Google Cloud

Partner Certification Academy

Professional Machine Learning Engineer

The information in this presentation is classified:

Google confidential & proprietary

⚠️ This presentation is shared with you under NDA.

- Do **not** record or take screenshots of this presentation.
- Do **not** share or otherwise distribute the information in this presentation with anyone **inside** or **outside** of your organization.

Thank you!

Google Cloud

Source Materials

Some of this program's content has been sourced from the following resources:

- Google Cloud certification site
- Google Cloud documentation
- Google Cloud console
- Google Cloud courses and workshops
- Google Cloud white papers
- Google Cloud Blog
- Google Cloud YouTube channel
- Google Cloud samples
- Google codelabs
- Google Cloud partner-exclusive resources

Google Cloud

Google Cloud Skills Boost for Partners

- Professional Machine Learning Engineer Certification
- Cloud Skills Boost for Partners Professional Machine Learning Engineer Learning Path
- Partner Learning Services Instructor-Led PMLE Curriculum

Google Cloud Partner Advantage

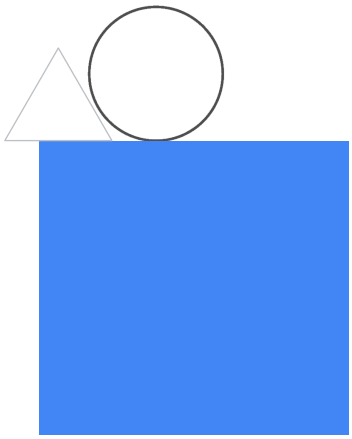
- Best practices for implementing machine learning on Google Cloud
- Artificial Intelligence
- End-to-End MLOps Go-to-Market Kit

Session Logistics

- When you have a question, please:
  - Click the Raise hand button in Google Meet.
  - Or add your question to the Q&A section of Google Meet.
  - Please note that answers may be deferred until the end of the session.
- These slides are available in the Student Lecture section of your Qwiklabs classroom.
- The session is **not recorded.**
- Google Meet does not have persistent chat.
  - If you get disconnected, you will lose the chat history.
  - Please copy any important URLs to a local text file as they appear in the chat.

Google Cloud Partner Learning Programs

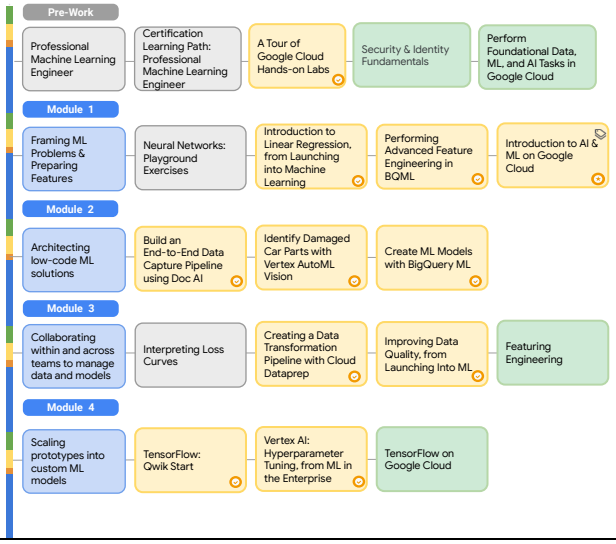
- Partner Certification Academy
- Partner Delivery Readiness Index (DRI)
- Cloud Skills Boost for Partners
- Partner Advantage



PARTNER CERTIFICATION ACADEMY

Professional Machine Learning Engineer

On-demand Course
Hands-on Skill Badge
Hands-on Lab
Resource
Live Virtual Workshop
required to earn exam voucher



A Professional Machine Learning Engineer builds, evaluates, productionizes, and optimizes ML models by using Google Cloud technologies and knowledge of proven models and techniques. The ML Engineer:

- handles large, complex datasets and creates repeatable, reusable code.
- considers responsible AI and fairness throughout the ML model development process, and collaborates closely with other job roles to ensure long-term success of ML-based applications.
- has strong programming skills and experience with data platforms and distributed data processing tools.
- is proficient in the areas of model architecture, data and ML pipeline creation, and metrics interpretation.
- is familiar with foundational concepts of MLOps, application development, infrastructure management, data engineering, and data governance.
- makes ML accessible and enables teams across the organization.

By training, retraining, deploying, scheduling, monitoring, and improving models, the ML Engineer designs and creates scalable, performant solutions.

Recommended candidate:

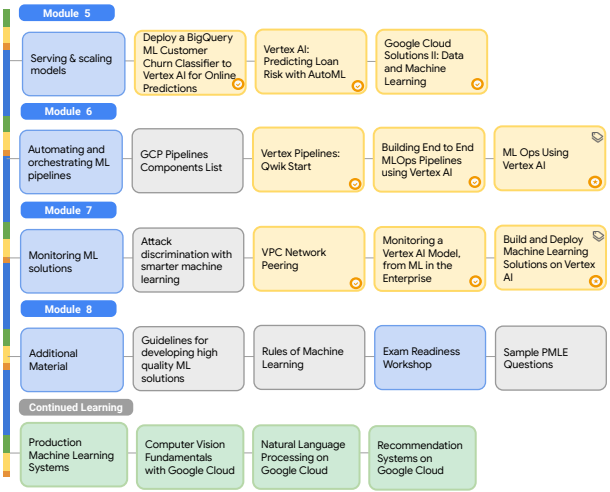
- Has in-depth experience setting up cloud environments for an organization
- Has experience deploying services and solutions based on business requirements

Google Cloud

PARTNER CERTIFICATION ACADEMY

Professional Machine Learning Engineer

On-demand Course
Hands-on Skill Badge
Hands-on Lab
Resource
Live Virtual Workshop
required to earn exam voucher



A Professional Machine Learning Engineer builds, evaluates, productionizes, and optimizes ML models by using Google Cloud technologies and knowledge of proven models and techniques. The ML Engineer:

- handles large, complex datasets and creates repeatable, reusable code.
- considers responsible AI and fairness throughout the ML model development process, and collaborates closely with other job roles to ensure long-term success of ML-based applications.
- has strong programming skills and experience with data platforms and distributed data processing tools.
- is proficient in the areas of model architecture, data and ML pipeline creation, and metrics interpretation.
- is familiar with foundational concepts of MLOps, application development, infrastructure management, data engineering, and data governance.
- makes ML accessible and enables teams across the organization.

By training, retraining, deploying, scheduling, monitoring, and improving models, the ML Engineer designs and creates scalable, performant solutions.

Recommended candidate:

- Has in-depth experience setting up cloud environments for an organization
- Has experience deploying services and solutions based on business requirements

Google Cloud

Learner Commitment

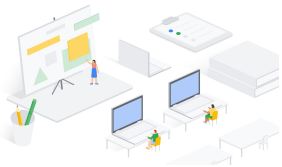
Each week, learners are to complete the learning path's course content, Cloud Skills Boost for Partner Quests/Challenge Labs and material that the mentor has recommended that will support learning.

- **Workshop Day:** Meet for the cohort's weekly 'general session'. **(≅ 2 hours)**
- **During the week:** Complete the week's course, perform hands-on labs, review any additional material suggested material for the week. **(≅ 8 - 16 hours)**

➔ **Important:** Learners must allocate time between each weekly session to study and familiarize themselves with any prerequisite knowledge they may lack. It is also recommended that learners complete the next week's course prior to the scheduled workshop.

Path to Service Excellence

Certification



Advanced Solutions Training

Delivery Readiness Index

Certification is just one step on your professional journey. Google Cloud also offers our partners access to advanced solutions training, and a new quality-focused program called Delivery Readiness Index (DRI) to help you achieve service excellence with your customers.

Benchmark your skills with DRI

DRI helps to benchmark partner proficiency and capability at any point during the customer journey however should be used primarily as a lead measure to predict and prepare for partner delivery success.

DRI assesses and analyzes Partner Consultant GCP proficiency by creating a DRI Profile inclusive of their GCP knowledge, skills, and experience.

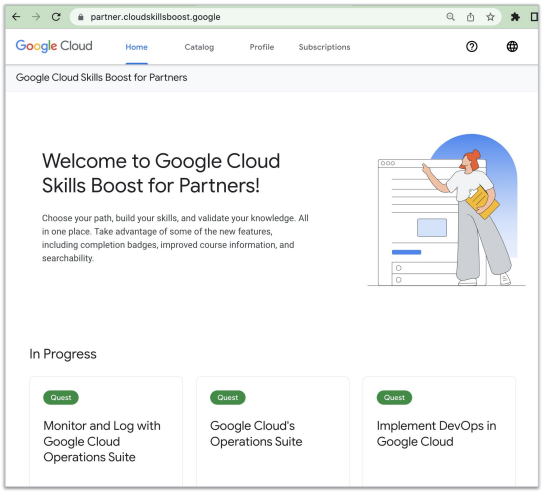
With the DRI insights, we can prescriptively advise the partner project team on the ground and bridge niche capability gaps.

DRI also takes action. For partner consultants, DRI generates a tailored L&D plan that prescribes personalized learning, training, and skill development to build GCP proficiency.

Google Cloud Skills Boost for Partners

https://partner.cloudskillsboost.google/

- On-demand course content
- Hands-on labs
- Skill Badges
- **FREE** to Google Cloud Partners!

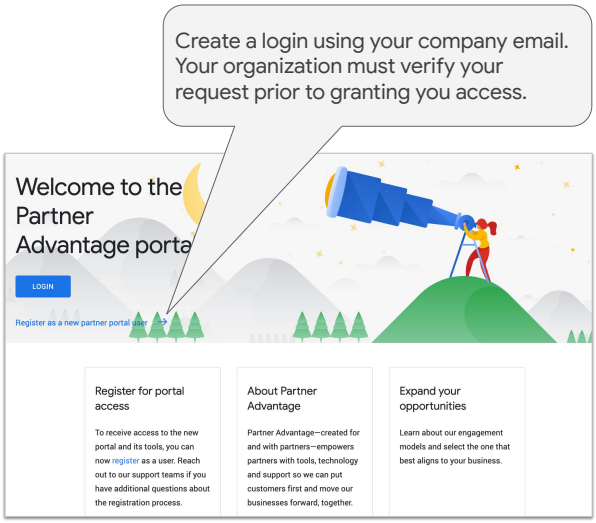


Proprietary + Confidential

Google Cloud

Partner Advantage

- Resources for Google Cloud partner organizations:
  - Recent announcements
  - Solutions/role-based training
  - Live/pre-recorded webinars on various topics
    - Partner Advantage Live Webinars
- Complements the certification self-study material presented on Google Cloud Skills Boost for Partners
- Helpful Links:
  - Getting started on Partner Advantage
  - Join Partner Advantage
  - Get help accessing Partner Advantage



https://www.partneradvantage.goog

Google Cloud

The getting started link:
https://support.google.com/googlecloud/topic/9198654#zippy=

Note the top section, **"Getting Started & User Guides"** and two key documents → Direct Partners to this if they need to enroll into Partner Advantage

1. Logging in to the Partner Advantage Portal - Quick Reference Guide
2. Enrolling in the Partner Advantage Program - Quick Reference Guide

Focus from this point on:

Some context on enrolling in PA:
Access to Partner Portal is given in 2 ways

- Partner Admin Led: Partner Administrator at Partner Company can set up users
- User Led: User can go through Self Registration
  - https://www.partneradvantage.goog/GCPPRM/s/partneradvantageportallogin?language=en_US
  - Or directly to the User Registration Form, https://www.partneradvantage.goog/GCPPRM/s/partnerselfregistration?language=en_US

Please Note

- After a user self-registers, they receive an email that essentially states:

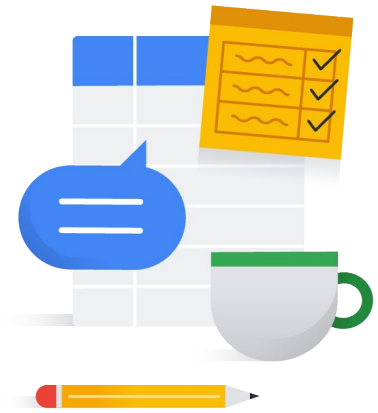
    - “Hi {Partner Name}, you are one step away from joining the Google Cloud Partner Advantage Community. Please click to continue with the user registration process. See you in the cloud, The Partner Advantage Team
- Once registered, they can access limited content until their **Partner Administrator approves the user**
- Their Partner Administrator also receive an email notifying them that a member of their organization has registered themselves on their organization’s Google Cloud Partner Advantage account.
    - It also states that this user has limited access to the portal
    - They are provided instructions on how to review and provision the appropriate access for the user that has registered
- Once their admin approves the user, they receive an email that states:
    - Hi {User Name}, Your Partner Administrator has updated your access to the Google Cloud Partner Advantage portal. You have been granted edit access to additional account information on the portal on behalf of your organization to help build your business. For additional access needs, please work with your Partner Administrator. See you in the cloud, The Partner Advantage Team

The net takeaway is, on the Support Page (the first link on this slide) Google Cloud Partner Advantage Support, there’s a section **“Issue accessing Partner Advantage Portal? Click here for troubleshooting steps”**

- The source of their issue can be related to the different items shown
- Additionally, there’s a Partner Administrator / Partner Adminstrator Team at their partner organization that has to approve their access.. Until that step is completed, they will have access issues/limitation. They will need to identify who this person or team is at their organization

Program issues or concerns?

- Problems with **accessing** Cloud Skills Boost for Partners
  - cloud-partner-training@google.com
- Problems with **a lab** (locked out, etc.)
  - support@qwiklabs.com
- Problems with accessing Partner Advantage
  - https://support.google.com/googlecloud/topic/9198654



- Problems with accessing **Cloud Skills Boost for Partners**
  - cloud-partner-training@google.com
- Problems with a **lab** (locked out, etc.)
  - support@qwiklab.com
- Problems with accessing **Partner Advantage**
  - https://support.google.com/googlecloud/topic/9198654

Module 1

Framing ML Problems & Preparing Features

Module Agenda

01 ML Problem Framing

02 A Sample of Machine Learning Algorithms

03 Introduction to Feature Engineering

ML Problem Framing

To build a machine learning model

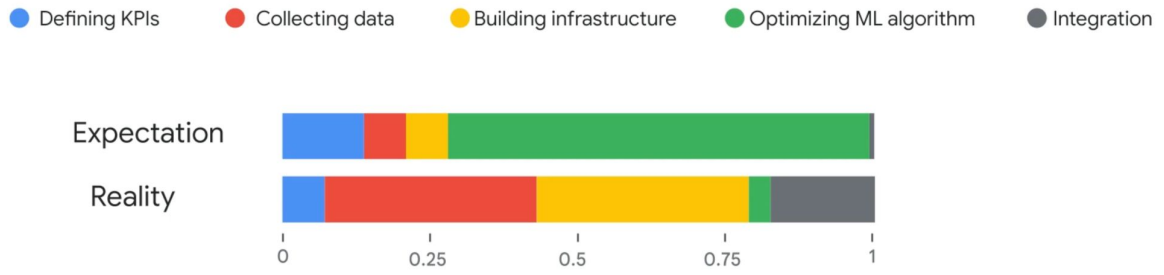
1	2	3	4	5	6
Identify business problem	Develop hypothesis	Acquire + explore data	Build a model	Train the model	Apply and scale

Source: Machine Learning on Google Cloud v3.0

Specifically you must collect information about your use case. Build a model to make sense of it. Train that model repeatedly and at scale with more data and then apply that model into a solution to make it useful.

Data quality, along with code quality, is key to success.

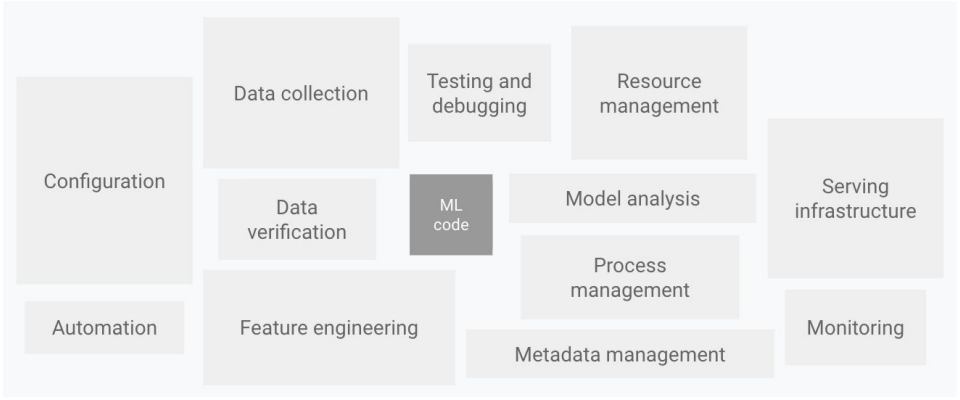
What does working on an ML project look like?



This helpful graphic urges us to consider how ML teams actually spend their time at Google.

From “How Google Does Machine Learning” Course

What goes into a complete ML project?



Additionally, only a small percentage of the code in an ML system is actually the ML code.

