
Binary classification −

Within classification problems, there are actually two types. The first is considered a binary classification problem. Think back to the example provided above about identifying fraudulent transactions. The target variable in this example is limited to two options: fraudulent or not fraudulent. This is an example of a binary classification problem: you are classifying an observation into one of two categories.

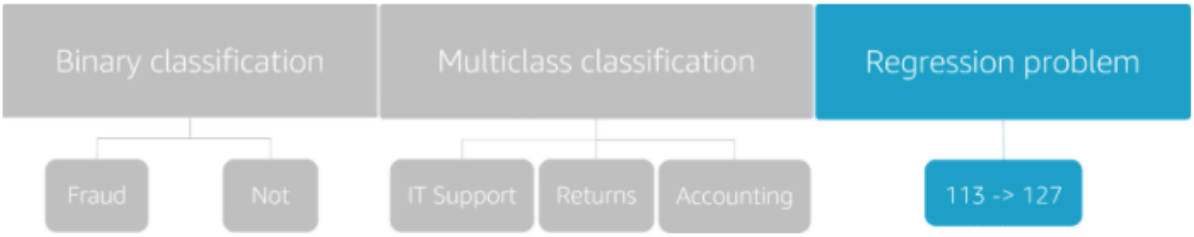


Multiclass classification

There are also multiclass classification problems. These ML problems classify an observation into one of three or more categories. Pretend you have an ML model that predicts why a customer is calling your store so that you can reduce the number of transfers needed before getting the customer to the right customer support department. The different customer support departments, in this case, represent the variety of potential target variables—which, needless to say, could be many different departments, far greater than just two.



There are also regression problems. In a regression problem, you're no longer mapping an input to a defined number of categories, but instead to a continuous value, like an integer. One example of an ML regression problem is predicting the price of a company's stock.

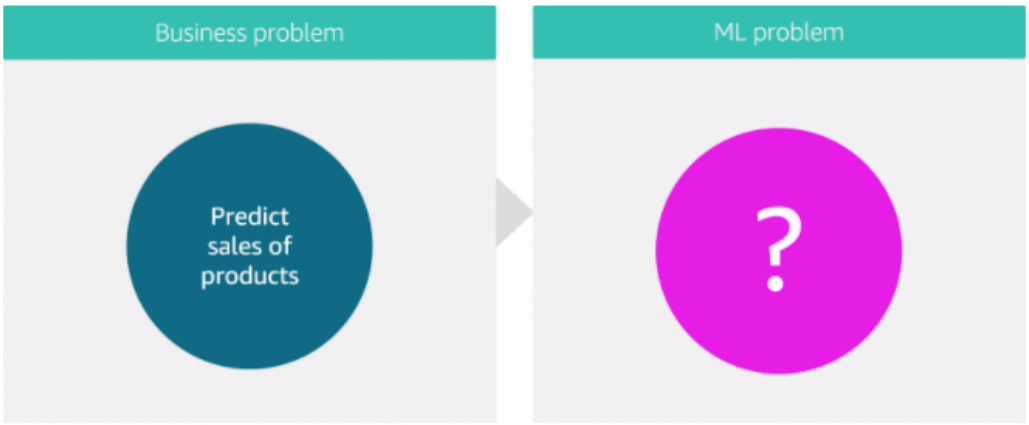


Now that you know the types of ML and when ML should be applied, reframe your business problem as an ML problem

Business problem

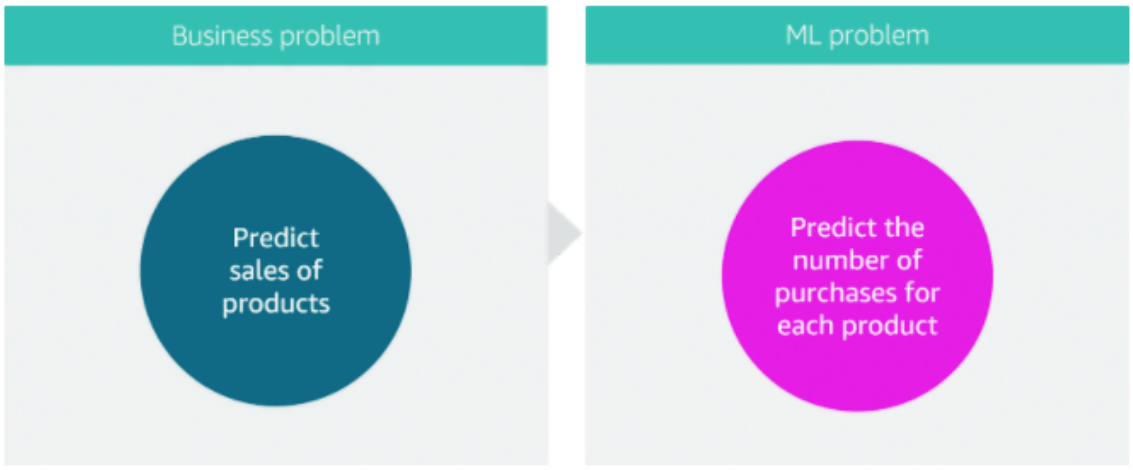
How you define the problem depends on the use case and business need

Imagine that you want to manufacture products, but your decision to manufacture each product depends on its number of potential sales. In this scenario, you want to predict how many times each product will be purchased (predict number of sales). There are multiple ways to define this problem by using ML. Choosing how to define the problem depends on your use case or business need.



