
PLURALSIGHT

Intro to AI/ML in Azure

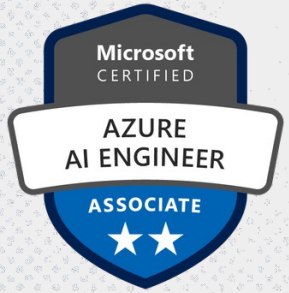
Welcome!

Tarek Atwan
Instructor, Pluralsight

Objectives

At the end of this course, you will be able to:

- Have a good understanding of Azure **Machine Learning Workspace** and **Azure AI Services**
- Be ready to continue your journey and explore taking **DP-100** and **AI-102** Certification



Microsoft Certified: Azure AI Engineer Associate

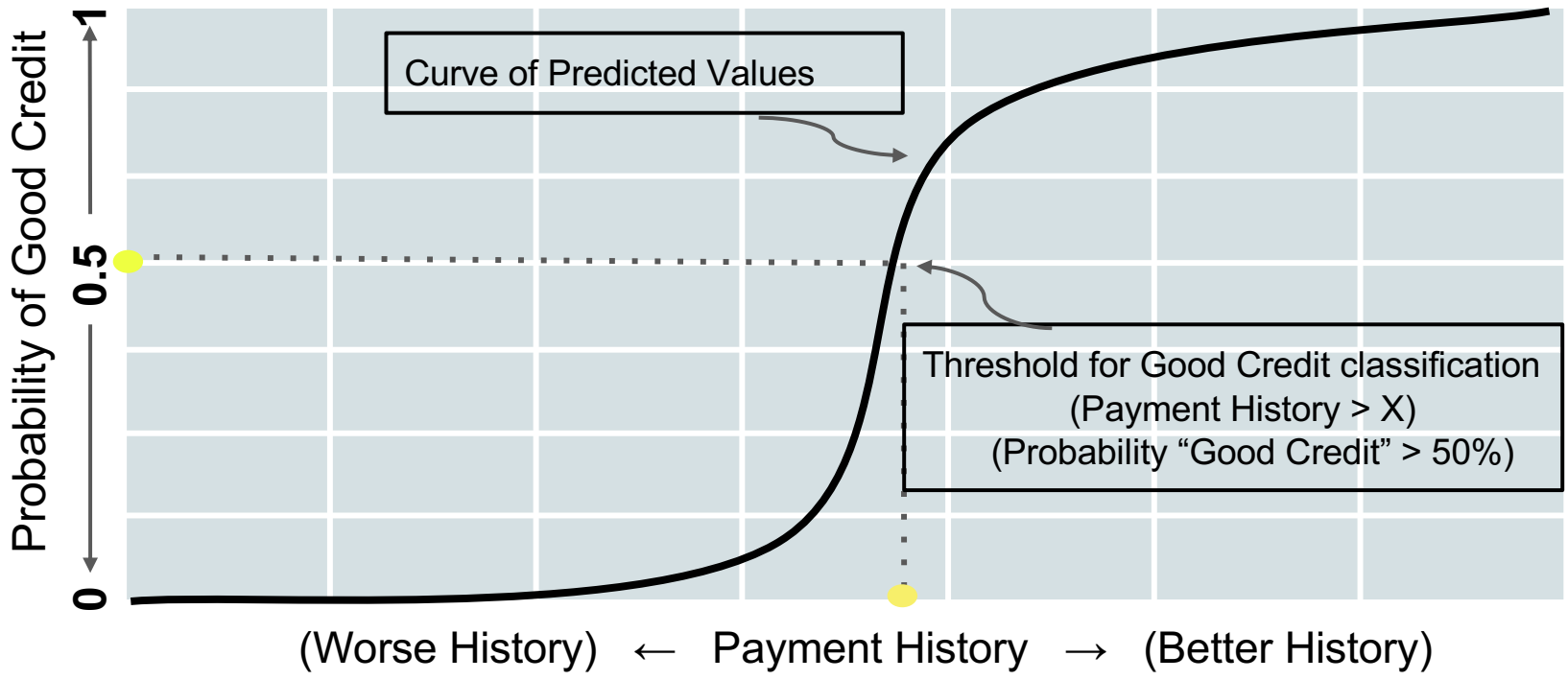
Microsoft Certified: Azure Data Scientist Associate

Making Predictions with Logistic Regression

Logistic regression is a statistical method for predicting binary outcomes from data.

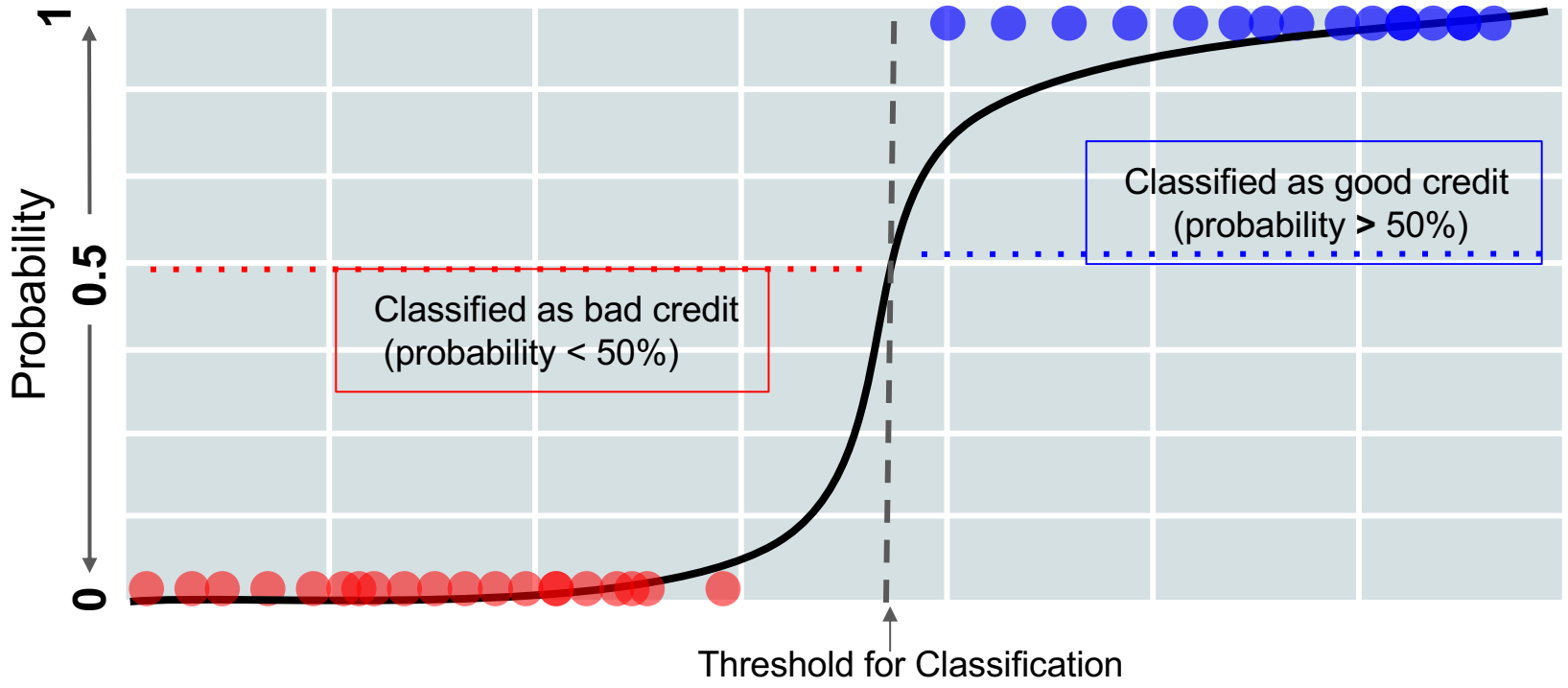
Making Predictions with Logistic Regression

Each data point receives a probability of being in the **1** category (e.g. “good credit”).



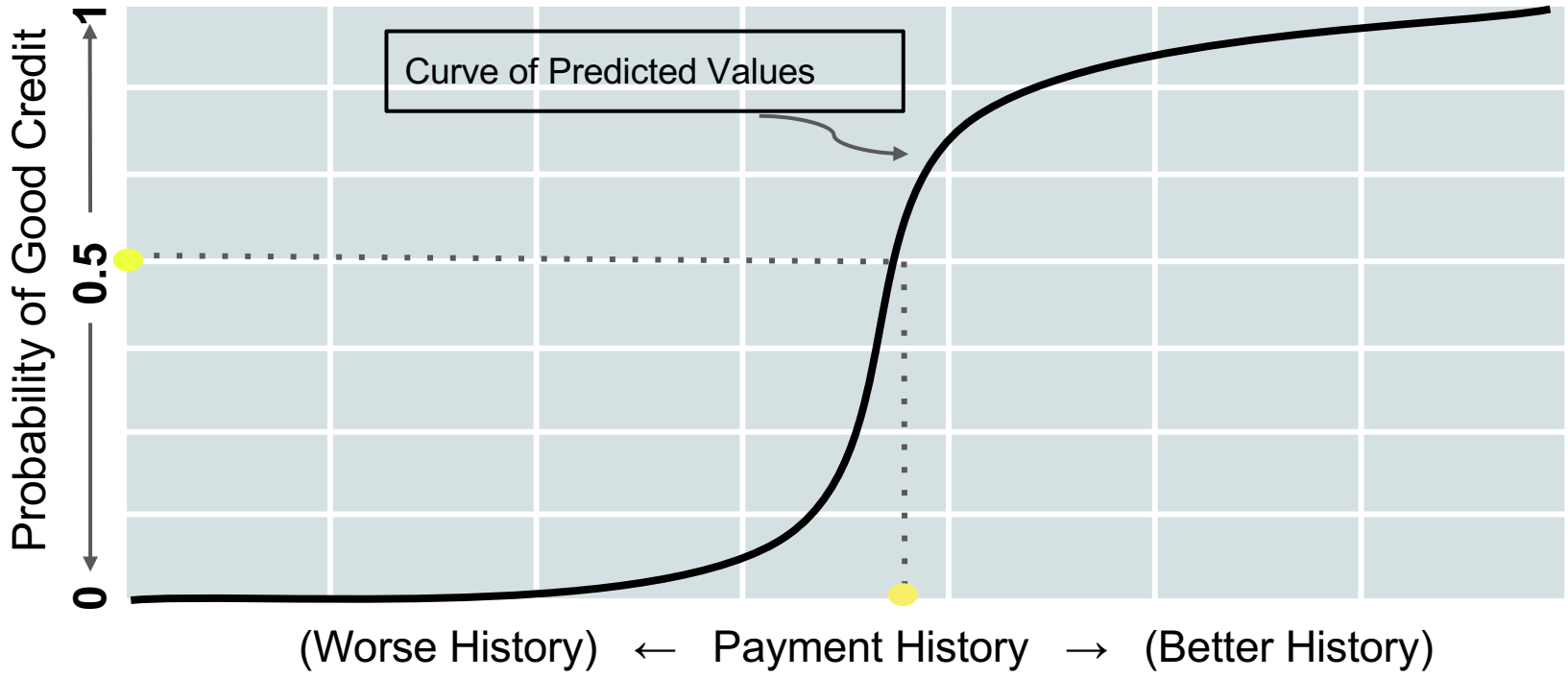
Making Predictions with Logistic Regression

If the probability is above a certain threshold, that data point is estimated to be a **1** ("good credit"). Below that threshold, the data point is a (**0**).



The Sigmoid Function

How do we translate continuous data like **payment history** into a probability of "good credit" ranging from 0 to 1?



Steps in Logistic Regression Modeling

We can use logistic regression to predict which category or class a new data point should go into.

01	02	03	04
Preprocess	**Train**	**Validate**	**Predict**
This step consists of cleaning the data (such as removing rows with missing values) and splitting it into subsets for training and testing the model.	Training is when we use a large subset of our labeled data to teach the model to recognize classification patterns.	In the validation step, we use a small subset of our labeled data to test how well the model is able to predict labels.	Finally, we use our model to predict labels for unclassified data.

Hold-Out Validation

We create two datasets where we hold out a subset of data that we refer to as the testing dataset.

This allows us to measure the performance of our model and parameter selections with a subset of data not used to train the model.

Evaluating Logistic Regression Predictions

Evaluating Logistic Regression Predictions

In this scenario, we used a logistic regression model to predict if an individual has diabetes, based on a set of diagnostic metrics provided as a dataset.

- We evaluated the logistic regression model by using a scoring feature.
- This revealed that the model is somewhat accurate.
- However, can you trust that its predictions are correct?

How sure are you that your models can actually predict diabetes?

75%

sure, as described by the scored accuracy.

Would you feel comfortable giving the diagnosis of diabetes based on the predictions of the model?

Answer

No.

The prediction is not 100% accurate.
There is room for error, and there are false positives.

	Test says you don't have it	Test says you do have it
You don't really have it	True Negative	False Positive
You do really have it	False Negative	True Positive

What is better:
the false-positive or false-negative?

Answer

Neither option is preferred.
Both leave opportunities for inaccuracy.

- However, in the case of a model that incorrectly flags patients as having diabetes, we can use additional tests to refine the prediction and filter out individuals who do not have diabetes.
- This way, anyone with the potential of having the disease can be given the treatment and attention they need.

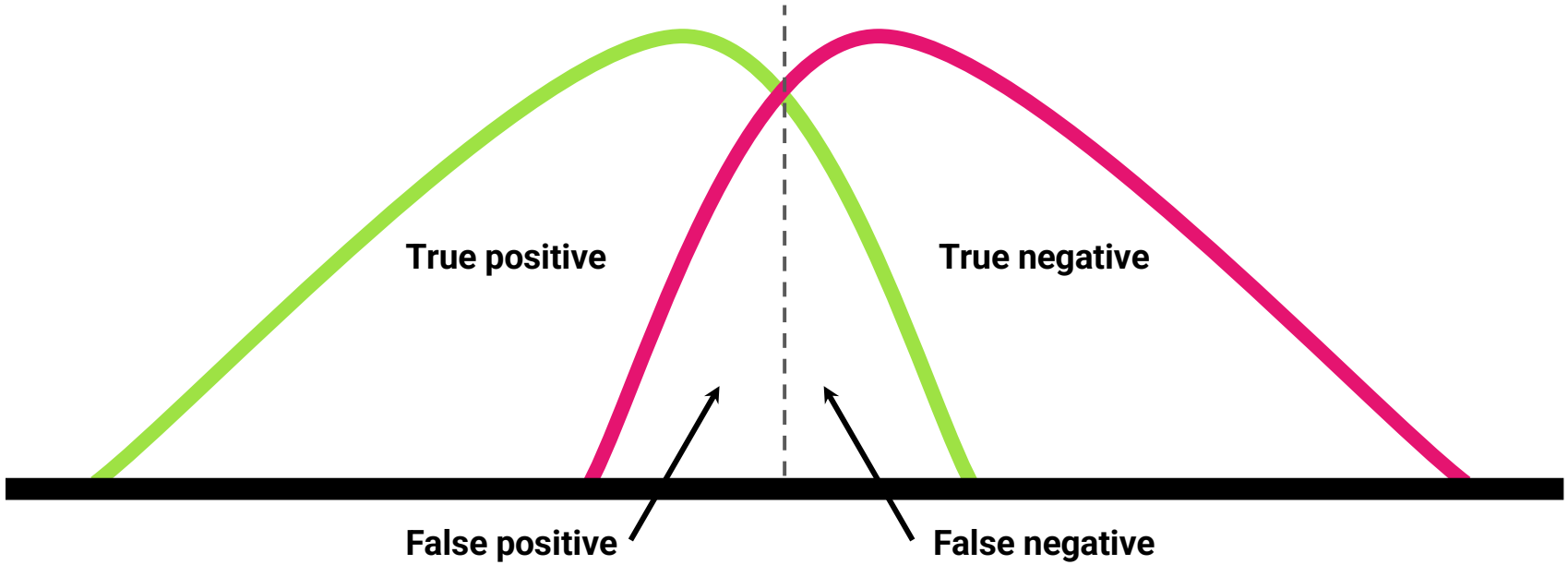
The process of evaluating a model requires more than simply scoring/measuring the model for accuracy.