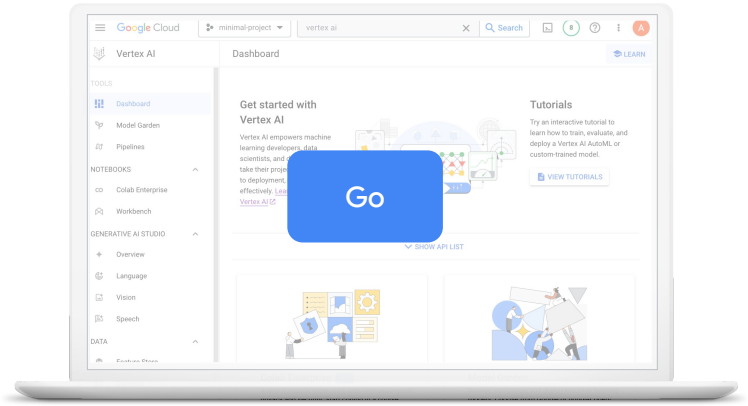
From: [MLOps: Continuous delivery and automation pipelines in machine learning]

Vertex AI

Vertex AI provides tools to help with all phases of the Machine Learning model data gathering, training, and serving process.

Demo

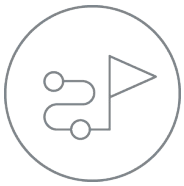
Let's take a tour of Vertex AI in the GCP Console.



Demo the Vertex AI Environment:
https://console.cloud.google.com/vertex-ai

Recommendation: Work through the Console from top to bottom on the left nav to describe Vertex AI products.

When faced with a problem for which we might build an ML solution, how do we begin?



To understand the problem, perform the following tasks:

- State the goal for the product you are developing or refactoring.
- Determine whether the goal is best solved using, predictive ML, generative AI, or a non-ML solution.
- Verify you have the data required to train a model if you're using a predictive ML approach.

https://developers.google.com/machine-learning/problem-framing/problem

After verifying that your problem is best solved using either a predictive ML or a generative AI approach, you're ready to frame your problem in ML terms by completing the following tasks:

- Define the ideal outcome and the model's goal.
- Identify the model's output.
- Define success metrics.

https://developers.google.com/machine-learning/problem-framing/ml-framing

After verifying that your problem is best solved using either a predictive ML or a generative AI approach, you're ready to frame your problem in ML terms. You frame a problem in ML terms by completing the following tasks:

Define the ideal outcome and the model's goal.
Identify the model's output.
Define success metrics.

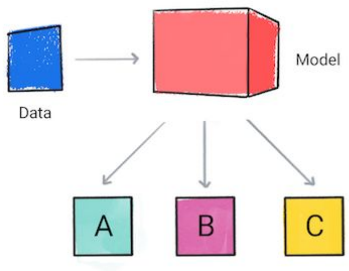
Defining the Task

Classification

A classification model predicts what category the input data belongs to, for example, whether an input should be classified as A, B, or C.

We can classify examples from:

- Tabular data
- Images
- Text documents
- Audio
- Video
- Medical imaging data
- etc.



ML Framing:

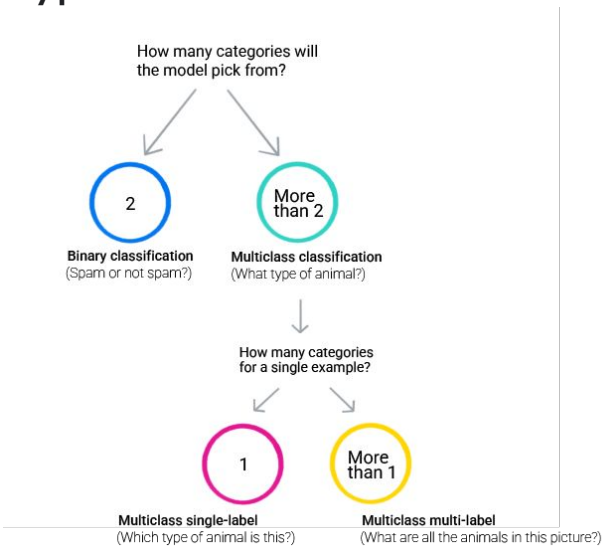
https://developers.google.com/machine-learning/problem-framing/ml-framing

In summary, depending on the problem you are trying to solve, the data you have, explainability, etc. will determine which machine learning methods you use to find a solution.

Your data isn't labeled? You won't be able to use supervised learning then, and will have to resort to clustering algorithms to discover interesting properties of the data.

Your data is labeled and the label is dog breed, which is a discrete quantity since there are a finite number of dog breeds? You should use a classification algorithm. If instead the label is dog weight, which is a continuous quantity, you should use a regression algorithm. The label, again, is the thing that you are trying to predict. In supervised learning, you have some data with the correct answers.

Classification Types



ML Framing:

https://developers.google.com/machine-learning/problem-framing/ml-framing

In summary, depending on the problem you are trying to solve, the data you have, explainability, etc. will determine which machine learning methods you use to find a solution.

Your data isn't labeled? You won't be able to use supervised learning then, and will have to resort to clustering algorithms to discover interesting properties of the data.

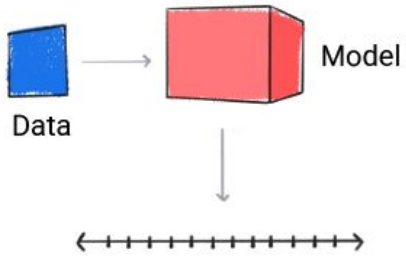
Your data is labeled and the label is dog breed, which is a discrete quantity since there are a finite number of dog breeds? You should use a classification algorithm. If instead the label is dog weight, which is a continuous quantity, you should use a regression algorithm. The label, again, is the thing that you are trying to predict. In supervised learning, you have some data with the correct answers.

Regression

A regression model predicts where to place the input data on a number line.

Like with classification, we can make numeric predictions based on any type of input data:

- Tabular data
- Images
- Text documents
- Audio
- Video
- Medical imaging data
- etc.



ML Framing:

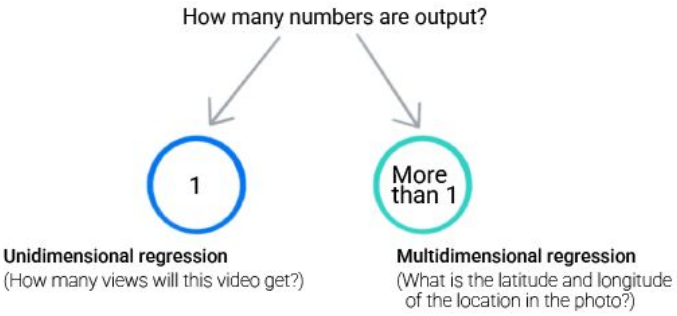
https://developers.google.com/machine-learning/problem-framing/ml-framing

In summary, depending on the problem you are trying to solve, the data you have, explainability, etc. will determine which machine learning methods you use to find a solution.

Your data isn't labeled? You won't be able to use supervised learning then, and will have to resort to clustering algorithms to discover interesting properties of the data.

Your data is labeled and the label is dog breed, which is a discrete quantity since there are a finite number of dog breeds? You should use a classification algorithm. If instead the label is dog weight, which is a continuous quantity, you should use a regression algorithm. The label, again, is the thing that you are trying to predict. In supervised learning, you have some data with the correct answers.

Regression Types



ML Framing:

https://developers.google.com/machine-learning/problem-framing/ml-framing

In summary, depending on the problem you are trying to solve, the data you have, explainability, etc. will determine which machine learning methods you use to find a solution.

Your data isn't labeled? You won't be able to use supervised learning then, and will have to resort to clustering algorithms to discover interesting properties of the data.

Your data is labeled and the label is dog breed, which is a discrete quantity since there are a finite number of dog breeds? You should use a classification algorithm. If instead the label is dog weight, which is a continuous quantity, you should use a regression algorithm. The label, again, is the thing that you are trying to predict. In supervised learning, you have some data with the correct answers.

Know Your Computer Vision Tasks

dog

Image Classification

Object Detection

Image Segmentation

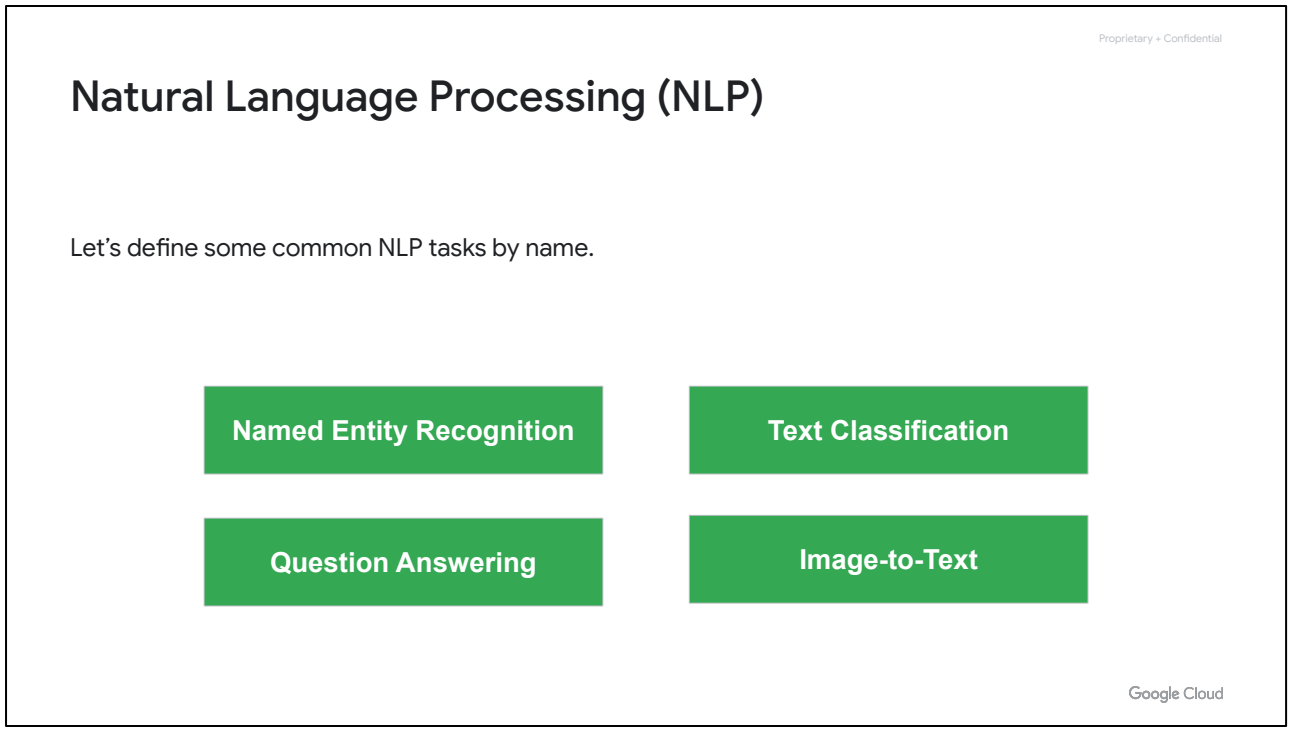
It's important to know the industry term for the task you are trying to accomplish.

One good place to explore tasks being researched is at Papers With Code.

Another is Hugging Face.

It's important from the beginning because they have different data labeling levels of effort, etc.

Image credit: "Image Classification vs. Object Detection vs. Image Segmentation" on Medium.



Named Entity Recognition:

Question Answering:

Text Classification: Text Classification is the task of assigning a label or class to a given text. Some use cases are sentiment analysis, natural language inference, and assessing grammatical correctness.

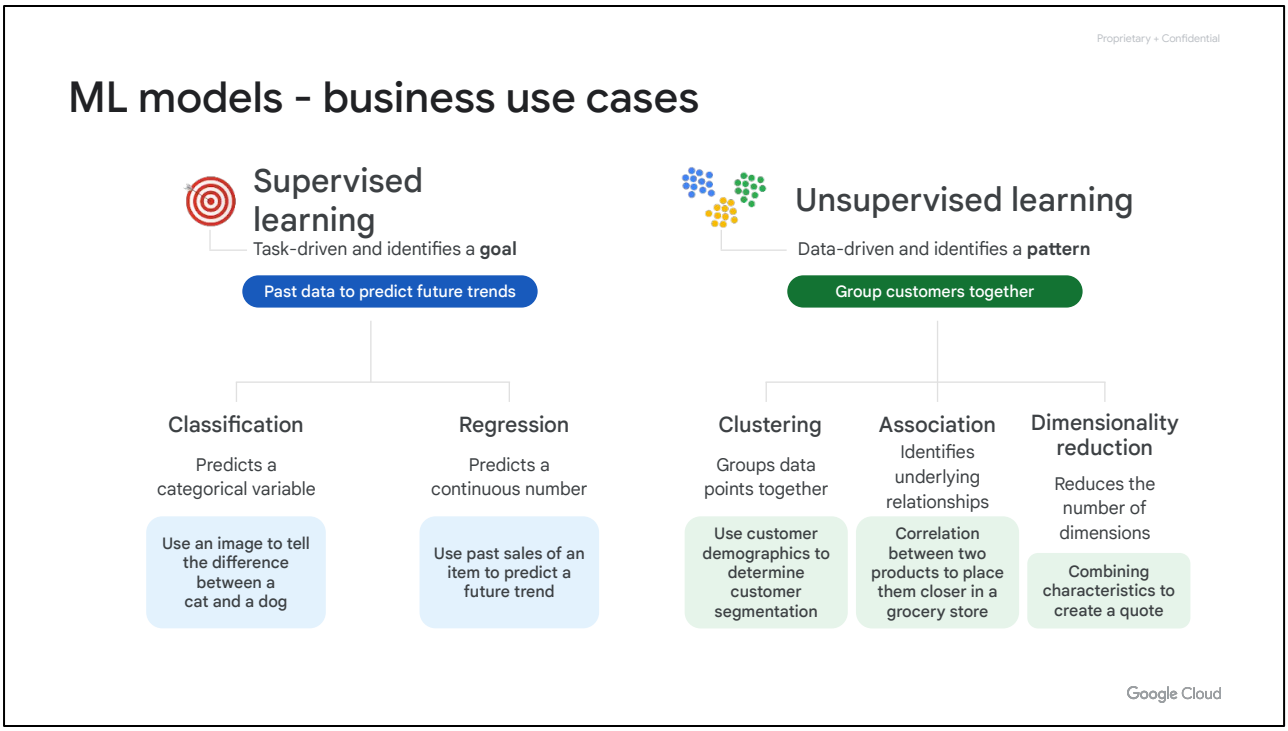
Image-to-Text: Image to text models output a text from a given image. Image captioning or optical character recognition can be considered as the most common applications of image to text.

Question Answering: Question Answering models can retrieve the answer to a question from a given text, which is useful for searching for an answer in a document.

Named Entity Recognition: NER models can identify specific entities in a text, such as dates, individuals and places. (A subset of token classification, which could also be used for part-of-speech tagging.)

Source for definitions: [Tasks - Hugging Face]

Types of Learning



Source: Google Cloud BigData and Machine Learning Fundamentals v3.0.0

So, what's the difference between supervised and unsupervised learning?

- **Supervised learning** is task-driven and identifies a goal.
- **Unsupervised learning**, however, is data-driven and identifies a pattern.

An easy way to distinguish between the two is that **supervised** learning provides each data point with a **label**, or an answer, while unsupervised does not.

For example, if we were given sales data from an online retailer, we could use supervised learning to predict the sales trend for the next couple of months and use unsupervised learning to group customers together based on common characteristics.

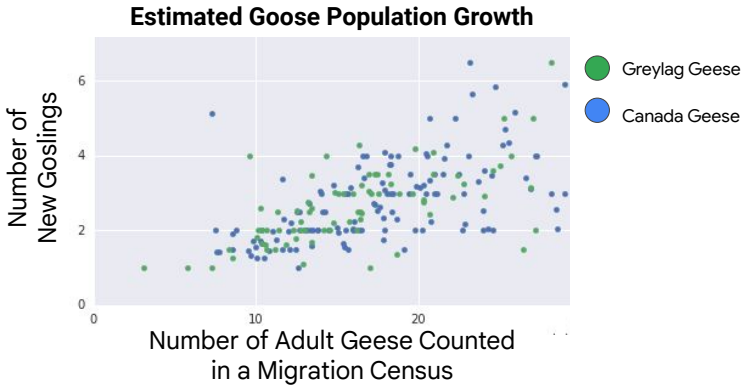
There are two major types of ***supervised* learning**:

- The first is **classification**, which predicts a categorical variable, like using an image to tell the difference between a cat and a dog.
- The second type is a **regression** model, which predicts a continuous number, like using past sales of an item to predict a future trend.

And then there are three major types of ***unsupervised* learning**:

- The first is **clustering**, which groups together data points with similar characteristics and assigns them to "clusters", like using customer demographics, to determine customer segmentation.
- The second is **association**, which identifies underlying relationships, like a correlation between two products to place them closer together in a grocery store for a promotion.
- And the third is **dimensionality reduction**, which reduces the number of dimensions, or features, in a dataset to improve the efficiency of a model. For example, combining customer characteristics like age, driving violation history, or car type, to create an insurance quote. If too many dimensions are included, it can consume too many compute resources, which might make the model inefficient.

In supervised learning we learn from past, labeled examples to predict future values.