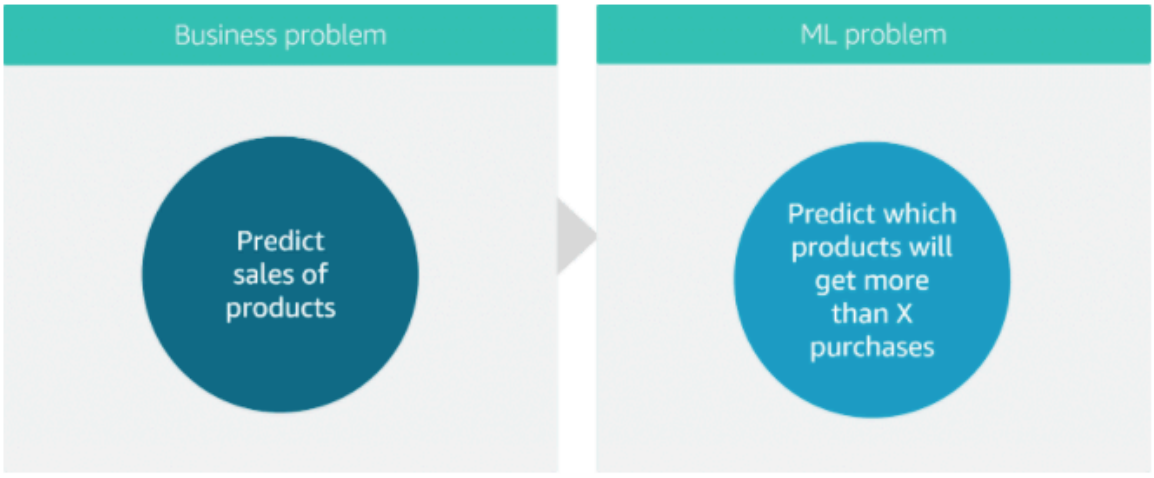
In this case, the target is numeric and it's a regression problem

Do you want to predict the number of purchases your customers will make for each product (in which case the target is numeric and you're solving a regression problem)?



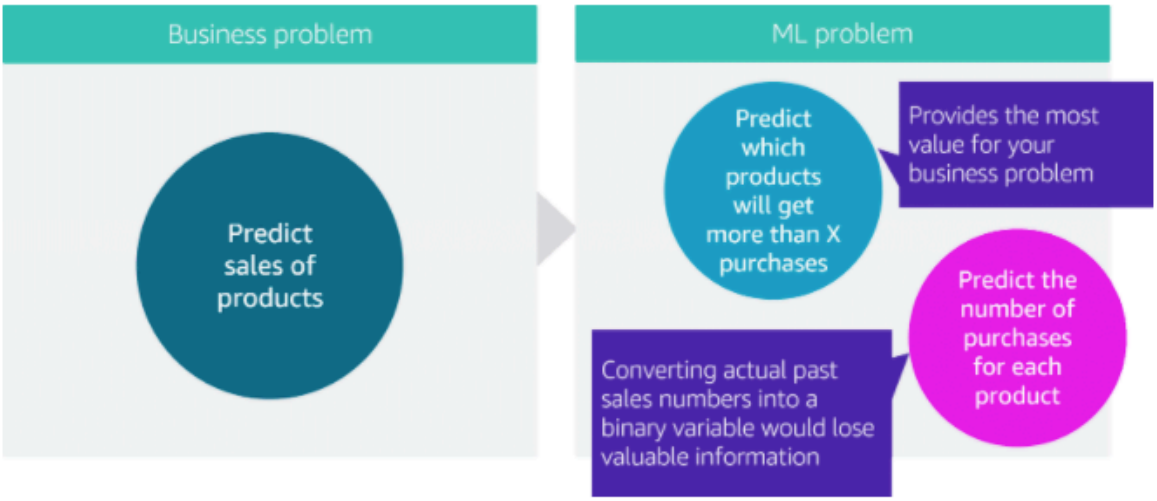
Here, the target is binary and it's a classification problem

Or do you want to predict which products will get more than 10 purchases (in which case the target is binary and you're solving a binary classification problem)?



Frame in the simplest form, but make sure you aren't losing important information

It is important to avoid overcomplicating the problem and to frame the simplest solution that meets your needs. However, it is also important to avoid losing information, especially information in the historical answers. Here, converting an actual past sales number into a binary variable like "over 10" would lose valuable information. Investing time in deciding which target makes most sense for you to predict will save you from building models that don't answer your question.



Topics related to this subdomain

Here are some topics you may want to study for more in-depth information related to this subdomain:

- [ ] Supervised learning
- [ ] Regression and classification
- [ ] Unsupervised learning
- [ ] Clustering
- [ ] Anomaly detection
- [ ] Deep learning
- [ ] Perceptron
- [ ] Components of an artificial neuron

Domain 3.2: Select the appropriate model(s) for an ML problem

Building on the last section, this subdomain focuses on mapping well-converted ML problems to appropriate algorithms. As mentioned, the format of the question you want to solve influences the algorithm that you choose. Therefore, it's critical to understand the particular use cases for a variety of algorithms, including the built-in Amazon SageMaker algorithms. Let's walk through an example.

