GCP PMLE certification refresh guide

Contents

GCP components

- **Data Storage**
- **Data Ingestion**
- **Data processing**
- **Model development**
- **Orchestration**

Data Storage

BigQuery

- **BigQuery** is a **fully managed** enterprise **data warehouse** that helps you **manage and analyze** your data with **built-in** features like **machine learning, geospatial analysis, and business intelligence**
- BigQuery's **serverless architecture** lets you use **SQL queries** to answer your organization's biggest questions with **zero infrastructure management.**
- BigQuery's **scalable, distributed** analysis engine lets you query **terabytes** in **seconds** and **petabytes** in **minutes.**
- BigQuery maximizes **flexibility** by **separating** the **compute engine** that analyzes your data from your **storage choices**.
- BigQuery ML **increases development speed** by **eliminating** the need to **move data.**

Components of Vertex AI:

➔ Manage data
➔ Analyze data
➔ Query data
➔ execute machine learning models

Tools to interact with Vertex AI:

➔ Google Cloud console
➔ BigQuery command-line tool
➔ client libraries
➔ REST API and RPC API
➔ ODBC and JDBC drivers
➔ third-party tools and utilitie

Advantages of BigQuery

- Empowers data analysts, to **build and run** models using **existing** business intelligence tools and spreadsheets using their language(**SQL**)
- BigQuery brings ML to the data, no need to **export data** from the data warehouse, increasing speed of model development and innovation.

The need to export and reformat data has the following disadvantages:

- Increases complexity because multiple tools are required.
- Reduces speed because moving and formatting large amounts of data for Python-based ML frameworks takes longer than model training in BigQuery.
- Requires multiple steps to export data from the warehouse, restricting the ability to experiment on your data.
- Can be prevented by **legal** restrictions such as HIPAA guidelines.

BigQueryML supported algos: Linear Regression, Binary Logistic Regression, Multiclass Logistic Regression, K-means, Matrix Factorization, Time Series, BoostedTrees, Deep Neural Network, AutoML Tables, **Tensorflow model importing,** autoencoder.

Cloud Storage

- Cloud Storage is a service for storing your **objects** in Google Cloud.
- An object is an immutable piece of data consisting of a file of any format.

Tools to interact with Vertex AI:

- ➔ Google Cloud console
- ➔ gsutil
- ➔ client libraries
- ➔ REST API and RPC API

BigTable

A fully managed, scalable NoSQL database service for large analytical and operational workloads with up to 99.999% availability.

Features:

- ➔ High throughput at low latency
- ➔ Cluster resizing without downtime
- ➔ Flexible, automated replication to optimize any workload

Data Ingestion: Pub/Sub

- **Pub/Sub** is used for streaming analytics and data integration pipelines to ingest and distribute data.
- Pub/Sub enables you to create systems of event producers and consumers, called **publishers** and **subscribers**. Publishers communicate with subscribers **asynchronously** by broadcasting events.

Types of Pub/Sub:

➔ Pub/Sub service : highly reliable, largest set of integrations, automatic capacity management
➔ Pub/Sub Lite: low cost, lower reliability, requires storage and capacity management

Can integrate with:

➔ Stream processing and data integration products
➔ Monitoring, Alerting and Logging
➔ Authentication and IAM
➔ APIs
➔ Triggers, notifications, and webhooks
➔ Orchestration

```
App -> PubSub -> Dataflow (streaming) -> BigQuery
```

Data Processing tools

Dataproc: Using Dataproc you can migrate your entire deployment of Spark/Hadoop to fully-managed services. Dataproc supports manual provision to clusters

Dataflow: Using Dataflow you can manage and operate batch and stream processing of data. Dataflow supports automatic provision to clusters.

Dataprep: Dataprep was created to solve three major problems, i.e., lack of data visualization, redundant data, and slow processing. Dataprep allows users to explore data visually by transforming the file into CSV, JSON, or in a graphical table format.

Choose:

➔ If systems are Hadoop dependent, or one prefers a hands-on Dev-ops approach, then choose Dataproc.
➔ if you prefer a serverless approach, then select Dataflow.
➔ Dataprep, on the other hand, is used for UI-driven data processing

Credits: https://wisdomplexus.com/blogs/dataproc-vs-dataflow-vs-dataprep/

Model development

Pre-packaged/API solutions

Recommendations AI: Recommendations AI draws on google's experience and expertise in machine learning to deliver personalized recommendations that suit each customer's tastes and preferences across all your touchpoints

Natural Language AI: Derive insights from unstructured text using Google machine learning.

