**Data wrangling** is the process of cleaning and transforming data to make it more appropriate for data analysis. The process generally follows these main steps:

* Explore the raw data and check the general quality of the dataset.
* Transform the raw data, by restructuring, normalizing, and cleaning the data. For example, this could involve handling missing values and detecting errors.
* Validate and publish the data.

Data wrangling is an *iterative* process where you do some data transformation then check the results and come back to the process to make improvements.

**Datastores** offer a layer of abstraction over the supported Azure storage services. They store all the information needed to connect to a particular storage service. Datastores provide an access mechanism that is independent of the computer resource that is used to drive a machine learning process.

**Datasets** are resources for exploring, transforming, and managing data in Azure ML. A dataset is essentially a reference that points to the data in storage. It is used to get specific data files in the datastores.

**The steps of the data access workflow are:**

1. **Create a datastore** so that you can access storage services in Azure.
2. **Create a dataset**, which you will subsequently use for model training in your machine learning experiment.
3. **Create a dataset monitor** to detect issues in the data, such as data drift.

In the video, we mentioned the concept of *data drift*. Over time, the input data that you are feeding into your model is likely to change—and this is what we mean by **data drift**. Data drift can be problematic for model accuracy. Since you trained the model on a certain set of data, it can become increasingly inaccurate and the data changes more and more over time. For example, if you train a model to detect spam in email, it may become less accurate as new types of spam arise that are different from the spam on which the model was trained.

As we noted in the video, you can set up dataset monitors to detect data drift and other issues in your data. When data drift is detected, you can have the system automatically update the input dataset so that you can retrain the model and maintain its accuracy. **Key points to remember about datasets:**

* They are used to interact with your data in the datastore and to package data into consumable objects.
* They can be created from local files, public URLs, Azure Open Datasets, and files uploaded to the datastores.
* They are not copies of the data but *references* that point to the original data. This means that no extra storage cost is incurred when you create a new dataset.
* Once a dataset is registered in Azure ML workspace, you can share it and reuse it across various other experiments without data ingestion complexities.

**In summary, here are some of the main things that datasets allow you to do**:

* Have a single copy of some data in your storage, but reference it multiple times—so that you don't need to create multiple copies each time you need that data available.
* Access data during model training without specifying connection strings or data paths.
* More easily share data and collaborate with other users.
* Bookmark the state of your data by using dataset versionin
* In many cases, the set of initial features in the data is not enough to produce high quality trained machine learning models. You can use **feature engineering** to derive new features based on the values of existing features. This process can be as simple as applying a mathematical function to a feature (such as adding 1 to all values in an existing feature ) or it can be as complex as training a separate machine learning model to create values for new features.
* Once you have the features, another important task is selecting the features that are most important or most relevant. This process is called **feature selection**.
* Many machine learning algorithms cannot accommodate a large number of features, so it is often necessary to do **dimensionality reduction** to decrease the number of features.

Feature engineering: ------------

Can be done at database/training

Classical ML more reliant in feature engineering

Which of the following are true statements about feature engineering?

(Select all that apply.)

* Feature engineering manipulates existing features in order to create new features, with the goal of improving model training.
* Feature engineering can be implemented in multiple places, such as at the data source or during model training.
* 

Deep learning depends on feature engineering much more than classical machine learning.

* Classical machine learning depends on feature engineering much more than deep learning.

Feature- extraction examples:

**EXAMPLE**

**TYPE OF FEATURE ENGINEERING**

Deriving a boolean (0/1 or True/False) value for each entity

Flagging

Getting a count, sum, average, mean, or median from a group of entities

Aggregation

Extracting the month from a date variable

Part-of

Grouping customers by age and then calculating average purchases within each group

Binning

SUBMIT

**Feature Selection**

There are mainly two reasons for feature selection. Some features might be highly irrelevant or redundant. So it's better to remove these features to simplify the situation and improve performance. Additionally, it may seem like engineering more features is always a good thing, but as we mentioned earlier, many machine learning algorithms suffer from the *curse of dimensionality*—that is, they do not perform well when given a large number of variables or features.

