ML OPs Module notes

Git And Github

Git

Git is a version control system that lets you keep track of all the modifications you make to your code.
This means if a new feature is causing some errors in your code, you can always roll back to a previous stable version.

But Git isn't any VCS, it's a distributed VCS which means that every collaborator of the project will have a history of the changes made on their local machine. It is distributed so you can access your code files from another computer – and so can other developers.

GitHub

GitHub is a widely used platform for version control that uses Git at its core.
It lets you host the remote version of your project from where all the collaborators can have access to it.

Differences between Git and GitHub
Git is a version control system that manages and keeps track of your code. GitHub, on the other hand, is a service that lets you host, share, and manage your code files on the internet.

GitHub uses Git underneath, and lets you manage your Git repositories or folders easily on its platform.
So Git is the actual version control system and GitHub is the platform where you host your code.

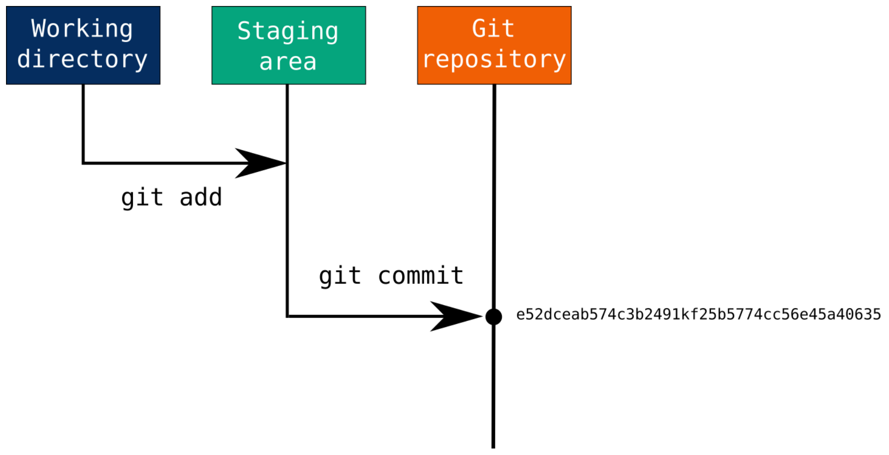
Git commands to type in terminal:

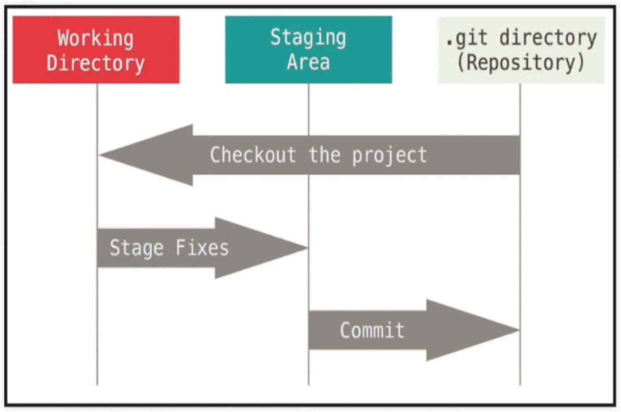
- git init (initializes repository (tells git to look for files in this folder))ONCE PER PROJECT
- git add . (stages files for a commit; the dot adds all files in the directory)
- git commit -m "comment" (creates a commit aka save. This is a save command in git; save often and frequently to track your changes)
- the -m "message" and -m :message commits to the master branch, e.g. the main branch
- git log (shows history of changes made by all contributing members)
- git status (shows you whats added ("staged") but not yet committed, and tells you if any files have changed since last commit)

- git commit -am "comment" (adds and commits in one step)
- git add -u (tells commit to remove files you've deleted)
- git push origin master (sends updates to github)
- git pull origin master (pulls updates from github)

Git Staging Environment
One of the core functions of Git is the concepts of the Staging Environment, and the Commit.

As you are working, you may be adding, editing and removing files. But whenever you hit a milestone or finish a part of the work, you should add the files to a Staging Environment.
Git has three main states that your files can reside in: modified, staged, and committed:
Modified means that you have changed the file but have not committed it to your database yet.
Staged means that you have marked a modified file in its current version to go into your next commit snapshot.
Staged files are files that are ready to be committed to the repository you are working on.





For more commands and what they do refer to [github cheat sheet]

Streamlit

Streamlit is an open source app framework in Python language. It helps us create web apps for data science and machine learning in a short time. All in pure Python. No front-end experience required. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc.
developing a dashboard for your machine learning solution has been made incredibly easy

Pros:

- Streamlit is really easy to start prototyping in, everything is done in python
- Streamlit is integrated with Python. The usual visualization libraries you may use like matplotlib and plotly can easily be used with Streamlit. it is compatible with major Python libraries used in Data Science such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc.
- Less code is needed to create a beautiful application
- No callbacks are needed since widgets are treated as variables

Cons:

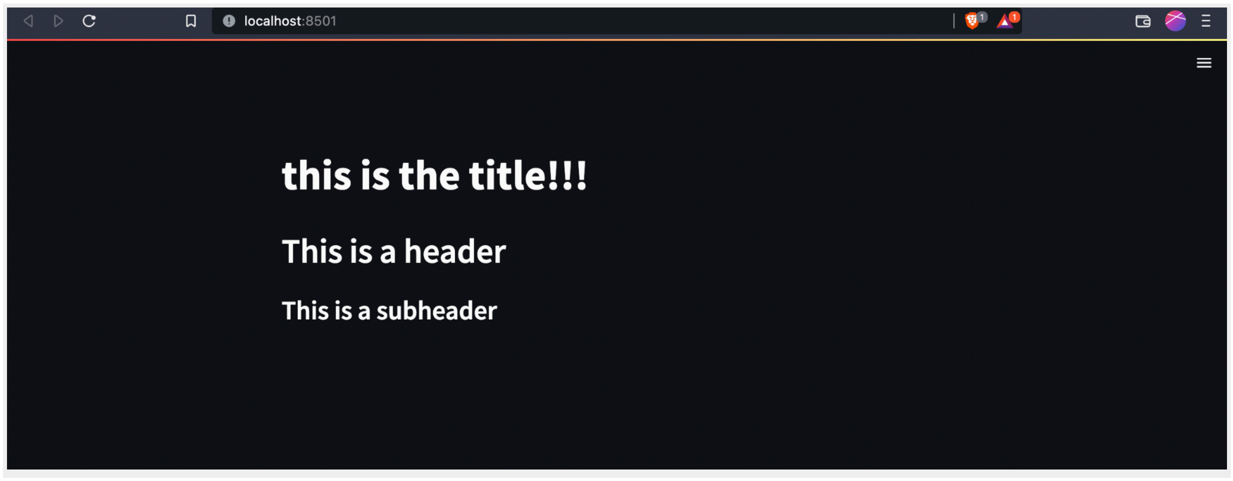
- Streamlit is constrained in adding options like session management and secure authentication.
- Every time you perform an action in Streamlit, such as clicking a button, all Python code is re-executed
- Cannot create apis, the only way to use it is with its own frontend

Understanding the Streamlit basic functions

Streamlit lays out everything in the form of elements and widgets. The elements that are interactable are widgets.

```python
# import module
import streamlit as st

# Title
st.title("this is the title!!!")
```

```python
# Header
st.header("This is a header")

# Subheader
st.subheader("This is a subheader")
```



Now to run a streamlit app run

```
streamlit run your_script.py
```

As soon as you run the script as shown above, a local Streamlit server will spin up and your app will open in a new tab in your default web browser. The app is your canvas, where you'll draw charts, text, widgets, tables, and more.