DialogFlow: Lifelike conversational AI with state-of-the-art virtual agents.

Translation AI: Make your content and apps multilingual with fast, dynamic machine translation.

Speech-to-Text: Accurately convert speech into text with an API powered by the best of Google's AI research and technology.

Video AI: Enable powerful content discovery and engaging video experiences.

AutoML

AutoML enables developers with limited machine learning expertise to train high-quality models specific to their business needs.

AutoML Tabular: Automatically build and deploy state-of-the-art machine learning models on structured data.

AutoML Image: Derive insights from object detection and image classification, in the cloud or at the edge.

AutoML Video: Enable powerful content discovery and engaging video experiences.

AutoML Text: Reveal the structure and meaning of text through machine learning.

AutoML Translation: Dynamically detect and translate between languages.

Vertex AI

- **AutoML** lets you train models on image, tabular, text, and video datasets **without writing code.**
- Training in **AI Platform** lets you run **custom** training code.
- **Vertex AI** brings AutoML and AI Platform together into a **unified API, client library, and user interface.**

Components of Vertex AI:

➔ Model training
➔ Model deployment for prediction
➔ Vertex AI Data Labeling
➔ Vertex AI Feature Store
➔ Vertex AI Workbench

Tools to interact with Vertex AI:

➔ Google Cloud console
➔ Cloud Client Libraries
➔ REST API
➔ Deep Learning VM Images
➔ Deep Learning Containers

ML Workflow Orchestration

Kubeflow Pipilines

Kubeflow Pipelines is a platform for building and deploying portable, scalable machine learning (ML) workflows based on **Docker containers**.

The following are the goals of Kubeflow Pipelines:

- End-to-end orchestration: enabling and simplifying the orchestration of machine learning pipelines.
- Easy experimentation: making it easy for you to try numerous ideas and techniques and manage your various trials/experiments.
- Easy re-use: enabling you to re-use components and pipelines to quickly create end-to-end solutions without having to rebuild each time.

Tensorflow Extended

- TFX is a Google-production-scale machine learning (ML) platform based on TensorFlow. It provides a configuration framework and shared libraries to integrate common components needed to define, launch, and monitor your machine learning system..
- When you're ready to move your models from research to production, use TFX to create and manage a production pipeline.
- A TFX pipeline is a sequence of components that implement an ML pipeline which is specifically designed for scalable, high-performance machine learning tasks. Components are built using TFX libraries which can also be used individually.

Vertex AI pipelines

- Vertex AI Pipelines helps you to automate, monitor, and govern your ML systems by orchestrating your ML workflow in a serverless manner, and storing your workflow's artifacts using Vertex ML Metadata.
- By storing the artifacts of your ML workflow in Vertex ML Metadata, you can analyze the lineage of your workflow's artifacts
- for example, an ML model's lineage may include the training data, hyperparameters, and code that were used to create the model.

Cloud Composer

- A fully managed workflow orchestration service built on Apache Airflow.
- Author, schedule, and monitor pipelines that span across hybrid and multi-cloud environments

Best Practices

Data preparation Best practices

- ➔ **Avoid target leakage:** when your training data includes predictive information that is not available when you ask for a prediction.
- ➔ **Avoid training-serving skew:** when you generate your training data differently than you generate the data you use to request predictions.
- ➔ **Training-serving skew and data distribution:** the difference in production data versus training data must be reflected in the difference between the validation data split and the training data split, and between the testing data split and the validation data split.
- ➔ **Provide a time-signal:** If the underlying pattern in your data is likely to shift over time (it is not randomly distributed in time), make sure you provide that information.
- ➔ Avoid missing values where possible
- ➔ Make sure your categorical features are accurate and clean.
- ➔ Provide sufficient training data for the minority class.
- ➔ If you have imbalanced classes, you might want to assign a manual split to make sure enough rows with the minority outcomes are included in every split.
- ➔ **Avoid bias:** Make sure that your training data is representative of the entire universe of potential data that you will be making predictions for.
- ➔ **Provide enough training data:** The more features (columns) you use to train your model, the more data (rows) you need to provide.