In addition to accuracy, we must measure for precision and recall, both of which can be used to eliminate false positives and false negatives.

Accuracy, Precision, Recall

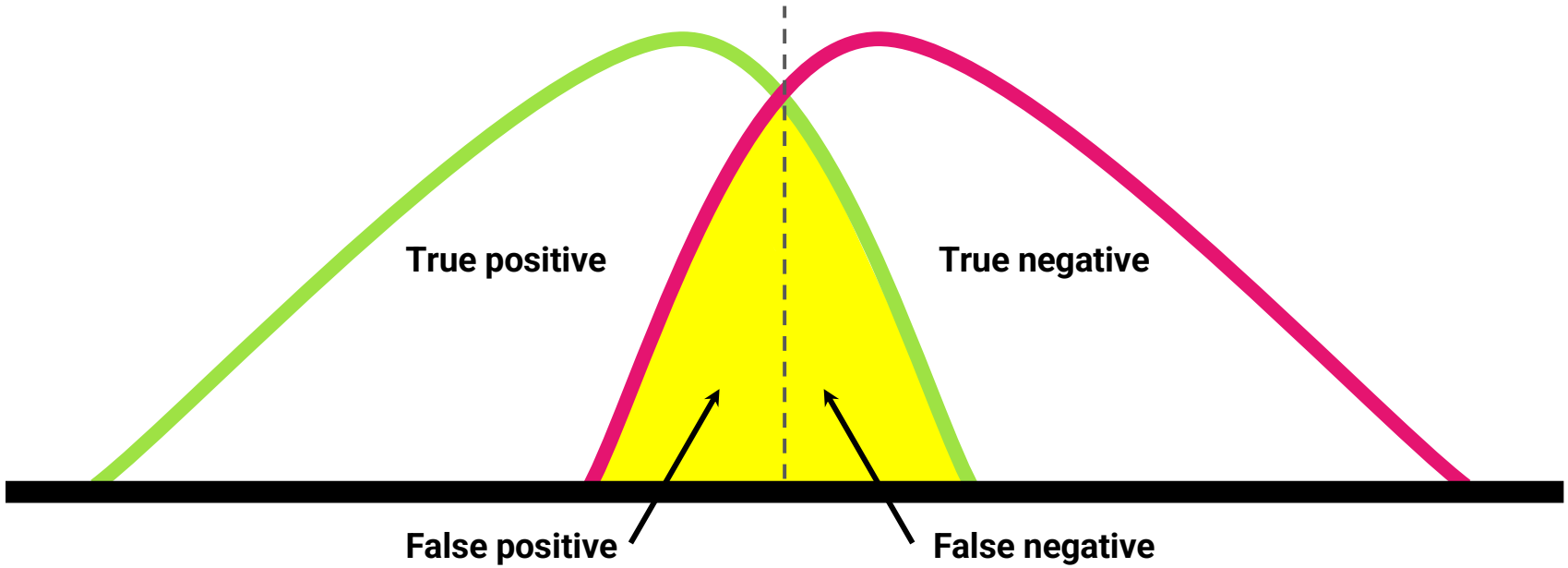
Accuracy, Precision, and Recall

Accuracy, precision, and recall are especially important for classification models that involve a binary decision problem. Binary decision problems have two possible correct answers: **True Positive** and **True Negative**.



Accuracy, Precision, and Recall

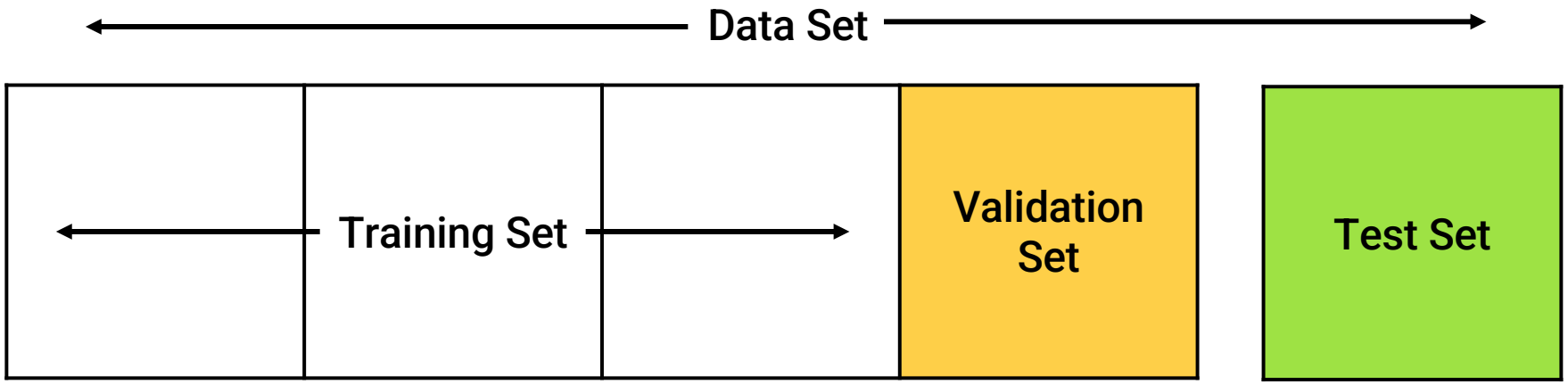
Inaccurate and imprecise models will return false positives and false negatives.



Accuracy is a measure of how often the model is correct: the ratio of correctly predicted observations to the total number of observations.

Accuracy

We can evaluate the accuracy of a model by scoring the model using training and testing datasets. However, accuracy does not communicate how precise the model is.



Accuracy

Accuracy can be very susceptible to imbalanced classes.

- Example: If the number of good loans greatly outweighs the number of at-risk loans.
- In this case, the model will primarily evaluate the good loans because that has the biggest impact on accuracy.
- However, we also care about at-risk loans, so we need a metric that can help us evaluate each class prediction.

Calculation:

```
(TP + TN) / (TP + TN + FP + FN)
```

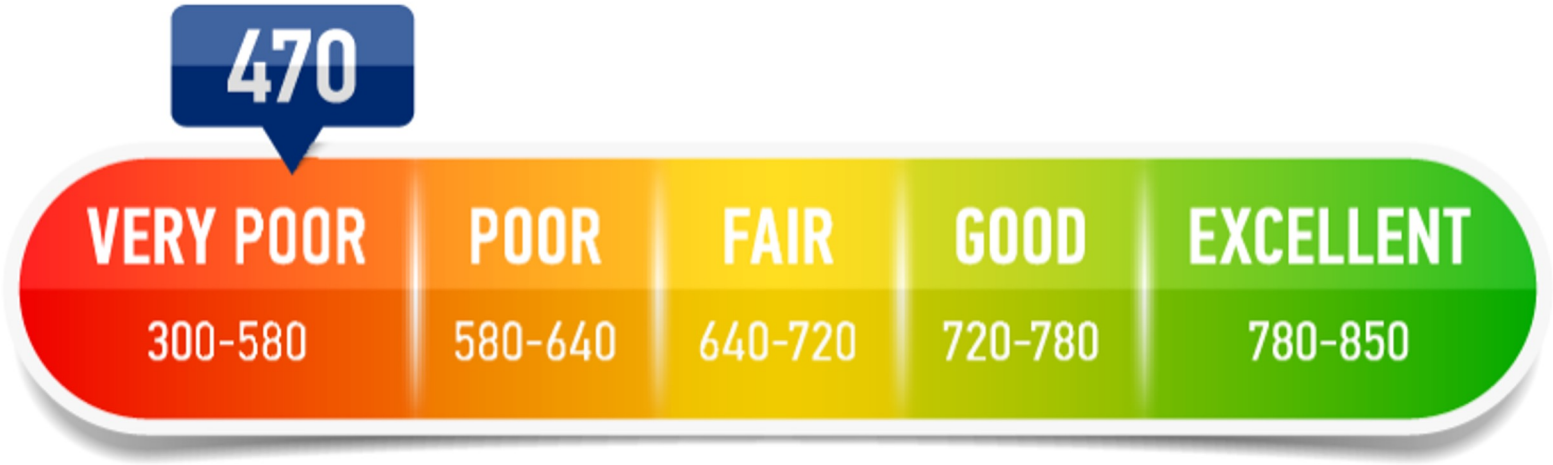
Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

(i.e., of all the samples classified as having diabetes, how many actually have diabetes?)

Precision

Another example: For all the individuals who were classified by the model as being a credit risk, how many actually were a credit risk?

Did we classify them comprehensively and correctly?



Precision

High precision relates to a low false positive rate.

Calculation:

$$TP / (TP + FP)$$

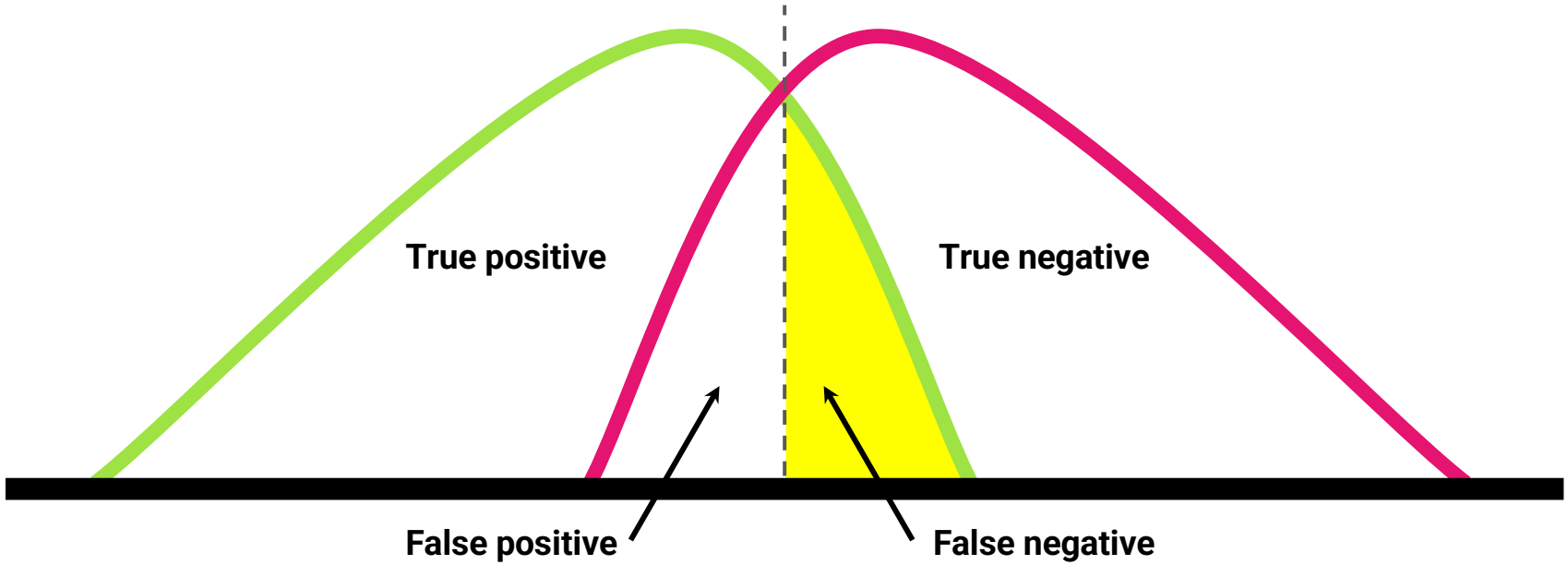
Recall is the ratio of correctly predicted positive observations to all predicted observations for that class.

(i.e., of all of the actual diabetes samples, how many were correctly classified as having diabetes?)

Recall

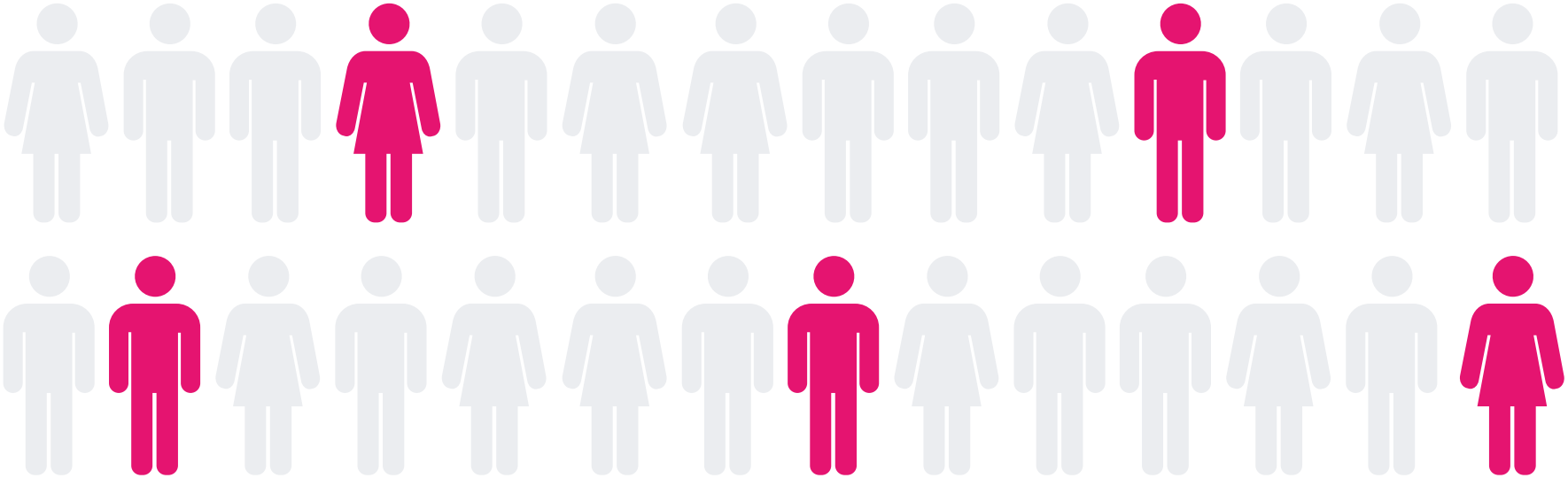
Of all of the actual diabetes samples, how many were correctly classified as having diabetes?

