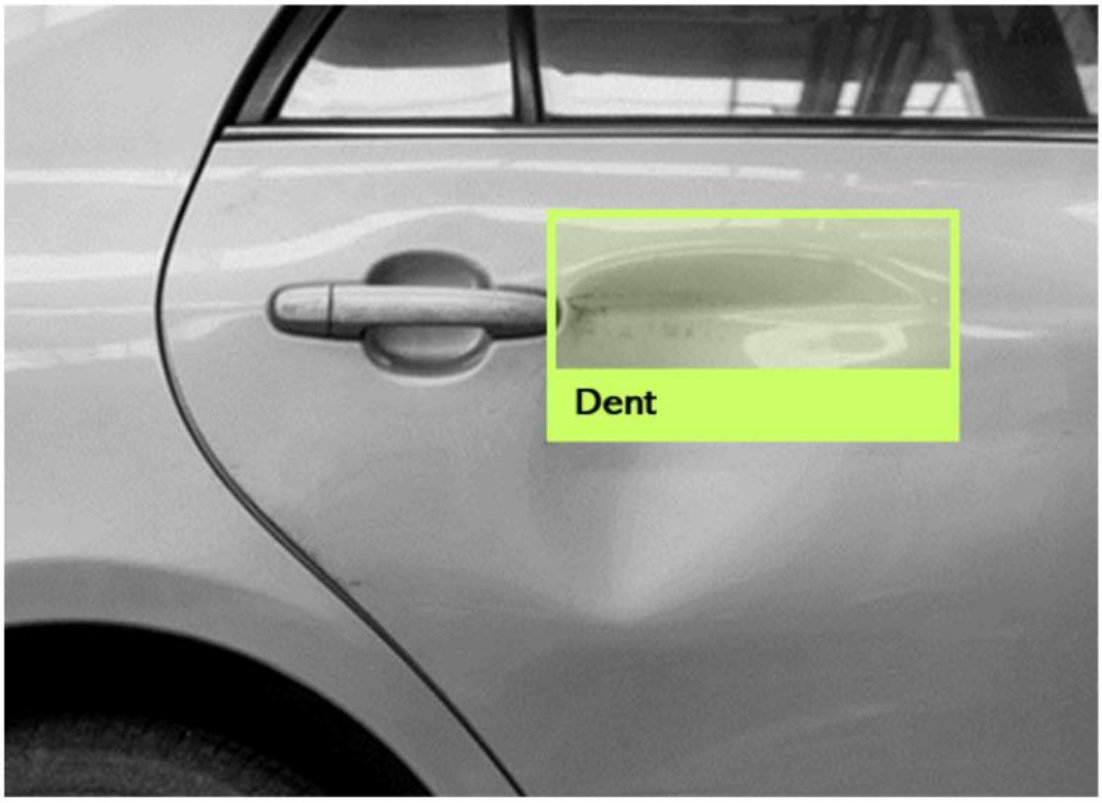


Car Damage Detector

CASE STUDY

Sudeep K S |EPGP ML | 11-03-2023

Problem statement

- You are working as an ML engineer for an e-commerce company Carsdepo.com, which provides an online marketplace for the purchase and sale of used/new cars.

- The company was founded in 2016 as an online platform to buy and sell used/new cars. The company offers end-to-end transactions, where it provides a smooth transition of ownership from the sellers to the buyers at the best rates. It sells more than 1 million cars worldwide annually and is considered one of the top three companies in the Asian market.

- While expanding operations internationally into several countries, it noticed that the market for used cars trumps over the new cars. However, this is a complicated problem to solve, as buyers want to be assured of the money they invest while buying.

- When a used car is listed, its worth must be effectively determined based on various features. A car can suffer from different types of damages, impacting its original shape and form, and this heavily impacts its resale value.
  Therefore, when a used car is bought or sold, its actual market value must be known. Also, It is often impossible to manually inspect every part of a car, as this can be quite labour intensive.

- In light of this, the company wants to build an automated damage-detection system that can detect any type of damage in cars. This solution can help determine the true resale value of a car. Based on what you learnt in the previous modules, you must understand the root problem that the company is trying to solve and build an ML system that provides an effective solution.

- Let's take a look at the problem statement:
  The company wants to provide an effective pricing strategy for any listing of used cars. Earlier, this problem was solved by a manual inspection conducted by a representative, who would inspect every feature of a used car. This solution was effective, as the representative considered every factor affecting the resale value (especially the damage); however, this solution is not scalable, as the company is growing.