What gets drawn in the app is up to you. For example st.text writes raw text to your app, and st.line_chart draws — you guessed it — a line chart. Refer to API documentation to see all commands that are available to you.

Data flow

Streamlit's architecture allows you to write apps the same way you write plain Python scripts. To unlock this, Streamlit apps have a unique data flow: any time something must be updated on the screen, Streamlit reruns your entire Python script from top to bottom.

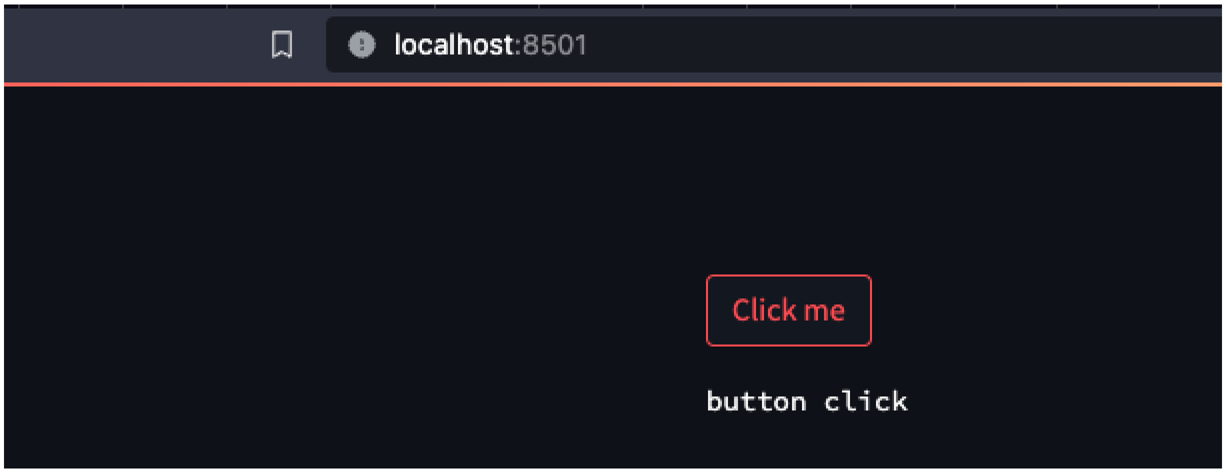
This can happen in two situations:

- Whenever you modify your app's source code.
- Whenever a user interacts with widgets in the app. For example, when dragging a slider, entering text in an input box, or clicking a button.

Button

Here we create a button which shows a text after it is clicked

```
# Create a button, that when clicked, shows a text
if(st.button("Click me")):
    st.text("button click")
```



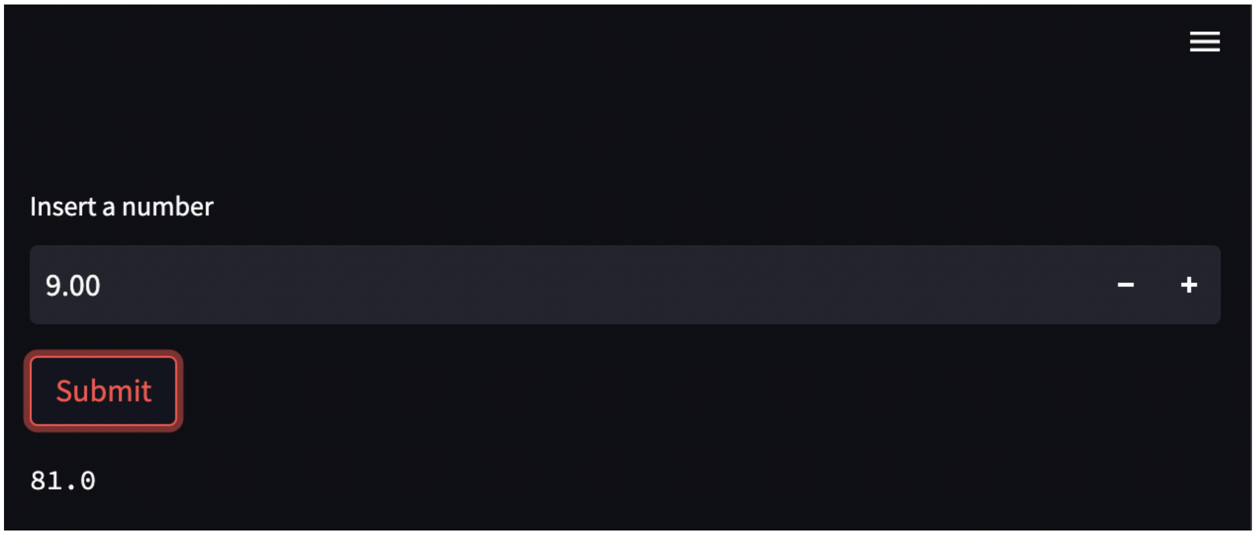
```
button click
```

Now we'll see how to interact with elements, make function calls and make elements work together.

- A very simple function that returns the square of the number entered.

Create a number input widget and store the value in num.
Create a button 'calculate square'. Which when pressed will pass the num to sqr function and create a text to show that result

```
def sqr(num):

     return num*num


num = st.number_input('Insert a number')

# display the name when the submit button is clicked
# .title() is used to get the input text string
if(st.button(Calculate Square)):
    result = sqr(num)
    st.text(result)
```