Binary classification example

Suppose you're a bank marketing manager and you want to conduct a direct mail campaign to attract new customers. You have taken this business situation and have formulated the following ML problem: "Based on past customer responses, should I mail this particular customer?" Answers to this question fall into two categories: yes or no. In other words, you've turned this into a binary classification problem.

Multiclass classification example

Pretend now that your business problem is crafted as: "Based on past customer segmentation, which segment does this customer fall into?" Answers might fall into categories such as "empty nester," "suburban family," or "urban professional." You could use these segments to decide who should receive the mailing. As you have probably noticed, this is a multiclass classification problem.

Both of these ML problems represent classification problems. Amazon SageMaker provides a few built-in algorithms that work for these situations: Linear Learner, XGBoost, and K-Nearest Neighbors. XGBoost, for instance, is an open-source implementation of the gradient-boosted trees algorithm. Gradient boosting is a supervised learning algorithm that attempts to accurately predict a target variable by combining an ensemble of estimates from a set of simpler, weaker models.

Regression example

Now, think about other ways you could potentially frame this ML problem. What if you worded it as: "Based on the return on investment (ROI) from past mailings, what is the ROI for mailing this customer?" In this case, you use the ROI to target customers for the mail campaign.

Given how you have now framed this problem, you are now dealing with a regression problem opposed to the above classification problem.

In terms of the built-in Amazon SageMaker algorithms you could choose, it's pretty similar. Again, you could choose Linear Learner and XGBoost. The difference is that you set the hyperparameters to direct these algorithms to produce quantitative results.

Amazon SageMaker built-in algorithms for NLP and CV

Algorithms for natural language processing (NLP)

There are Amazon SageMaker built-in algorithms for natural language processing:

- BlazingText algorithm provides highly optimized implementations of the Word2vec and text classification algorithms.
- Sequence2sequence is a supervised learning algorithm where the input is a sequence of tokens (for example, text, audio) and the output generated is another sequence of tokens.
- Object2Vec generalizes the well-known Word2Vec embedding technique for words that are optimized in the Amazon SageMaker BlazingText algorithm.

Algorithms for computer vision (CV)

There are Amazon SageMaker built-in algorithms for computer vision:

- Image classification is a supervised learning algorithm used to classify images.
- Object detection algorithm detects and classifies objects in images using a single deep neural network. It is a supervised learning algorithm that takes images as input and identifies all instances of objects within the image scene. The object is categorized into one of the classes in a specified collection with a confidence score that it belongs to the class. Its location and scale in the image are indicated by a rectangular bounding box.
- Semantic segmentation algorithm tags every pixel in an image with a class label from a predefined set of classes.

Other options for training algorithms

Up to now, the focus has been exclusively on Amazon SageMaker built-in algorithms, but there are other options for training algorithms:

- Use Apache Spark with Amazon SageMaker
- Submit custom code to train a model with a deep learning framework like TensorFlow or Apache MXNet
- Use your own custom algorithm and put the code together as a Docker image
- Subscribe to an algorithm from AWS Marketplace

Topics related to this subdomain

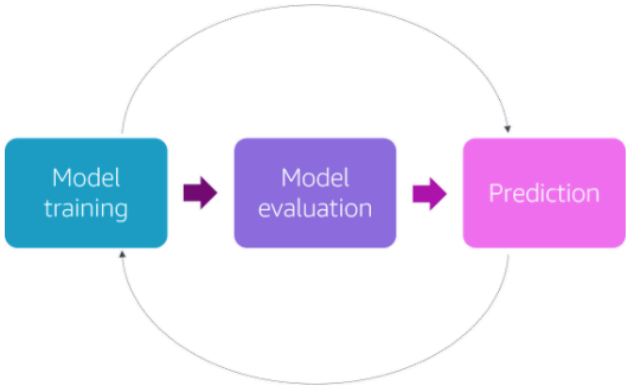
Here are some topics you may want to study for more in-depth information related to this subdomain:

- [ ] Linear learner
- [ ] XGBoost
- [ ] K-means
- [ ] Decision trees
- [ ] Random forest
- [ ] Image classification
- [ ] Object detection
- [ ] Semantic segmentation

Domain 3.3: Train ML models

Modeling training, tuning, and evaluation is an iterative process

Now that you have chosen an algorithm, you are ready to train, tune, and evaluate your ML model. This is an iterative process—these are not discrete phases in the pipeline. This subdomain focuses on how to set up and run model training jobs in Amazon SageMaker; the next two focus on tuning and evaluation.



Split data to ensure a proper division between training and evaluation

A big part of preparing for that training process is to first split your data to ensure a proper division between your training and evaluation efforts.

A common strategy is to split all available labeled data into training, validation, and testing subsets, usually with a ratio of 80%:10%:10%. (Another common ratio is 70%:15%:15%.)

Here's how you use that data in a technique called the holdout method.

Validation

Use the validation data to give you an estimate of model performance while tuning its hyperparameters and/or to compare performance across different models you may be considering.