We can improve the situation of having too many features through **dimensionality reduction**.

Commonly used techniques are:

* PCA (Principal Component Analysis)
* t-SNE (t-Distributed Stochastic Neighboring Entities)
* Feature embedding
* A linear dimensionality reduction technique based mostly on exact mathematical calculations.
* PCA (Principal Component Analysis)
* Encodes a larger number of features into a smaller number of "super-features."
* Feature embedding
* A dimensionality reduction technique based on a probabilistic approach; useful for the visualization of multidimensional data.
* t-SNE (t-Distributed Stochastic Neighboring Entities)
* SUBMIT
* NEXT

# Data Drift

As we mentioned earlier, **data drift** is change in the input data for a model. Over time, data drift causes degradation in the model's performance, as the input data drifts farther and farther from the**EXAMPLE**

**CAUSE OF DATA DRIFT**

A change in customer behavior over time.

Natural drift in the data

A sensor breaks and starts providing inaccurate readings.

Data quality issues

Two features that used to be correlated are no longer correlated.

Covariate shift / Change in relationship between features

A sensor is replaced, causing the units of measurement to change (e.g., from minutes to seconds).

Upstream process changes

SUBMIT

data on which the model was trained.

Remember, the process of monitoring for data drift involves:

* Specifying a **baseline dataset** – usually the training dataset
* Specifying a **target dataset** – usually the input data for the model
* Comparing these two datasets over time, to monitor for differences

Here are a couple different types of comparisons you might want to make when monitoring for data drift:

* **Comparing input data vs. training data.** This is a proxy for model accuracy; that is, an increased difference between the input vs. training data is likely to result in a decrease in model accuracy.
* **Comparing different samples of time series data.** In this case, you are checking for a difference between one time period and another. For example, a model trained on data collected during one season may perform differently when given data from another time of year. Detecting this seasonal drift in the data will alert you to potential issues with your model's accuracy.
* **DESCRIPTION**
* **TERM**
* Data used to tune the values of the *hyperparameters*.
* Validation data
* Variables whose value is not learned during training, but rather set as a "best guess" and then tuned.
* Hyperparameters
* Data used to learn the values of the *parameters*.
* Training data
* Variables whose value is learned during training.
* Parameters
* Data used to check the performance of the final, fully trained model.
* Test Data
* SUBMIT
* NEXT

Collect data

Prepare

Train

Evaluate

*In a****classification****problem, the outputs are categorical or discrete.*Deploy

Classify an image as one (and only one) of five possible fruits.

Multi-class single-label classification

Classify medical test results as "positive" or "negative".

Binary classification

Classify music as belonging to multiple groups (e.g., "upbeat", "jazzy", "pop").

Multi-class multi-label classification

SUBMIT

NEXT

Logistic Regression

SVM

*In a****regression****problem, the output is numerical or continuous.*

Types of regression:

Regression to values between 0 and 1

Regression to arbiitray values

Algos: Linear regression

Decision vectors

# Model Performance

SEND FEEDBACK

# Evaluating Model Performance

It is not enough to simply train a model on some data and then assume that the model will subsequently perform well on future data. Instead, as we've mentioned previously, we need to split off a portion of our labeled data and reserve it for evaluating our model's final performance. We refer to this as the test dataset.

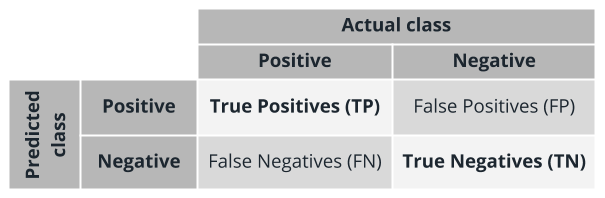
*The****test dataset****is a portion of labeled data that is split off and reserved for model evaluation.*

If a model learns to perform well with the training data, but performs poorly with the test data, then there may be a problem that we will need to address before putting our model out into the real world. In practice, we will also need to decide what metrics we will use to evaluate performance, and whether there are any particular thresholds that the model needs to meet on these metrics in order for us to decide that it is "good enough."