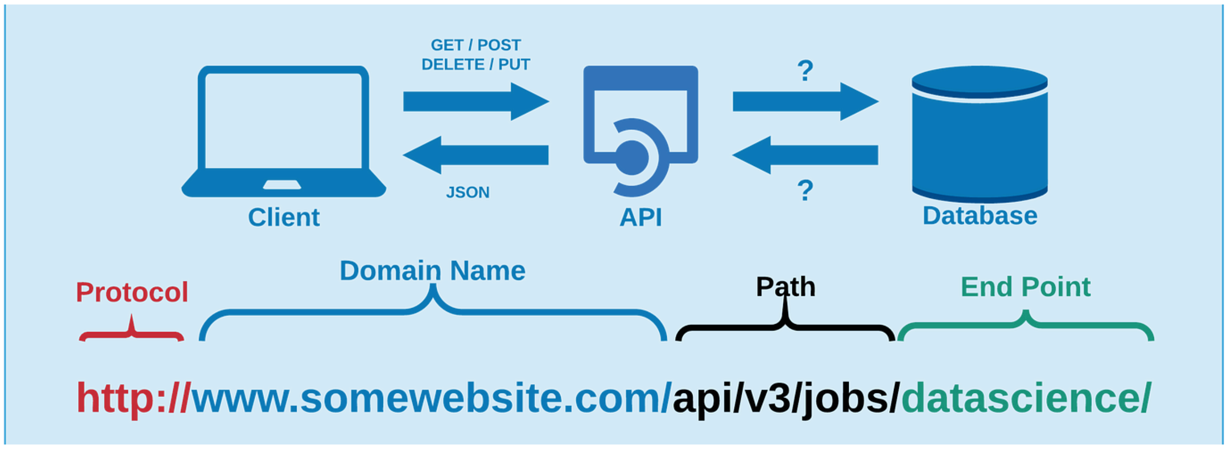


```
81.0
```

Streamlit cheat sheet:
https://dev.to/ramanbansal/the-ultimate-streamlit-cheatsheet-for-2023-1pln
Streamlit docs: https://docs.streamlit.io/library/cheatsheet

Web APIs - Flask

Web APIs are tools for making information and application functionality accessible over the internet.
In programming more generally, the term API, short for Application Programming Interface, refers to a part of a computer program designed to be used or manipulated by another program, as opposed to an interface designed to be used or manipulated by a human. Computer programs frequently need to communicate amongst themselves or with the underlying operating system, and APIs are one way they do it. In this tutorial, however, we'll be using the term API to refer specifically to web APIs.



API Terminology

When using or building APIs, you will encounter these terms frequently:

- **HTTP (Hypertext Transfer Protocol)** is the primary means of communicating data on the web. HTTP implements a number of “methods,” which tell which direction data is moving and what should happen to it. The two most common are GET, which pulls data from a server, and POST, which pushes new data to a server.
- **URL (Uniform Resource Locator)** - An address for a resource on the web, such as https://programminghistorian.org/about. A URL consists of a protocol (http://), domain (programminghistorian.org), and optional path (/about). A URL describes the location of a specific resource, such as a web page. When reading about APIs, you may see the terms URL, request, URI, or endpoint used to describe adjacent ideas. This tutorial will prefer the terms URL and request to avoid complication. You can follow a URL or make a GET request in your browser, so you won’t need any special software to make requests in this tutorial.
- **JSON (JavaScript Object Notation)** is a text-based data storage format that is designed to be easy to read for both humans and machines. JSON is generally the most common format for returning data through an API, XML being the second most common.
- **REST (REpresentational State Transfer)** is a philosophy that describes some best practices for implementing APIs. APIs designed with some or all of these principles in mind are called REST APIs. While the API outlined in this lesson uses some REST principles, there is a great deal of disagreement around this term. For this reason, I do not describe the example APIs here as REST APIs, but instead as web or HTTP APIs.

Flask

is a web framework for Python, meaning that it provides functionality for building web applications, including managing HTTP requests and rendering templates.
To install flask, use the command: **`pip install flask`**

```
from flask import Flask
```

```
import pickle

app = Flask(__name__)
```

- We've imported `Flask` class, to which we'll pass the name of the app.We can now develop API endpoints as per our need.
- For a particular request we need to send it via some `url`, and corresponding to that `url`, we need some piece of code, that will respond when invoked.
- For this, we've written a function `ping()`, in which we will write a simple message written
- To define the `url`, we'll use a **decorator** `@app.route` to which you'll pass the url. You can also specify the type of url request such as **get**, **post**, etc.
- A decorator in python allows a user to add new functionality to an existing object without modifying its structure.

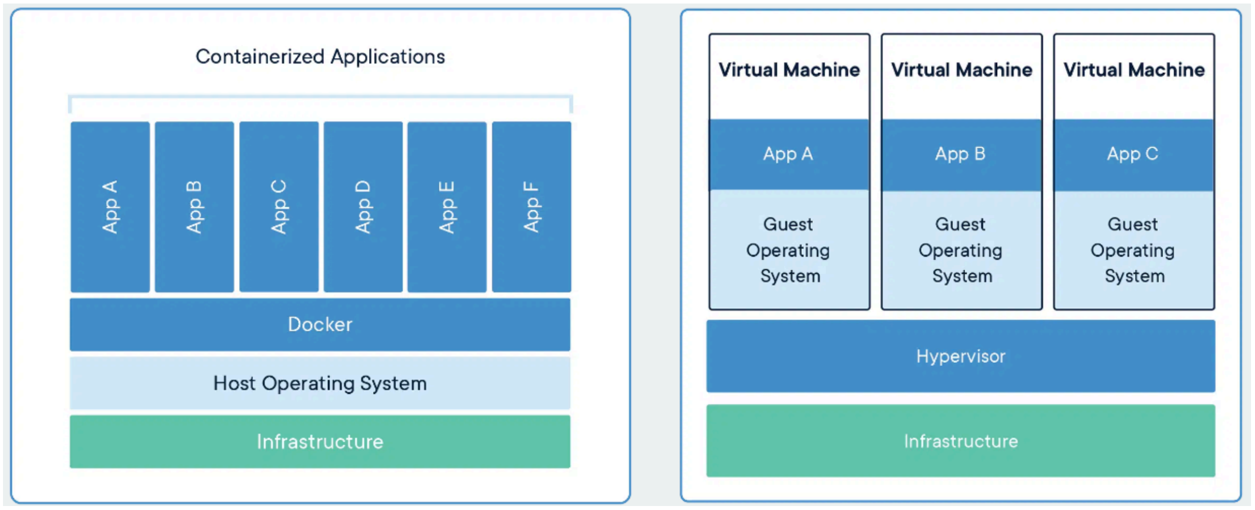
```
from flask import Flask
import pickle

app = Flask(__name__)

@app.route("/ping", methods=['GET'])
def ping():
    return {"message": "Hi there, I'm working!!"}
```

Flask cheatsheet:
https://drive.google.com/file/d/1705hDOsBCShGwEGSOetvfejYeUFtWFzl/view?usp=sharing
Flask docs: https://flask.palletsprojects.com/en/2.3.x/

Containerization & Docker

containerization

In recent years, a technology known as "containerization" has become very popular. As a concept, it refers to packaging up all that is required to run an application, including runtime dependencies, configuration files, and settings, and then running the application in an isolated, pre prepared environment. This is similar to running separate applications in entirely separate virtual machines, but is generally much more lightweight, as the containerized application is run directly on the host operating system, and is simply isolated using features the host already provides.



Docker

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

Container images become containers at runtime and in the case of Docker containers – images become containers when they run on Docker Engine. Available for both Linux and Windows-based applications, containerized software will always run the same, regardless of the infrastructure. Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging.

Containers can be created and built on-site when used to run your own software, or you can make use of already published containers for third party dependencies, hosted on public container registries, the most well known of which is likely Docker's own hub.