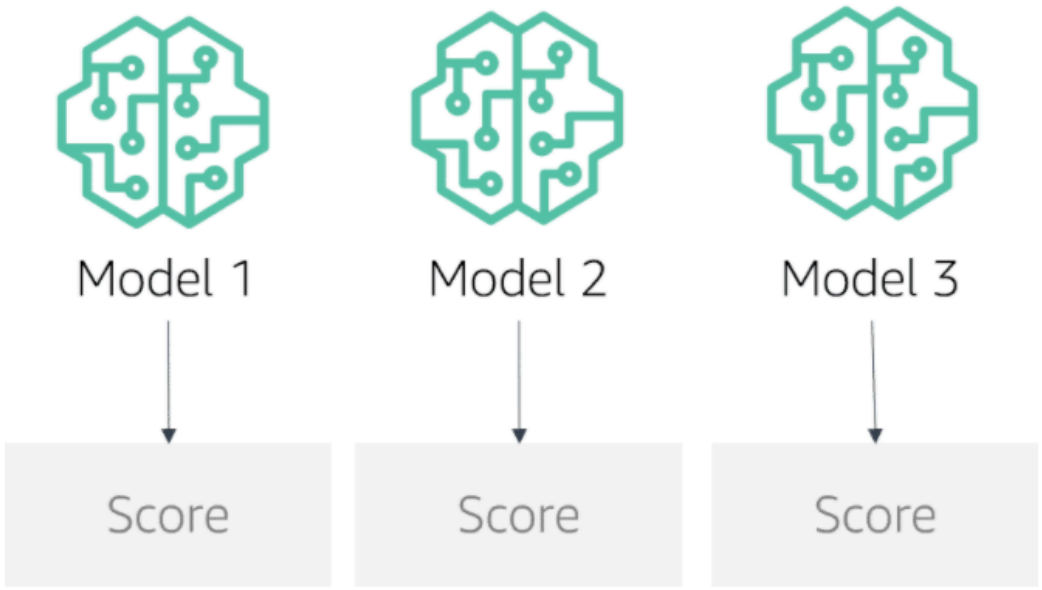
Testing (or evaluation)

Use the test (or evaluation) data to evaluate the predictive quality of the trained model you eventually decide to go with.

Other techniques for splitting data

Cross-validation

Use cross-validation methods to compare the performance of multiple models. The goal behind cross-validation is to help you choose the model that will eventually perform the best in production.



K-fold cross-validation is a common validation method

K-fold cross-validation is a common validation method. In k-fold cross-validation, you split the input data into k subsets of data (also known as folds). You train your models on all but one (k-1) of the subsets, and then evaluate them on the subset that was not used for training. This process is repeated k times, with a different subset reserved for evaluation (and excluded from training) each time.

For instance, performing a 5-fold cross-validation generates four models, four datasets to train the models, four datasets to evaluate the models, and four evaluations, one for each model. In a 5-fold cross-validation for a binary classification problem, each of the evaluations reports an area under curve (AUC) metric. You can get the overall performance measure by computing the average of the four AUC metrics.

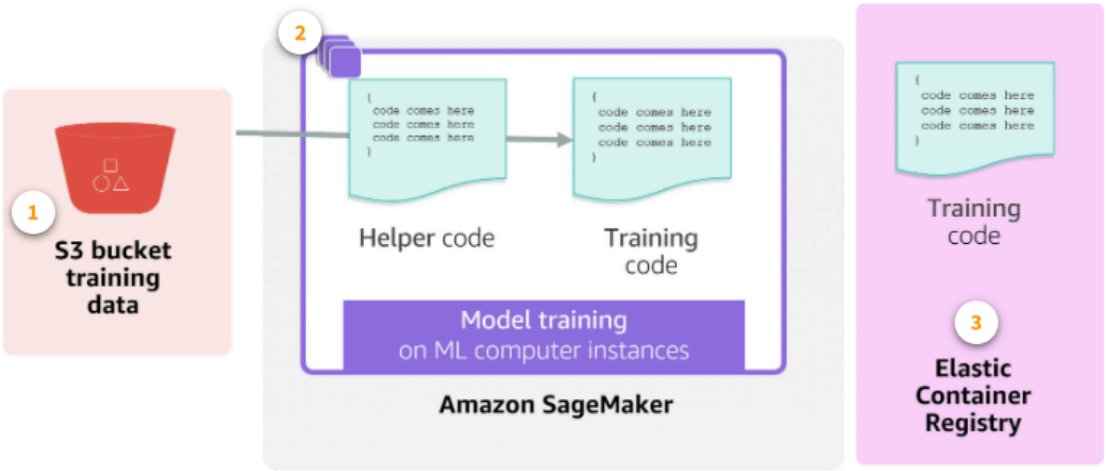
Other cross-validation methods

You can use a host of other cross-validation methods, depending on your requirements. If you have a small dataset, for instance, consider Leave-one-out cross-validation or, as mentioned above, K-fold cross-validation. Or you might use the Stratified K-fold cross-validation when you have imbalanced data. Just remember that these techniques increase the computational power needed during training.

Cross-validation techniques will increase the computational power needed for your training.

Creating a training job in Amazon SageMaker

With an algorithm chosen and your data split up, you can now run the actual training job. Creating a training job in Amazon SageMaker typically requires the following steps (click on the image below to learn more):



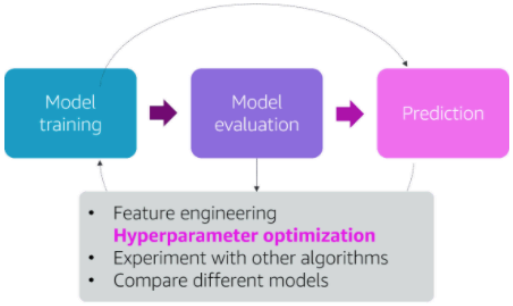
After you create the training job, Amazon SageMaker launches the compute instances and uses the training code and the training dataset to train the model. It saves the resulting model artifacts and other output in the Amazon S3 bucket you specified for that purpose.