Source: Best practices for creating training data | AutoML Tables | Google Cloud

Data Processing Best Practices

- ➔ Use TensorFlow Extended when leveraging TensorFlow ecosystem .
- ➔ Use BigQuery to process tabular data.
- ➔ Use Dataflow to process unstructured data.
- ➔ Use managed datasets to link data to your models.
- ➔ If systems are Hadoop dependent, or one prefers a hands-on Dev-ops approach, then choose **Dataproc**.
- ➔ if you prefer a serverless approach, then select **Dataflow**.
- ➔ **Dataprep**, on the other hand, is used for UI-driven data processing

ML development environment Best Practices

BigQuery ML:

- All of your data is contained in BigQuery. BigQuery ML requires tabular data.
- You are comfortable with SQL.
- The set of models available in BigQuery ML matches the problem you are trying to solve.

AutoML:

- Your problem fits into one of the types that AutoML supports.
- Your data matches the format and fits within the limits set by each type of AutoML model.
- For AutoML models, your model can be served from Google Cloud.For text, video, or tabular models, your model can tolerate inference latencies > 100ms.

Vertex AI custom-trained models:

- Your problem does not match the criteria listed in this table for BigQuery ML or AutoML.
- You are already running training on-premises or on another cloud platform, and you need consistency across the platforms.
- (Optional) You are using TensorFlow Extended.

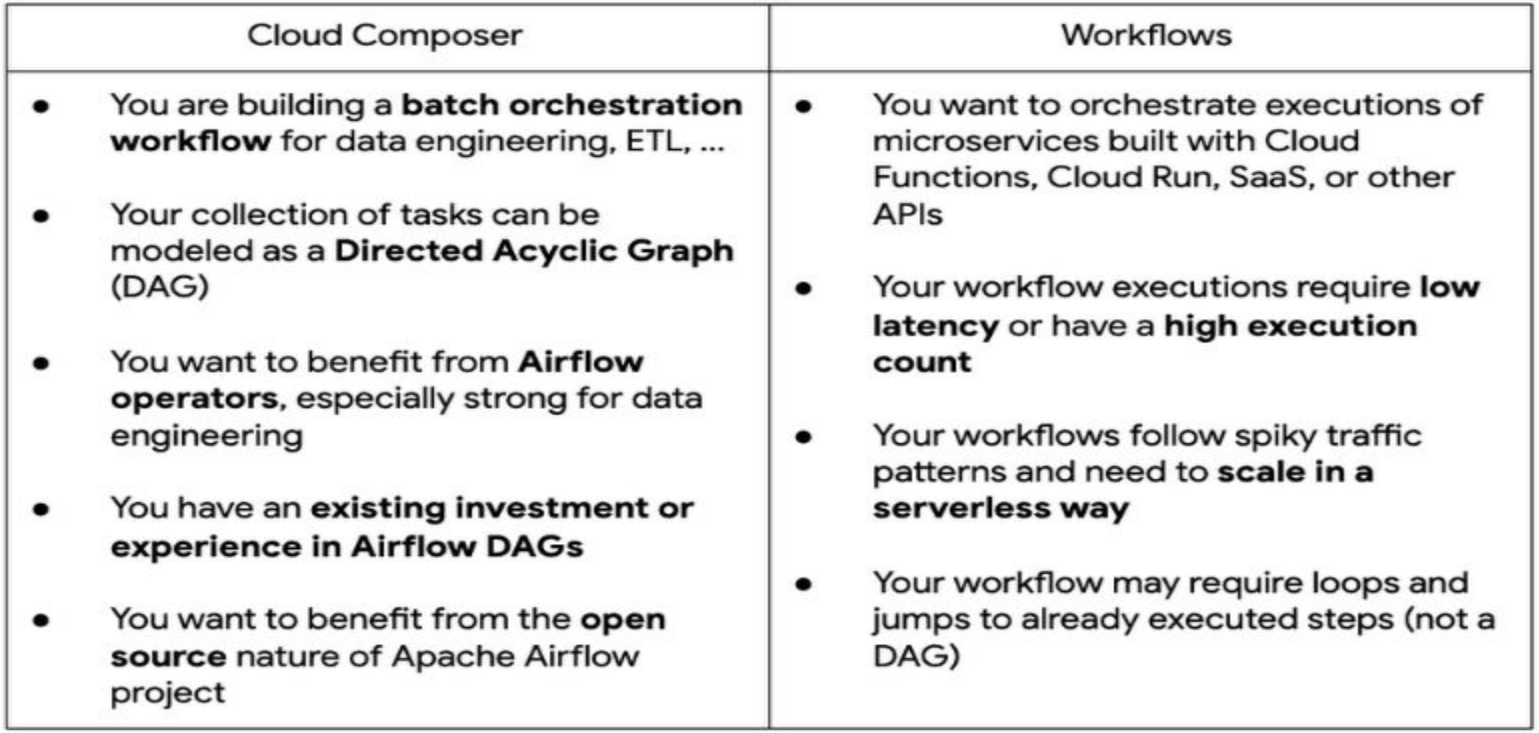
Machine learning development Best Practices