We must be familiar with some terminologies to get started with Docker.

- Container - For encapsulating the required software, we need this running instance.
- Image - images are pre-built packages containing an operating system, software, and configurations - like a blueprint for your app.
- Port - A TCP/UDP port. This can be exposed to the outer world (accessible from the host OS) or connected to the other containers.
- Volume - The shared folder can be described through this.
- Registry - It is the docker image storing server

Workflow of Using Docker::

- When starting using Docker, we would have an application ready to be deployed, the data, and the trained model.
- We would then write the code for something known as Dockerfile, which is nothing but a plain text file that contains instructions that docker uses to pack your app into an image.
- Once a dockerfile is ready, the next step would be to build an image.
- For using this image, we would run the image that will basically create a container that will run your app.

Dockerfile

To build a Docker image, you need to create a Dockerfile. It is a plain text file with instructions and arguments. Here is the description of the instructions we're going to use in our next example:

- FROM — set base image, docker offers a lot of pre-built containers for the developers to use, so that they don't have to start again from scratch.
- RUN — execute command in container like upgrade pip and pip install
- ENV — set environment variable
- WORKDIR — set working directory
- VOLUME — create mount-point for a volume
- CMD — set executable for container

You can check the Dockerfile reference for more details.
Example:

```
FROM python:3.8-slim-buster

WORKDIR /flask-docker

RUN python3 -m pip install --upgrade pip
COPY requirements.txt requirements.txt
RUN pip3 install -r requirements.txt

COPY . .

CMD ["python3", "-m", "flask", "run", "--host=0.0.0.0"]
```

Deploy docker to ecs

ECS - Elastic Container Service

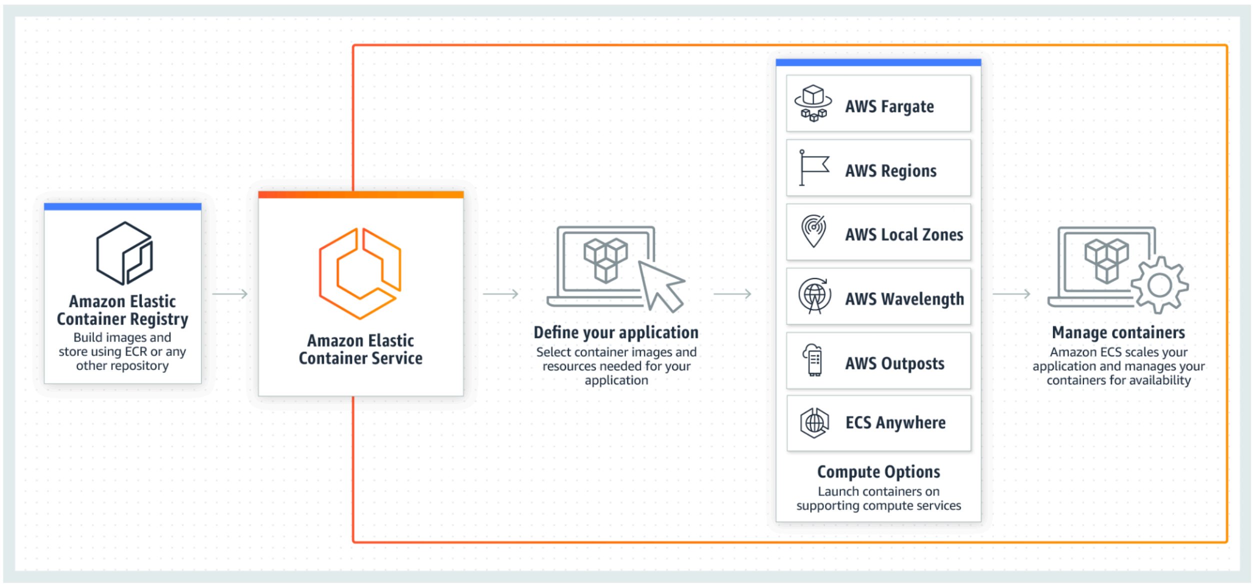
Amazon Elastic Container Service (ECS) is a cloud computing service in Amazon Web Services (AWS) that manages containers and lets developers run applications in the cloud without having to configure an environment for the code to run in. It enables developers with AWS accounts to deploy and manage scalable applications that run on groups of servers called clusters through API calls and task definitions.

ECR - Elastic container repository

Amazon Elastic Container Registry (Amazon ECR) is an AWS managed container image registry service that is secure, scalable, and reliable. Amazon ECR supports private repositories with resource-based permissions using AWS IAM.

This is where we will store our docker image, and all other services will retrieve the resources and docker images from here itself.

To create an image repository
A repository is where you store your Docker or Open Container Initiative (OCI) images in Amazon ECR. Each time you push or pull an image from Amazon ECR, you specify the repository and the registry location which informs where to push the image to or where to pull it from.


Amazon Elastic Container Registry
Build images and store using ECR or any other repository
Amazon Elastic Container Service
Define your application
Select container images and resources needed for your application
AWS Fargate
AWS Regions
AWS Local Zones
AWS Wavelength
AWS Outposts
ECS Anywhere
Compute Options
Launch containers on supporting compute services
Manage containers
Amazon ECS scales your application and manages your containers for availability

AWS Fargate

AWS Fargate is a serverless, pay-as-you-go compute engine that lets you focus on building applications without managing servers, it ia a serverless compute for containers, which eliminates the need to configure and manage control plane, nodes, and instances.

CI/CD with github actions to ECS

Continuous integration (CI)

Continuous integration is the practice of integrating all your code changes into the main branch of a shared source code repository early and often, automatically testing each change when you commit or merge them, and automatically kicking off a build. With continuous integration, errors and security issues can be identified and fixed more easily, and much earlier in the software development lifecycle.

continuous delivery/continuous deployment (CD)

Continuous delivery is the automated delivery of completed code to environments like testing and development. CD provides an automated and consistent way for code to be delivered to these environments.

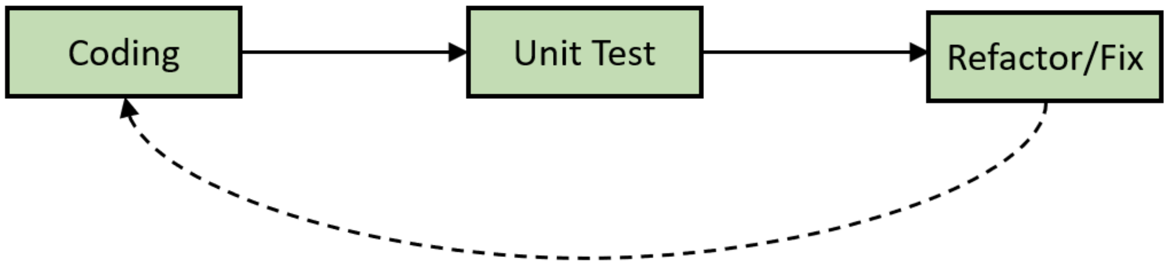
CD is the next step of CI. Every change that passes the automated tests is automatically placed in production, resulting in many production deployments.