In this chapter though, we'll be focused on supervised machine learning problems, like this one. The critical difference is that with supervised learning, we have some notion of a "label," or one characteristic of each data point that we care about a lot.

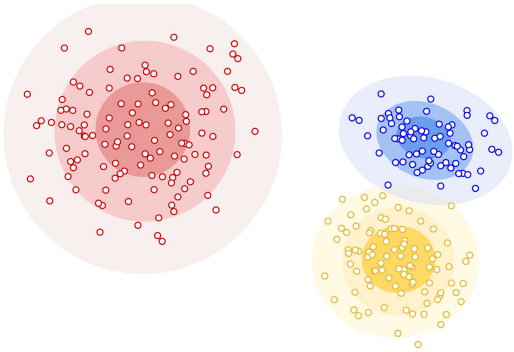
Typically, this is something you know about in historical data, but you don't know in real time. You know other things, which you call predictors, and you want to use those predictors to predict the thing you don't know.

For example, let's say you are tracking populations of migratory birds. You have historical data on how many new eggs were laid based on population counts from previous censuses. Now, you are conducting the current census. You want to know how many new eggs to prepare for this year.

In the historical data, the number of babies are labeled next to the population count. You create a model to predict the number of new babies.

You can create a different model for each species you are tracking.

In unsupervised learning, we are looking to find useful patterns in unlabeled data.

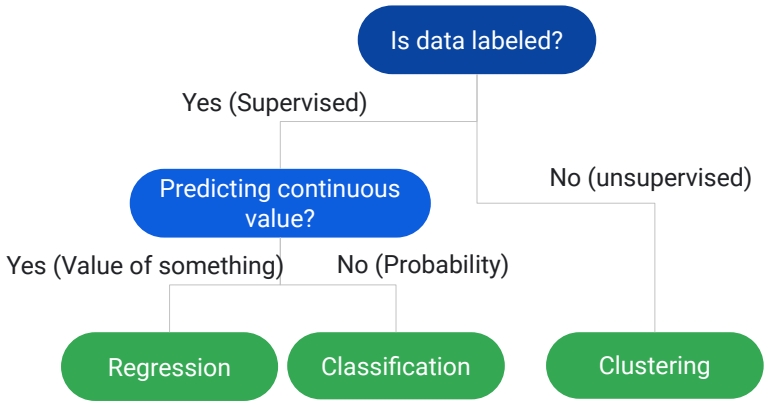


In unsupervised learning, the data does not have labels, but it does still have features. Some common kinds of problems where we might deploy an unsupervised learning solution include:

- Anomaly detection: Unsupervised clustering can process large datasets (for example server logs) and discover data points that are atypical in a dataset (in the server example, this might represent a cybersecurity event or a failing server).
- Recommendation engines: Using association rules, unsupervised machine learning can help explore transactional data to discover patterns or trends that can be used to drive personalized recommendations for online retailers.
- Customer segmentation: Unsupervised learning is also commonly used to generate buyer persona profiles by clustering customers' common traits or purchasing behaviors. These profiles can then be used to guide marketing and other business strategies.
- Fraud detection: Unsupervised learning is useful for anomaly detection, revealing unusual data points in datasets. These insights can help uncover events or behaviors that deviate from normal patterns in the data, revealing fraudulent transactions or unusual behavior like bot activity.
- Natural language processing (NLP): Unsupervised learning is commonly used for various NLP applications, such as categorizing articles in news sections, text translation and classification, or speech recognition in conversational interfaces.
- Genetic research

Source:
https://cloud.google.com/discover/what-is-unsupervised-learning#:~:text=Imagine%20that%20you%20have%20a,temperature%20or%20similar%20weather%20patterns.
Image from:
https://developers.google.com/machine-learning/clustering/clustering-algorithms

The right algorithm depends on whether your data is (or can be) labeled and what output you would like.



In summary, depending on the problem you are trying to solve, the data you have, explainability, etc. will determine which machine learning methods you use to find a solution.

Your data isn't labeled? You won't be able to use supervised learning then, and will have to resort to clustering algorithms to discover interesting properties of the data.

A good example of a clustering problem might be "how should we size our clothing line?" If we have measurements of hundreds or thousands of people, we could cluster those measurements to determine clothing sizes that would give most people a best-fit option.

Your data is labeled and the label is dog breed, which is a discrete quantity since there are a finite number of dog breeds? You should use a classification algorithm. If instead the label is dog weight, which is a continuous quantity, you should use a regression algorithm. The label, again, is the thing that you are trying to predict. In supervised learning, you have some data with the correct answers.

Regression and classification are tasks solved by supervised ML model types.

	total_bill	tip	sex	smoker	day	time
2	16.99	1.01	Female	No	Sun	Dinner
3	10.34	1.66	Male	No	Sun	Dinner
4	21.01	3.5	Male	No	Sun	Dinner
5	23.68	3.31	Male	No	Sun	Dinner
6	24.59	3.61	Female	No	Sun	Dinner
7	25.29	4.71	Male	No	Sun	Dinner
8	8.77	2	Male	No	Sun	Dinner
9	26.88	3.12	Male	No	Sun	Dinner

Continuous value
Regression Model
Predict the tip amount

Probability
Classification Model
Predict the gender of the customer

Within supervised ML, two types of problems are regression and classification. To explain them, let's dive a little deeper into this data.

In this dataset of tips, an example dataset that comes with the Python package *seaborn*, each row has many characteristics, such as total bill, tip, and sex. In machine learning, we call each row an "example." You'll choose one of the columns as the characteristic we want to predict, called the "label," and you'll choose a set of the other columns, which are called the "features."

- In model option 1, you want to predict the tip amount, therefore the column *tip* is your label. You can use one, all, or any number of the other columns as my features to predict the tip. This will be a regression model because *tip* is a continuous label.
- In model option 2, you want to predict the sex of the customer, therefore the column *sex* is the label. Once again, you will use some set of the rest of the columns as your features, to try and predict the customer's sex. This will be a classification model because our label *sex* has a discrete number of values or classes.

Quiz

Supervised Learning

Imagine you are in banking and you are creating an ML model for detecting if transactions are fraudulent or not. Is this classification or regression and why?

A. Regression, categorical label
B. Regression, continuous label
C. Classification, categorical label
D. Classification, continuous label

Question: Imagine you are in banking and you are creating an ML model for detecting if transactions are fraudulent or not. Is this classification or regression and why?

Quiz

Supervised Learning

Imagine you are in banking and you are creating an ML model for detecting if transactions are fraudulent or not. Is this classification or regression and why?

A. Regression, categorical label
B. Regression, continuous label
C. Classification, categorical label
D. Classification, continuous label

Answer: The correct answer is classification, categorical label.

This is a binary classification problem because there are two possible classes for each transaction: fraudulent or not fraudulent. In practice, you may actually have a third class, uncertain. This way depending on your classification threshold it could send any cases that it can't firmly place into the fraudulent or not fraudulent buckets to a human to have a closer look. It is often good practice to have a human in the loop when performing machine learning.

You can eliminate Regression, categorical label and Classification, continuous label because the model types have the opposite label type than they should.

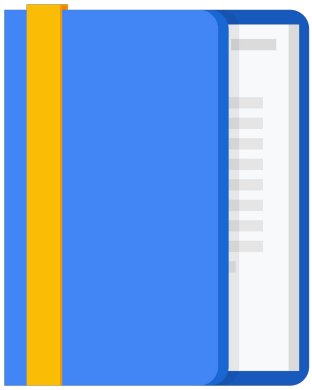
Regression, continuous label at least is a correct pairing however it is incorrect because this is a classification problem so you would not use regression. You could, also create a regression model such as predicting the number of fraudulent transactions, fraudulent transaction amounts, etc.

A Sample of Machine Learning Algorithms

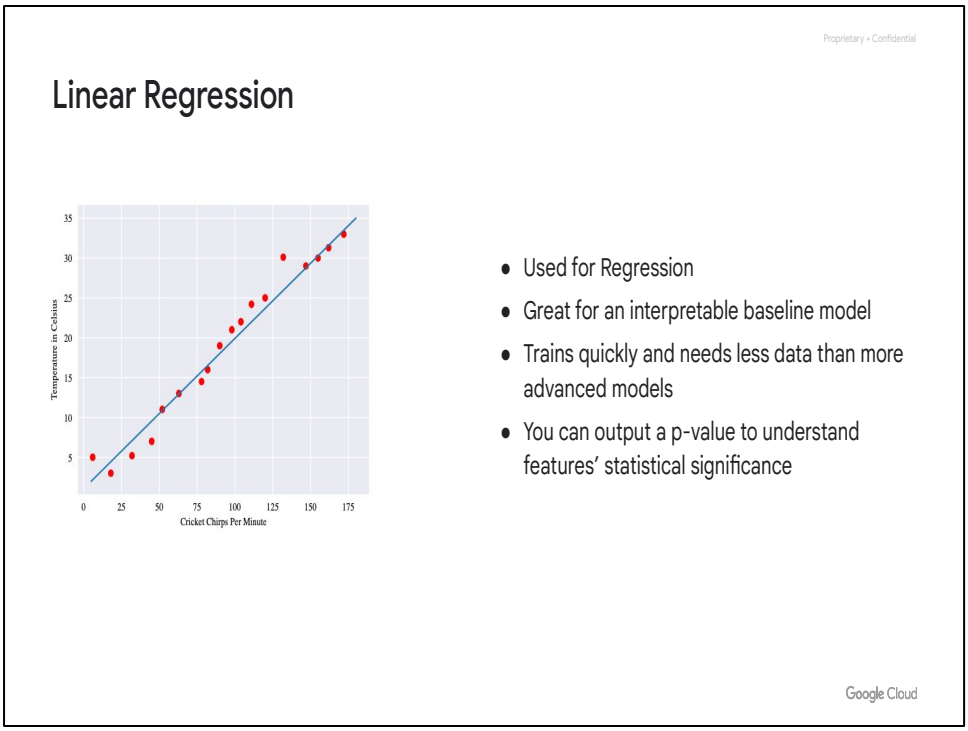
Algorithms Section Agenda

Supervised Learning Algorithms

Unsupervised Learning Algorithms



These are the modules that make up this course.



Linear regression can be thought of as fitting a straight line to a set of data points, where the goal is to find the line that best represents the relationship between the variables.

Use Case: predict a student's exam score based on the number of hours they studied. The number of hours studied would be the independent variable, while the exam score would be the dependent variable.

To create a linear regression model for this use case, we would gather data on the number of hours each student studied and their corresponding exam score. We can then use this data to find the best-fit line that represents the relationship between the two variables. We can use this model to make predictions about a student's exam score based on the number of hours they studied.

For example, if a student studies for 5 hours, the linear regression model can predict that their exam score would be around 80%.

Linear regression was “invented” for predicting the movement of planets and the size of pea pods based on their parents.

LAB: Introduction to Linear Regression:
https://partner.cloudskillsboost.google/course_sessions/2980558/labs/377707

Image Source:
https://developers.google.com/machine-learning/crash-course/descending-into-ml/linear-regression
Good reference on R-Squared:
https://statisticsbyjim.com/regression/interpret-r-squared-regression/

Linear Regression

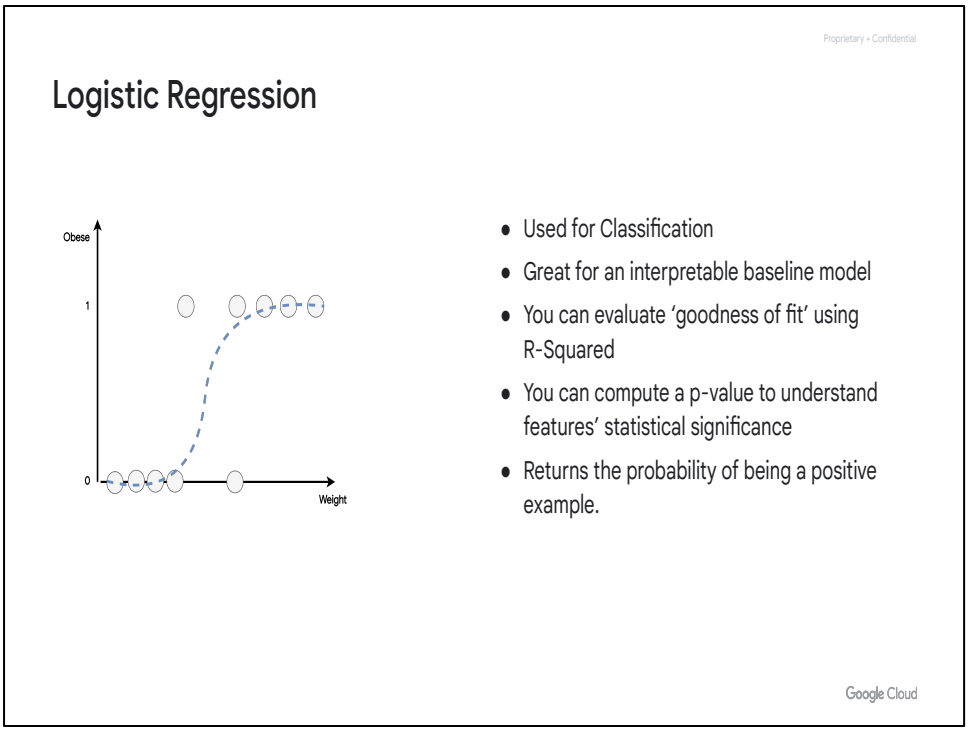
$$y' = b + w_1 x_1 + w_2 x_2 + w_3 x_3$$

bias

feature 2 weight

feature 2 value

A linear regression model yields an interpretable model consisting of a bias term and a weight for each feature variable.



The goal of logistic regression is to find the best-fit curve or line that represents the relationship between the independent variables and the probability of the binary outcome. Unlike linear regression, logistic regression uses a sigmoid function to model the relationship between the variables, which results in an S-shaped curve

Use Case 2:
Another simple use case for logistic regression is to predict whether a customer will churn or not based on their demographic and usage information. Churn refers to customers who discontinue their relationship with a company or service. The dependent variable in this case is binary, where 1 indicates churn and 0 indicates no churn.

LAB: Basic Introduction to Logistic Regression:
https://partner.cloudskillsboost.google/catalog_lab/4062

old:
https://partner.cloudskillsboost.google/catalog_lab/2876

explain a simple use case for using linear regression over logistic regression and vice versa:

A simple use case for using linear regression over logistic regression would be to predict a continuous outcome, such as the price of a house, Linear regression is

appropriate when the dependent variable is continuous and has a linear relationship with the independent variables. In this case, we would use linear regression to find the best-fit line that represents the relationship between the independent variables and the continuous outcome.

On the other hand, a simple use case for using logistic regression over linear regression would be to predict a binary outcome, such as whether a customer will purchase a product or not, based on one or more independent variables. Logistic regression is appropriate when the dependent variable is binary and has a non-linear relationship with the independent variables. In this case, we would use logistic regression to find the best-fit curve that represents the relationship between the independent variables and the probability of the binary outcome.

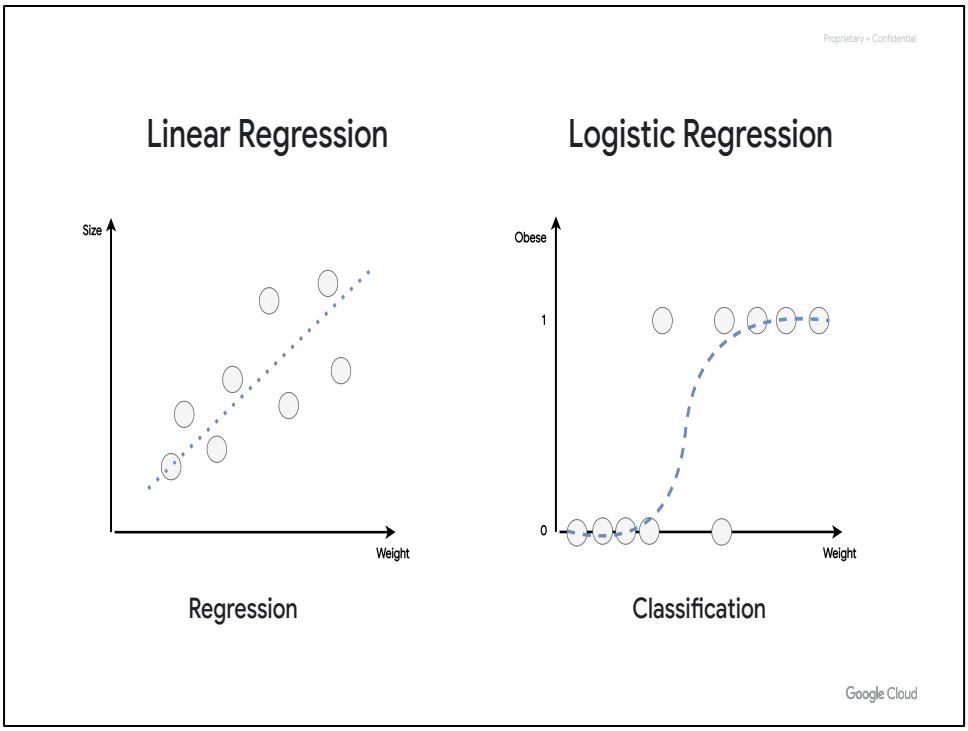
In summary, the choice between linear regression and logistic regression depends on the nature of the dependent variable and the relationship it has with the independent variables. If the dependent variable is continuous and has a linear relationship with the independent variables, we use linear regression. If the dependent variable is binary and has a non-linear relationship with the independent variables, we use logistic regression.