Topics related to this subdomain

Here are some topics you may want to study for more in-depth information related to this subdomain:

- [ ] Amazon SageMaker workflow for training jobs
- [ ] Running a training job using containers
- [ ] Build your own containers
- [ ] P3 instances
- [ ] Components of an ML training job for deep learning

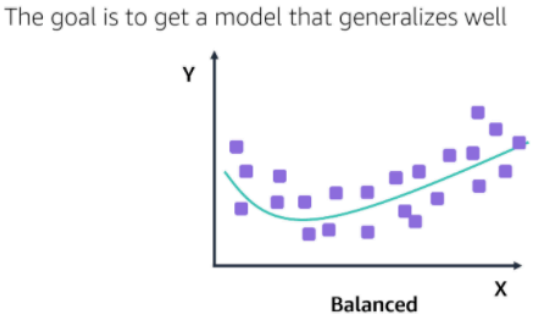
Domain 3.4: Perform hyperparameter optimization



Model tuning involves optimizing hyperparameters

It is rare that your model will perform at the level you need for production the first time around. To find the right solution for your business problem, often you have to tune your model after training by performing additional feature engineering, experimenting with new algorithms, or making other changes. You will probably need to train and evaluate multiple models that include different data setup and algorithms. This model tuning also involves tweaking your model's hyperparameters.

This subdomain is about the process of model tuning—specifically how and why you go about performing hyperparameter optimization.



What are hyperparameters?

Hyperparameters are the knobs or settings that can be tuned before running a training job to control the behavior of an ML algorithm. They can have a big impact on model training as it relates to training time, model convergence, and model accuracy. Unlike model parameters that are derived from the training job, the values of hyperparameters do not change during the training.

There are different categories of hyperparameters

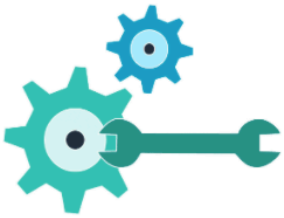
Learn more by clicking below.

Model hyperparameters define the model itself—Attributes of a neural network architecture like filter size, pooling, stride, padding

Optimizer hyperparameters, are related to how the model learn the patterns based on data and are used for a neural network model. These types of

Data hyperparameters are related to the attributes of the data, often used when you don't have enough data or enough variation in data—Data augmentation techniques

Tuning hyperparameters can be very labor-intensive

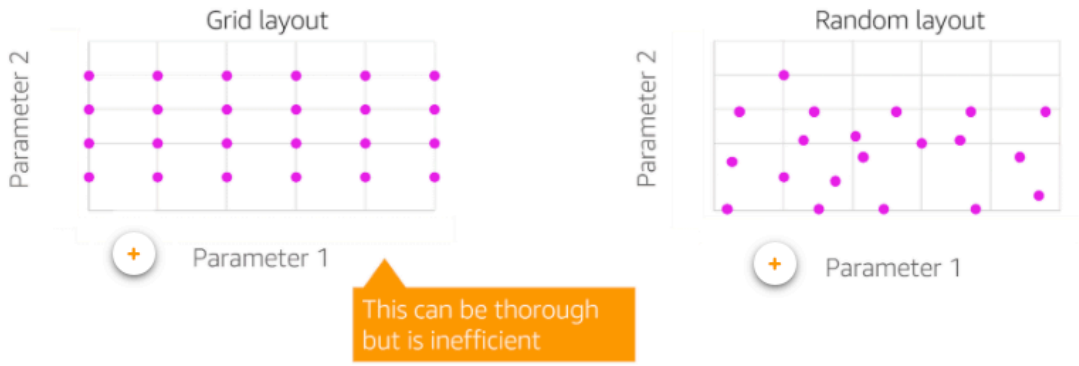


Manually select hyperparameters based on one's intuition/experience

Traditionally, this was done manually: someone who has domain experience related to that hyperparameter and the use case would manually select the hyperparameters based on their intuition and experience. Then they would train the model and score it on the validation data. This process would be repeated over and over again until satisfactory results are achieved.

Needless to say, this is not always the most thorough and efficient way of tuning your hyperparameters. As a result, several other methods for hyperparameter tuning have been developed.

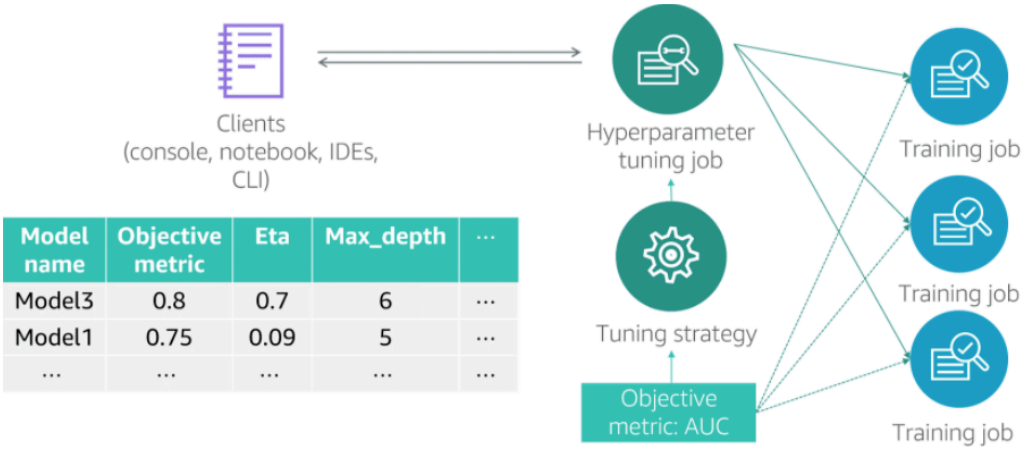
A better way is to use search methods to tune hyperparameters



Alternatively, Amazon SageMaker offers automated hyperparameter tuning

Then there's automated hyperparameter tuning, which uses methods like gradient descent, Bayesian optimization, and evolutionary algorithms to conduct a guided search for the best hyperparameter settings.