- ➔ Prepare training data.
- ➔ Store tabular data in BigQuery.
- ➔ Store image, video, audio and unstructured data on Cloud Storage.
- ➔ Use Vertex Data Labeling for unstructured data.
- ➔ Use Vertex AI Feature Store with structured data.
- ➔ Avoid storing data in block storage like Network File Systems or on virtual machine (VM) hard disks.
- ➔ Use Vertex AI TensorBoard to visualize experiments.
- ➔ Train a model within a user-managed notebooks instance for small datasets.
- ➔ For larger datasets or for distributed training. Use the Vertex training service.
- ➔ Maximize your model's predictive accuracy with hyperparameter tuning.
- ➔ Use a user-managed notebooks instance to evaluate and understand your models.
- ➔ Use feature attributions(Vertex Explainable AI) to gain insights into model predictions.

Workflow orchestration

- ➔ **Vertex AI pipelines** is a fully managed service that allows you to retrain your models as often as necessary. While retraining enables your models to adapt to changes and maintain performance over time, consider how much your data will change when choosing the optimal model retraining cadence.
- ➔ If you're using TensorFlow, use **TensorFlow Extended** to define your pipeline and the operations for each step, then execute it on Vertex AI's serverless pipelines system.
- ➔ For all other frameworks, use the **Kubeflow Pipelines** with Vertex AI Pipelines. Use Vertex AI to launch and interact with the platform.

Workflow orchestration



Cloud Composer	Workflows
• You are building a **batch orchestration workflow** for data engineering, ETL, ... • Your collection of tasks can be modeled as a **Directed Acyclic Graph** (DAG) • You want to benefit from **Airflow operators**, especially strong for data engineering • You have an **existing investment or experience in Airflow DAGs** • You want to benefit from the **open source** nature of Apache Airflow project	• You want to orchestrate executions of microservices built with Cloud Functions, Cloud Run, SaaS, or other APIs • Your workflow executions require **low latency** or have a **high execution count** • Your workflows follow spiky traffic patterns and need to **scale in a serverless way** • Your workflow may require loops and jumps to already executed steps (not a DAG)

Responsible AI Practices

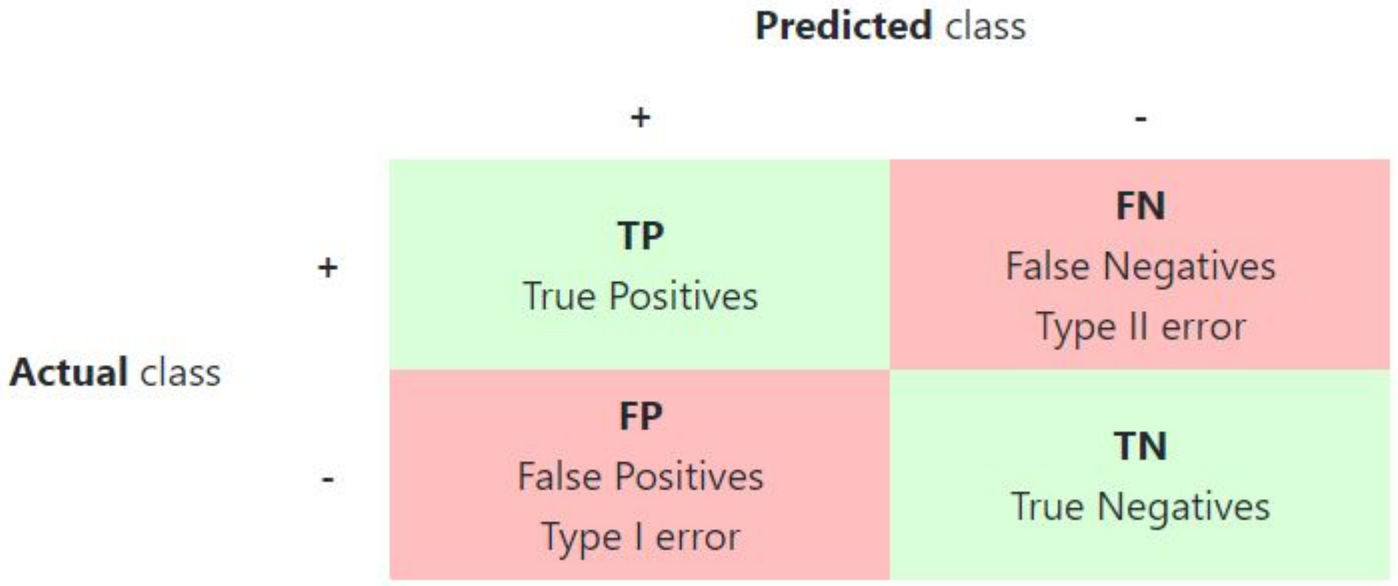
The development of AI is creating new opportunities to improve the lives of people around the world, from business to healthcare to education. It is also raising new questions about the best way to build fairness, interpretability, privacy, and security into these systems.