Logistic Regression

$$y' = \frac{1}{1 + e^{-(w^T x + b)}}$$

Linear model inserted into a sigmoid function

The sigmoid function gives the plot its shape of trying to define a boundary between a “0” (not an example of this class) and a “1” (an example of this class).



A simple use case for using linear regression over logistic regression and vice versa:

A simple use case for using linear regression over logistic regression would be to predict a continuous outcome, such as the price of a house, Linear regression is appropriate when the dependent variable is continuous and has a linear relationship with the independent variables. In this case, we would use linear regression to find the best-fit line that represents the relationship between the independent variables and the continuous outcome.

On the other hand, a simple use case for using logistic regression over linear regression would be to predict a binary outcome, such as whether a customer will purchase a product or not, based on one or more independent variables. Logistic regression is appropriate when the dependent variable is binary and has a non-linear relationship with the independent variables. In this case, we would use logistic regression to find the best-fit curve that represents the relationship between the independent variables and the probability of the binary outcome.

explain a simple use case for using linear regression over logistic regression and vice versa:

Linear Regression:
A simple use case for using linear regression over logistic regression would be to predict a continuous outcome, such as the price of a house, Linear regression is appropriate when the dependent variable is continuous and has a linear relationship

with the independent variables. In this case, we would use linear regression to find the best-fit line that represents the relationship between the independent variables and the continuous outcome.

Logistic regression:
to predict a binary outcome, such as whether a customer will purchase a product or not, based on one or more independent variables. Logistic regression is appropriate when the dependent variable is binary and has a non-linear relationship with the independent variables. In this case, we would use logistic regression to find the best-fit curve that represents the relationship between the independent variables and the probability of the binary outcome.

More Scenarios:
Linear Regression: It's like trying to draw a straight line through a bunch of points on a graph. The line is supposed to be the best representation of the relationship between those points. In practical terms, you might use linear regression to predict something like the price of a house based on its size. The bigger the house, the higher the price, and this relationship can be represented by a straight line.

Logistic Regression: This one is a bit different. Instead of predicting a value like house price, logistic regression is used for classification - basically figuring out "which group does this thing belong to?" An example might be whether an email is spam or not. We can take various factors (like the presence of certain words, the time it was sent, etc.) and logistic regression helps us say, "Given all these factors, is this email likely to be spam or not?"

So, in short, linear regression is used for predicting a continuous outcome (like house prices) and logistic regression is used for binary classification (like whether an email is spam or not).

Introduction to Linear Regression:

Enroll on this Course: Launching into Machine Learning
https://partner.cloudskillsboost.google/course_templates/8

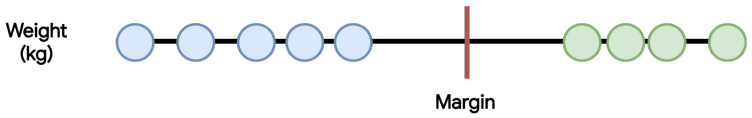
LAB: Introduction to Linear Regression:
https://partner.cloudskillsboost.google/course_sessions/2980558/labs/377707

Getting started with the built-in linear learner algorithm:
https://cloud.google.com/ai-platform/training/docs/algorithms/linear-star t

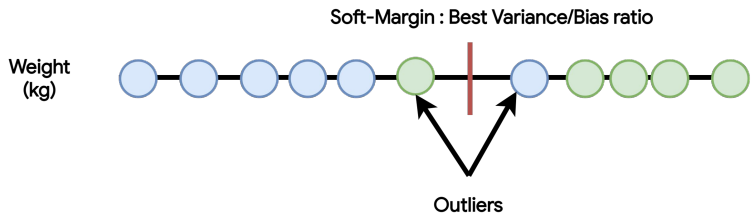
Introduction to Logistic Regression:
[LAB]: Basic Introduction to Logistic Regression:
https://partner.cloudskillsboost.google/catalog_lab/2876

Support Vector Machines (SVMs)

We want to classify people based on the weight. We can start with using the Margin.



However, Margin cannot handle outliers. We can use Cross Validation to find the Soft-Margin.



SVMs can try to find the hyperplane of best fit (regression) or best boundary (classification). They use a ‘kernel’ for an idea

From: [Unlocking the True Power of Support Vector Regression]

Unlike other Regression models that try to minimize the error between the real and predicted value, the SVR tries to fit the best line within a threshold value.

Until 2006 they were the best general purpose algorithm for machine learning.

Use Case 1:

Imagine you have a collection of apples and oranges, and you know the weight of each fruit. You want to find a way to distinguish between apples and oranges based solely on their weight.

Let's say the weights of the apples are generally lower, between 100 to 150 grams, while the weights of the oranges are generally higher, between 150 to 200 grams. There's some overlap in the weights around 150 grams, where it could either be a very heavy apple or a very light orange.

A support vector machine (SVM) in this scenario would be like a smart digital scale that not only measures the weight, but also tells you whether the fruit

placed on it is more likely to be an apple or an orange based on the weight.

In the simplest terms, an SVM is an algorithm that finds the "best" line in higher dimensions, it could be a margin (also called a hyperplane) that separates two classes of data. In this case, it would find the best weight to separate apples from oranges.

Use Case 2:

in a simple use case is to classify people as underweight, normal weight, or overweight based on their weight.

Imagine a group of people with different weights, and we want to classify them into three categories: underweight, normal weight, and overweight. SVM and SVC can be used to create a model that separates the people into these categories by finding the hyperplane that maximizes the margin between the weight groups.

The hyperplane is like a line that separates the people into the three weight categories. The margin is like the distance between the line and the nearest people from each weight category. By finding the hyperplane that maximizes the margin, SVM and SVC can create a model that is robust to noise and can generalize well to new people with different weights.

Once the model is created, it can be used to classify new people as underweight, normal weight, or overweight based on their weight. For example, if a person weighs 70 kg, the SVM or SVC model can predict that they are in the normal weight category.

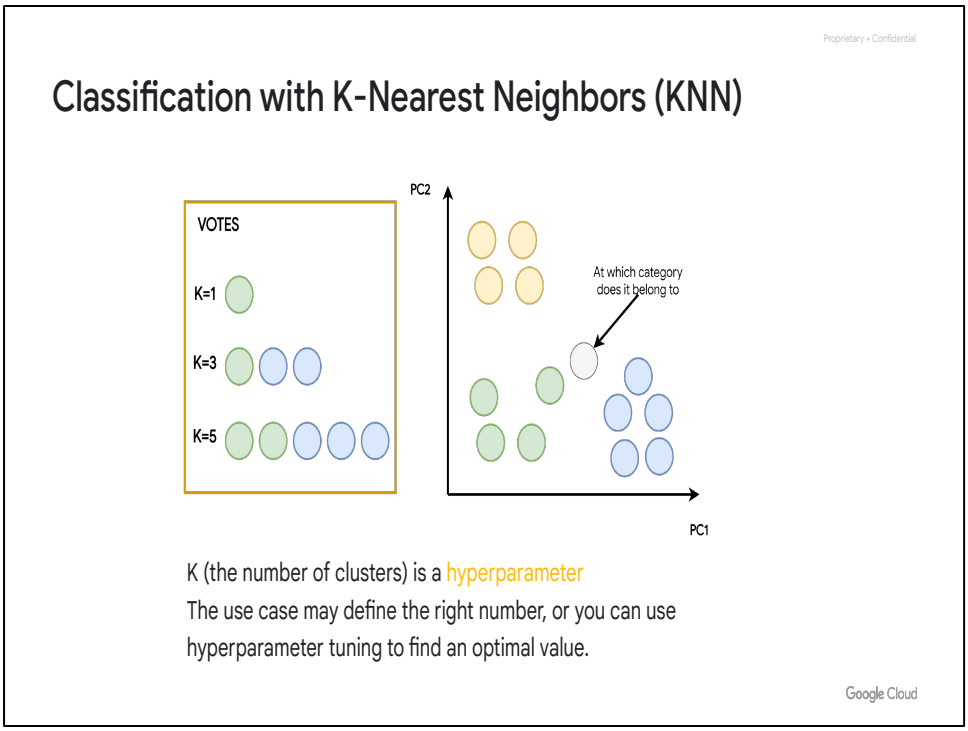
SVM and SVC can also be used in other classification tasks where the data is not linearly separable, such as identifying spam emails, detecting fraudulent transactions, or classifying images.

Colab Link:Support Vector Machines
https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.07-Support-Vector-Machines.ipynb#scrollTo=1_sOo_U5RAzB

Margin:

The trade-off between using a hard margin or a soft margin in SVM depends on the nature of the data and the degree of separation between the classes. If the data is noise-free and the classes are well-separated, a hard margin SVM can

be used to find the hyperplane that perfectly separates the classes. However, if the data is noisy or the classes overlap, a soft margin SVM can be used to find a hyperplane that separates the classes with a certain degree of error.



Use Case:

K-Nearest Neighbors

KNN is a type of machine learning algorithm used for both classification and regression analysis.

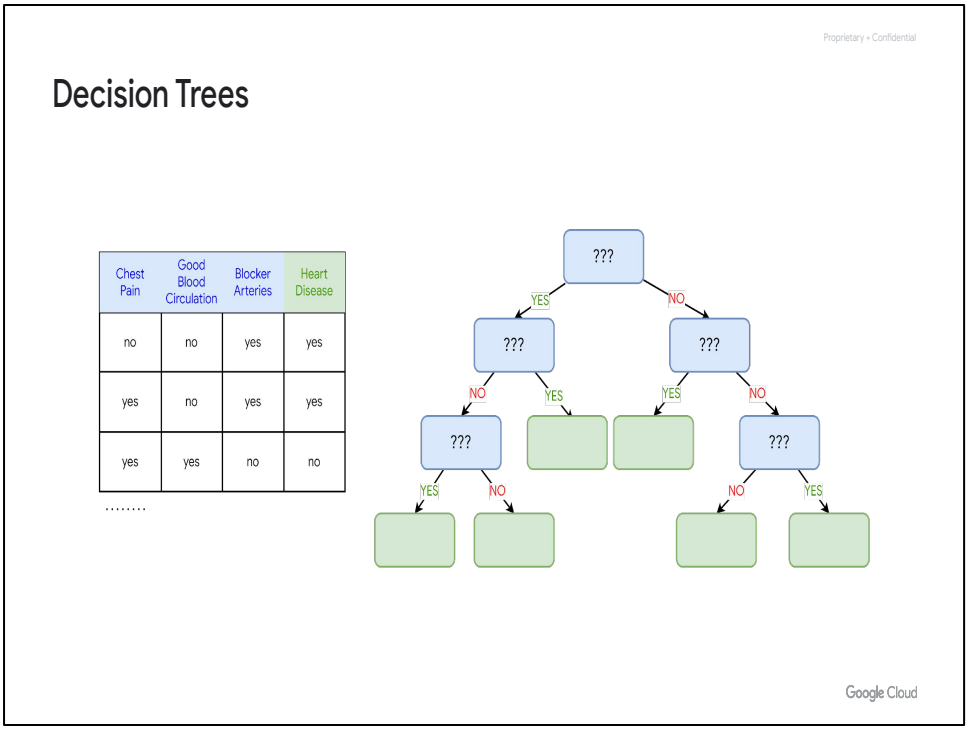
For example, let's say we have a dataset of fruits that are classified into two classes: apples and oranges. Each fruit has two features: weight and color. Now, if we have a new fruit that we don't know the class of, we can use KNN to classify the fruit based on the class of its nearest neighbors.

In KNN, the number of neighbors to consider is determined by the value of K. If we set K=3, for example, KNN will find the three nearest fruits to the new fruit based on their weight and color. The new fruit will then be classified based on the class of the majority of the three nearest fruits.

Once the KNN model is created, it can be used to classify new fruits as apples or oranges based on their weight and color. KNN can also be used in other classification tasks, such as identifying spam emails, predicting customer behavior, or diagnosing medical conditions.

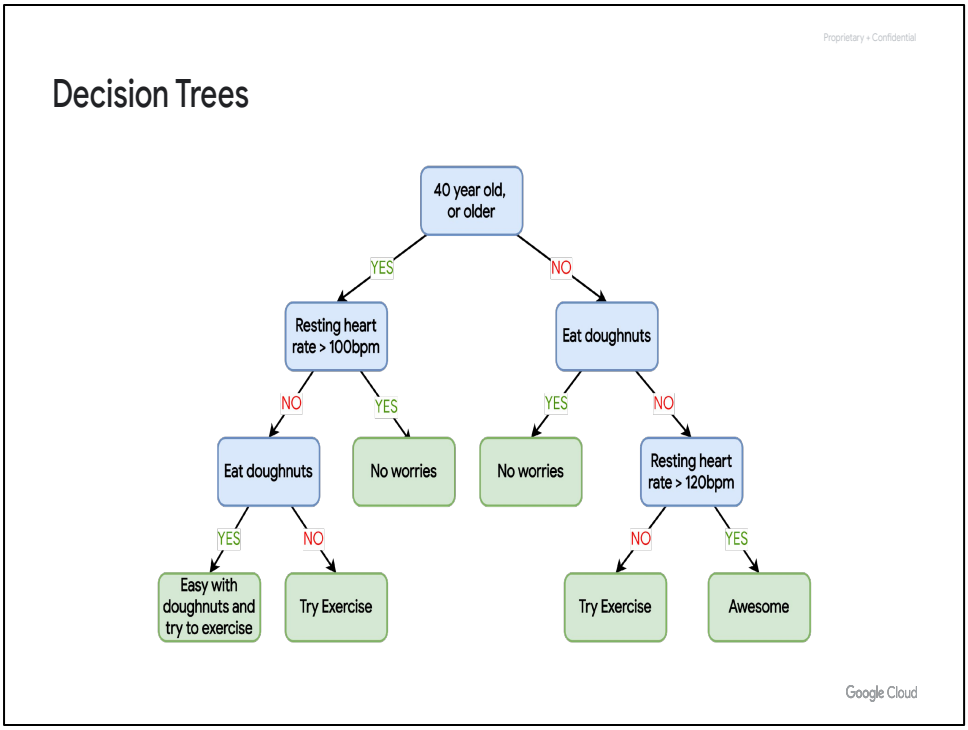
In summary, KNN is a simple and intuitive algorithm that can be used to classify data based on the class of its nearest neighbors.

Colab: KNN:
https://colab.research.google.com/github/akshayrb22/playing-with-data/blob/master/supervised_learning/KNN/KNN.ipynb#:~:text=K%2DNearest%20Neighbors&text=It%20is%20a%20lazy%20learning,anything%20about%20the%20underlying%20data.



Decision trees can also be used for regression or classification.