Amazon SageMaker lets you perform automated hyperparameter tuning. Amazon SageMaker automatic model tuning, also known as hyperparameter tuning, finds the best version of a model by running many training jobs on your dataset using 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.



Model name	Objective metric	Eta	Max_depth	···
Model3	0.8	0.7	6	···
Model1	0.75	0.09	5	···
···	···	···	···	···

Topics related to this subdomain

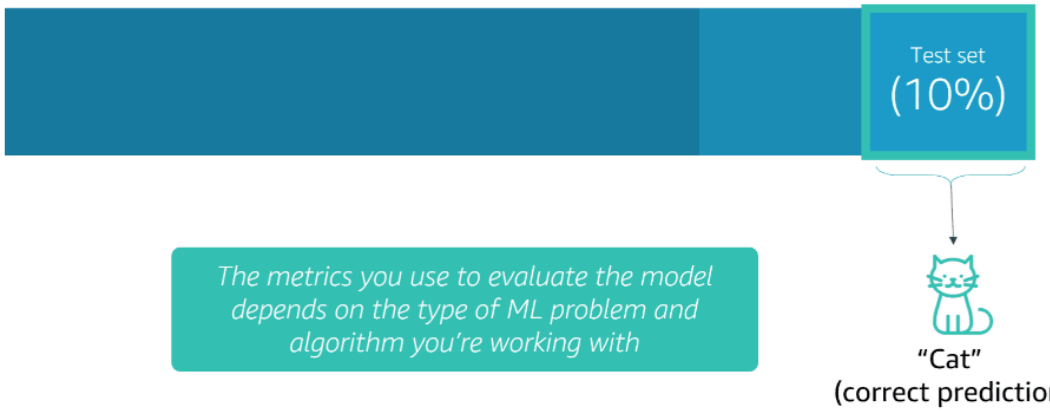
Here are some topics you may want to study for more in-depth information related to this subdomain:

- [ ] Amazon SageMaker hyperparameter tuning jobs
- [ ] Common hyperparameters to tune:
    - Momentum
    - Optimizers
    - Activation functions
    - Dropout
    - Learning rate
- [ ] Regularization:
    - Dropout
    - L1/L2

Domain 3.5: Evaluate ML models

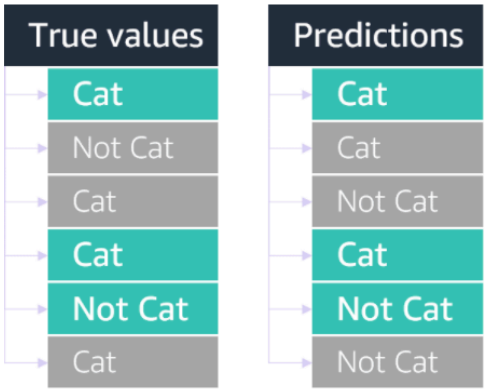
Once you have trained and tuned your models and decided which model is the best for your business problem, it's time to evaluate that model to determine if it will do a good job predicting the target on new and future data. Because future instances have unknown target values, you need to check the accuracy metric of the model on data for which you already know the target answer, and use this assessment as a proxy for predictive accuracy on future data. This is why you hold out a sample of your data for evaluation or testing purposes. The process and metrics by which you evaluate your model is what this subdomain is all about.

Run the test set through the trained model and compare predictions to true values