- ➢ Use a human-centered approach
- ➢ Identify multiple metrics to assess training and monitoring
- ➢ When possible, directly examine your raw data
- ➢ Understand the limitations of your dataset and model
- ➢ Test, Test, Test
- ➢ Continue to monitor and update the system after deployment

Machine Learning Concepts

Machine Learning Theory

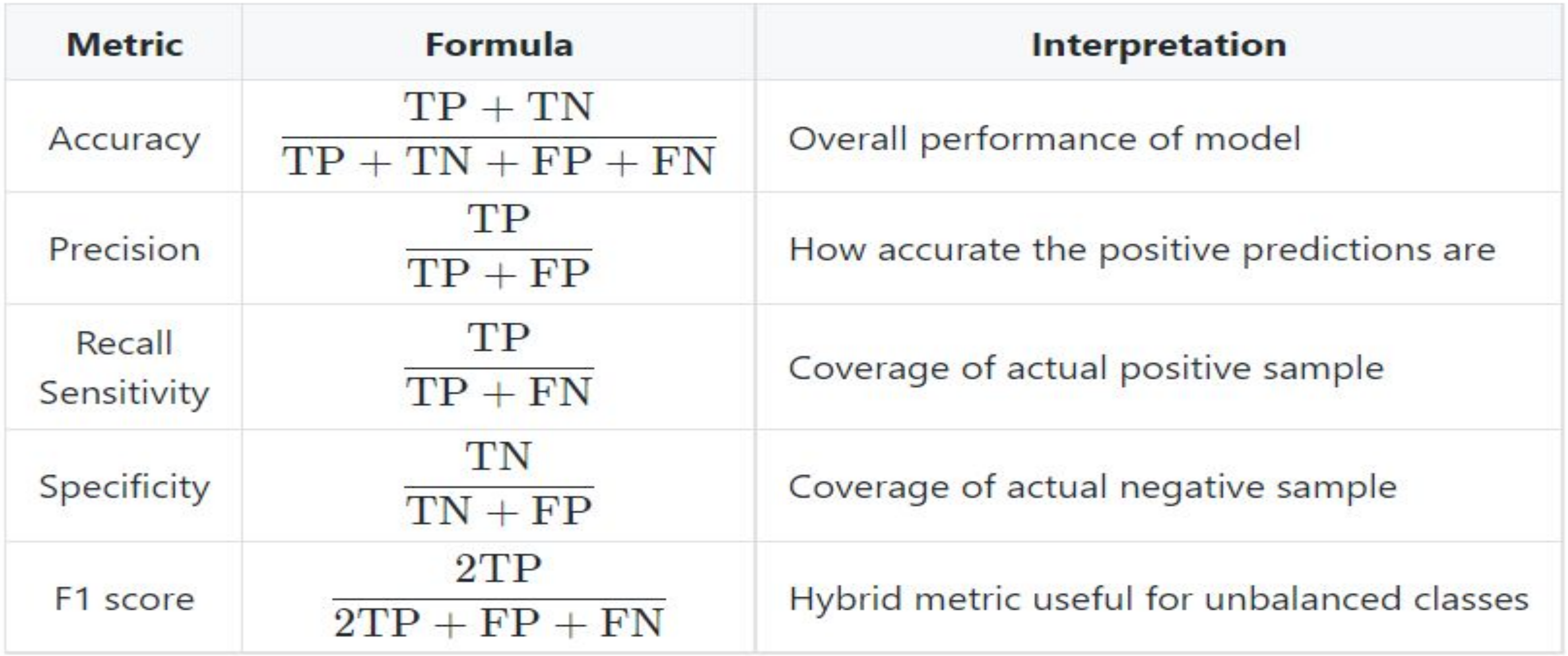
- **Confusion matrix**



		Predicted class	
		+	-
Actual class	+	**TP** True Positives	**FN** False Negatives Type II error
	-	**FP** False Positives Type I error	**TN** True Negatives