Did we classify all samples correctly, leaving little room for false negatives?



Recall

Another example:
For all the individuals who are a credit risk, how many were classified by the model as being a credit risk?



Recall

High recall correlates to a more comprehensive output and a low false negative rate.

$$TP / (TP + FN)$$

We use a **confusion matrix** to measure and gauge the success of a model.

Confusion Matrix

Confusion matrixes reveal the number of true negatives and true positives (actuals) for each categorical class and compare them to the number of predicted values for each class.

n=165	**Predicted: No**	**Predicted: Yes**
Actual=No	50	10
Actual=Yes	5	100

Confusion Matrix

These values are then individually summed by column and row. The aggregate sums are then compared to assess accuracy and precision. If the aggregates match, we can consider the model to be accurate and precise.

n=165	**Predicted: No**	**Predicted: Yes**	
Actual=No	50	10	**=60**
Actual=Yes	5	100	**=105**
	=55	**=110**	

Confusion Matrix

Confusion matrixes are great because they describe the performance of the classification model.

- By looking at the confusion matrix, we can determine whether the model has been correctly trained to produce comprehensive, accurate, and precise predictions.
- For binary classifiers like the logistic regression classifier, a confusion matrix will measure the number of positive and negative predictions.
- These will then be compared to the actuals.

Classification Report

Classification report identifies the **precision**, **recall**, and **accuracy** of a model for each given class.

	precision	recall	f1-score	support
No Diabetes	0.77	0.90	0.83	125
Diabetes	0.72	0.49	0.58	67
accuracy			0.76	192
macro avg	0.74	0.69	0.71	192
weighted avg	0.75	0.76	0.74	192

KNN: Logistic Regression Friendly Neighbor

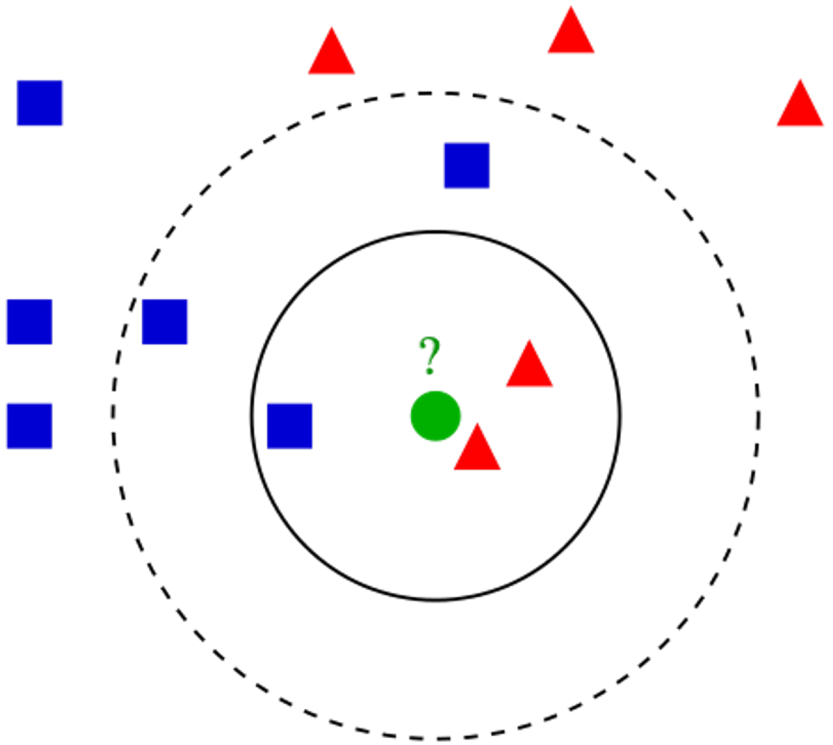
One of the most popular and flexible supervised machine learning models is the k-nearest neighbor model.

The **k-nearest neighbor** compares known data points to determine the classification of a novel data point.

KNN: Logistic Regression Friendly Neighbor

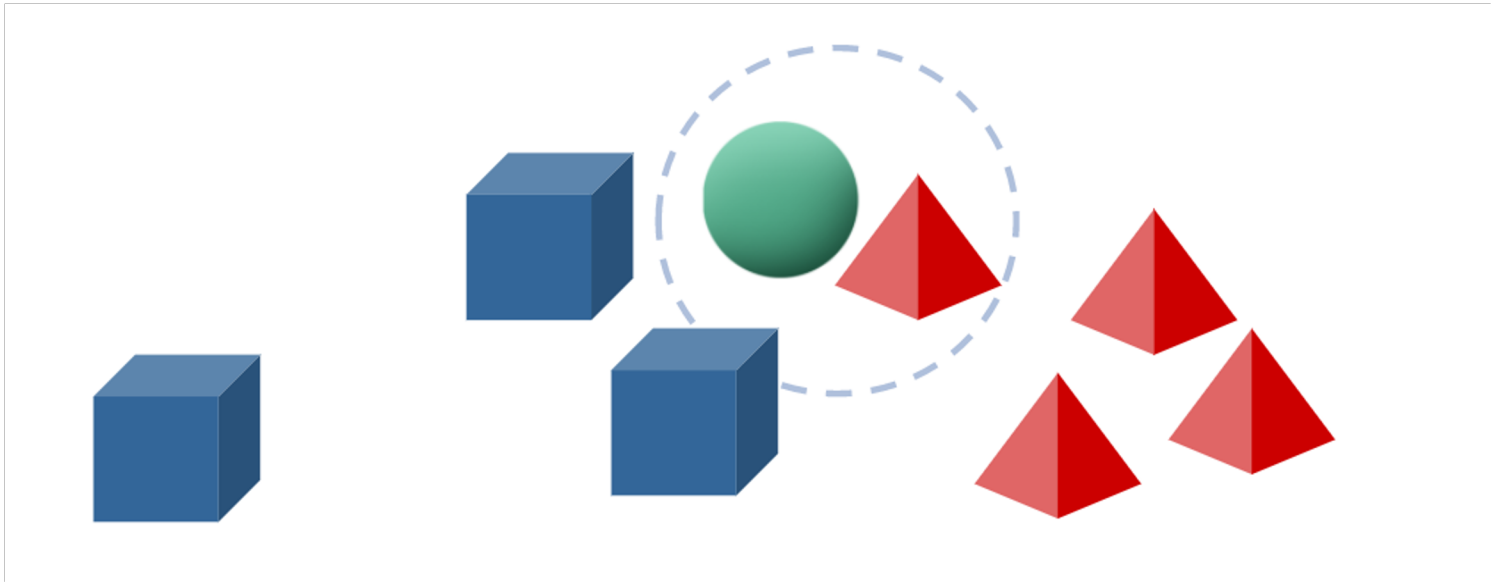
We set the k-value parameter to tell the algorithm to find the k-number of closest known data points.

Then, the algorithm determines what the majority of surrounding data points are classified to determine the class of the new data point.



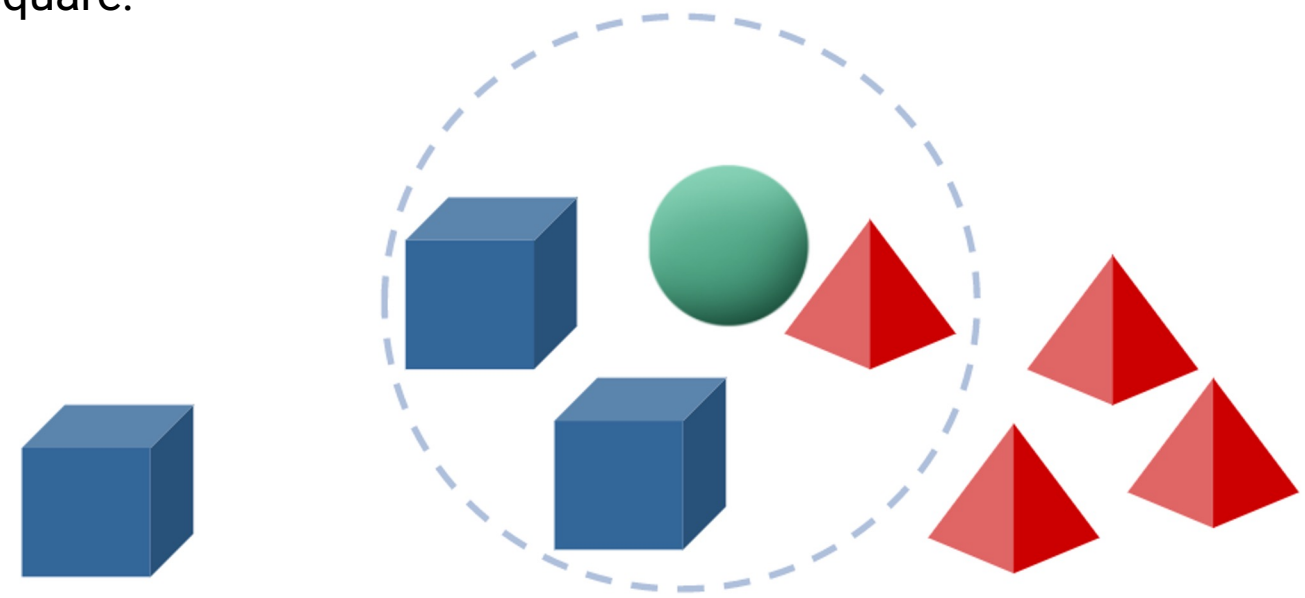
KNN: Logistic Regression Friendly Neighbor

Consider the example we wanted to classify a new data point (green circle) from known square and triangle data points. If **k = 1**, then the algorithm classifies our new data point to whatever is the closest known single neighbor. In this case our green circle would be classified as a red triangle.



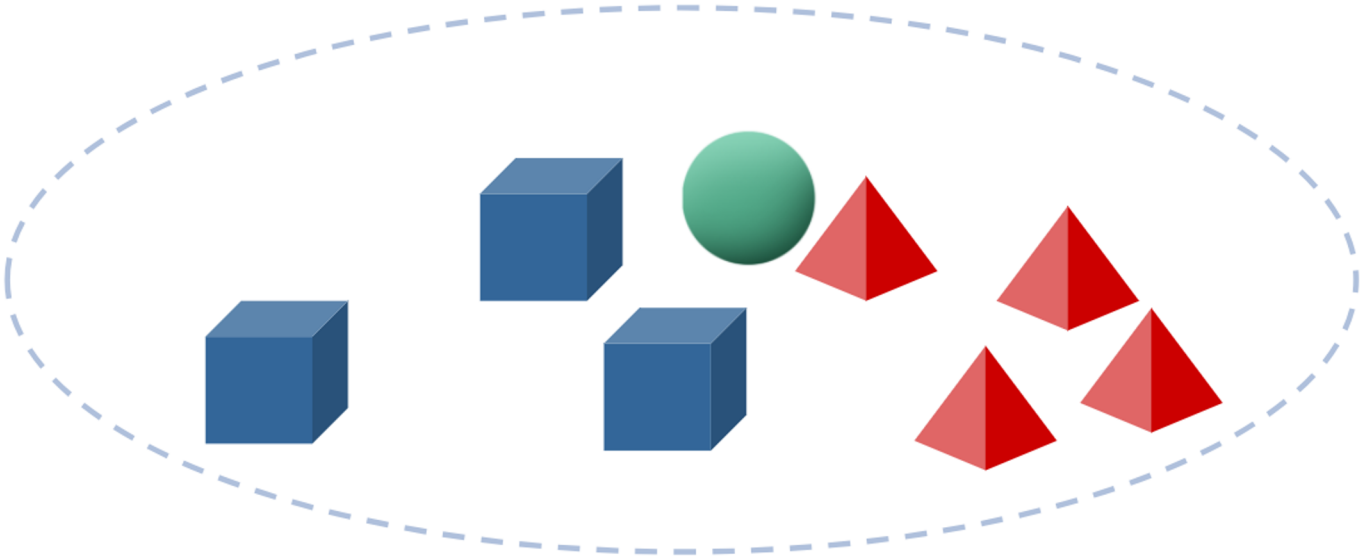
KNN: Logistic Regression Friendly Neighbor

As the number of k-nearest neighbors increase, the distribution of nearest classifications can change. If in our example we used **k = 3**, there are two squares over the one triangle, so our model would classify the new data point as a blue square.



KNN: Logistic Regression Friendly Neighbor

If we make our k-value too large, our k-nearest neighbor classification will classify all new data types as the most dominant classification. In this final part to our example, **k = 7** means that all known data points are considered and our green circle would be classified as a red triangle.

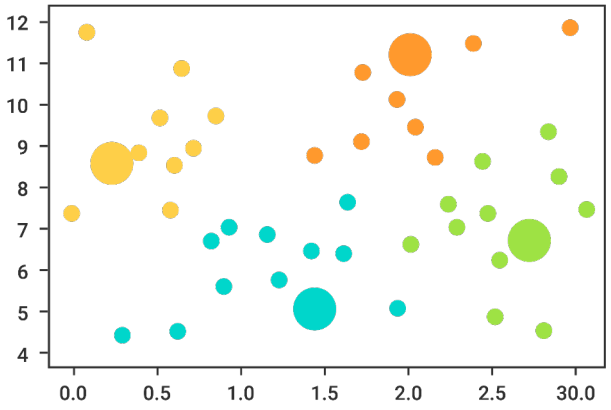


K-Means

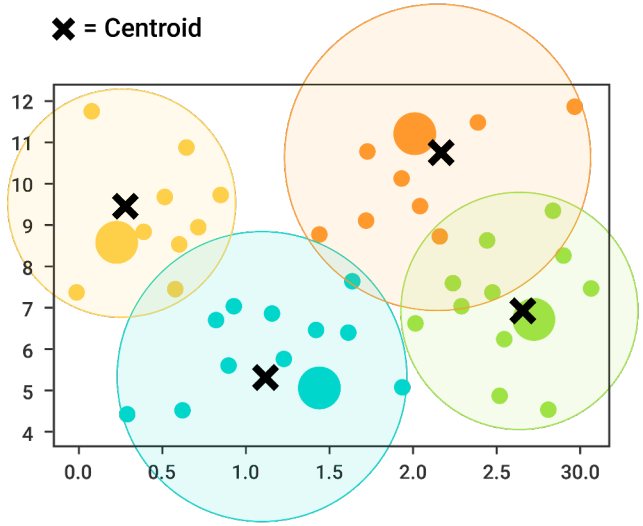
K-means is an unsupervised learning algorithm used to identify clusters and solve clustering issues. k represents the number of clusters.