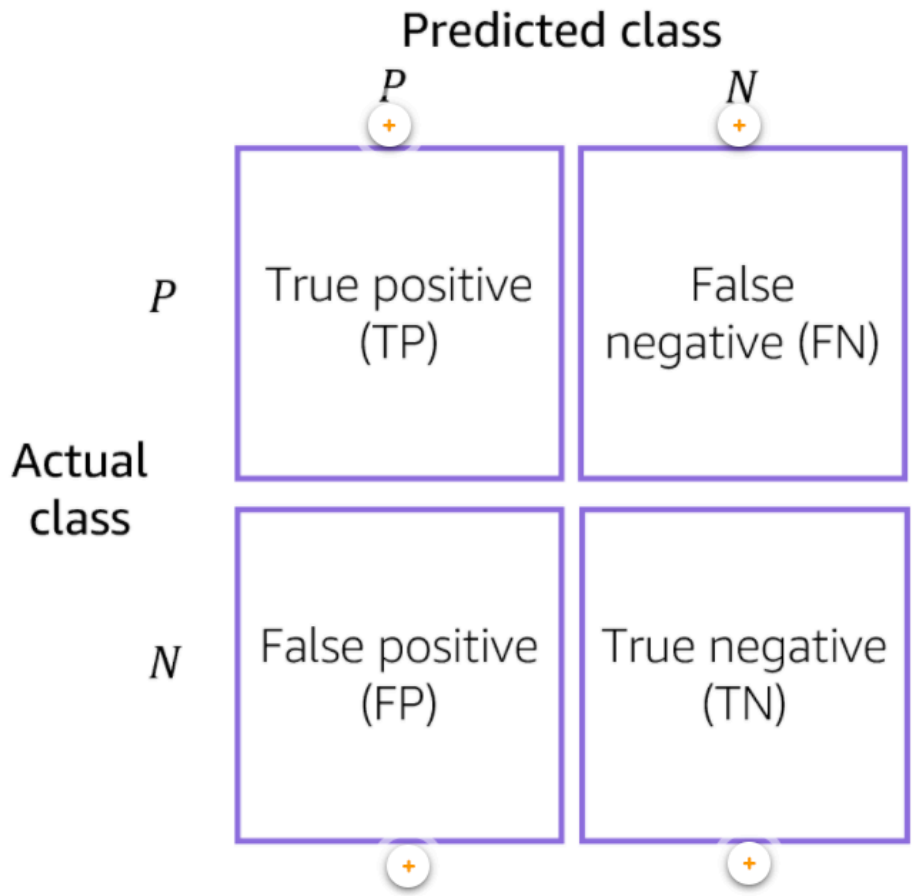
For classification problems, a confusion matrix is the building block for your model evaluation.

For instance, with binary classification problems, two different classes or target predictions are associated with each instance of your training data. Pretend this is a simple image recognition model that is labeling data as either "cat" or "not a cat." These classes represent the actual predictions the model should be making. After running the model on your hold-out test data, you now have another set of classes that represent the predicted outcomes.

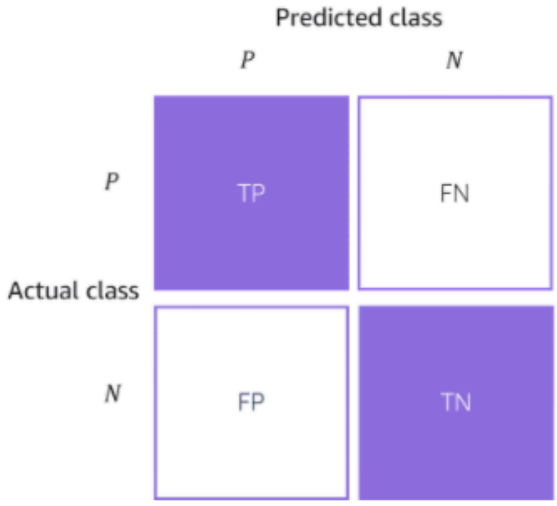


Explaining a confusion matrix

Click on the various points of the matrix below to learn more.



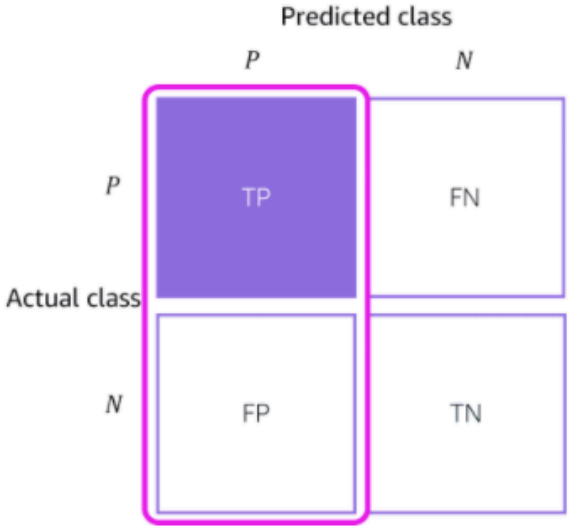
Accuracy is the ratio of correct predictions to total number of predictions



$$Accuracy = \frac{TP+TN}{TP+FP+FN+FP}$$

Accuracy is less effective when there are lots of true negatives (say, predicting fraud with little or no fraud data)

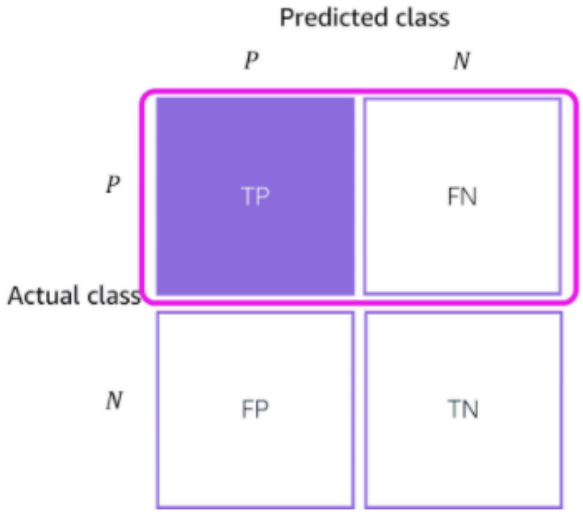
Precision is the proportion of positive predictions that are actually correct



$$Precision = \frac{TP}{TP+FP}$$

Precision is a good metric when the cost of false positives are high

Recall is the proportion of correct sets that are identified as positive



$$Recall = \frac{TP}{TP+FN}$$

Recall is a good metric when the cost of false negatives are high

Additional metrics include:

- F1 score = $\frac{2 \cdot Precision \cdot Recall}{Precision+Recall}$
- Area under the curve: Receiver operator curve (AUC-ROC)

Topics related to this subdomain

Here are some topics you may want to study for more in-depth information related to this subdomain:

- [ ] Metrics for regression: sum of squared errors, RMSE
- [ ] Sensitivity
- [ ] Specificity
- [ ] Neural network functions like Softmax for the last layer

Walk-through of sample questions

In this section, you'll have a chance to answer and walk through the solution to two different sample questions. The questions reflect some of the design and technical content you may see on the exam. The videos below will give you the answers to each question by walking you through the test-taking strategies presented to you earlier in this course.