CI/CD allows organizations to ship software quickly and efficiently. CI/CD facilitates an effective process for getting products to market faster than ever before, continuously delivering code into production, and ensuring an ongoing flow of new features and bug fixes via the most efficient delivery method.

Unit tests

"Unit tests are typically automated tests written and run by software developers to ensure that a section of an application (known as the "unit") meets its design and behaves as intended"

Essentially, a unit test is a method that instantiates a small portion of our application and verifies its behavior independently from other parts. A typical unit test contains 3 phases: First, it initializes a small piece of an application it wants to test (also known as the system under test, or SUT), then it applies some stimulus to the system under test (usually by calling a method on it), and finally, it observes the resulting behavior. If the observed behavior is consistent with the expectations, the unit test passes, otherwise, it fails, indicating that there is a problem somewhere in the system under test.



Basic Unit Test Life Cycle

Pytest

Pytest is possibly the most widely used Python testing framework around - this means it has a large community to support you whenever you get stuck. It's an open-source framework that enables developers to write simple, compact test suites while supporting unit testing, functional testing, and API testing.

- In Pytest, we can use the fixture function as an input parameter of the test function, and that input parameter is already the return object.
- We have to indicate that the function is a fixture with @pytest.fixture. These specific Python decorations let us know that the next method is a pytest fixture.

Github actions

GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows you to automate your build, test, and deployment pipeline. You can create workflows that build and test every pull request to your repository, or deploy merged pull requests to production.

GitHub Actions goes beyond just DevOps and lets you run workflows when other events happen in your repository. For example, you can run a workflow to automatically add the appropriate labels whenever someone creates a new issue in your repository.

GitHub provides Linux, Windows, and macOS virtual machines to run your workflows, or you can host your own self-hosted runners in your own data center or cloud infrastructure.

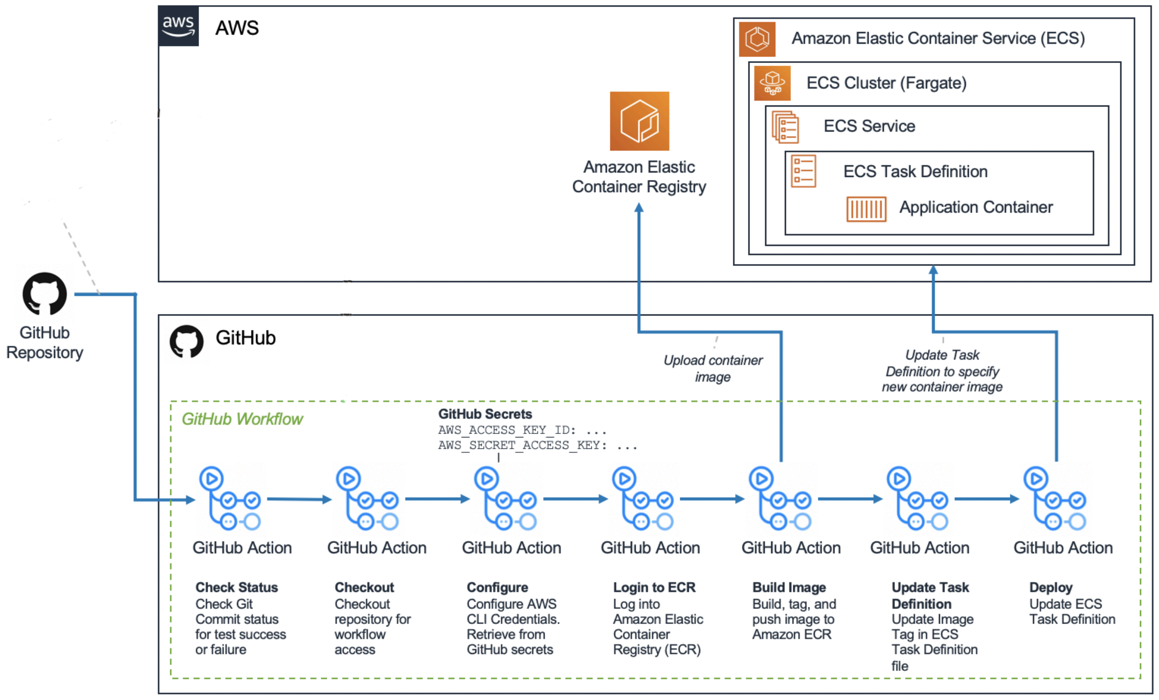
The components of GitHub Actions

- You can configure a GitHub Actions workflow to be triggered when an event occurs in your repository, such as a pull request being opened or an issue being created. Your workflow contains one or more jobs which can run in sequential order or in parallel. Each job will run inside its own virtual machine runner, or inside a container, and has one or more steps that either run a script that you define or run an action, which is a reusable extension that can simplify your workflow.

- **Workflows** A workflow is a configurable automated process that will run one or more jobs. Workflows are defined by a YAML file checked in to your repository and will run when triggered by an event in your repository, or they can be triggered manually, or at a defined schedule.Workflows are defined in the .github/workflows directory in a repository, and a repository can have multiple workflows, each of which can perform a different set of tasks. For example, you can have one workflow to build and test pull requests, another workflow to deploy your application every time a release is created, and still another workflow that adds a label every time someone opens a new issue.

- **Events** An event is a specific activity in a repository that triggers a workflow run. For example, activity can originate from GitHub when someone creates a pull request, opens an issue, or pushes a commit to a repository. You can also trigger a workflow run on a schedule, by posting to a REST API, or manually.

- **Jobs** A job is a set of steps in a workflow that execute on the same runner. Each step is either a shell script that will be executed, or an action that will be run. Steps are executed in order and are dependent on each other. Since each step is executed on the same runner, you can share data from one step to another. For example, you can have a step that builds your application followed by a step that tests the application that was built.