Decision Trees are a building process
Are Rules based

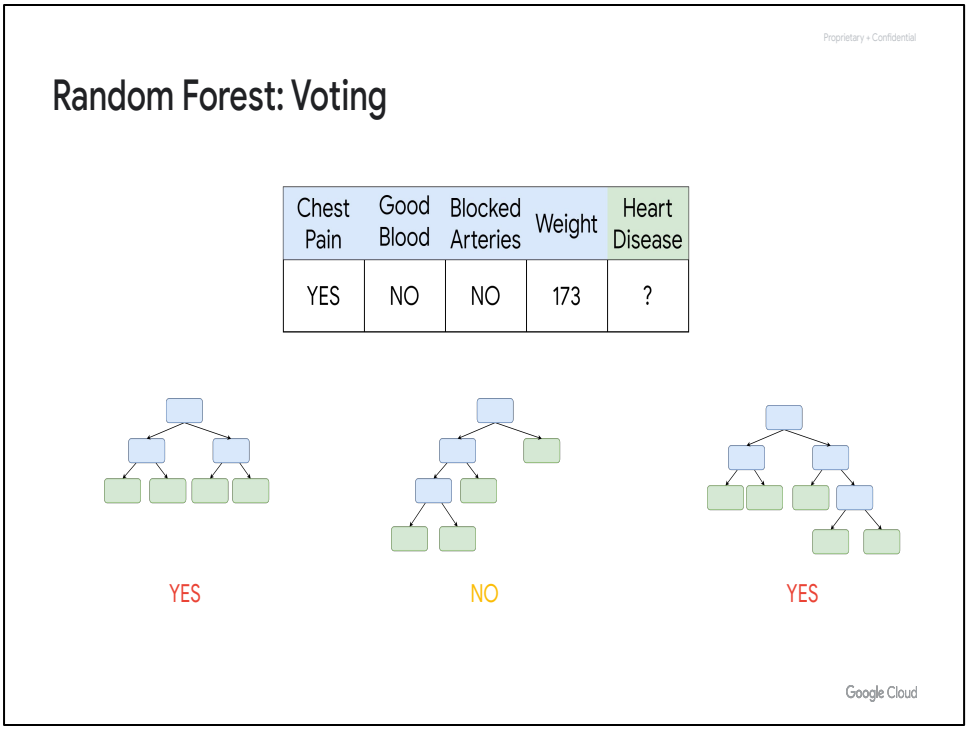


Colab: Decision Trees:
https://colab.research.google.com/github/daniyal9538/GeneralProjects/blob/master/Decision_Tree_Tutorial.ipynb

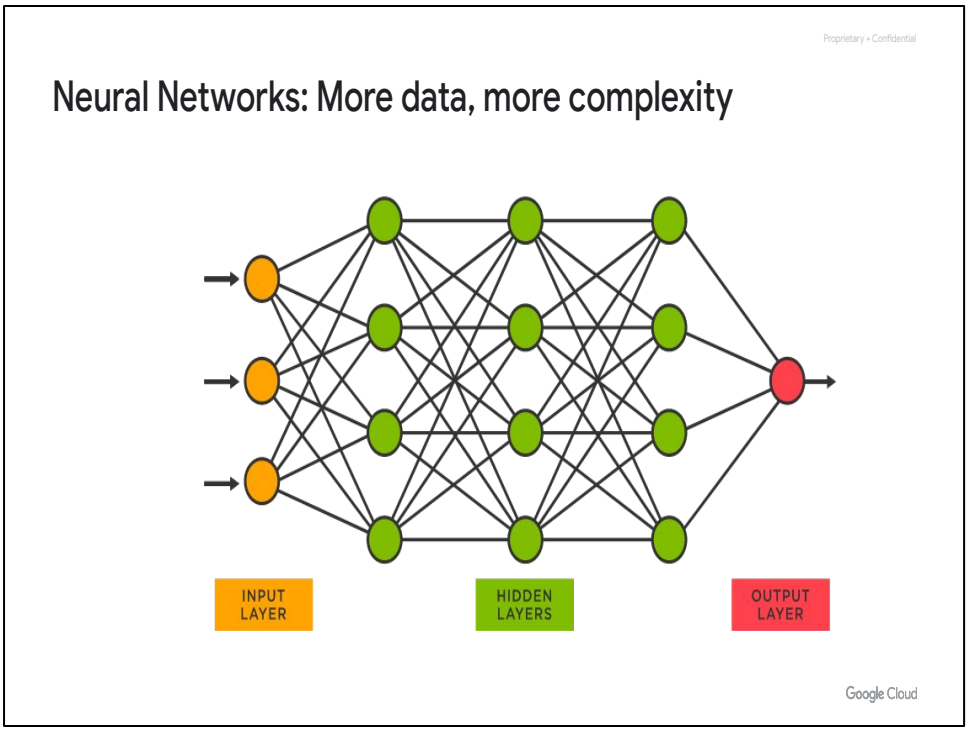
Decision Trees are great and simple.
However, they can overfit.

So we combine several, each with a different subset of features, into Random Forests.

Colab Random Forests:
https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.08-Random-Forests.ipynb



Google Colab: In Depth Decision Trees and Random Forests
https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.08-Random-Forests.ipynb



input layer, which receives the initial input data.

hidden layers, perform complex computations on the input data by applying mathematical functions to the data and passing it through multiple interconnected nodes.

output layer, which generates the final prediction or output based on the computations performed in the hidden layers. During the training process, the neural network adjusts the weights and biases of the connections between the neurons in each layer to minimize the difference between the predicted output and the actual output, thereby improving the accuracy of the predictions.

In summary,
a neural network uses multiple layers of interconnected nodes to process input data and generate output predictions by adjusting the weights and biases of the connections between the neurons during the training process.

Play with a Neural network Yourself with this link:

https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=3,5,3,3&seed=0.85204&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=true&ySquared=true&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false

&hideText=false

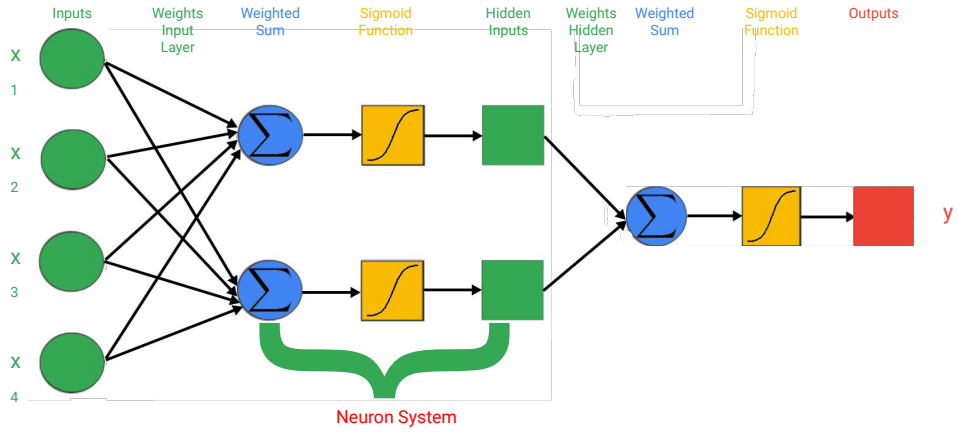
Dosage:
Dosage in neural networks typically refers to the quantity or amount of training data that is used to train the network.

Hidden Layer:
Refers to a layer of neurons that is not directly connected to either the input or output layer. Each hidden layer is made up of a number of neurons, which perform computations on the input data by applying mathematical

LAB: Classifying Images with a NN and DNN Model:
https://partner.cloudskillsboost.google/catalog_lab/5003

Colab: Neural Networks:
https://colab.research.google.com/github/google/eng-edu/blob/main/ml/cc/exercises/intro_to_neural_nets.ipynb?utm_source=mlcc&utm_campaign=colab-external&utm_medium=referral&utm_content=intro_to_nn_tf2-colab&hl=en#scrollTo=TL5y5fY9Jy_x

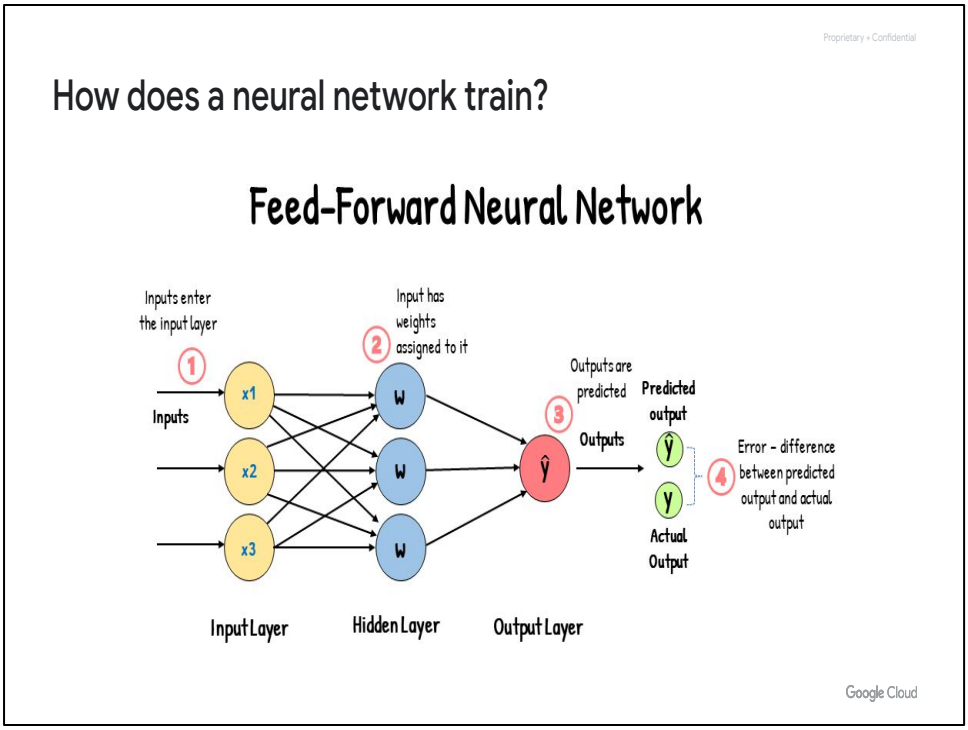
Neural networks: Multi-layer perceptron



Building off of the perceptron, just like the brain, we can connect many of them together to form layers to create feedforward neural networks. Really not much has changed in components from the single layer perceptron. There are still inputs, weighted sums, activation functions, and outputs.

One difference is that the inputs to neurons not in the input layer are not the raw inputs but the outputs of the previous layer. Another difference is that the weights connecting the neurons between layers are no longer a vector but now a matrix because of the completely connected nature of all neurons between layers. For instance, in the diagram the input layer weights matrix is 4 x 2 and the hidden layer weights matrix is 2 x 1. We will learn later that neural networks don't always have complete connectivity which has some amazing applications and performance like with images.

Also, there are different activation functions than just the unit step function such as the sigmoid and the hyperbolic tangent or tanh activation functions. Each non-input neuron, you can think of as a collection of three steps packaged up into a single unit. The first component is a weighted sum, the second component is the activation function, and the third component is the output of the activation function.



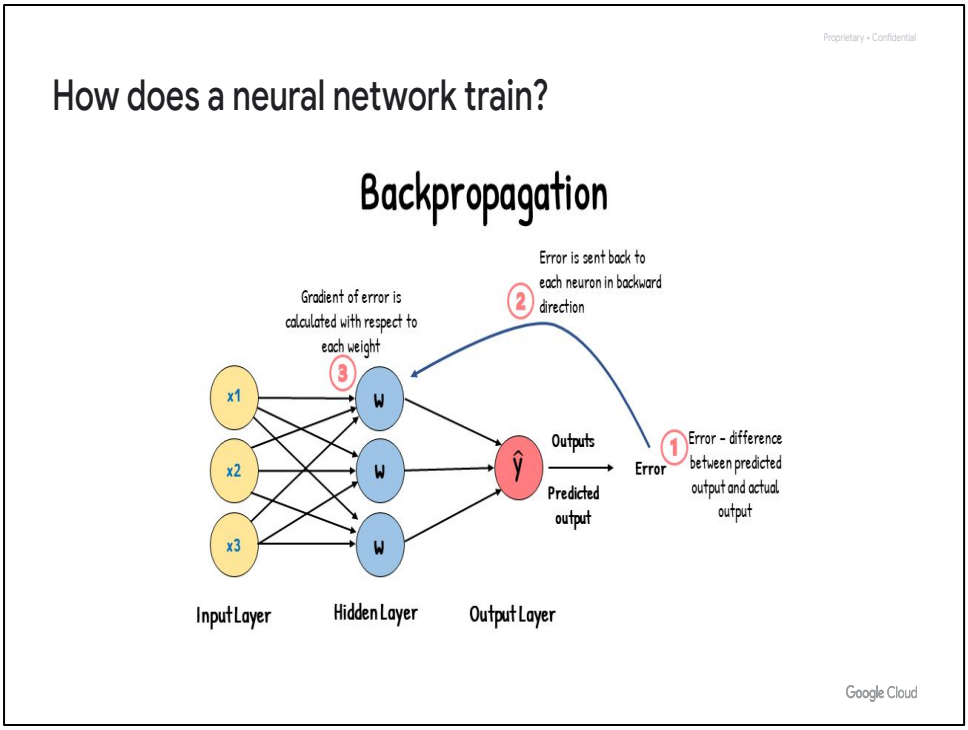
Essentially, backpropagation aims to calculate the negative gradient of the cost function. This negative gradient is what helps in adjusting of the weights. It gives us an idea of how we need to change the weights so that we can reduce the cost function.

Backpropagation uses the chain rule to calculate the gradient of the cost function. The chain rule involves taking the derivative. This involves calculating the partial derivative of each parameter. These derivatives are calculated by differentiating one weight and treating the other(s) as a constant. As a result of doing this, we will have a gradient.

Since we have calculated the gradients, we will be able to adjust the weights.

LAB: Classifying Images with a NN and DNN Model:
https://partner.cloudskillsboost.google/catalog_lab/5003

Colab: Feed Forward: Neural Network
https://colab.research.google.com/github/tdmdal/nn-workshop/blob/master/docs/2.2%20A%20First%20Feedforward%20Neural%20Network.ipynb



Essentially, backpropagation aims to calculate the negative gradient of the cost function. This negative gradient is what helps in adjusting of the weights. It gives us an idea of how we need to change the weights so that we can reduce the cost function.

Backpropagation uses the chain rule to calculate the gradient of the cost function. The chain rule involves taking the derivative. This involves calculating the partial derivative of each parameter. These derivatives are calculated by differentiating one weight and treating the other(s) as a constant. As a result of doing this, we will have a gradient.

Since we have calculated the gradients, we will be able to adjust the weights.

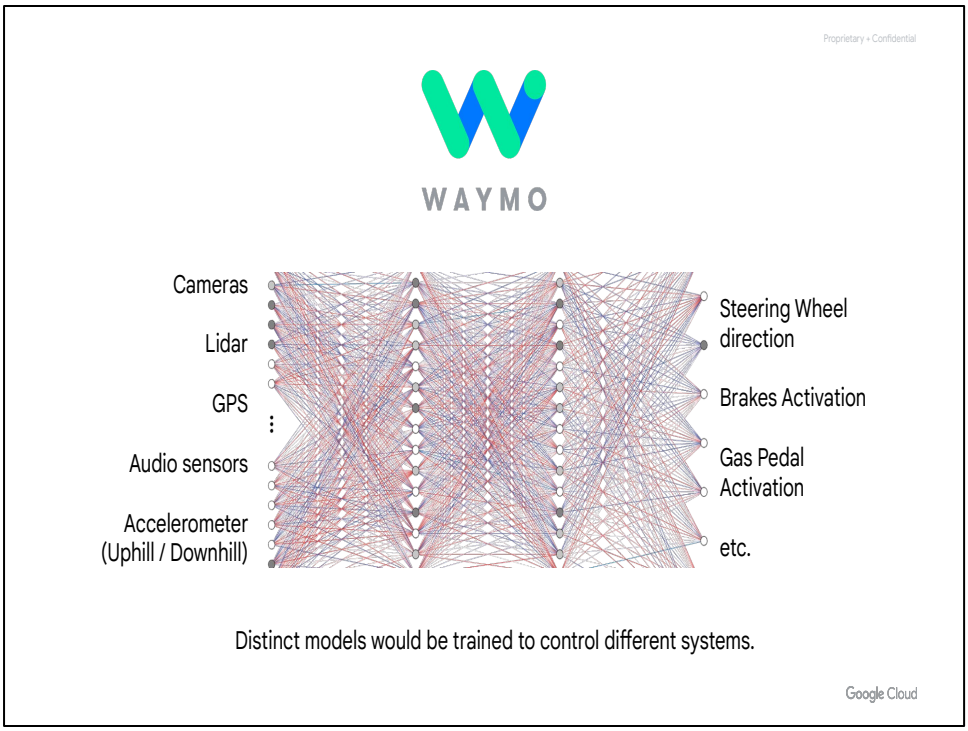
LAB: Classifying Images with a NN and DNN Model:
https://partner.cloudskillsboost.google/catalog_lab/5003

Colab: Neural Networks: backpropagation
https://colab.research.google.com/github/AmanDaVinci/DeepOrigins/blob/master/notebooks/03_Neural-Network-Backpropagation.ipynb

Let’s consider an example from the Autonomous Driving company Waymo within the Alphabet family (Google’s parent company).

A cool demo video. Play 1:30 to 3:05 from:
https://youtu.be/hA_-MkU0Nfw?feature=shared&t=89

Images from: https://waymo.com/about/



Source: Custom Screenshots from lab [Image Understanding with TensorFlow on Google Cloud]

CNNs are DNNs with some convolution and pooling layers. This allows to extract only the important features from images and feed it to a DNNs. Features are extracted using filters called kernels.

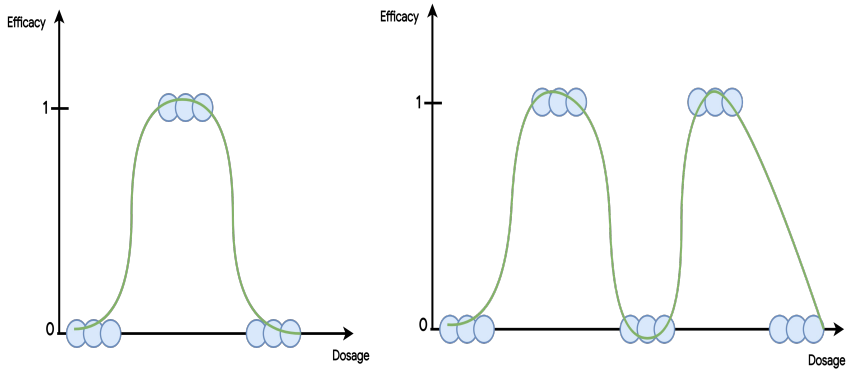
Waymo
Waymo's system relies on a combination of cameras, ultrasonic sensors, radar, and advanced machine learning algorithms to interpret the surrounding environment, identify objects, and make driving decisions.

CNN's play a role:
CNNs play a crucial role in the perception and object recognition components of the system. The neural networks are trained on large datasets containing labeled images and videos to recognize and classify various objects, such as cars, pedestrians, traffic signs, and road markings.