Question 1

Answer the question below before watching the corresponding solution video.

An oil and natural gas company is using machine learning to discover prime locations for drilling. The company has chosen Amazon SageMaker as its service for creating machine learning models. The company's data scientists are using notebook instances to develop those models. However, the data scientists spend a long time waiting for the training jobs to complete.

The company wants improve this idle time to more effectively iterate on the models with minimal change to the code to enable data scientists to quickly experiment with their models without having to wait for the training job to load the data and train the model.

What should the team of data scientists do to solve this issue?

- ✗ Use Amazon SageMaker in-built algorithms.
- ✓ Use Amazon SageMaker Estimators in local mode to train the models.
- ✗ Change the training job to use Pipe Mode to improve the time it takes to train the model.
- ✗ Create the models on local laptops. Then, port the code over to use Amazon SageMaker.

A oil and natural gas company is using machine learning to discover prime locations for drilling. The company has chosen Amazon SageMaker as its service for creating machine learning models. The company's data scientists are using notebook instances to develop those models. However, the data scientists spend a long time waiting for the training jobs to complete.

The company wants improve this idle time to more effectively iterate on the models with minimal change to the code to enable data scientists to quickly experiment with their models without having to wait for the training job to load the data and train the model.

What should the team of data scientists do to solve this issue?

A. Use Amazon SageMaker in-built algorithms

Question 2

Answer the question below before watching the corresponding solution video.

A data scientist trained an XGBoost model to classify internal documents for further inquiry, and now wants to evaluate the model's performance by looking at the results visually.

What technique should the data scientist use in this situation?

- ✗ Scatterplot to visualize the predicted labels versus the true label
- ✗ Correlation matrix to visualize the predicted labels versus the true label
- ✓ Confusion matrix to visualize the predicted labels
- ✗ Box plot to visualize the predicted labels (X axis) versus the true labels (Y axis)

Domain quiz

Take this short quiz to test your understanding of some of the topics related to this domain and experience the types of questions that will be on the exam. We recommend you use the test-taking strategies presented earlier in this course when completing the questions. Click the link below to begin.

aws training and certification

A real estate company wants to provide its customers with a more accurate prediction of the final sale price for houses they are considering in various cities. To do this, the company wants to use a fully connected neural network trained on data from the previous ten years of home sales, as well as other features.

What kind of machine learning problem does this situation represent?

- (●) Regression
- ( ) Recommender system
- ( ) Reinforcement learning
- ( ) Classification

A manufacturing company wants to increase the longevity of its factory machines by predicting when a machine part is about to stop working, jeopardizing the health of the machine. The company's team of Data Scientists will build an ML model to accomplish this goal. The model will be trained on data made up of consumption metrics from similar factory machines, and will span a time frame from one hour before a machine part broke down to five minutes after the part degraded.

What kind of machine learning algorithm should the company use to build this model?

- ○ Scikit Learn Random Forest
- ◉ Amazon SageMaker DeepAR
- ○ SciKit Learn Regression
- ○ Convolutional neural network (CNN)

A Data Scientist working for an autonomous vehicle company is building an ML model to detect and label people and various objects (for instance, cars and traffic signs) that may be encountered on a street. The Data Scientist has a dataset made up of labeled images, which will be used to train their machine learning model.

What kind of ML algorithm should be used?

- (●) Image classification
- ( ) Semantic segmentation
- ( ) Instance segmentation
- ( ) Image localization

=> Incorrect

Instance segmentation

A Data Scientist is training a convolutional neural network model to detect incoming employees at the company's front gate using a camera so that the system opens for them automatically. However, the model is taking too long to converge and the error oscillates for more than 10 epochs.

What should the Data Scientist do to improve upon this situation? (Select TWO.)

- [ ] Add more epochs
- [ ] Decrease weight decay
- [ ] Normalize the images before training
- [x] Increase batch size
- [x] Add batch normalization

=> partially incorrect

Normalize the images before training

✗ **Increase batch size**

✓ **Add batch normalization**

A Data Scientist at a waste recycling company trained a CNN model to classify waste at the company's sites. Incoming waste was classified as either trash, compost, or recyclable to make it easier for the machines to split the incoming waste into the appropriate bins.

During model testing, the F1 score was 0.918. The company's senior leadership originally asked the Data Scientist to reach an F1 score of at least 0.95.

What should the Data Scientist do to improve this score without spending too much time optimizing the model?

- ○ Run the Amazon SageMaker training job for more epochs
- ◉ Use Amazon SageMaker tuning jobs to tune the hyperparameters used
- ○ Increase the batch size to improve the score in the Amazon SageMaker training job
- ○ Use momentum to improve the training in the Amazon SageMaker training job

4/6	66.7%