AWS
Amazon Elastic Container Service (ECS)
ECS Cluster (Fargate)
ECS Service
ECS Task Definition
Application Container
Amazon Elastic Container Registry
GitHub Repository
GitHub
Upload container image
Update Task Definition to specify new container image
GitHub Workflow
GitHub Secrets
AWS_ACCESS_KEY_ID: ...
AWS_SECRET_ACCESS_KEY: ...
GitHub Action
Check Status
Check Git Commit status for test success or failure
GitHub Action
Checkout
Checkout repository for workflow access
GitHub Action
Configure
Configure AWS CLI Credentials. Retrieve from GitHub secrets
GitHub Action
Login to ECR
Log into Amazon Elastic Container Registry (ECR)
GitHub Action
Build Image
Build, tag, and push image to Amazon ECR
GitHub Action
Update Task Definition
Update Image Tag in ECS Task Definition file
GitHub Action
Deploy
Update ECS Task Definition

Experiment tracking

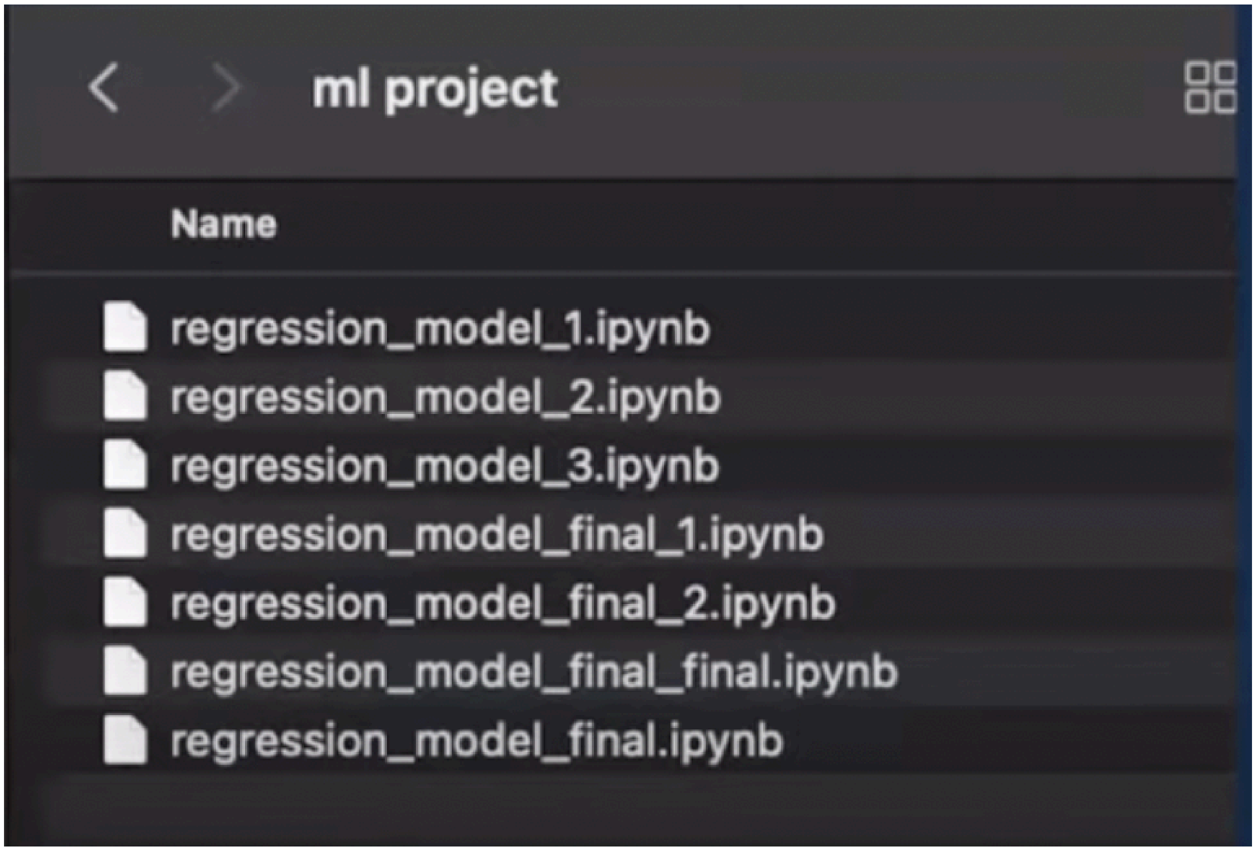
You can think of experiments as the process of building an ML model. When we say experiment run, we mean each trial in an ML experiment. So the ML experiment is actually the whole process that a data scientist may start playing with some data, models and hyperparameters. Each of these trials is an experiment run.

Experiment tracking is the process of keeping track of all the relevant information from ML experiments.

- **Organize** all the necessary components of a specific experiment. It's important to have everything in one place and know where it is so you can use them later.
- **Reproduce** past results (easily) using saved experiments.
- **Log** iterative improvements across time, data, ideas, teams, etc.

We have created a model successfully, but now we have new set of data how do you proceed on working with it

- we can change the data and run the code again
  - but we will loose the output and results from the old data
- we can create new cells below these to create a new model with the new data
  - but then when we have a lot of experiments in one file it will be really difficult finding the one we want to look at
- We can create new files for each experiment
  - but for actually comparing the results and outputs you'll still have to open each file and look into it closely



```
regression_model_1.ipynb
regression_model_2.ipynb
regression_model_3.ipynb
regression_model_final_1.ipynb
regression_model_final_2.ipynb
regression_model_final_final.ipynb
regression_model_final.ipynb
```

These are not the best ways of keeping track of the work and experiments that you perform, we need to create something that is easy to manage, clearly shows the results and metrics, logs the changes and hyperparameters for us.

ML Flow

https://mlflow.org/

MLflow is an open-source platform to manage Machine Learning Lifecycle. In layman's terms, it can track and store data, parameters, and metrics to be retrieved later or displayed nicely on a web interface.Furthermore, MLflow is a framework-agnostic tool, so any ML / DL framework can quickly adapt to the ecosystem that MLflow proposes.

MLflow emerges as a platform that offers tools for tracking metrics, artifacts, and metadata.

ML flow Tracking

MLflow Tracking is an API-based tool for logging metrics, parameters, model versions, code versions, and files. MLflow Tracking is integrated with a UI for visualizing and managing artifacts, models, files, etc.

Each MLflow Tracking session is organized and managed under the concept of runs.