Next slides: Also in addition
In addition to CNNs, Waymo also employs other machine learning algorithms and techniques to handle tasks like path planning, control, and decision-making. The combination of these models and techniques allows Tesla's driverless cars to navigate complex driving scenarios and respond to real-world situations effectively

Colab: Neural Networks:
https://colab.research.google.com/github/google/eng-edu/blob/main/ml/cc/exercises/intro_to_neural_nets.ipynb?utm_source=mlcc&utm_campaign=colab-external&utm_medium=referral&utm_content=intro_to_nn_tf2-colab&hl=en#scrollTo=TL5y5fY9Jy_x

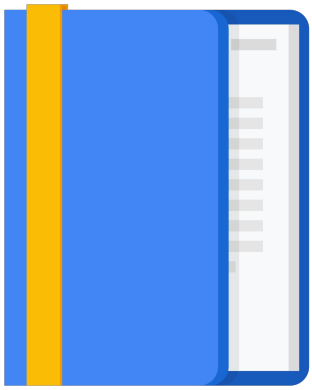
Neural networks... can solve nonlinear problems



Algorithms Section Agenda

Supervised Learning Algorithms

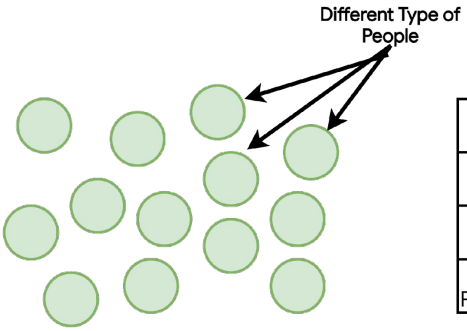
Unsupervised Learning Algorithms



These are the modules that make up this course.

Proprietary + Confidential

Clustering with K-Means



	Type1	Type2	Type3	Type4	Type5	....
Height	1.73	1.84	1.54	1.68	1.88	
Weight	78	65	69	75	90	
Blood Pressure	120	110	90	130	150	

Google Cloud

In simple terms, clustering is a process of grouping similar data points together based on their features or characteristics.

A common algorithm for clustering is K-Means. (K-Means entry from Google ML Glossary).
It is similar to K-Nearest Neighbors, only now since we do not have labels, we are not looking for the nearest neighbor example to predict a label. We are instead trying to identify boundaries that group

Clustering Algorithms Google Doc

# # #

Clustering: this helps to identify patterns or structures within the data that might not be immediately apparent.

Principal Component Analysis (PCA) is a technique used to reduce the dimensionality of data while preserving as much information as possible. It transforms the original set of features into a new set of features called principal components, which are linear combinations of the original features.

When you use clustering with PCA, you're combining these two techniques to achieve the following goals:

- **Reduce the dimensionality of the data using PCA:** By applying PCA first, you reduce the number of features in the dataset, making it easier to analyze and visualize. This process can also help remove noise and improve the efficiency of the clustering algorithm.

- **Perform clustering on the transformed data:** After reducing the dimensions using PCA, you apply a clustering algorithm (e.g., K-means,, or hierarchical clustering) to the transformed data. This step groups similar data points together based on their principal components, which are the new features generated by PCA.

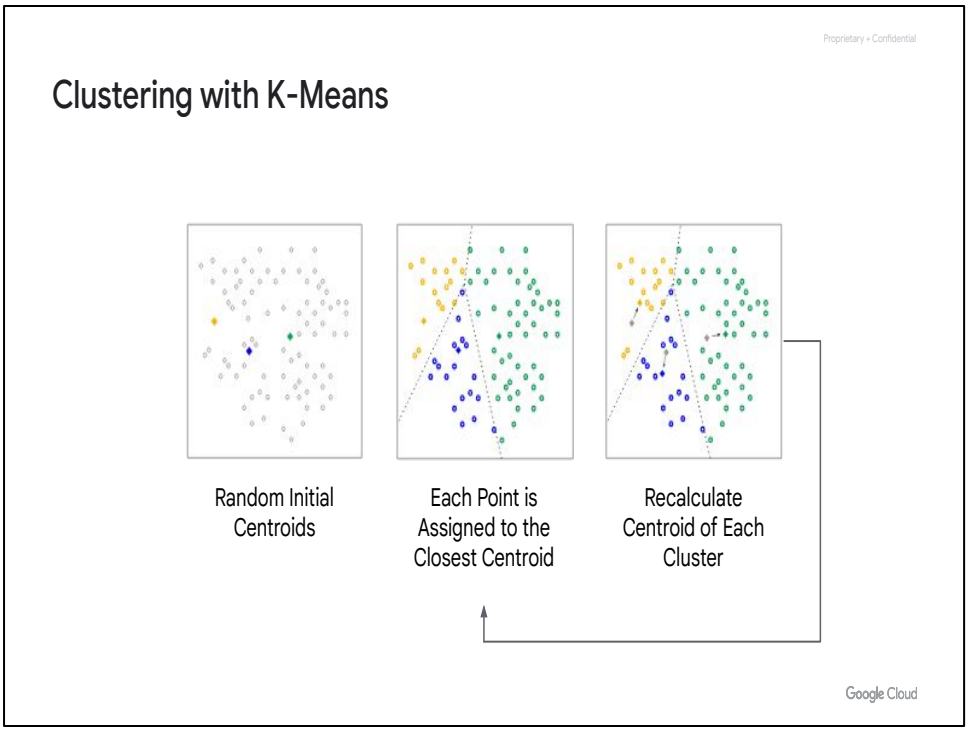
In summary, clustering with PCA involves reducing the dimensionality of data using PCA and then applying a clustering algorithm to the transformed data. This combination helps to simplify the analysis, improve efficiency, and reveal patterns or structures in the data more effectively.

Dimension Reduction: - Definition
Reducing the dimensionality of the data is the process of transforming a dataset with many features (dimensions) into a new dataset with fewer features while retaining as much of the original information as possible.

In very simplistic terms, think of it as compressing the data into a more compact form without losing too much of its essence. This is done to make the data easier to understand, visualize, and analyze.

Simplify the data, making it easier to understand and work with.
Reduce the computational complexity, making analysis faster and more efficient.
Remove noise and irrelevant features, leading to better results in machine learning and data analysis tasks

Colab:
https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.09-Principal-Component-Analysis.ipynb#scrollTo=xdyApijwQDAn



K-means is a popular clustering algorithm in machine learning that is used to partition a dataset into K distinct clusters. A simple use case of K-means in ML is in **customer segmentation for a retail business.**

Images from: [Run the Clustering Algorithm]

Optional clustering programming exercise:
https://developers.google.com/machine-learning/clustering/programming-exercise

A Simple use case:

- Suppose a retail business has a large database of customer transaction data, including information such as the customer's age, income, purchase history, and other relevant features. The business wants to segment its customers into different groups based on their buying behavior to create more targeted marketing campaigns.

- **Answer:**
  To accomplish this task, the business can use K-means clustering. They would start by selecting a value for K, the number of clusters they want to create. Then, they would apply the K-means algorithm to the customer data, using the features as inputs. The algorithm would group similar customers together into clusters based on their feature values. This will to group th customers

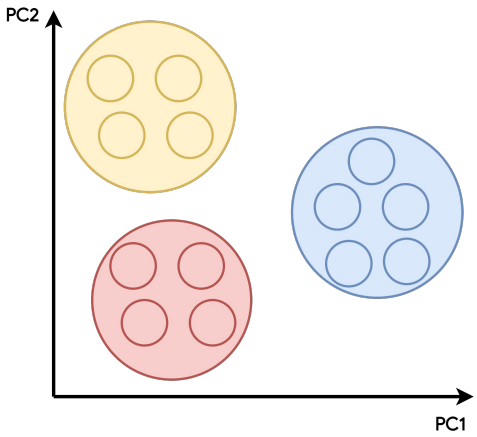
Overall, K-means clustering is a useful technique in machine learning for segmenting large datasets into distinct clusters based on their feature values. It can help businesses gain insights into customer behavior and create more targeted marketing campaigns.

Colab: Kmean:
https://colab.research.google.com/github/SANTOSHMAHER/Machine-Learning-Algorithams/blob/master/K_Means_algorithm_using_Python_from_scratch_.ipynb

Dimensionality Reduction with PCA

	Type1	Type2	Type3	Type4	Type5
Height	1.73	1.84	1.54	1.68	1.88
Weight	78	65	69	75	90
Blood Pressure	120	110	90	130	150

- Correlation matters
- Features that are highly correlated cluster together



Principal Component Analysis (PCA)
is a dimensionality reduction technique commonly used in machine learning to reduce the number of features in a dataset while retaining as much information as possible. A simple use case of PCA in ML is in image recognition tasks.

This can be used before visualizing data in a number of dimensions we can visualize.

It can also be used before clustering, regression, classification, or other technique if there are too many features per example for a model to generalize well.

Correlation:
PCA can be especially useful when dealing with high-dimensional data that exhibits multicollinearity, meaning that some variables are highly correlated with each other. In such cases, PCA helps to reduce redundancy in the data by identifying the most important linear combinations of the original variables.

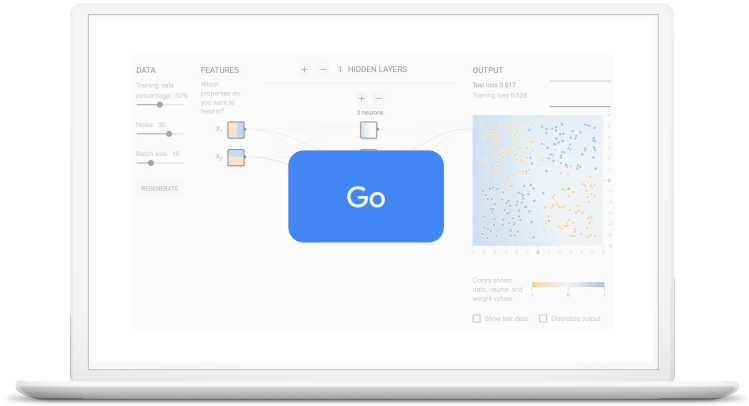
High Dimension data:
High-dimensional data refers to datasets that have a large number of features, variables, or dimensions. In other words, the data has many attributes or characteristics associated with each observation. High-dimensional datasets can be challenging to work with and analyze for several reasons:

Principal Component Analysis Colab - interactive

- https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.09-Principal-Component-Analysis.ipynb#scrollTo=p5aR1jV2yypE

TF Playground Exercise

Neural Networks: Playground Exercises



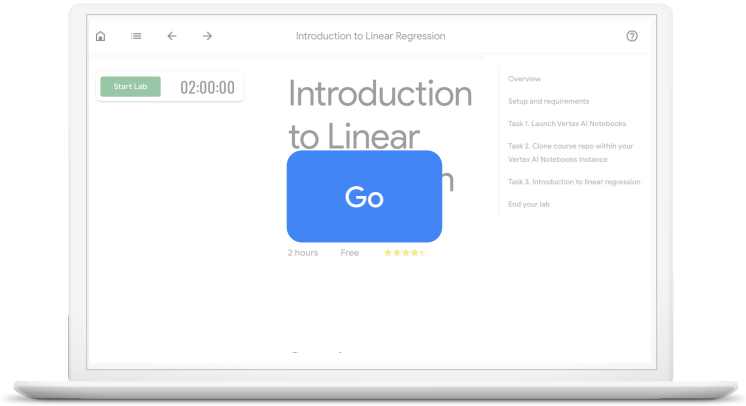
Neural Networks: Playground Exercises
Estimated Time: 20 minutes

Recommended Lab

Introduction to Linear Regression

From the Course
Launching Into Machine Learning

Select this lab from the section:
“Machine Learning in Practice”



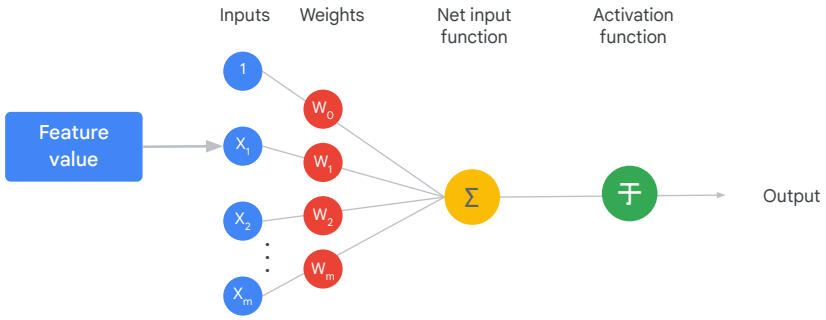
Direct link to the notebook on GitHub

https://partner.cloudskillsboost.google/course_sessions/4505642/labs/411385

Go to the Course: https://partner.cloudskillsboost.google/course_templates/8
then within the section "Machine Learning in Practice", select "Intro to Linear Regression"

Introduction to Feature Engineering

What is a feature?

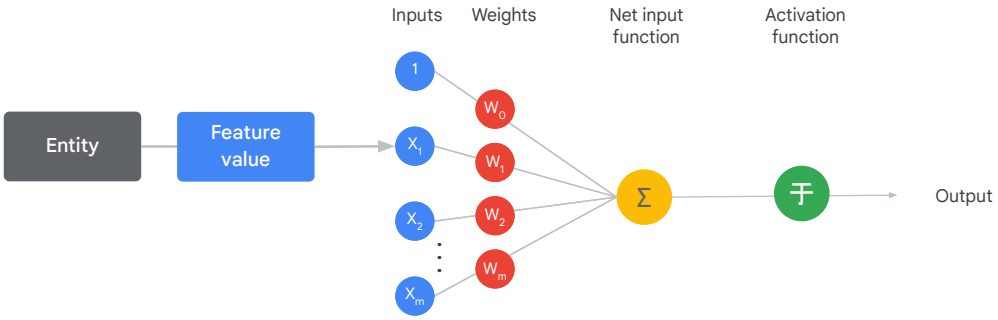


A measurable or recordable property of an entity,
which is passed as input to a machine learning model

Simply stated, a feature is a value that is passed as input to a model.

Note also that Feature Store can store scalars and arrays/tensors.

A feature describes some entity

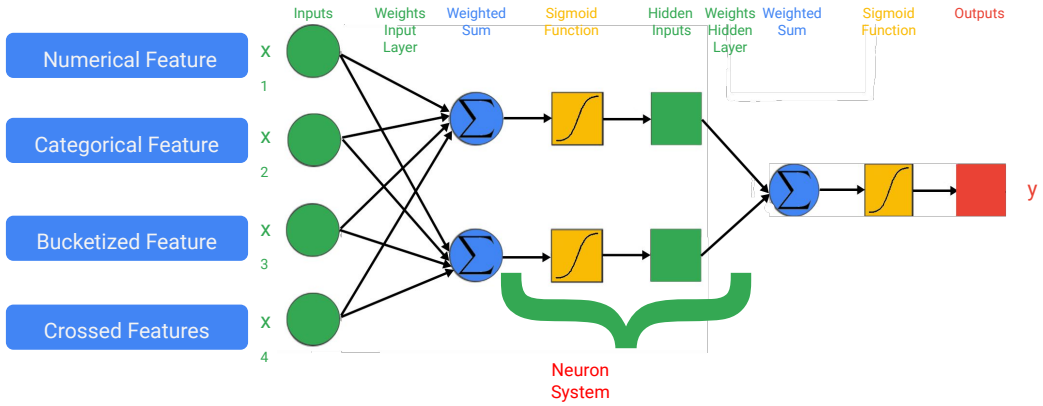


Examples: Age of user, price of product, category of web page

Essentially, a feature describes some entity.

For example, age of user, price of product, or category of web-page.

Features are represented numerically and fed into our mathematical modes



Machine learning models, such as neural networks, accept a feature vector and provide a prediction. These models learn in a supervised fashion where a set of feature vectors with expected output is provided.

From a practical perspective, many machine learning models must represent the features as real-numbered vectors because the feature values must be multiplied by the model weights. In some cases, the data is raw and must be transformed into feature vectors.

Features, the columns of your dataframe, are key in assisting machine learning models to learn. Better features result in faster training and more accurate predictions. As the diagram shows, feature columns are input into the model—not as raw data, but as feature columns.

Engineering new features from a provided feature set is a common practice. Such engineered features will either augment or replace portions of the existing feature vector. These engineered features are essentially calculated fields based on the values of the other features. As you will see later in the labs, feature vectors can be numerical, categorical, bucketized, crossed, and hashed.

Some types of features:

(**TODO**: Turn this into a series of slides exploring these different features.)

Numerical Feature:
Ex, Age: The age of a person is a numerical feature that can take on continuous values, such as 22.5 years old, 40 years old, or 65.2 years old. This feature can be used in a machine learning model to predict various outcomes, such as the likelihood of purchasing a certain product or developing a certain health condition.

Categorical features include:
Color: The color of a product is a categorical feature that can take on a limited set of values, such as red, green, or blue. This feature can be used in a machine learning model to predict various outcomes, such as the likelihood of a product selling well in a particular market or the demand for certain products during specific seasons.

Education level: The education level of a person is a categorical feature that can take on a limited set of values, such as high school, college, or graduate school. This feature can be used in a machine learning model to predict various outcomes, such as the likelihood of a person getting a certain job or earning a certain salary.