K-Means

The K-means algorithm groups the data into k clusters, where each piece of data is assigned to a cluster based on some similarity or the distance measured to a centroid.

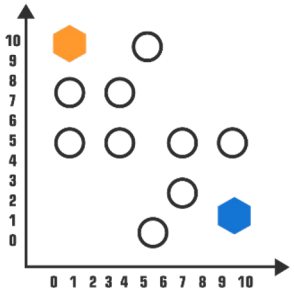


A centroid represents a data point that is the arithmetic-mean position of all the points in a cluster.

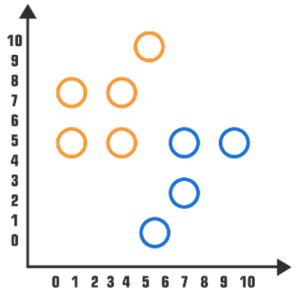


K-Means

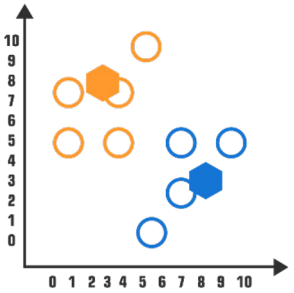
The K-means algorithm then repeats this process, again and again, each time getting a little bit better at separating the data points into distinct groups.



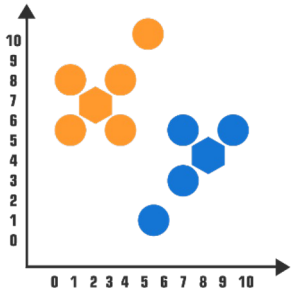
Randomly initialize the k starting centroids.



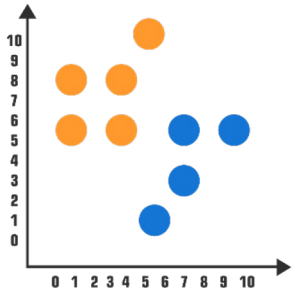
The starting centroids are assigned to random locations.



Each data point is assigned to its nearest centroid.



The centroids are recomputed as the mean of the data points assigned to the respective cluster.

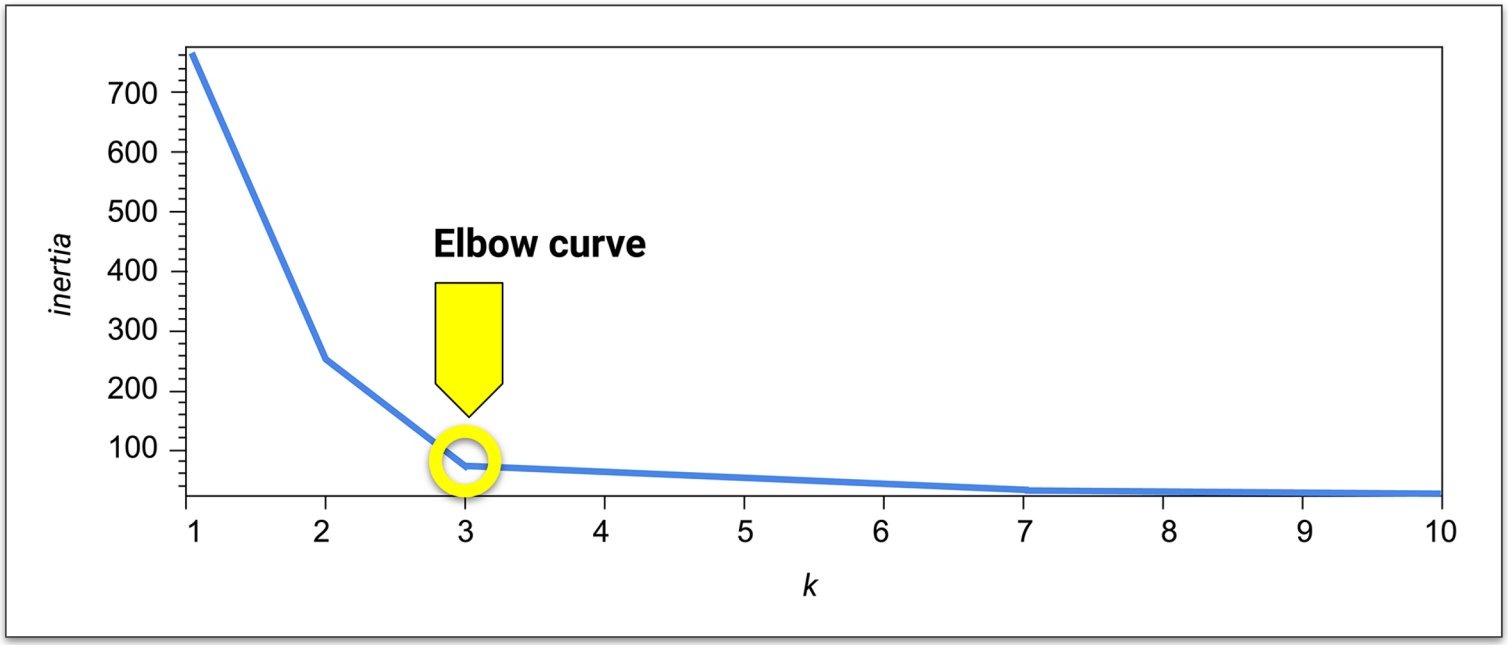


Repeat steps 1 through 3 until the values of the centroids are stable.

The K-means algorithm does not determine the best number of clusters. The user must assign the value of k.

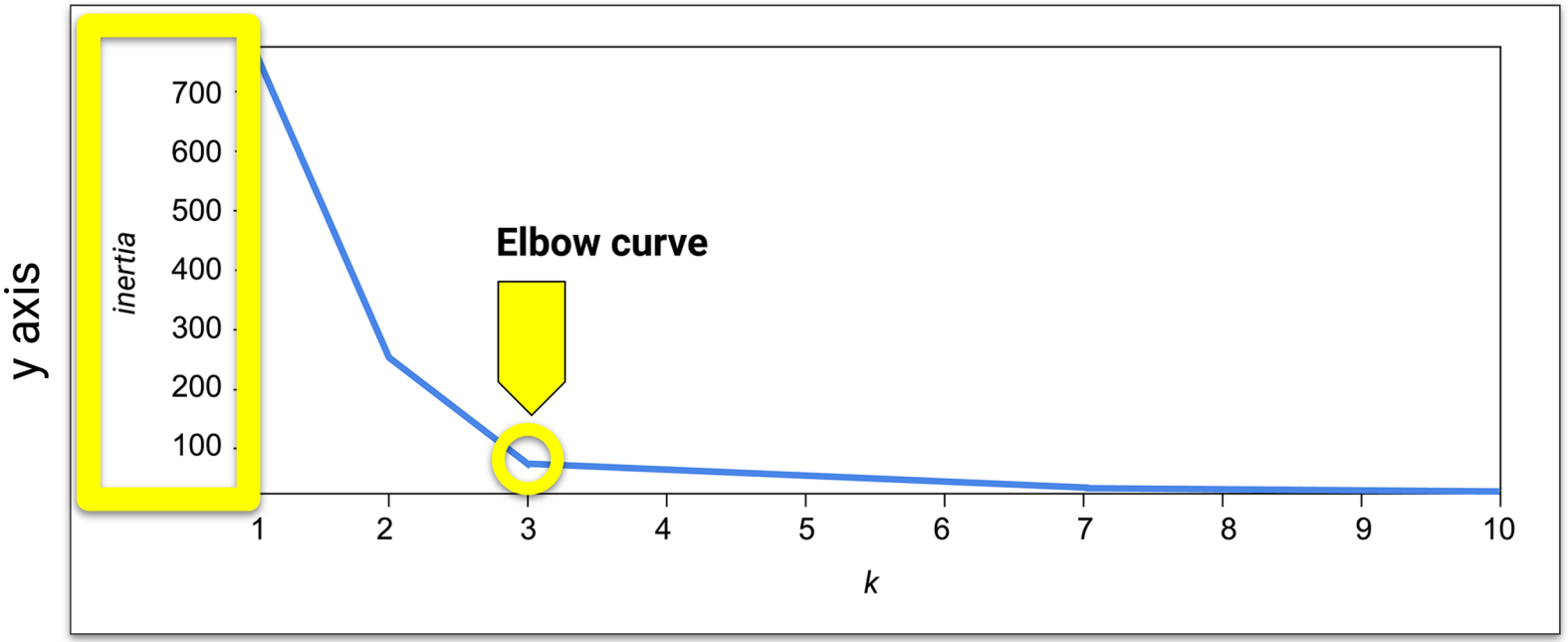
Elbow Curve

The best number for k is the number of clusters where the curve turns like an elbow. More specifically, it's the inflection point where the slope takes a sharp turn and flattens out.

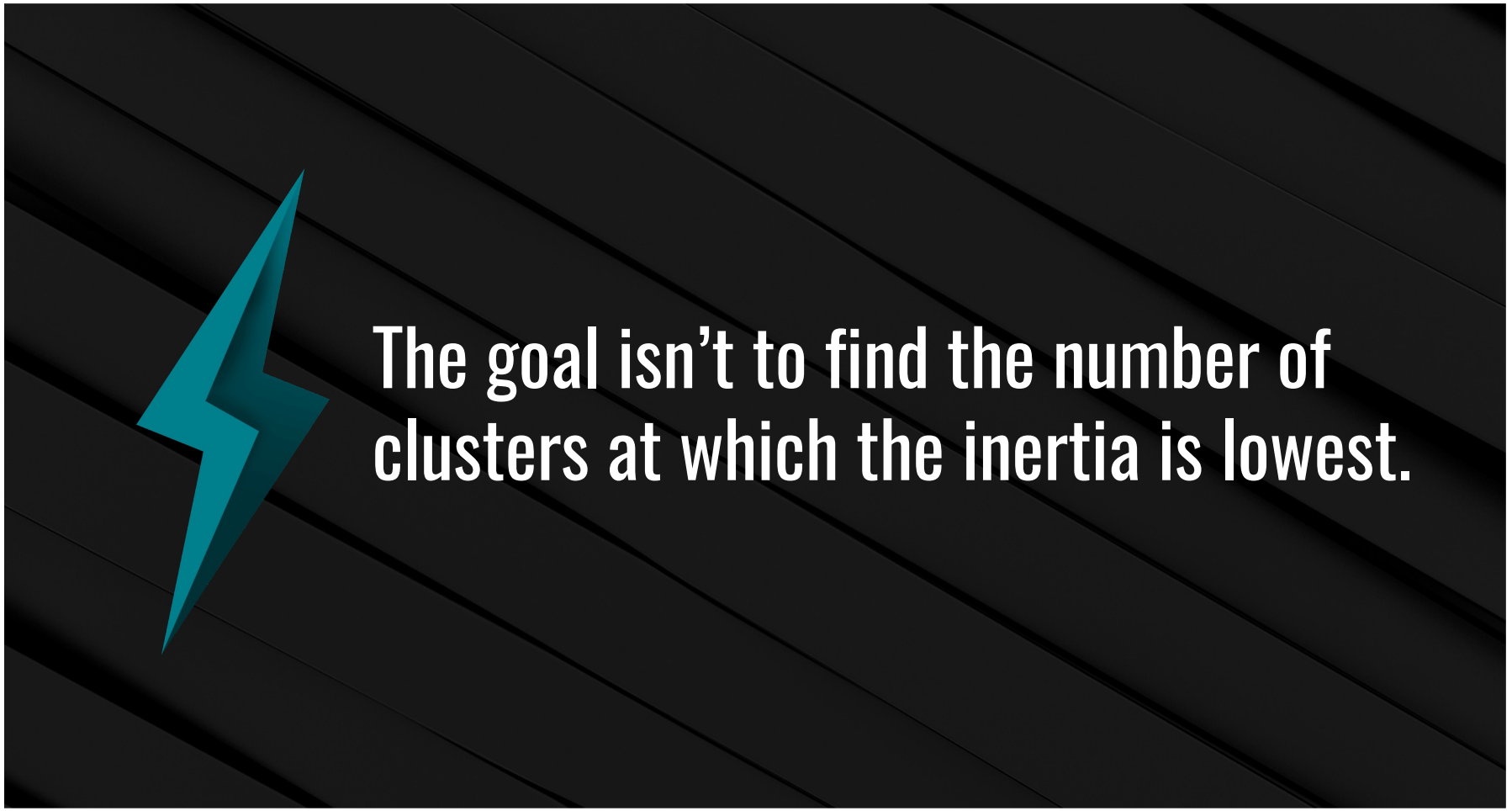


Elbow Curve

Alternatively, we can take a numerical approach. Using the inertia, we choose the best k, the value at which adding more clusters only marginally decreases the inertia.