When splitting the available data, it is important to preserve the statistical properties of that data. This means that the data in the training, validation, and test datasets need to have similar statistical properties as the original data to prevent bias in the trained model.

This type of table is called a **confusion matrix**. A confusion matrix gets its name from the fact that it is easy to see whether the model is getting *confused* and misclassifying the data.

You will often see the confusion matrix represented in a more general, abstract form that uses the terms *positive* and *negative*:



* **True positives** are the *positive* cases that are *correctly* predicted as *positive* by the model
* **False positives** are the *negative* cases that are *incorrectly* predicted as *positive* by the model
* **True negatives** are the *negative* cases that are *correctly* predicted as *negative* by the model
* **False negatives** are the *positive* cases that are *incorrectly* predicted as *negative* by the model

# Evaluation Metrics for Classification

### QUESTION 1 OF 2

As we just saw, the confusion matrix gives us several different metrics we can use to measure the performance of our model. See if you can remember which formula is used to calculate each metric.

### ***Submit to check your answer choices!***

**FORMULA**

**METRIC**

\frac{TP + TN}{TP+FP+FN+TN}*TP*+*FP*+*FN*+*TNTP*+*TN*​

Accuracy

\frac{TP}{TP+FP}*TP*+*FPTP*​

Precision

\frac{TP}{TP+FN}*TP*+*FNTP*​

Recall

2\*\frac{Precision \* Recall}{Precision + Recall}2∗*Precision*+*RecallPrecision*∗*Recall*​

F1 score

SUBMIT

**Receiver Operating Characteristics (ROC)** chart that we just looked at, the **Area Under the Curve (AUC)** for the diagonal line is 0.5. What does this indicate?

* A classifier that performs no better than random guessing

# Evaluation Metrics for Regression

If you recall, classification yields discrete outputs (e.g., cat vs dog or positive vs. negative), while regression yields continuous, numerical outputs (e.g., 3.229, 23 minutes, $17.78).

Not surprisingly then, we need a different set of metrics for evaluating regression models. Let's have a look.

Again, note that with regression metrics, we are using functions that in some way calculate the numerical difference between the predicted vs. expected values.

### QUIZ QUESTION

Below are the regression metrics we just discussed. Can you match each one with the description of what it measures?

### ***Submit to check your answer choices!***

**WHAT IT MEASURES**

**METRIC**

How close the regression line is to the true values.

MAE

Square root of the squared differences between the predicted and actual values.

RMSE

Average of the absolute difference between each prediction and the true value.

R-Squared

Strength and direction of the relationship between predicted and actual values.

Spearman correlation

SUBMIT

NEXT

# Strength in Numbers

Remember, no matter how well-trained an individual model is, there is still a significant chance that it could perform poorly or produce incorrect results. Rather than relying on a single model, you can often get better results by training multiple models or using multiple algorithms and in some way capturing the collective results. As we mentioned, there are two main approaches to this: **Ensemble learning** and **automated machine learning**.

**ensemble learning** combines multiple machine learning models to produce one predictive model. There are three main types of ensemble algorithms:

**Bagging** or **bootstrap aggregation**

* Helps reduce overfitting for models that tend to have high variance (such as *decision trees*)
* Uses random subsampling of the training data to produce a *bag* of trained models.
* The resulting trained models are homogeneous
* The final prediction is an average prediction from individual models

**Boosting**

* Helps reduce bias for models.
* In contrast to bagging, boosting uses the same input data to train multiple models using different hyperparameters.
* Boosting trains model in *sequence* by training weak learners one by one, with each new learner correcting errors from previous learners
* The final predictions are a *weighted* average from the individual models

**Stacking**

* Trains a large number of completely different (heterogeneous) models
* Combines the outputs of the individual models into a meta-model that yields more accurate predictions
* **Automated machine learning**, like the name suggests, automates many of the iterative, time-consuming, tasks involved in model development (such as selecting the best features, scaling features optimally, choosing the best algorithms, and tuning hyperparameters). Automated ML allows data scientists, analysts, and developers to build models with greater scale, efficiency, and productivity—all while sustaining model quality.