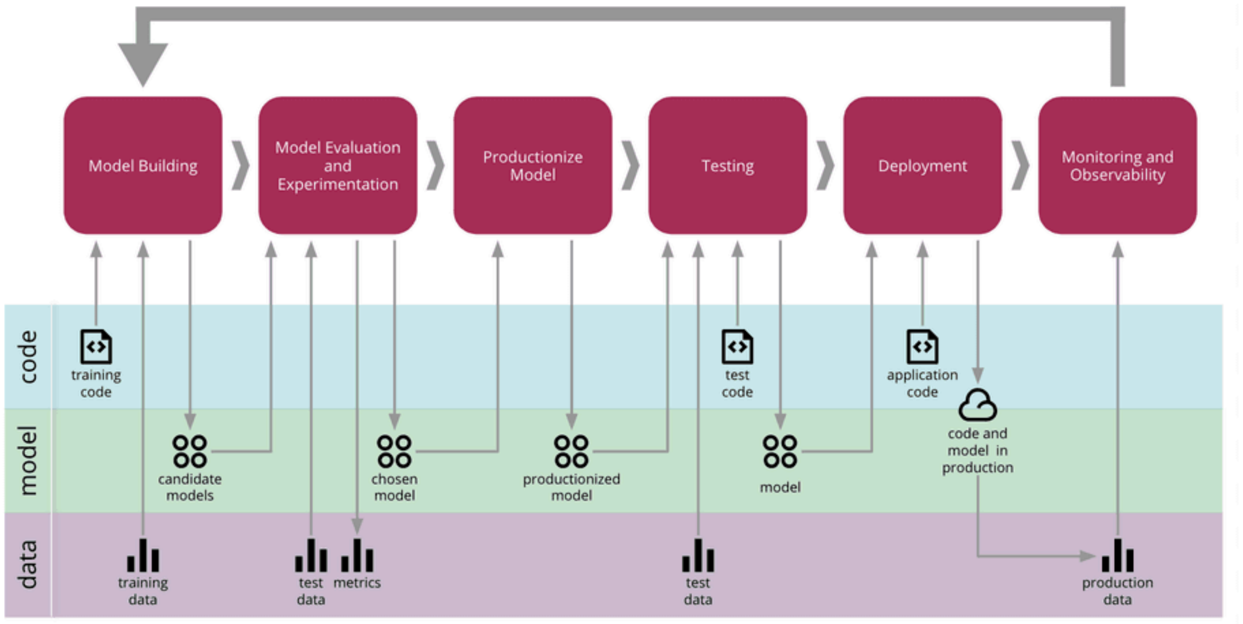
- A run refers to the execution of code where the artifact log is performed explicitly.
- An MLflow experiment is the primary unit of organization and access control for MLflow runs; all MLflow runs belong to an experiment. Experiment:{run,run.....run}
- Experiments let you visualize, search for, and compare runs, as well as download run artifacts and metadata for analysis in other tools.
- An MLflow run corresponds to a single execution of model code. Each run records the some information about that particulr trial

MLflow categorizes these into:

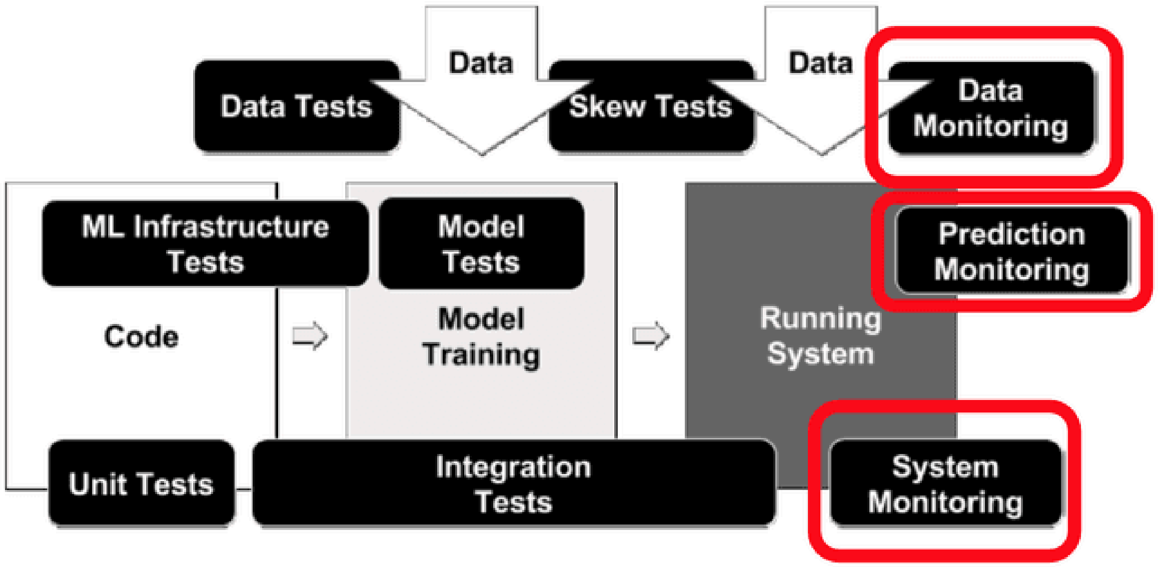
- Parameters (via mlflow.log_param()). Parameters are variables that you change or tweak when tuning your model.
- Metrics (using mlflow.log_metric()). Metrics are values that you want to measure as a result of tweaking your parameters. Typical metrics that are tracked can be items like F1 score, RMSE, MAE etc.
- Artifacts (using mlflow.log_artifact()). Artifacts are any other items that you wish to store. Typical artifacts to keep track of are PNGs of graphs,plots, confusion matrix, and also pickled model files
- Params are something you want to tune based on the metrics, whereas tags are some extra information that doesn't necessarily associate with the model's performance. there's no hard constraint on which to use to log which; they can be used interchangeably without error.

Monitoring Machine Learning Systems

Even though we've trained and thoroughly evaluated our model, the real work begins once we deploy to production. "Training and deploying ML models is relatively fast and cheap, but maintaining, monitoring and governing them over time is difficult and expensive.”



Different testing in ML



Drift

Drift is the change in an entity with respect to a baseline. Machine learning models are trained with historical data, but once they are used in the real world, they may become outdated and lose their accuracy over time due to a phenomenon called drift. Drift is when Real life data distribution changes from the one that it was trained on. This can cause the model to become less accurate or perform differently than it was designed to.

It refers to quantifying the changes in the observed data with respect to the training data.

Why does it drift

There are several reasons why machine learning models can drift over time.

- Outdated data: One common reason is simply that the data that the model was trained on becomes outdated or no longer represents the current conditions.

For example, consider a machine learning model trained to predict the stock price of a company based on historical data. If we train the model with data from a stable market, it might do well at first. However, if the market becomes more volatile over time, the model might not be able to accurately predict the stock price anymore because the statistical properties of the data have changed.

- Inefficient Model: Another reason for model drift is that the model was not designed to handle changes in the data. Some machine learning models can handle changes in the data better than others, but no model can avoid drift completely.