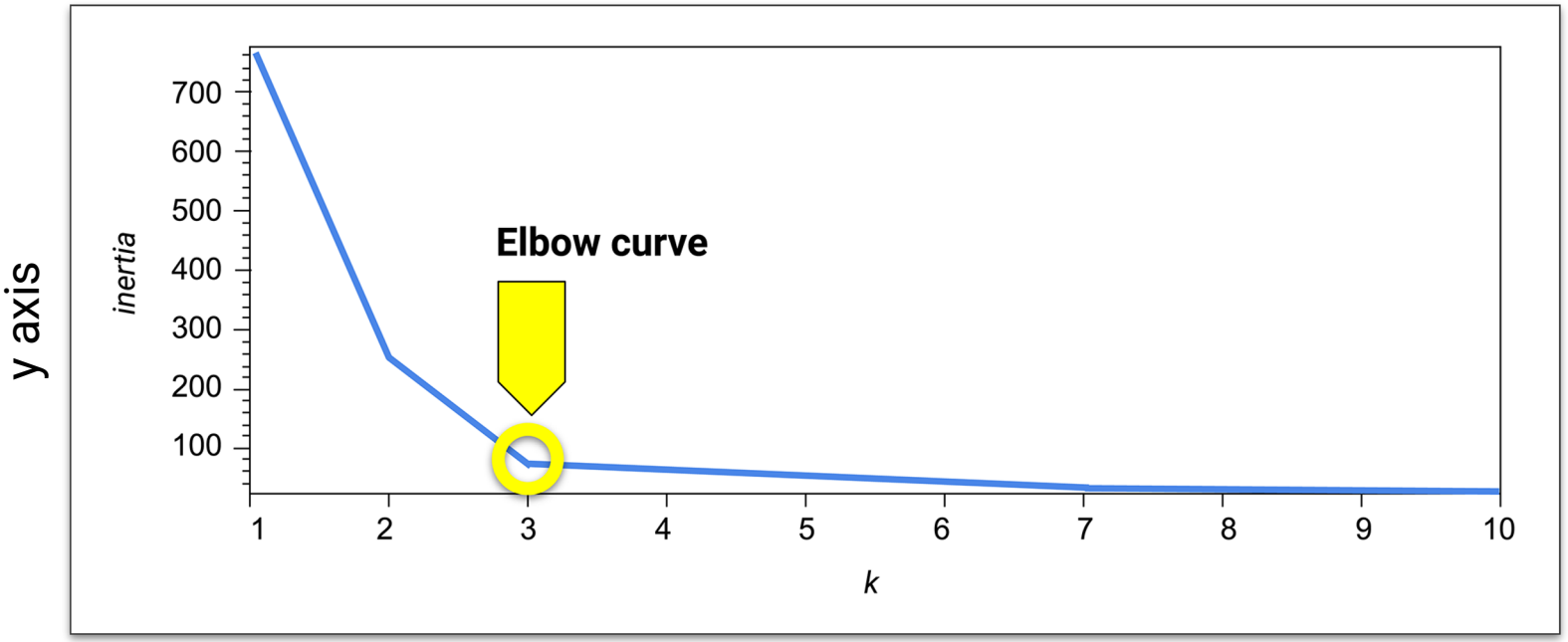


x axis


The goal isn’t to find the number of clusters at which the inertia is lowest.

Elbow Curve

If we had as many clusters as data points, each cluster would be the most tightly packed. We want to find the **optimal number** of clusters.

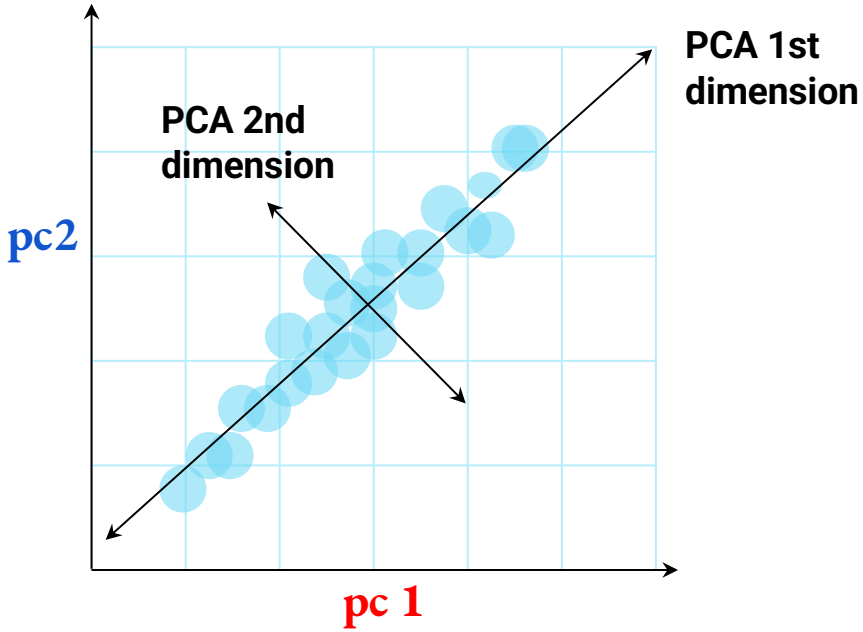


We can often enhance and optimize machine learning algorithms by applying Principal Component Analysis, or PCA.

Principal Component Analysis (PCA) is a statistical technique for streamlining the machine learning process when too many factors exist in the data.

Principal Component Analysis (PCA)

PCA reduces the number of factors by transforming a large set of features into a smaller one that contains MOST of the information of the original larger dataset.



Principal Component Analysis (PCA)

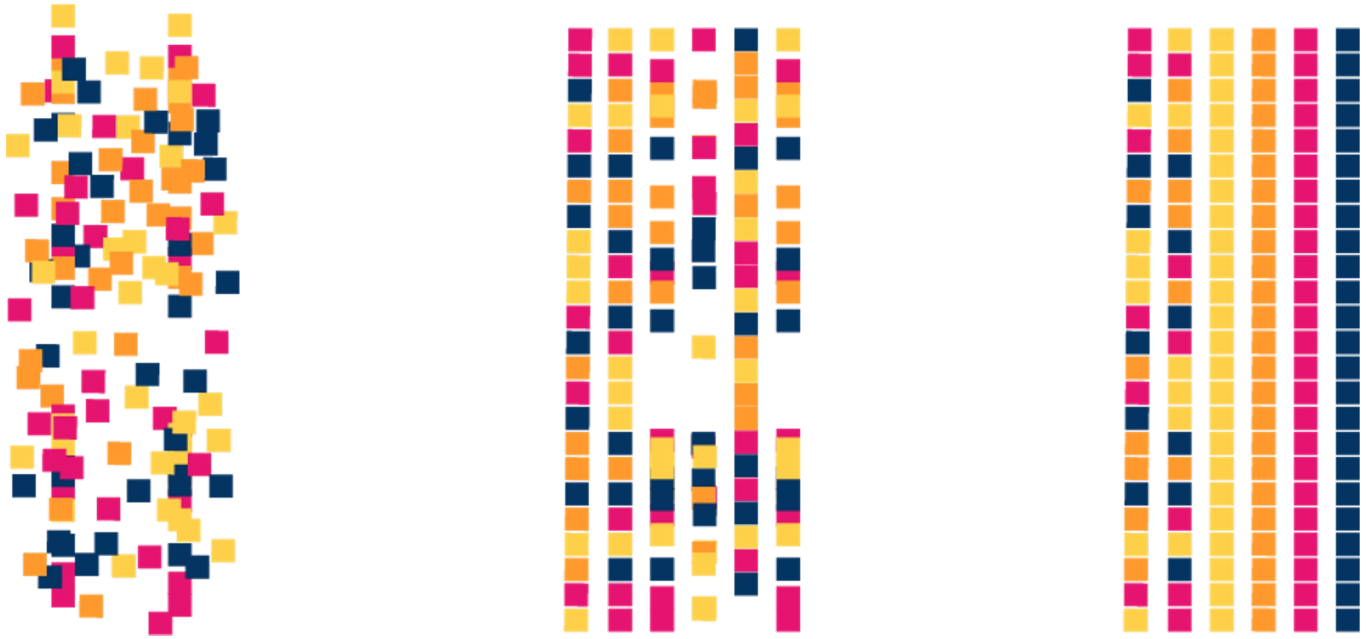
PCA is a dimensionality-reduction method that:

- Looks at all the dimensions (or data columns) in a dataset.
- Analyzes the weight of their contribution to the variance in the dataset.
- Reduces the dimensions to a smaller set that still contains as much of the information (the maximum variance) of the original dataset as possible.

PCA will NOT capture all the information from the original dataset, but it will capture as much as possible to maintain the predictive power and the meaning of the original dimensions.

Principal Component Analysis (PCA)

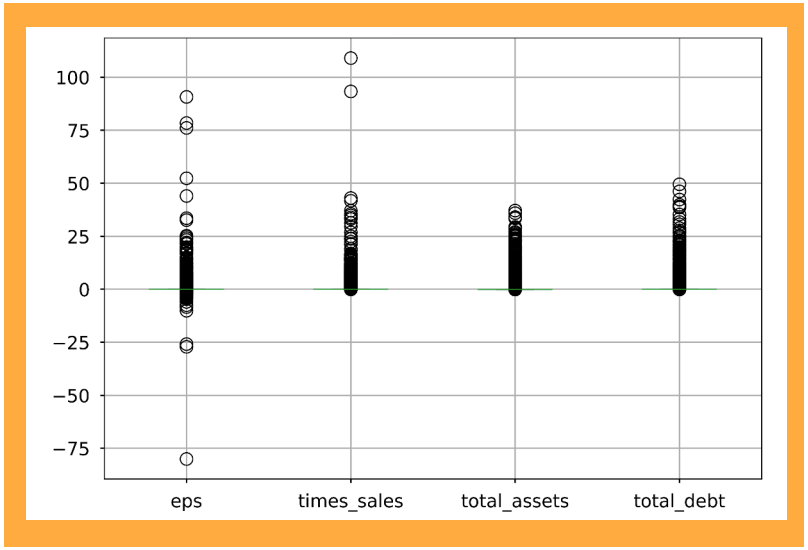
Reducing the number of factors, or **dimensional reduction**, comes at the expense of some accuracy, but the goal is to trade a little accuracy for simplicity.



Standard Scaling

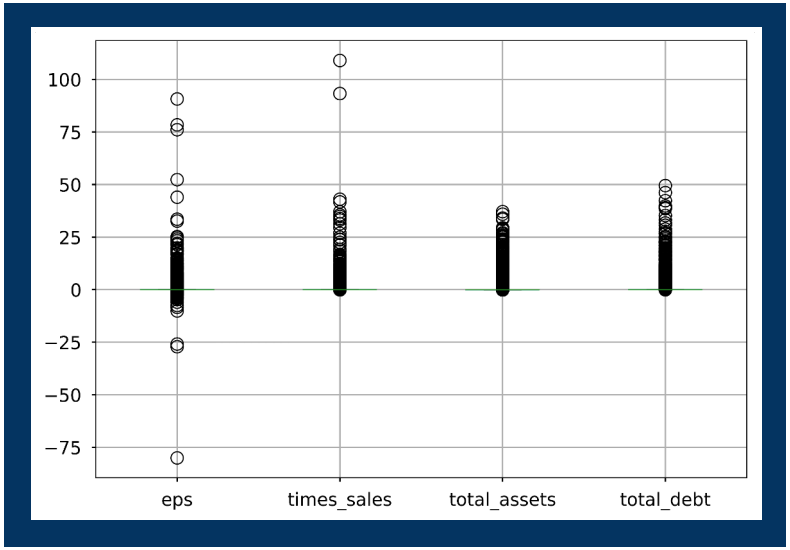
Before

Before using PCA, we'll apply standard scaling to learn how to transform the features of data.



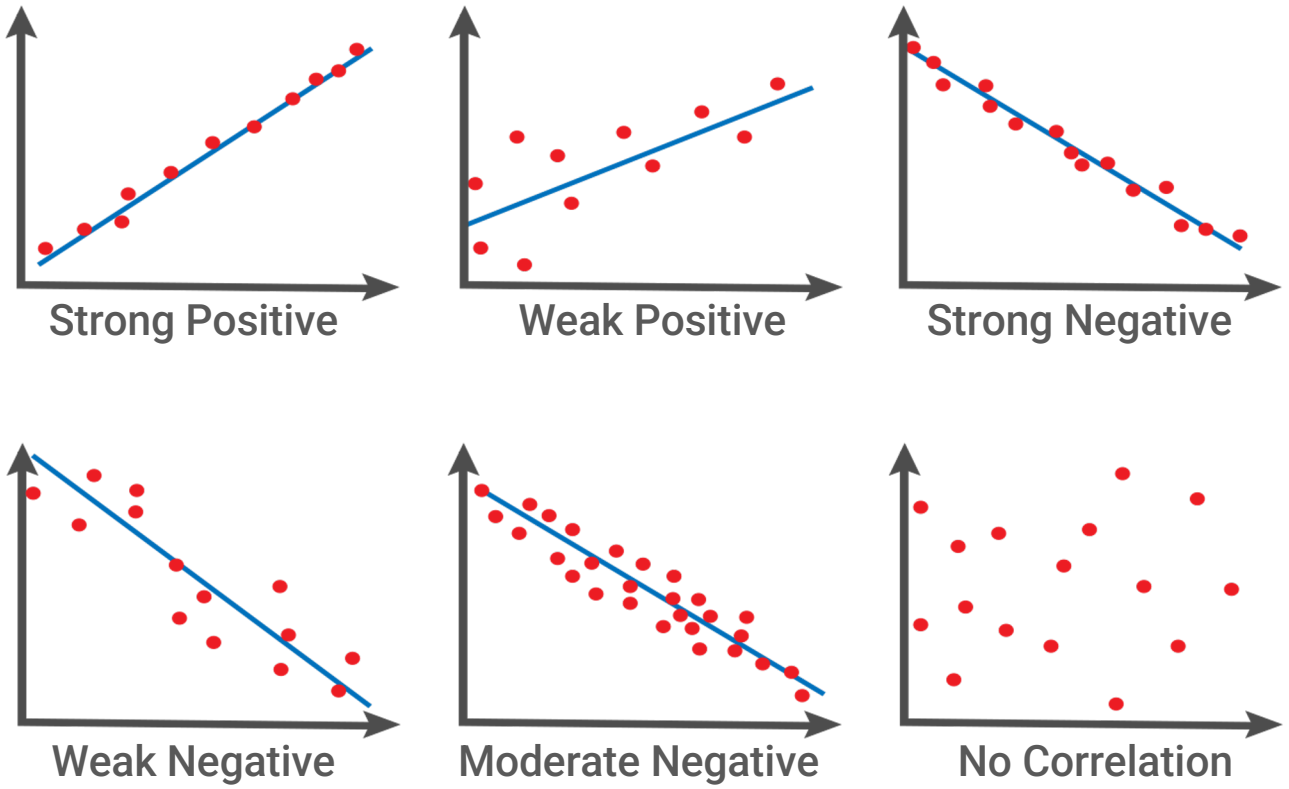
After

After scaling, we'll combine PCA with the K-means algorithm. This will give us a strategy to better handle extremely large financial datasets.



In statistics, a **correlation** defines the relationship between two or more variables.