Credits: CS 229 - Machine Learning Tips and Tricks Cheatsheet (stanford.edu)

Machine Learning Theory

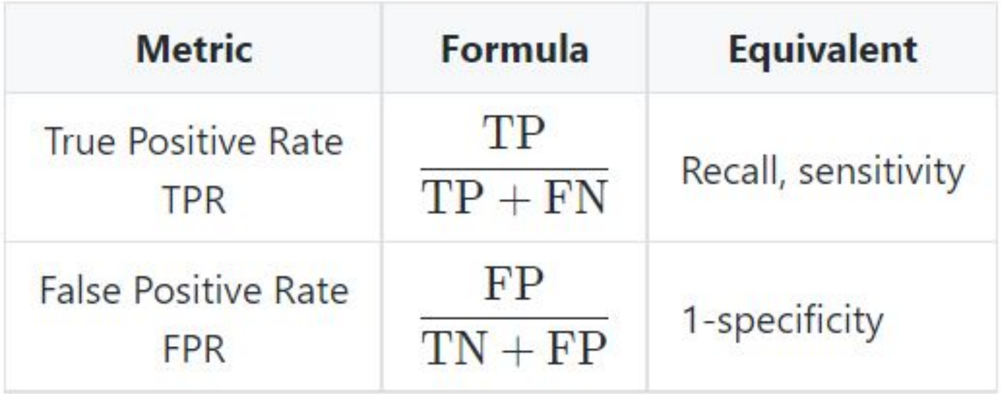


Metric	Formula	Interpretation
Accuracy	$\frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$	Overall performance of model
Precision	$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$	How accurate the positive predictions are
Recall Sensitivity	$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$	Coverage of actual positive sample
Specificity	$\frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$	Coverage of actual negative sample
F1 score	$\frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$	Hybrid metric useful for unbalanced classes

Machine Learning Theory

ROC: The receiver operating curve, also noted ROC, is the plot of TPR versus FPR by varying the threshold. These metrics are are summed up in the table below:

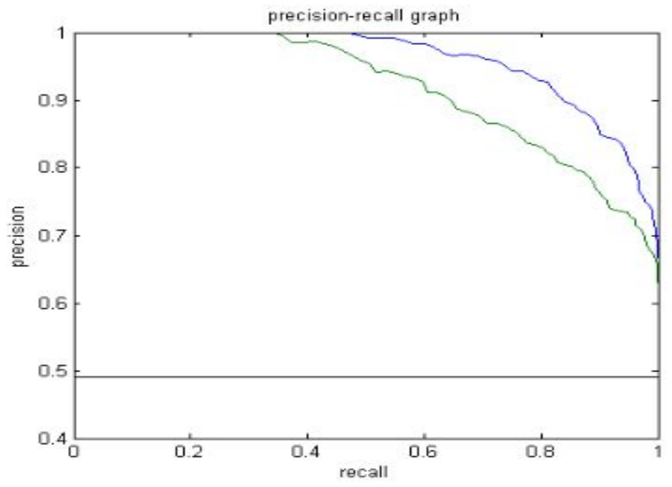
AUC: The area under the receiving operating curve, also noted AUC or AUROC, is the area below the ROC as shown in the following figure:

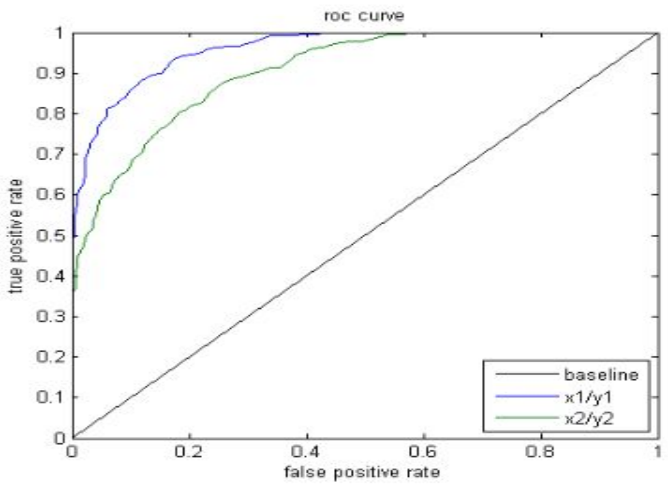


Metric	Formula	Equivalent
True Positive Rate TPR	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$	Recall, sensitivity
False Positive Rate FPR	$\frac{\mathrm{FP}}{\mathrm{TN}+\mathrm{FP}}$	1-specificity