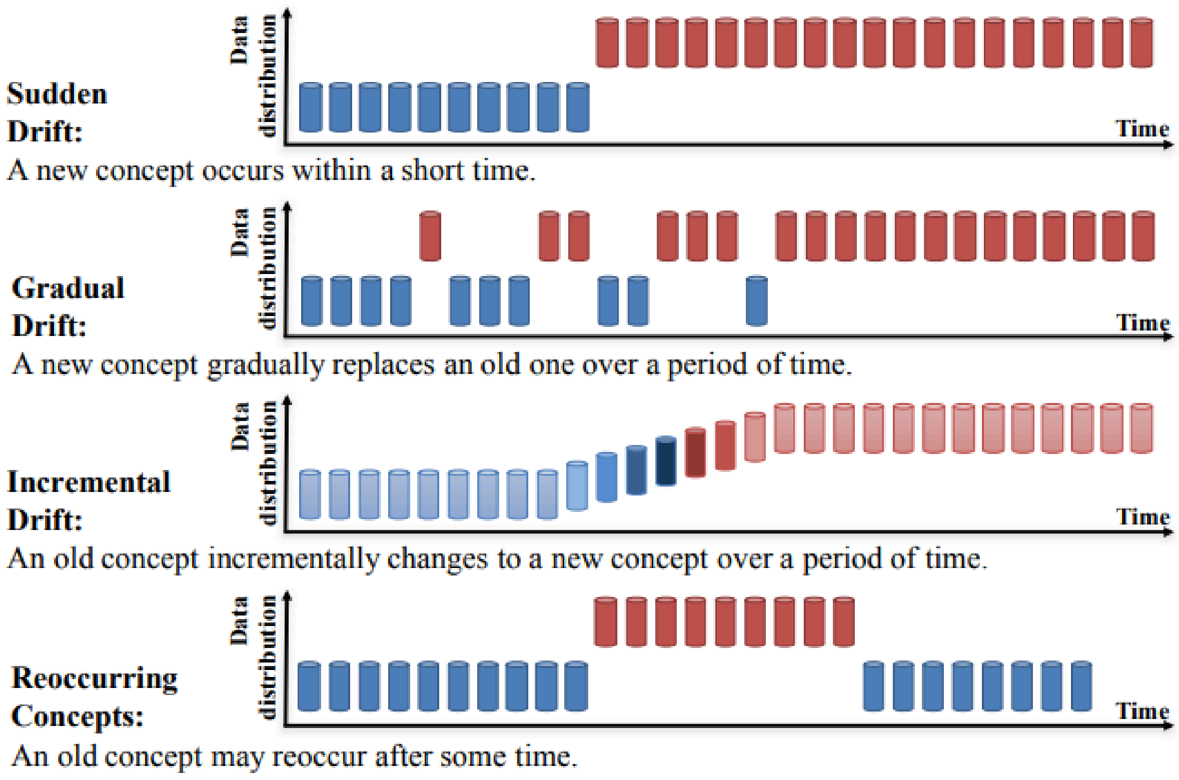
Types of drift

Two major drifts that often occur after a model has been deployed are concept drift and data drift.

Concept drift

Concept drift, also known as model drift, occurs when the task that the model was designed to perform changes over time.

It refers to a change in relationships between input data, ‘x’ and output data ‘y’ over a period of time. Eg- Consider a housing price prediction model. In an anomalous situation such as the pandemic, there is a sudden change in real estate prices and hence, the model may not make accurate predictions anymore. A home that costs 50 lakh INR in a normal scenario might now cost 75 lakh INR for the same set and quantity of features like bedrooms, ACs, area in square feet, etc.


Sudden Drift:
A new concept occurs within a short time.
Gradual Drift:
A new concept gradually replaces an old one over a period of time.
Incremental Drift:
An old concept incrementally changes to a new concept over a period of time.
Reoccurring Concepts:
An old concept may reoccur after some time.
Data distribution
Time

Data drift

Data drift refers to a change in the input features. Mathematically, it is a change in the distribution of variables that causes their meaning to change.

The change in input data or independent variable leads to poor performance of the model. Microsoft has stated data drift to be one of the top reasons model accuracy degrades over time.

Data drift is generally a consequence of changes in consumer preferences over time.

For instance,

- educational data collected before Covid shows a lesser preference for online learning than post-covid.
- Similarly, the demand for lipsticks has reduced considerably after Covid while face masks became a norm. As a result,
- Models trained on previous data will be useless. Since the input data has changed, the distribution of the variables becomes different and confuses the model.
- Data drift, feature drift, population, or covariate shift. Quite a few names to describe essentially the same thing.

Reasons for data drift

- changes in the data collection process
- changes in the data sources which provide the inputs
- changes in the business needs.
- Change in the distribution of the label/features in the data

Methods Available to detect Data Drift:

There are several methods available for detecting feature drift in machine learning:

- **Visual inspection:** One simple method is to visually inspect the input data over time to see if there are any noticeable changes in the distribution of the features. This can be done by plotting histograms or scatter plots of the data at different times and comparing them.
- **Statistical tests:** Statistical tests such as the Chi-Squared Test can be used to compare the input data distribution at different times and detect significant differences. These tests can provide a quantitative measure of the degree of drift in the data.
- **Model performance monitoring:** Another approach is to monitor the performance of the model over time and look for significant changes in accuracy or other performance metrics. If the model's performance begins to degrade, this could indicate that there is a drift in the data.
- **Data quality checks:** Regularly checking the quality of the input data can also help detect feature drift. For example, if there are sudden changes in the range, mean, median or variance of the features, this could indicate that there is a drift in the data.

The Kolmogorov-Smirnov Test

The KS test is a test of the equality between two univariate probability distributions. It can be used to compare a sample with a reference probability distribution or compare two samples.

When comparing two samples, we are trying to answer the following question:

“What is the probability that these two sets of samples were drawn from the same probability distribution?”

The null hypothesis is that the two samples come from the same distribution. The KS-test is applied to reject or accept it.

It returns the p-value. If the p-value is less than 0.05, you can usually declare that there is strong evidence to reject the null hypothesis and consider the two samples different.

You can also set a different significance level and, for example, react only to p-values less than 0.01. It is good to remember that the p-value is not a "measurement" of drift size but a declaration of the statistical test significance.

Chi-square

Chi-square test in another popular divergence test well-suited for categorical data.

The chi square statistic is a statistical hypothesis testing technique to test how two distributions of categorical variables are related to each other. Specifically, the chi-square statistic is a single number that quantifies the difference between the observed counts versus the counts that are expected if there was no relationship between the variables at all.

The divergence can range from zero to infinity. A value of zero means there is no difference between the data sets.

The null and alternative hypotheses of the Chi-Square Test of Fit Test are:

Null Hypothesis: The null hypothesis is: two groups have no significant difference.

Sagemaker

Amazon SageMaker is a widely used cloud machine-learning platform and is defined as a platform that enables developers to create, train, and deploy machine learning (ML) models in the cloud

- The Amazon SageMaker Studio provides a single, web-based visual interface where all Machine Learning development steps can be performed
- At the most basic level, SageMaker provides Jupyter notebooks like interface. You can use these notebooks for building, training and deploying ML models.
    - What is the advantage of using SageMaker notebooks instead of local or notebooks hosted on an EC2 server somewhere? Well, SageMaker lets you decide the type of machine you prefer so you don’t need to manage any complex AMIs or security groups — this makes it very easy to get started. SageMaker also provides access to GPUs and big machines with high amounts of RAM that might not be possible on a local setup.

- Yet another advantage is how you can use Amazon's own pre-built models that have been highly optimized to run on AWS services. These models come pre-built and you do not need to do much to build and check the model. You can use prebuilt XGBoost or LDA or PCA or Seq to Seq models — all these are available via high level Python SDK called sagemaker
    - One great thing about SageMaker is its modular design. If you prefer to train elsewhere and just use SageMaker for deployment, you can do that. If you just prefer to train your model and use its hyper-parameter tuning capability you can do that as well.
- It delivers High Performance.
- great integration with other AWS services like Dynamo DB or S3
- a very big reason to use sagemaker is that if your data is stored in s3 and you train or explore in colab there will be significant cost of data retrival, but within AWS services this is not the case.