Comparison of Correlation Relationships



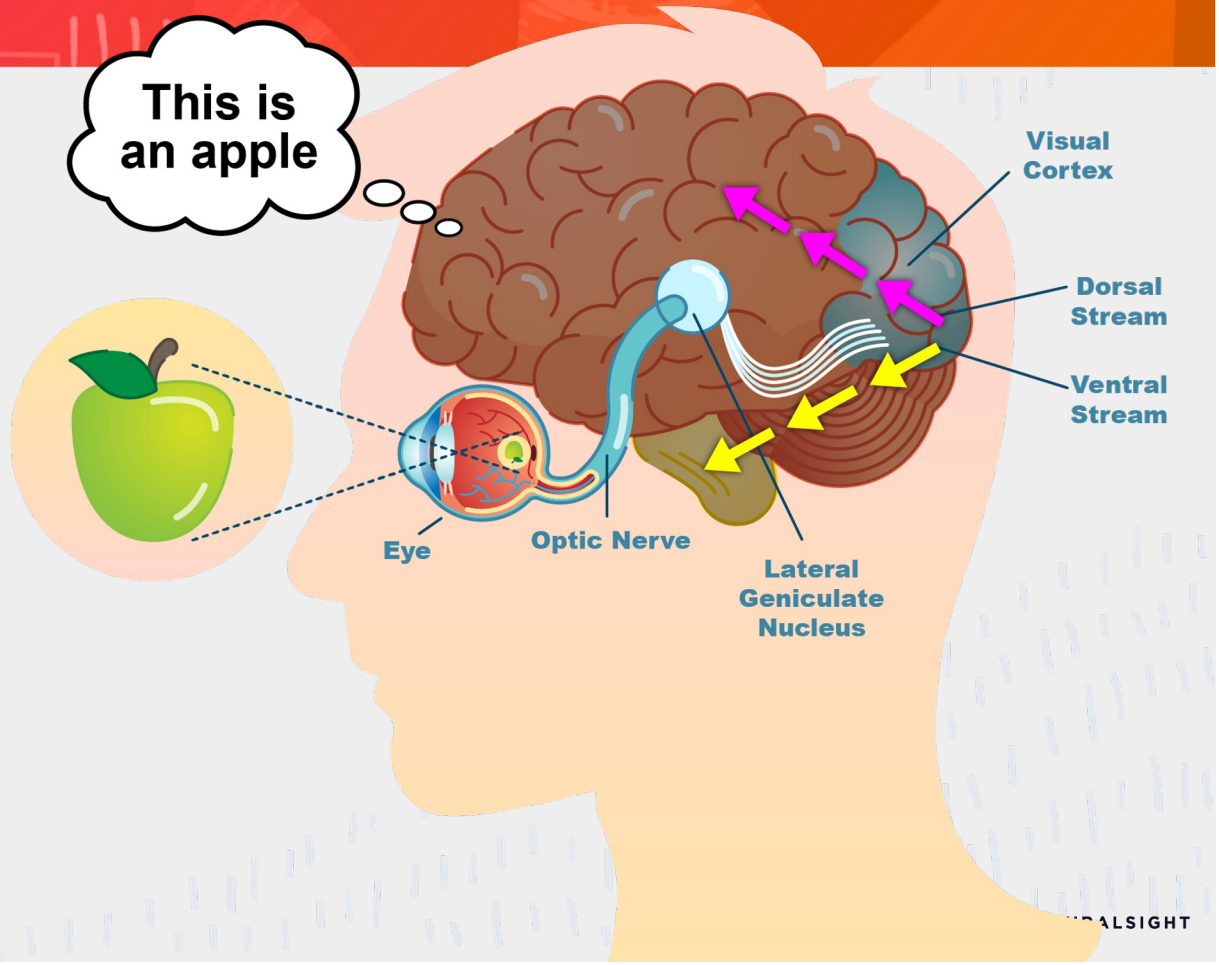
Correlations can be helpful,
but they don’t provide enough
information to infer the relationship
between two variables.

Artificial Neural Networks (ANN)

Neural Networks

How our brain works:

In order to recognize an image, our brain uses thousands
of neuron connections to find a match between the visual input and a mental representation of an object.

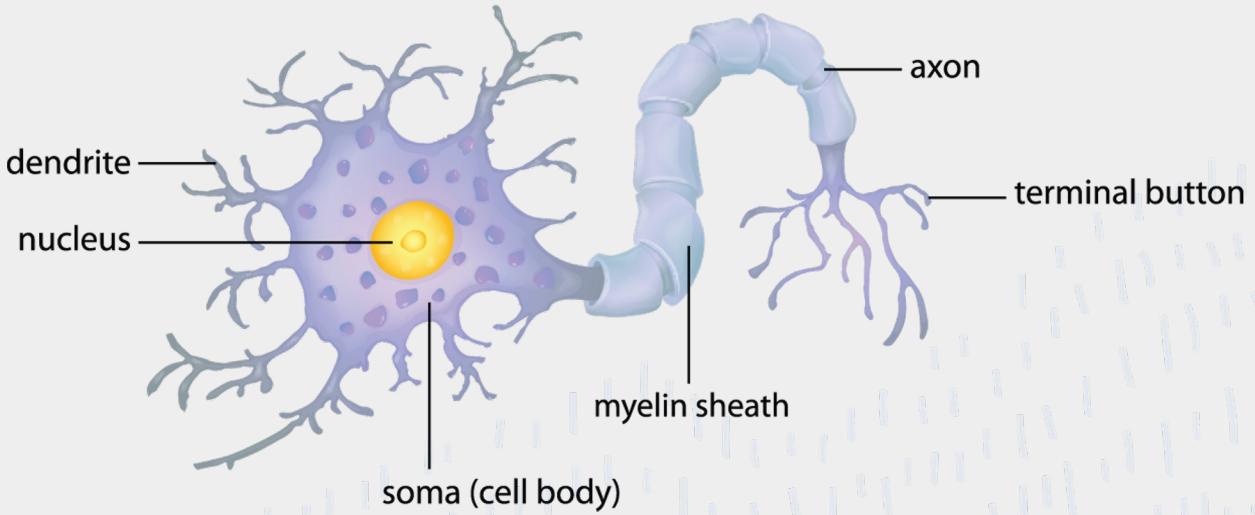


Neural Networks

The ability of the brain to process information and make predictions or interpretations is what inspired neurophysiologists and mathematicians to start the development of artificial neural networks (ANN).

In the same way that biological neurons receive input signals through the dendrites, an ANN receives input variables and processes them by using an activation function.

The output of an ANN is similar to the neuron nucleus in the brain.



History of Neural Networks

1943

Neurophysiologist **Warren McCulloch** and mathematician **Walter Pitts** wrote a paper on how neurons might work.

1949

Donald Hebb wrote *The Organization of Behavior*, which pointed out the fact that neural pathways are strengthened each time they are used.

1959

Bernard Widrow and **Marcian Hoff** of Stanford developed models called ADALINE and MADALINE.

1962

Widrow and **Hoff** developed a learning procedure that examines the value before the weight adjusts it (i.e., 0 or 1) according to the rule: Weight Change = (Pre-Weight line value).

1972

Teuvo Kohonen and **James A. Anderson** each developed a similar network independently of one another. They both used matrix mathematics to describe their ideas but did not realize that what they were doing was creating an array of analog ADALINE circuits.

History of Neural Networks

1982

John Hopfield of Caltech presented a paper to the National Academy of Sciences. His approach was to create more useful machines by using bidirectional lines. Previously, the connections between neurons was only one way.

1982

Joint US-Japan conference on **Cooperative/ Competitive Neural Networks**. Japan announced a new Fifth Generation effort on neural networks, and US papers generated worry that the US could be left behind in the field.

1986

Three independent groups of researchers, including **David Rumelhart**, a former member of Stanford's psychology department, came up with similar ideas which are now called back propagation networks.

1997

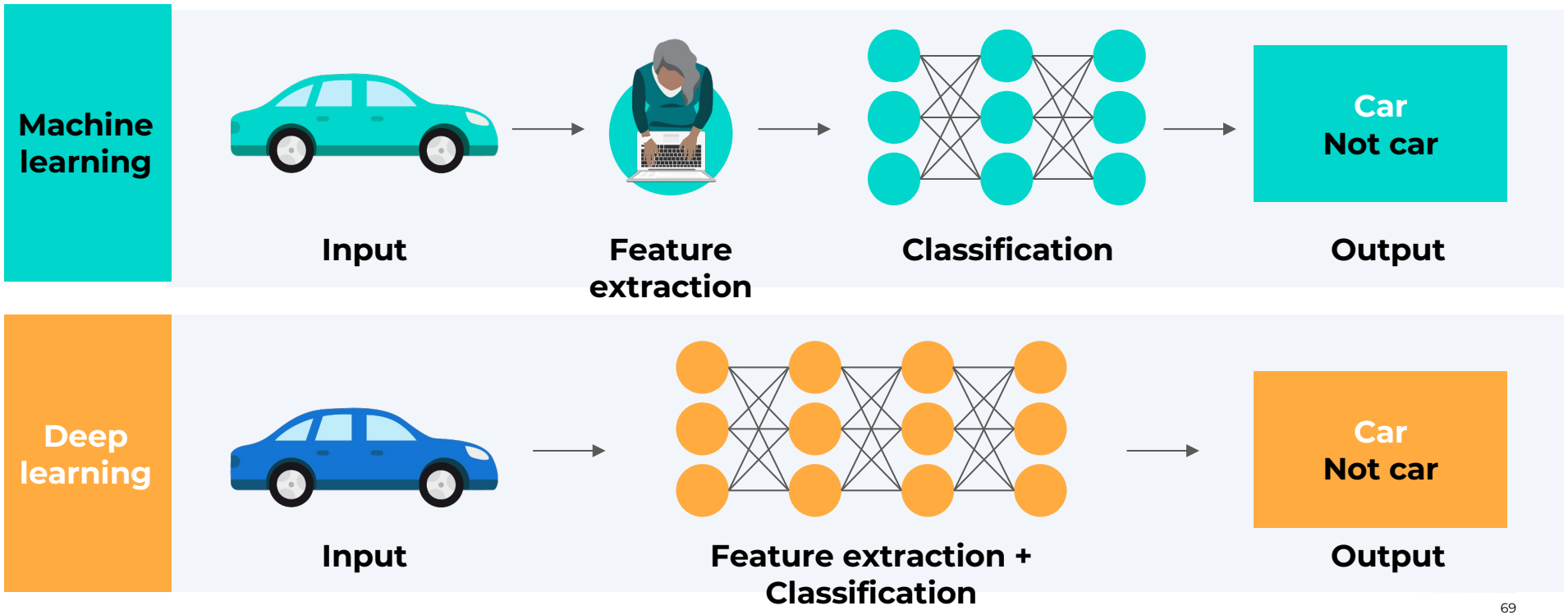
A recurrent neural network framework, LSTM was proposed by **Jürgen Schmidhuber** and **Sepp Hochreiter**.

2000s

Transformers were introduced. Followed by **GANs**, **VAEs**, and **Autoregressive** models which pushed the boundaries of **Generative AI**.

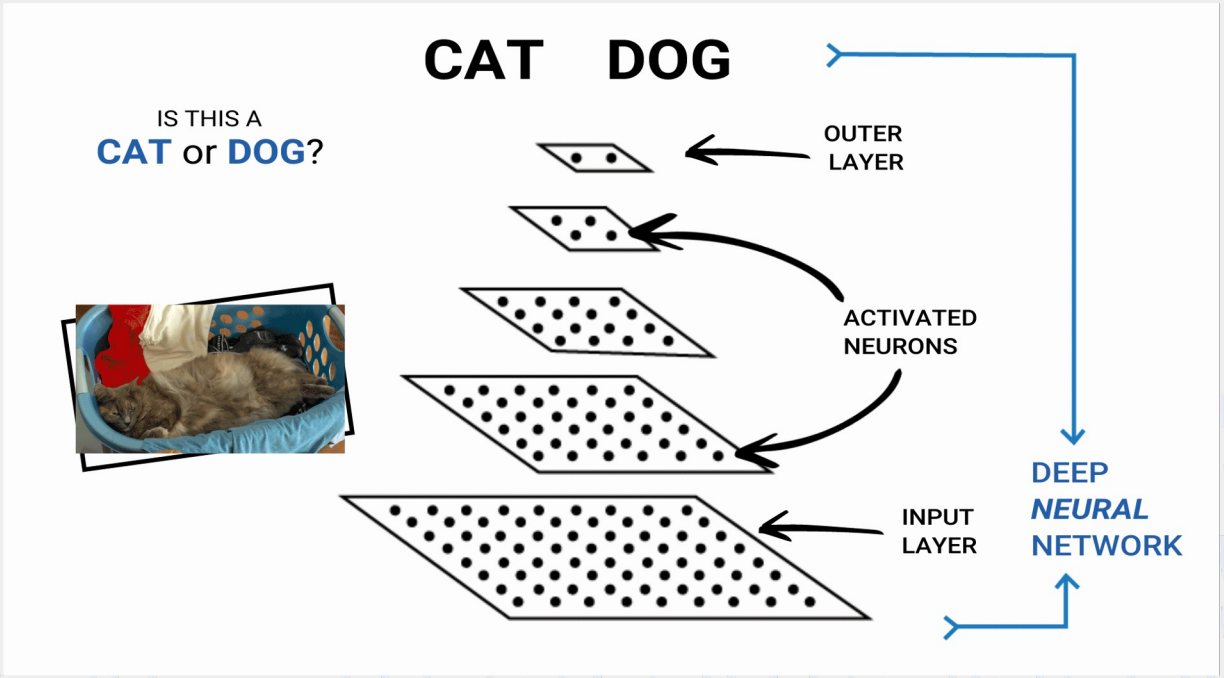
Machine Learning vs. Deep Learning

Deep neural networks are much more effective than traditional machine-learning approaches at discovering nonlinear relationships among data.



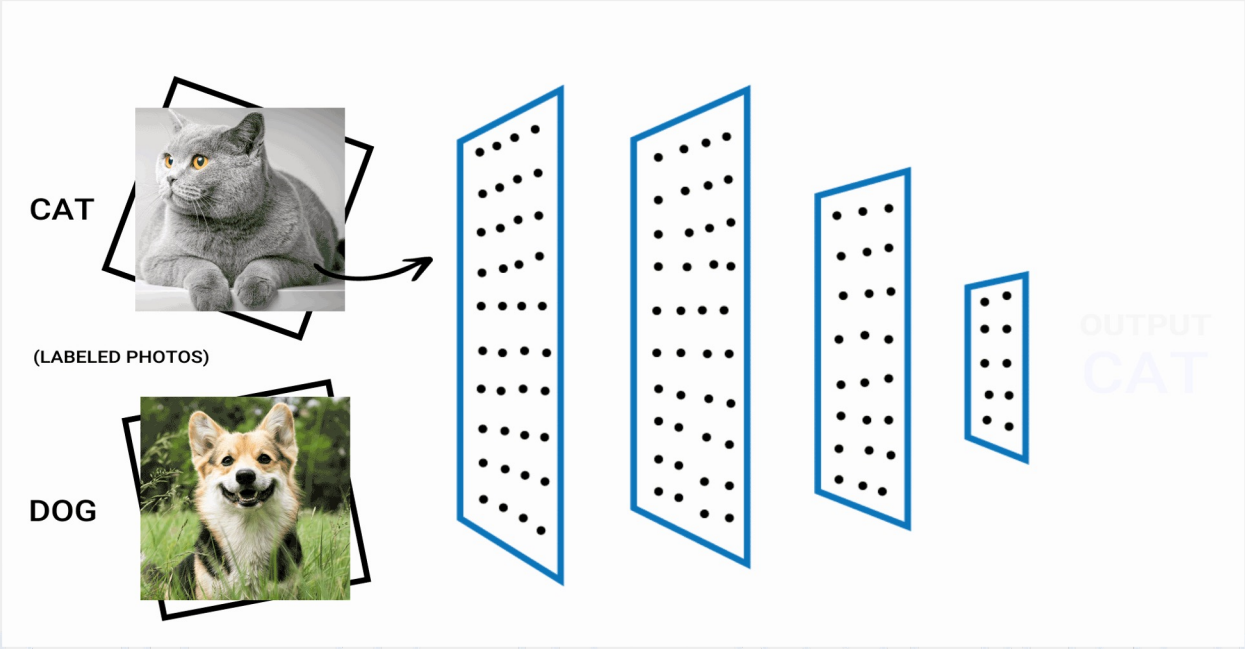
Neural Networks

Neural networks calculate the weights of various input data and pass them to the next layer of neurons. This process continues until the data reaches the output layer, which makes the final decision on the predicted category or numerical value of an instance.