Bucketised Feature:
Other examples of bucketized features include:

Temperature range: The temperature of a location is a numerical feature that can take on continuous values, such as 20.5 degrees Celsius or 75 degrees Fahrenheit. To create a bucketized feature from the temperature feature, we can divide the temperature range into discrete buckets, such as very cold, cold, mild, warm, hot, and very hot. This bucketized feature can be used in a machine learning model to predict various outcomes, such as the likelihood of rainfall, the risk of wildfires, or the demand for heating or cooling services.

Crossed Feature:
represents the interaction between two or more categorical features. It is typically represented as a new categorical feature that captures the joint effects of the original features. Here is an example of a crossed feature:

Zip code and occupation: Let's say we have two categorical features, zip code and occupation, and we want to capture their interaction to predict the likelihood of a person purchasing a certain product. We can create a crossed feature by concatenating the two features, such as "90210-Engineer" or "10001-Teacher". This crossed feature can be used in a machine learning model to predict various outcomes, such as the likelihood of purchasing certain products based on their occupation and geographic location.
Other examples of crossed features include:

Color and size: Let's say we have two categorical features, color and size, and we want to capture their interaction to predict the demand for a certain product. We can create a crossed feature by concatenating the two features, such as "Red-Small" or "Blue-Large". This crossed feature can be used in a machine learning model to

predict various outcomes, such as the likelihood of a product selling well in a particular market.

One-hot encoding turns categoricals into numeric values

Original Categorical Field

Country
Portugal
Italy
Portugal
null

One-hot Encoded Representation

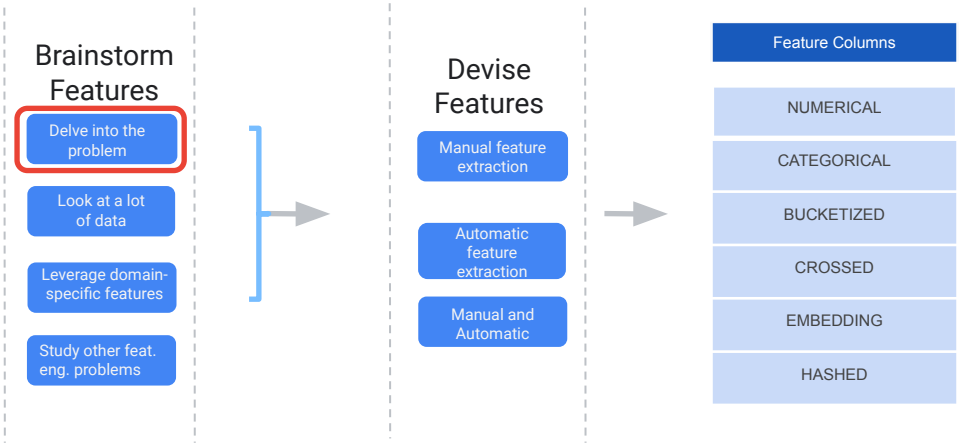
Country_Portugal	Country_Italy	Country_null
1	0	0
0	1	0
1	0	0
0	0	1

Categorical features are converted to numbers through one-hot encoding (a column that is a 1 for a particular value of a categorical, or 0 for another value).

This can lead to an explosion of features if there are many possible categorical values.

This can be solved by hashing, which can act like bucketizing categorical values.

How do we develop features?



Delve into the problem.

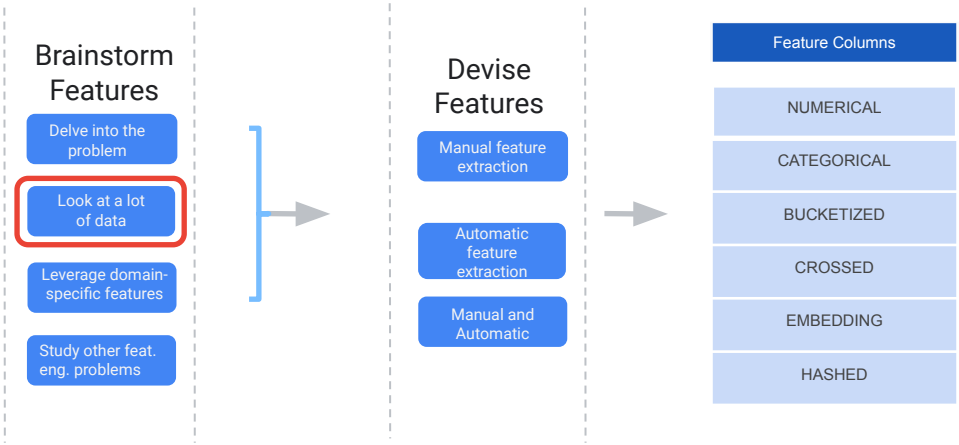
So, what does it mean to “delve into the problem?”

If your goal is to solve a problem where you need to “predict” an outcome and get the best possible results from a predictive model, you need to determine if your source data can aide in this goal.

In other words, how can you get the most out of your data for predictive modeling? This is the problem that the process and practice of feature engineering solves.

So you start by...

How do we develop features?



...looking at a lot of data.

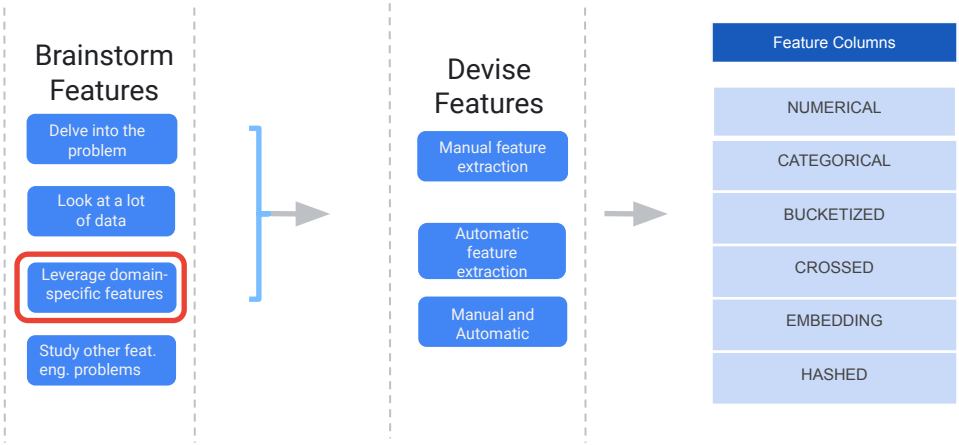
If you work as an academic researcher, you may have access to a plethora of data sources - from government data to industry data.

If you work in a corporation, your data sources are the data your organization uses to manage the business. For example, you may be looking at customer data, sales data, product data, inventory data, operational data, and people management data.

And, this data could all be in different formats. The customer data could be a .csv file, the sales data could come from a .JSON file, the operational data could be in an .XML format...you get the picture.

Depending on your problem and your domain, you then need to...

How do we develop features?

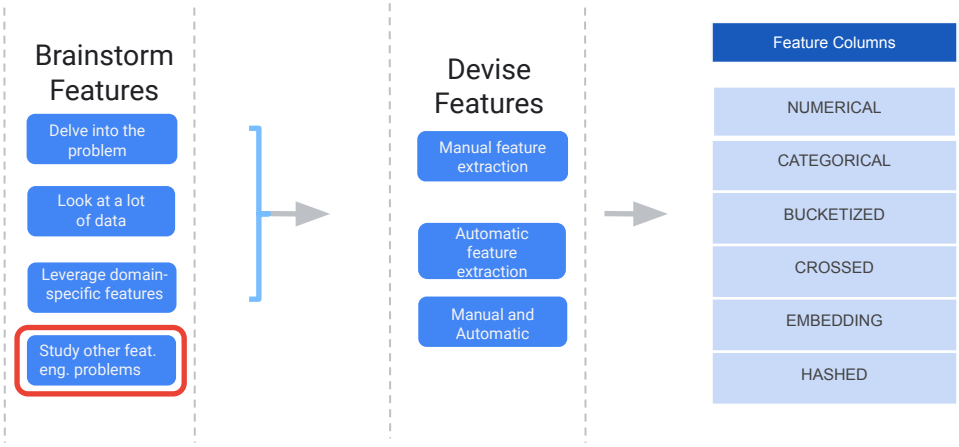


...leverage domain-specific engineered features.

For example, if your machine learning problem is to predict seasonal sales based on past customer purchases in a geographic location, then you must either have domain specific knowledge or find domain specific features to solve the problem.

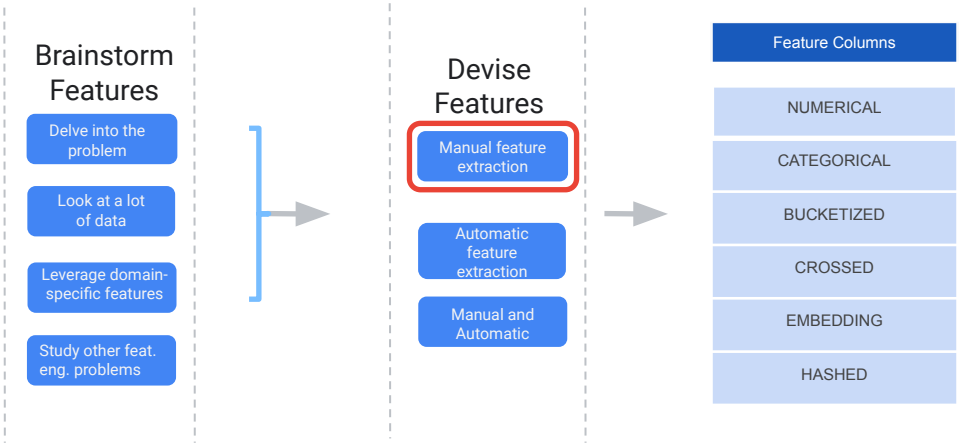
In this case, you may be looking for data that allows you to create features for customer purchases at a specific date/time in a specific geo-location (or spatial) region.

How do we develop features?



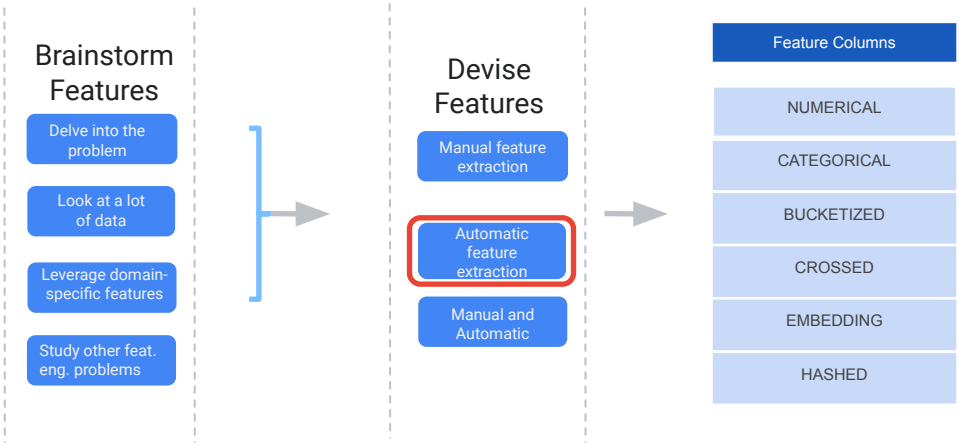
Studying other feature engineering problems is highly recommended. The example of seasonal sales, for example, lends itself to time series features and longitude and latitude features.

How do we develop features?



In manual feature extraction, features are constructed manually. In this module, we use manual feature extraction, where we manually devise features using Python and TensorFlow code libraries.

How do we develop features?

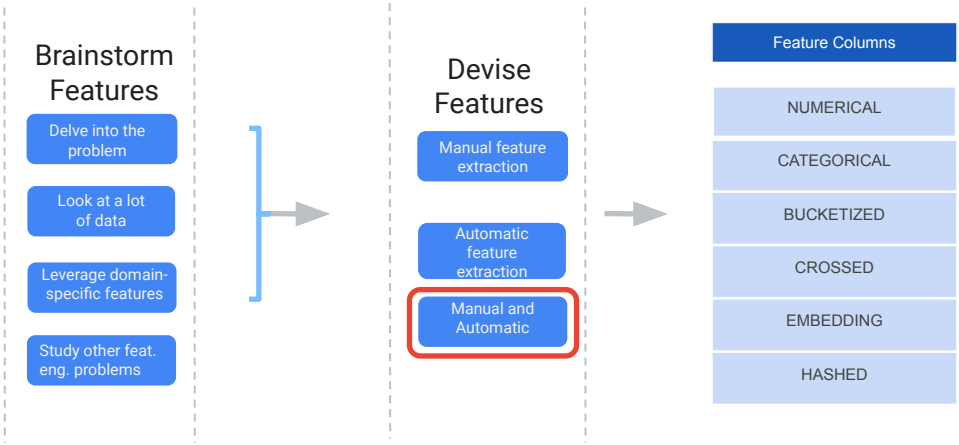


Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing.

For example, Principal Component Analysis (or PCA) is a way to reduce the number of variables or features in your dataset. As the name states, you are simply analyzing the principal components (your independent variables or features) to determine how well they predict your dependent variable or “target” (the final output you are trying to predict). In technical terms, you want to “reduce the dimension of your feature space.”

By reducing the dimension of your feature space, you have fewer relationships between variables to consider and you are less likely to overfit your model. In automatic feature extraction, the idea is to automatically learn a set of features from potentially noisy, raw data that can be useful in supervised learning tasks such as in computer vision.

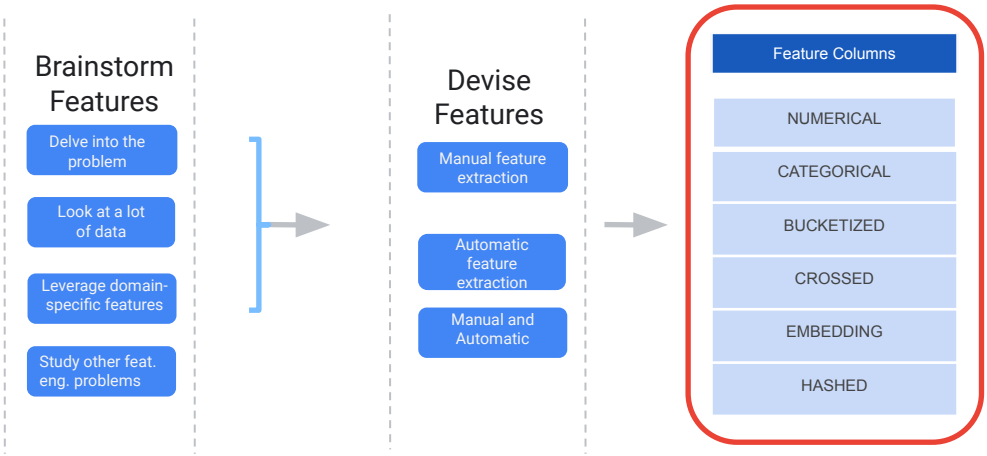
How do we develop features?



You can also devise features using a combination of both Automatic and Manual methods.

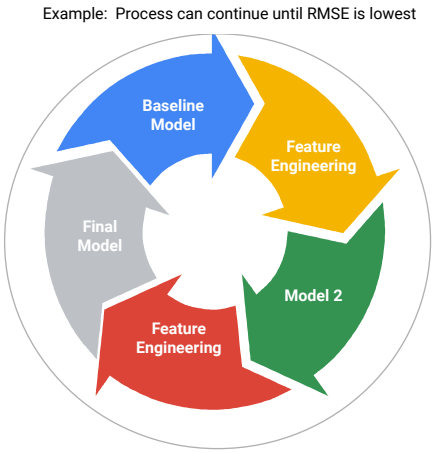
As you can see...

How do we develop features?



...in this example here, the feature columns derived from this process can include: numerical, categorical bucketized, crossed, embedding, and hashed feature columns.

Developing features can be an iterative process



Recall that feature engineering can be defined as the process of transforming raw data into features that are input to the final model.

However, feature engineering in reality is an iterative process. For example, you can improve the accuracy of models by increasing the predictive power of learning algorithms through iteration; in other words, you create a baseline model with little to no feature engineering (as determined by your data types), and then add feature engineering to see how your model improves.

Introduction to Feature Engineering module:

[Course]:
Feature Engineering:
https://partner.cloudskillsboost.google/course_templates/11

[LAB]
Using Feature Store:
https://partner.cloudskillsboost.google/course_sessions/2598087/labs/319204

Performing Basic Feature Engineering in Keras:
https://partner.cloudskillsboost.google/course_sessions/2598087/labs/319231

Performing Advanced Feature Engineering in Keras:

https://partner.cloudskillsboost.google/course_sessions/2598087/labs/319233

Types of features

Type	Example
Using indicator variables to isolate key information.	Isolates a specific area for our training dataset.
Highlighting interactions between two or more features.	Sum of two features, product of two features, etc.
Representing the same feature in a different way.	Create a new feature “grade” with "Elementary School," "Middle School," and "High School" as classes.
	Group similar classes, and then group the remaining ones into a single "Other" class.
	Transform categorical features into dummy variables.

Feature engineering types include:

1. Using indicator variables to isolate key information: For example, geolocation features for New York city taxi service area isolates a specific area for our training dataset.