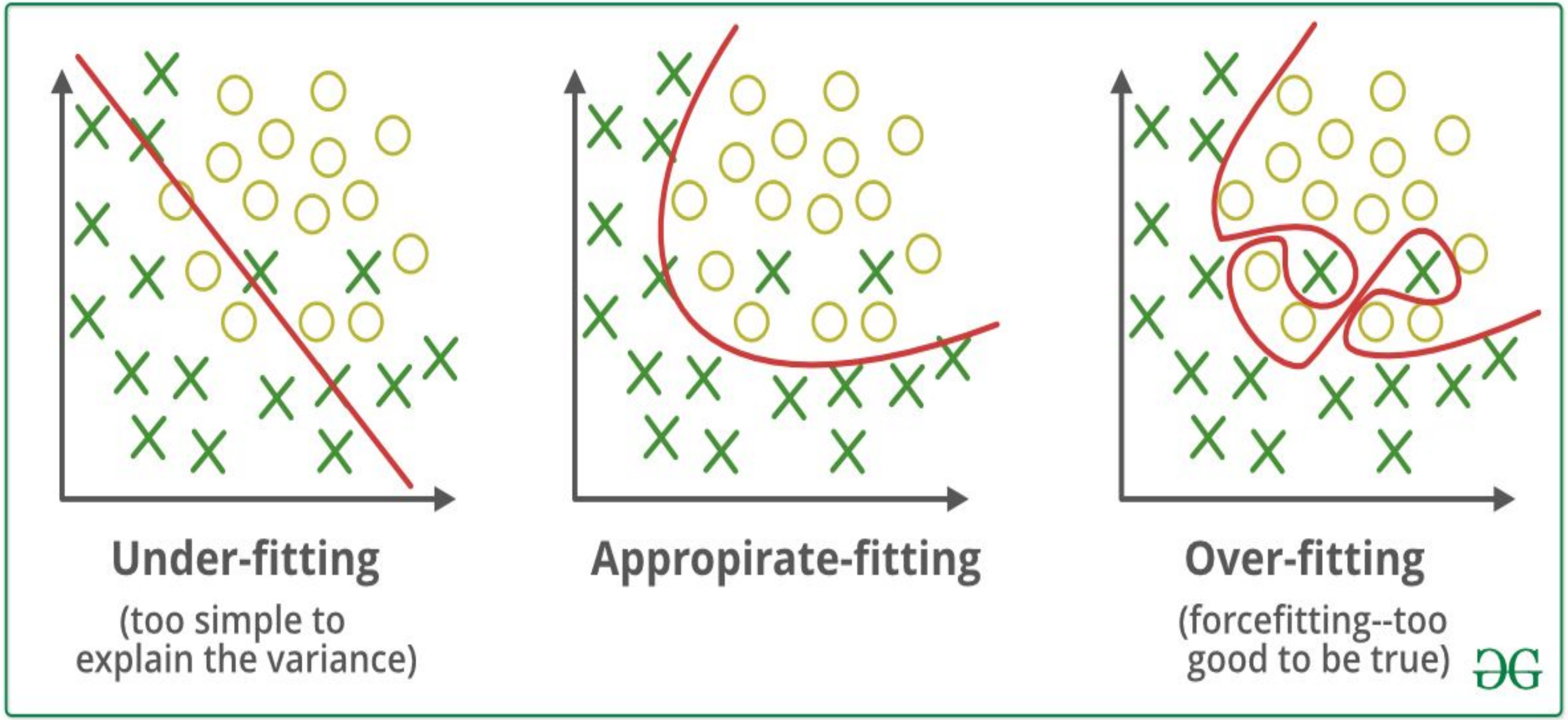
Machine Learning Theory

Precision-recall curve: A PR curve is simply a graph with Precision values on the y-axis and Recall values on the x-axis.