Neural Networks

While definitions vary, we can consider neural networks with more than one hidden layer to be deep learning models. The decreasing cost and greater availability of computing power has increased our ability to create and use these models.

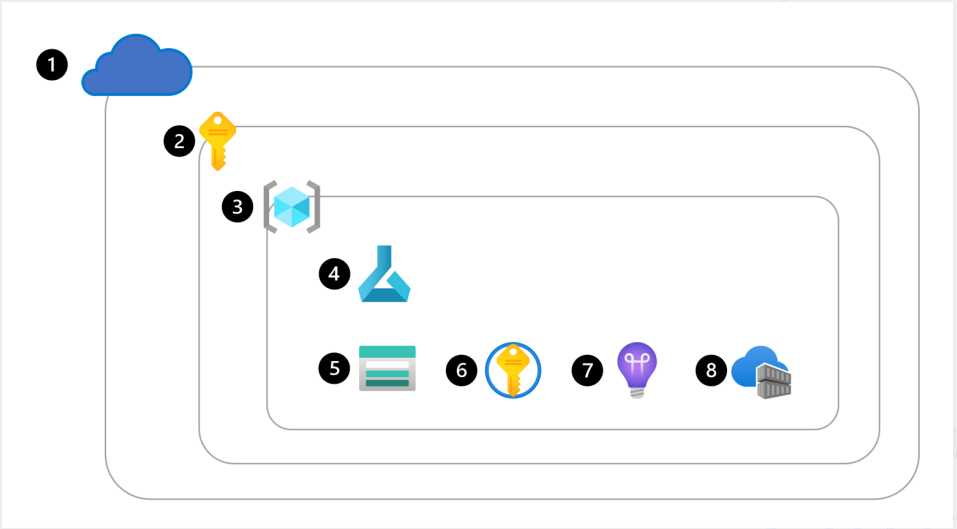


Azure AI and Machine Learning Services

https://azure.microsoft.com/en-us/products

Azure ML Services

- When a **workspace** is provisioned, Azure will automatically create other Azure resources within the same resource group to support the workspace:
- **Azure Storage Account**: To store files and notebooks used in the workspace, and to store metadata of jobs and models.
- **Azure Key Vault**: To securely manage secrets such as authentication keys and credentials used by the workspace.
- **Application Insights**: To monitor predictive services in the workspace.
- **Azure Container Registry**: Created when needed to store images for Azure Machine Learning environments.



Azure ML Compute Instances

A compute instance is a fully managed cloud-based workstation optimized for your machine learning development environment.

- **Compute instances**: Similar to a virtual machine in the cloud, managed by the workspace. Ideal to use as a development environment to run (Jupyter) notebooks.
- **Compute clusters**: On-demand clusters of CPU or GPU compute nodes in the cloud, managed by the workspace. Ideal to use for production workloads as they automatically scale to your needs.
- **Kubernetes clusters**: Allows you to create or attach an Azure Kubernetes Service (AKS) cluster. Ideal to deploy trained machine learning models in production scenarios.
- **Attached computes**: Allows you to attach other Azure compute resources to the workspace, like Azure Databricks or Synapse Spark pools.
- **Serverless compute**: A fully managed, on-demand compute you can use for training jobs.

Datastores

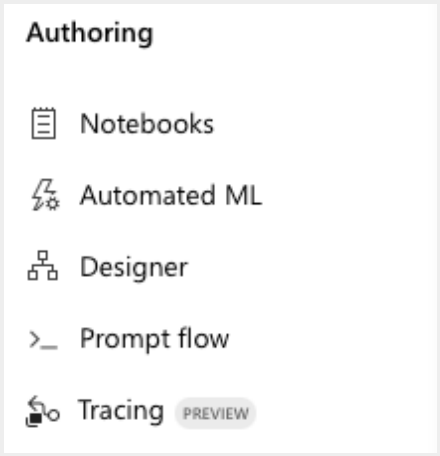
The workspace doesn't store any data itself. Instead, all data is stored in **datastores**, which are references to Azure data services.

- workspaceartifactstore: Connects to the azureml container of the Azure Storage account created with the workspace. Used to store compute and experiment logs when running jobs.
- workspaceworkingdirectory: Connects to the file share of the Azure Storage account created with the workspace used by the **Notebooks** section of the studio. Whenever you upload files or folders to access from a compute instance, it's uploaded to this file share.
- workspaceblobstore: Connects to the Blob Storage of the Azure Storage account created with the workspace. Specifically the azureml-blobstore-... container. Set as the default datastore, which means that whenever you create a data asset and upload data, it's stored in this container.
- workspacefilestore: Connects to the file share of the Azure Storage account created with the workspace. Specifically the azureml-filestore-... file share.