Types of features

Type	Example
Using indicator variables to isolate key information.	Isolates a specific area for our training dataset.
Highlighting interactions between two or more features.	Sum of two features, product of two features, etc.
Representing the same feature in a different way.	Create a new feature “grade” with "Elementary School," "Middle School," and "High School" as classes.
	Group similar classes, and then group the remaining ones into a single "Other" class.
	Transform categorical features into dummy variables.

Highlighting interactions between two or more features: Interactions include the sum of two features, the difference between two features, the product of two features, and the quotient of two features.

Types of features

Type	Example
Using indicator variables to isolate key information.	Isolates a specific area for our training dataset.
Highlighting interactions between two or more features.	Sum of two features, product of two features, etc.
Representing the same feature in a different way.	Create a new feature “grade” with "Elementary School," "Middle School," and "High School" as classes.
	Group similar classes, and then group the remaining ones into a single "Other" class.
	Transform categorical features into dummy variables.

Representing the same feature in a different way: For example, in numeric to categorical mappings, where you have “grade,” you can create categories and create a new feature “grade” with "Elementary School," "Middle School," and "High School" as classes.

Types of features

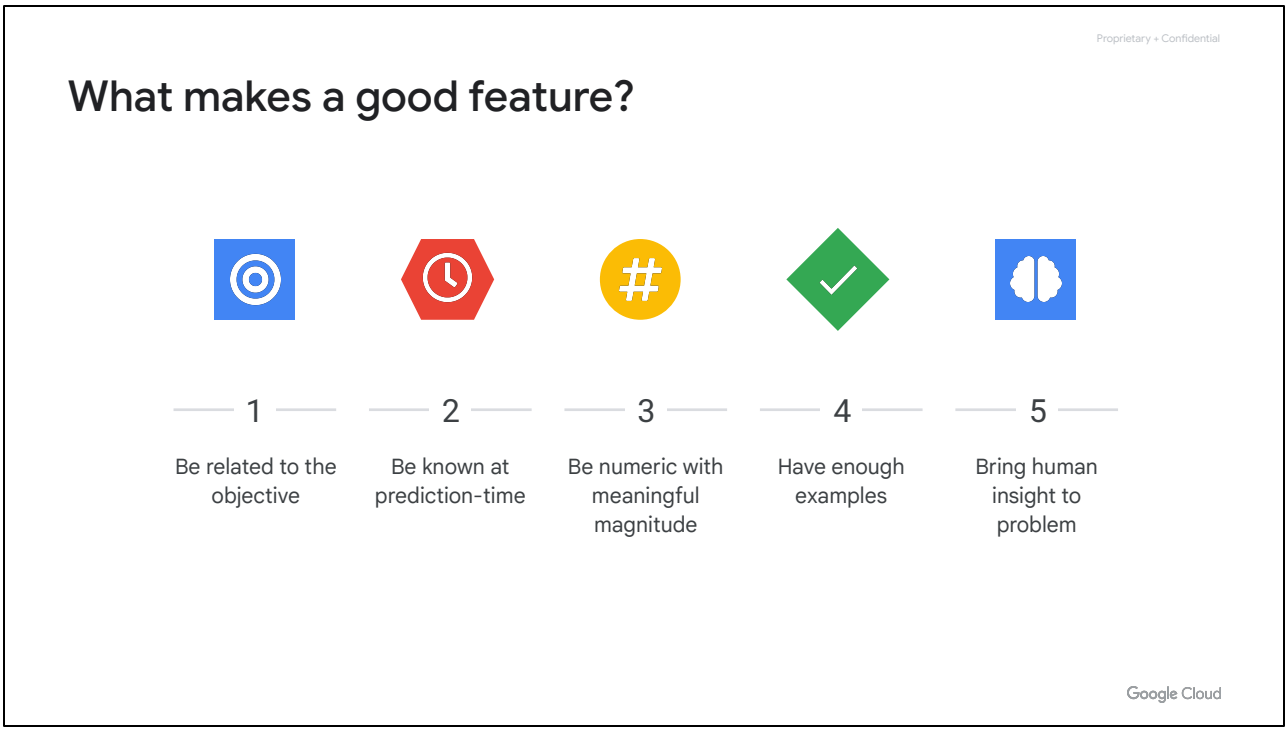
Type	Example
Using indicator variables to isolate key information.	Isolates a specific area for our training dataset.
Highlighting interactions between two or more features.	Sum of two features, product of two features, etc.
Representing the same feature in a different way.	Create a new feature “grade” with "Elementary School," "Middle School," and "High School" as classes.
	Group similar classes, and then group the remaining ones into a single "Other" class.
	Transform categorical features into dummy variables.

Another example is to group sparse classes, where you group similar classes and then group the remaining ones into a single “other” class.

Types of features

Type	Example
Using indicator variables to isolate key information.	Isolates a specific area for our training dataset.
Highlighting interactions between two or more features.	Sum of two features, product of two features, etc.
Representing the same feature in a different way.	Create a new feature “grade” with "Elementary School," "Middle School," and "High School" as classes.
	Group similar classes, and then group the remaining ones into a single "Other" class.
	Transform categorical features into dummy variables.

And another example is to transform categorical features into dummy variables.



Module: Feature Engineering

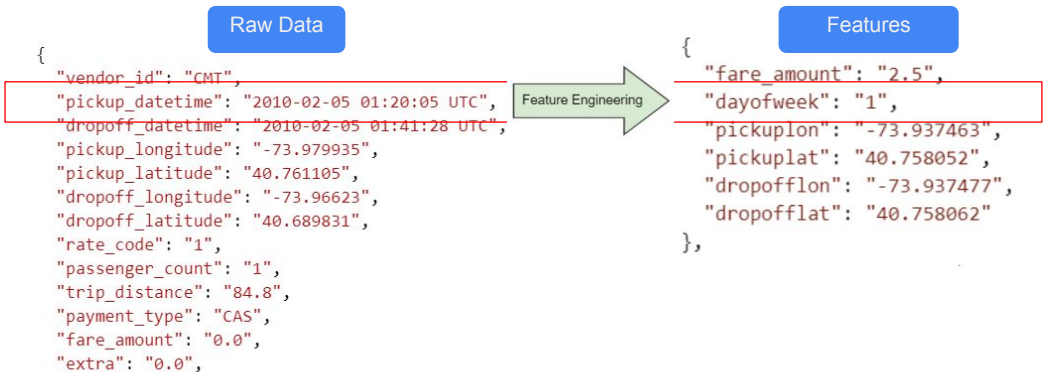
Now, what makes a good feature, right? You want to basically take your raw data, and you want to represent it in a form that's amenable to machine learning.

So it has to be related to the objective, you don't want to just throw random data in there. That just makes the ML problem harder and the idea is to make the ML problem easier, right, make it easier for you to find the solution. If something is not related, throw it away.

You have to make sure that it's known at **prediction-time**. This can be surprisingly tricky.

- You need to make sure it is **numeric.**
- It has to have **enough examples**.
- And you need to have some **human insights**.

Raw data does not come as a perfect set of features.



Often raw data needs to be cleaned up or transformed to be ready to be used.

You also don't want to just throw every possible feature at the model, but have at least a hypothesis for how each value might actually relate to the desired output.

In this example, of all the raw features on the left, we choose only six features for our machine learning problem. The label (or the value we are predicting) is the fare amount.

Some tools for exploring correlation between features and labeled instances of the label you will predict.

1. Pearson Coefficient (Between Two Numerical Features)
2. Chi-Squared Test (Between Two Categorical Features)
3. Mutual Information – Information Gain (Between categorical to categorical)
4. T-Test / ANOVA - (between multi-category and numerical feature)
5. PCA

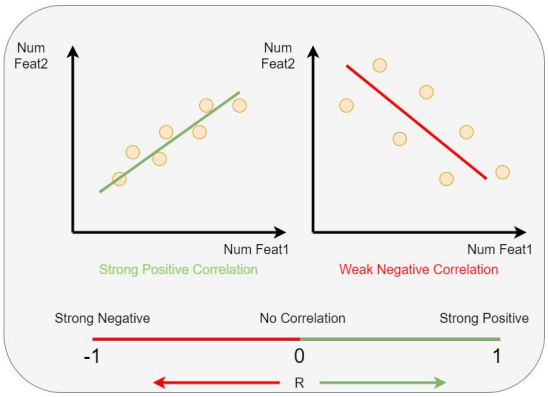
Source: Custom

A good feature must be **related to what you're predicting**. You need to have a reasonable hypothesis about why a particular feature might matter for this particular problem.

These are the 5 main methods used for feature selection.

1. Pearson Coefficient

- Returned values (R, p-value)
- Between numeric to numeric
- For p-value < 0.5% gives us strong confidence



Source Custom:

Pearson Coefficient allows to find the correlation between two numerical features. It returns a value that ranges from -1 and 1 and indicates how correlated two features are. The more R is close to -1 or 1 the stronger would be the correlation. On the other hand, if the R is close to 0 the feature are independent.

2. Chi-Squared Test

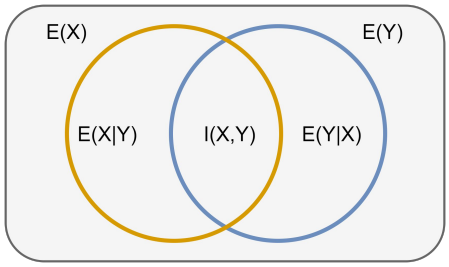
- Returned values (X2, p-value)
- Between categorical to categorical
- Test of Independence between features
- Test of dependence between a feature and the target

Source: Custom

Chi-Squared Test is similar to the Pearson Coefficient but works between categorical features

3. Mutual Information – Information Gain

- Share entropy between a feature and the target
- Between categorical to categorical
- Test of dependence between a feature and the target

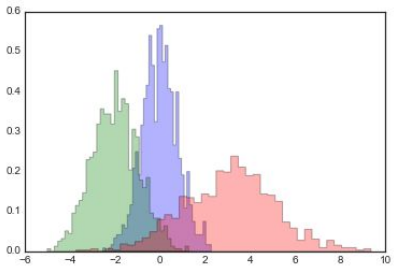


Source Custom

The Information Gain is another test that can measure how much information the feature shares with the target (label). It can be used to find out much a feature is correlated with the target

4. T-Test / ANOVA: Analysis of Variance

- Use T-Test between binary category and numerical feature
- Use ANOVA between multi-category and numerical feature
- Test of dependence between a feature and the target

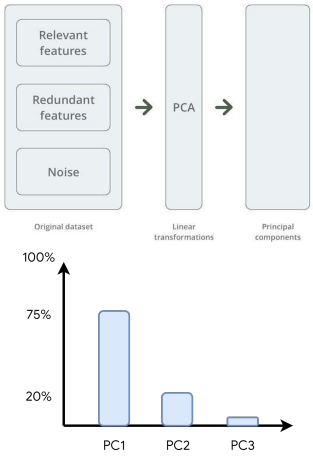


Source: Custom

Similar to Information Gain, T-Test and ANOVA can show how much numerical features and categorical features can have in common. Throughout the analysis of distribution of means you can see how much difference there is between feature and/or target means.

5. PCA: Principal Component Analysis

- PCA is used for dimension reduction.
- PCA finds out the most important features
- Use also for clustering features



Source: Custom

Principal Component Analysis allows data scientists to find out the most important features in a dataset. This happens by computing the total variation of each feature (transformed in the Principal Component) as shown in the graph.

Dimension Reduction:

Dimension reduction is a technique used in machine learning to reduce the number of input variables or features in a dataset. The aim of dimension reduction is to simplify the data while retaining as much of its important information as possible.

Dimensionality reduction is particularly useful when dealing with high-dimensional data, where there are many variables. In such cases, it can be difficult to visualize and analyze the data, and machine learning models may also struggle to find patterns and make accurate predictions.

There are several approaches to dimensionality reduction, including:

Feature selection: Selecting a subset of the original features based on their importance or relevance to the target variable.

Feature extraction: Transforming the original features into a new set of

features, usually using linear algebra techniques such as Principal Component Analysis (PCA) or Singular Value Decomposition (SVD).

Manifold learning: Finding a low-dimensional representation of the data that preserves the underlying structure and relationships among the data points.

Overall, dimensionality reduction can improve the performance of machine learning models by reducing noise, simplifying the data, and speeding up the training process.

Use Case:

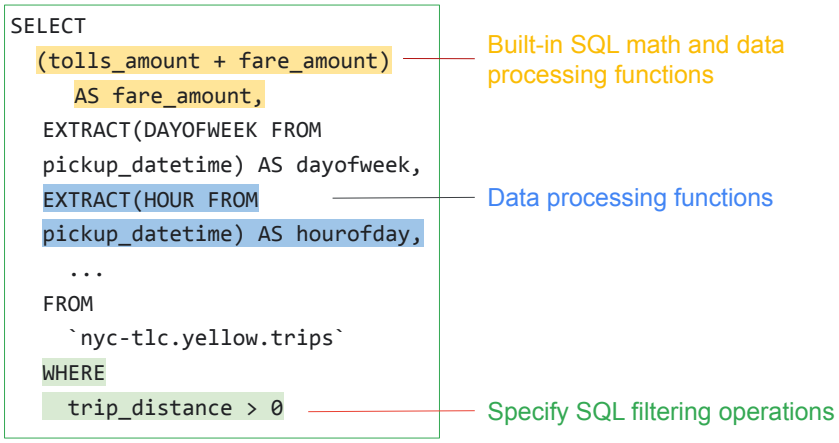
Principal Component Analysis (PCA) is a popular dimensionality reduction technique used in machine learning, and it can be applied to a wide range of use cases. One common use case for PCA is in image processing.

For example, let's say you have a dataset of images that are 100x100 pixels in size, each represented by 10,000 features (one for each pixel). This high-dimensional dataset may be difficult to work with and could lead to overfitting if used to train a machine learning model.

Using PCA, you could reduce the number of features in the dataset while retaining the important information. PCA works by identifying the principal components of the dataset, which are linear combinations of the original features that explain the most variance in the data.

By selecting the top k principal components, you can create a lower-dimensional representation of the images, where k is much smaller than the original number of features. This reduced dataset can be used to train machine learning models with improved performance and reduced risk of overfitting.

Basic feature engineering in BigQuery



BigQuery can help with feature engineering because it lets you use SQL to implement common preprocessing tasks. For example, if you are preprocessing a dataset with records of taxi rides in New York City, you can specify SQL filtering operations to exclude bogus data from your training examples datasets, like the rides with a distance of 0 miles.

Built-in SQL math and data processing functions are also valuable for simple calculations additions over source data.

Data processing functions are also valuable for parsing common data formats, like timestamps, to extract details about the time of day.

Filter out bogus data; for example trip distances must be above zero. Extract hourly data. Perform calculations to get a new field: fare_amount.

Advanced feature engineering

BigQuery ML preprocessing functions

- ML.FEATURE_CROSS(STRUCT(features)) does a feature cross of all the combinations.
- ML.POLYNOMIAL_EXPAND(STRUCT(features), degree) creates x, x2, x3, etc.
- ML.BUCKETIZE(f, split_points) where split_points is an array.

Here are some of the preprocessing functions in BigQuery ML:

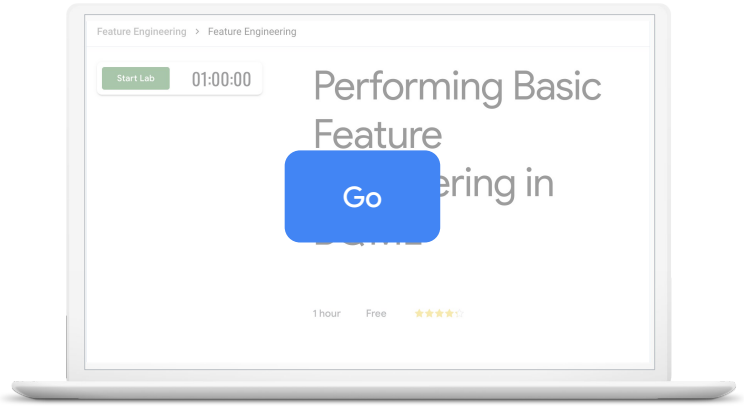
- ML.FEATURE_CROSS(STRUCT(features)) does a feature cross of all the combinations.
- ML.POLYNOMIAL_EXPAND(STRUCT(features), degree) creates x, x2, x3, etc.
- ML.BUCKETIZE(f, split_points) where split_points is an array.

Recommended Lab

Performing Advanced Feature Engineering in BQML

From the Course
Feature Engineering

Select the basic lab in the section: “Feature Engineering”, then open the advanced notebook instead of the basic one.



Performing Basic Feature Engineering in BQML (also uses Workbench)
part of Feature Engineering Course

Within the same folder where this notebook lies, there is an advanced bigquery version as well. The Qwiklab version of that advanced one seems to be gone, but a thing you may consider is assigning this Basic one and then when reviewing the lab with students, talk through the solution notebook from the advanced one to look at a more advanced and completed feature engineering process.

An alternate:
Interactive exploratory analysis of BigQuery data in a notebook
(Click "Open in Vertex AI Workbench managed notebooks". Requires their own GCP project)

Questions and answers

Thank you for attending this training!

We love your feedback! Please take a minute to complete the survey and help us improve our courses.


Google Cloud