Underfitting, Overfitting


Under-fitting
(too simple to explain the variance)
Appropirate-fitting
Over-fitting
(forcefitting--too good to be true)

Regularization

Regularization is a technique used to reduce the errors by **fitting** the function appropriately on the given training set and **avoid overfitting**.

L1 reguralization(Lasso):

`L = y log (wx + b) + (1 - y)log(1 - (wx + b)) + lambda*||w||`$_1$

L2 reguralization(Ridge):

`L = y log (wx + b) + (1 - y)log(1 - (wx + b)) + lambda*||w||`$_2$

Data transformation

Numeric data: You may need to apply two kinds of transformations to numeric data:

- **Normalizing** - transforming numeric data to the same scale as other numeric data.
- **Bucketing** - transforming numeric (usually continuous) data to categorical data.

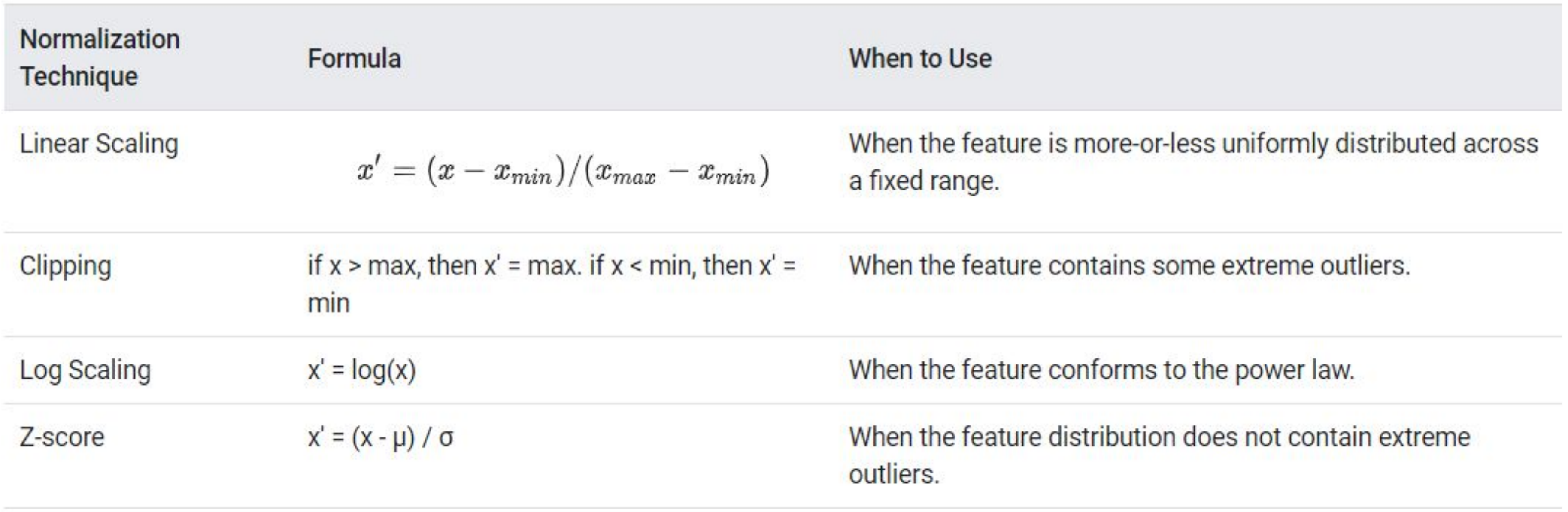
Categorical data: You may need to apply two kinds of transformations to categorical data:

- A unique feature for each category(one-hot encoding)
- Indexed features(label encoding)
- Embeddings

Feature-crossing: A **feature cross** is a **synthetic feature** formed by multiplying (crossing) two or more features. Crossing combinations of features can provide predictive abilities beyond what those features can provide individually. eg:X1*X2

Data Normalization

The goal of normalization is to transform features to be on a similar scale. This improves the performance and training stability of the model.



Normalization Technique	Formula	When to Use
Linear Scaling	$x' = (x - x_{min})/(x_{max} - x_{min})$	When the feature is more-or-less uniformly distributed across a fixed range.
Clipping	if $x > max$, then $x' = max$. if $x < min$, then $x' = min$	When the feature contains some extreme outliers.
Log Scaling	$x' = \log(x)$	When the feature conforms to the power law.
Z-score	$x' = (x - \mu) / \sigma$	When the feature distribution does not contain extreme outliers.

Machine Learning Theory

- ML algorithms cheat sheet

Sample Questions