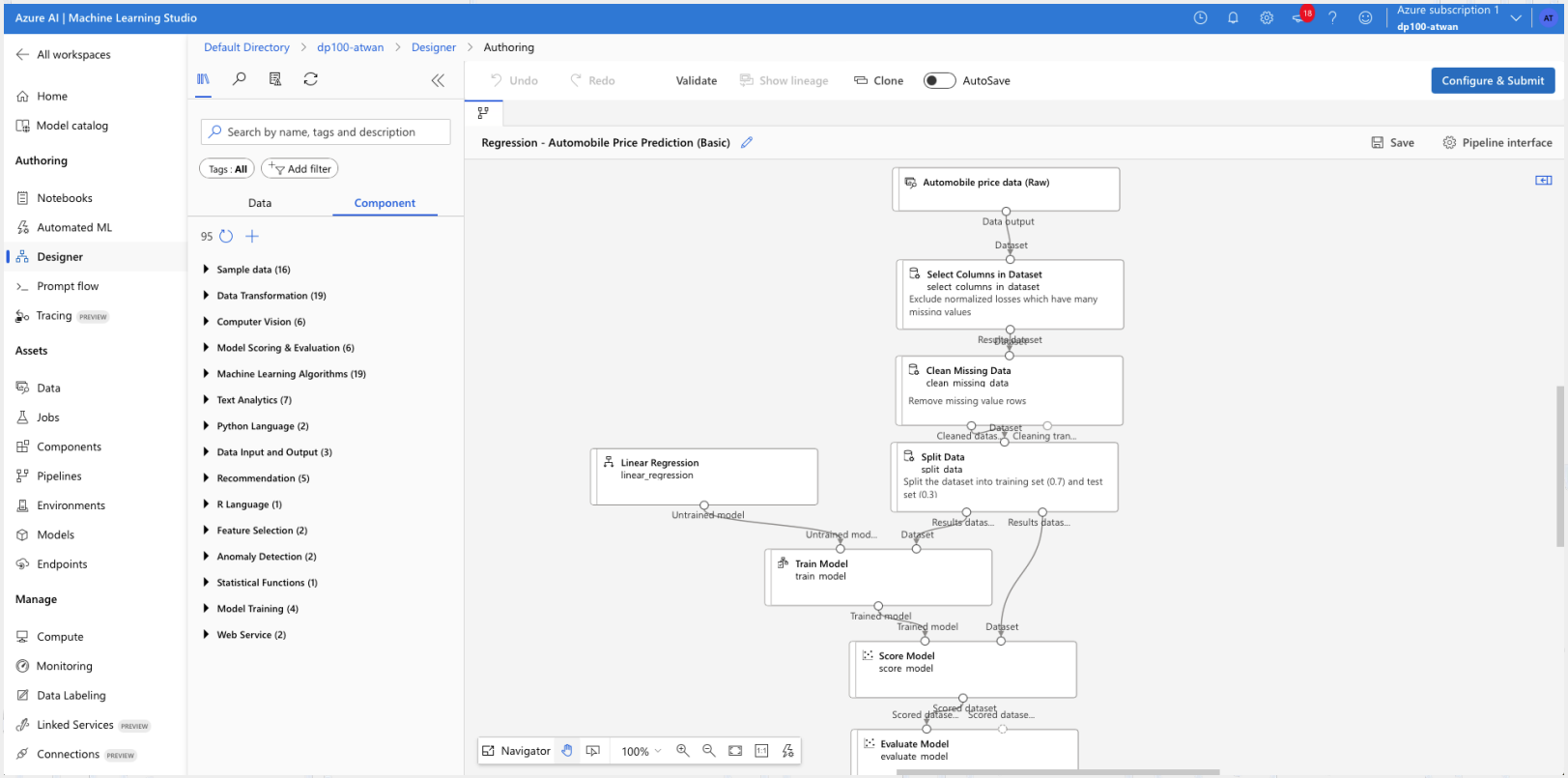
ML Assets

- Models
- Environments
- Data
- Components

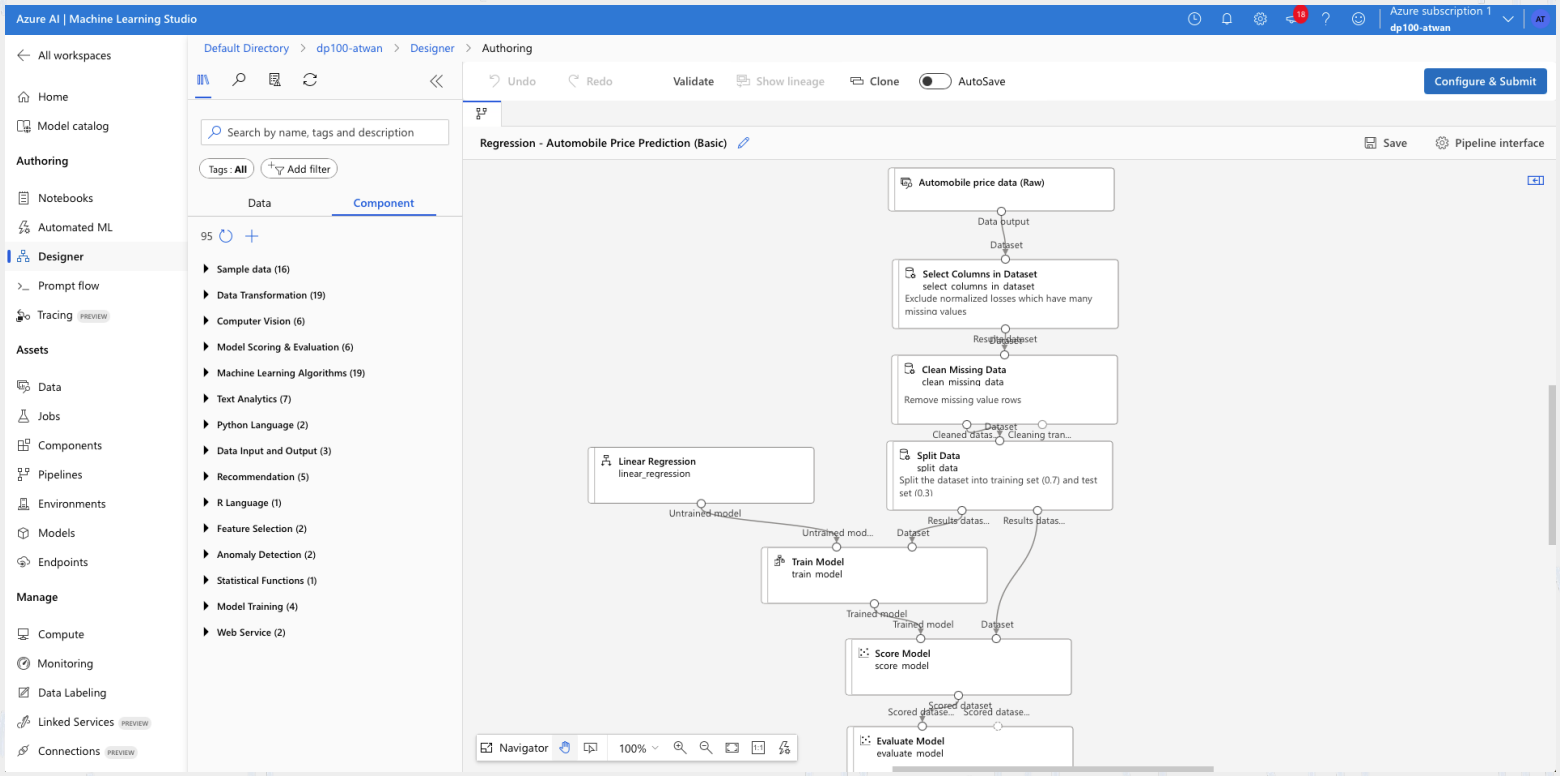
Authoring within ML Workspace



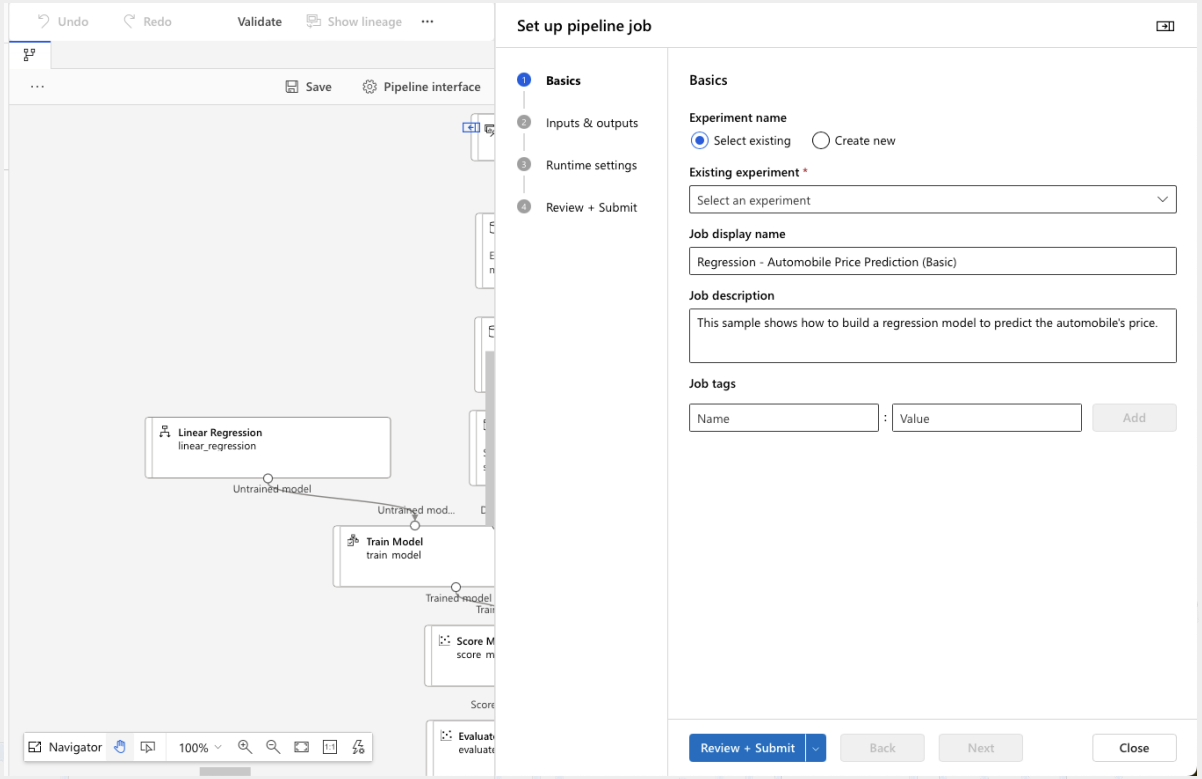
Designer



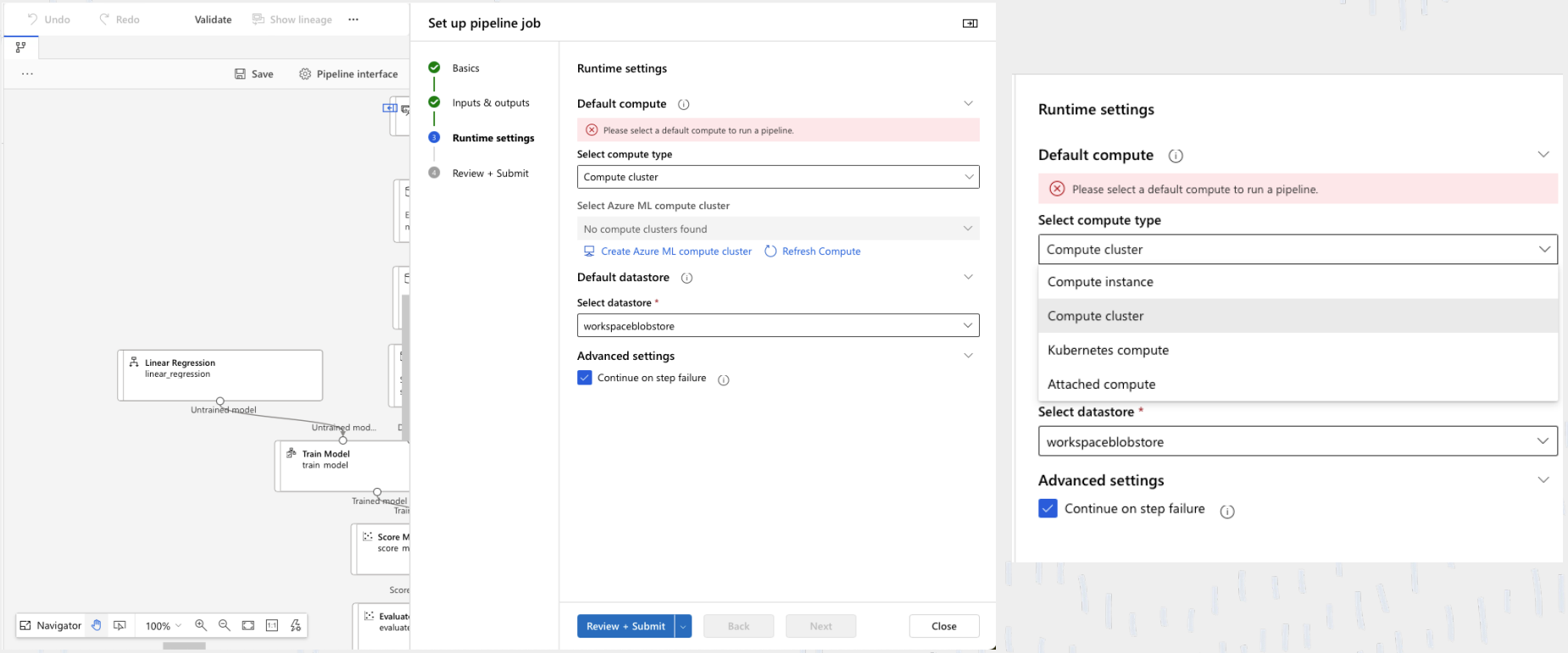
Designer



Designer

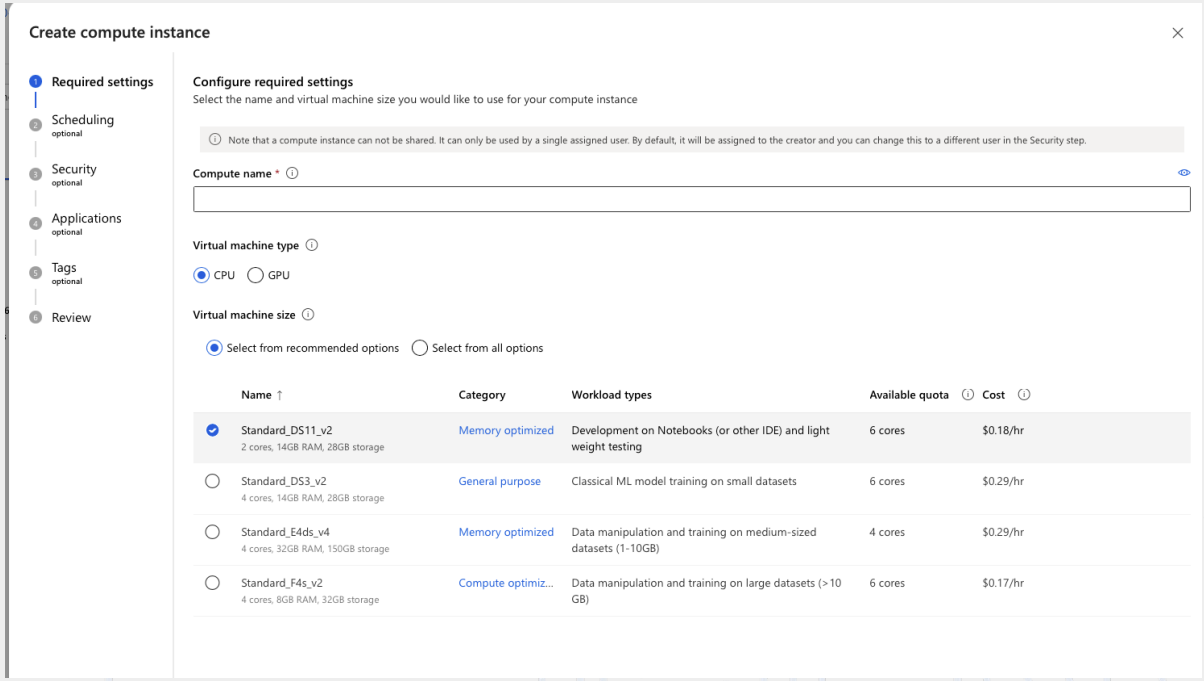


Designer – A Compute Cluster



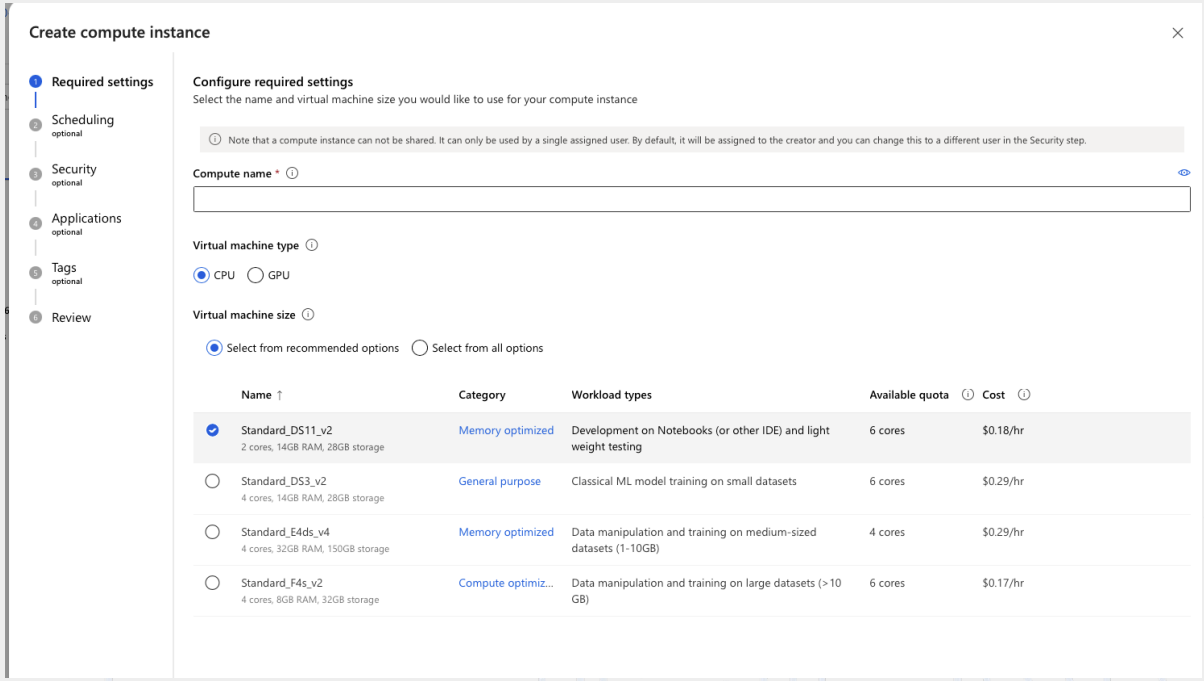
What is a Computer Instance in ML Workspace

- A compute instance is a fully managed cloud-based workstation optimized for your machine learning development environment.



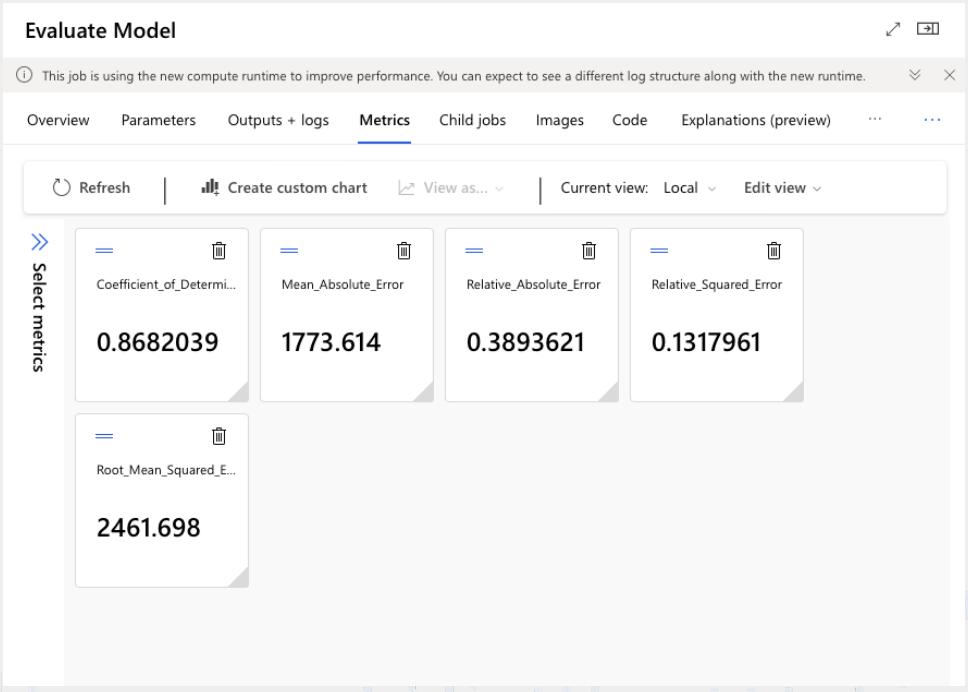
What is a Computer Instance in ML Workspace

- A compute instance is a fully managed cloud-based workstation optimized for your machine learning development environment.



What is a Computer Instance in ML Workspace

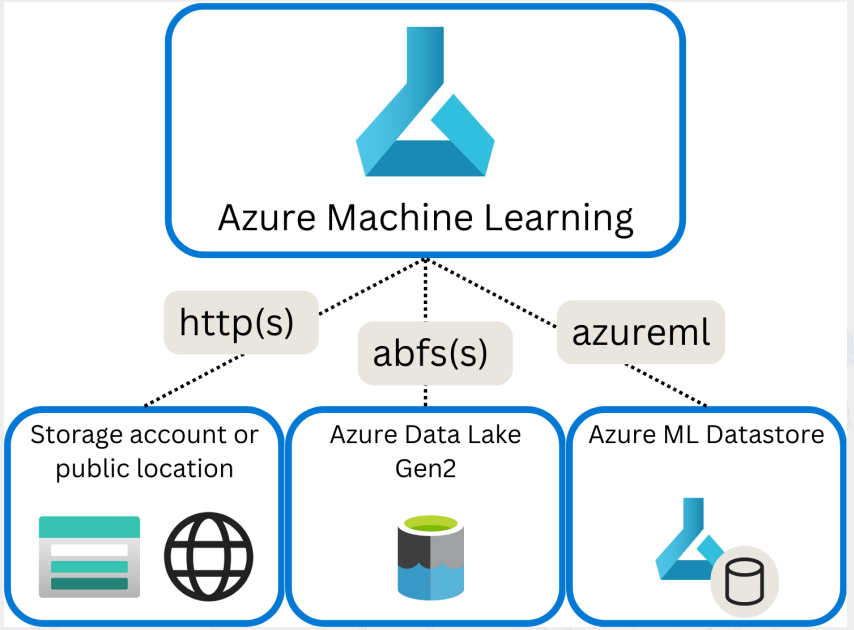
- A compute instance is a fully managed cloud-based workstation optimized for your machine learning development environment.



Data in Azure ML Workspace

To find and access data in Azure Machine Learning, you can use **Uniform Resource Identifiers** (**URIs**).

- http(s): Use for data stores publicly or privately in an Azure Blob Storage or publicly available http(s) location.
- abfs(s): Use for data stores in an Azure Data Lake Storage Gen 2.
- azureml: Use for data stored in a datastore.



Thank you!

If you have any additional questions, please ask!