- Using the millions of images of used cars the company has collected over the years, you must build an object-detection model that can detect the presence of any damage in cars.

- Among the various types of damage, the model should be able to detect two: scratches and dents. It should return "None" if no damage is detected. The product team will use these results to map the resale value of a car based on the type of damage detected.

- Note: You have a large repository of images, where the count of each damage is almost equal. However, some of the images are not yet labelled.
- You must create an ML system that has the features of a complete production stack, from experiment tracking to automated model deployment and monitoring. The framework should also be able to train if additional annotations are available later.

System design: Business KPI

Based on the above information, describe the KPI that the business should track.

One possible KPI that the business should track is the **accuracy** of the object detection model for car damage detection. Accuracy measures how well the model can correctly identify and classify car damages based on images1. A high accuracy means that the model can provide reliable and consistent results for pricing strategy.

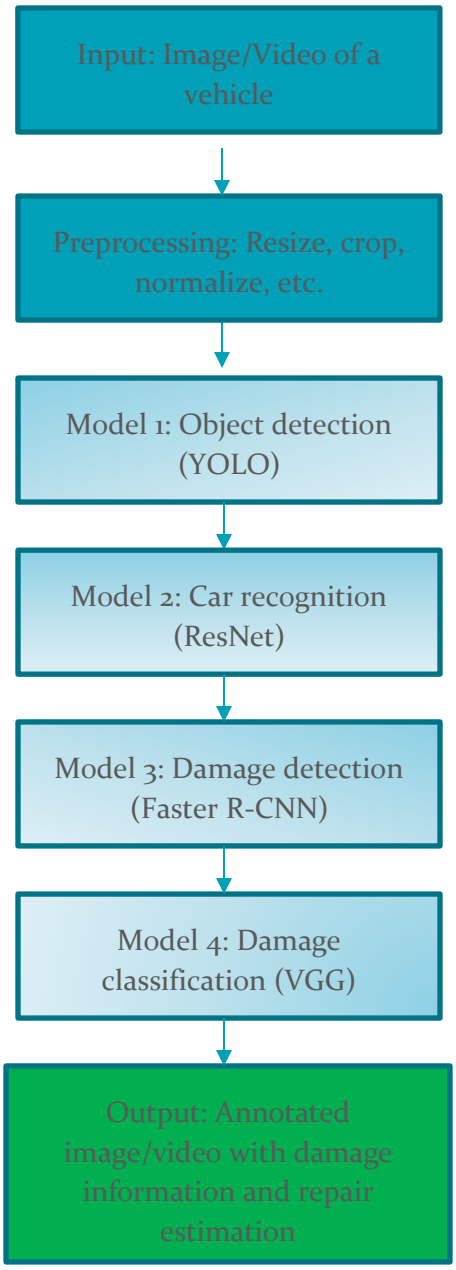
Another possible KPI that the business should track is the **latency** of the object detection model for car damage detection. Latency measures how fast the model can process an image and return a result3. A low latency means that the model can provide quick and timely feedback for customers and insurance agents.

System design: MLOps system

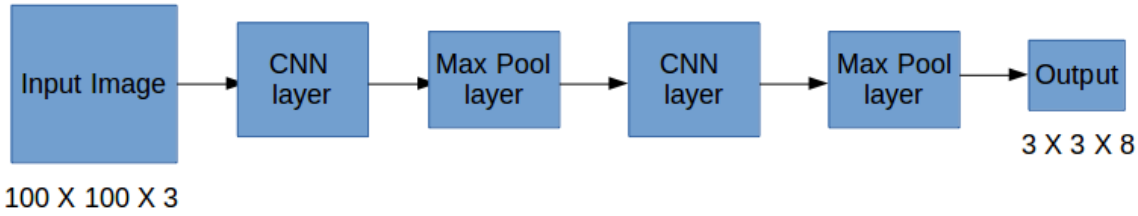
Some advantages of building an MLOps system rather than a simple model are:

- You can automate and monitor all steps of ML system development and operation, such as data collection, data preprocessing, model training, model testing, model deployment, model inference, etc.
- You can ensure reproducibility and traceability of your ML experiments by tracking parameters, metrics, artifacts, code versions, etc.
- You can scale up your ML system to handle large volumes of data and requests by using cloud services and distributed computing
- You can improve your ML system performance and quality by using continuous integration (CI) and continuous delivery (CD) pipelines to test and deploy new models or updates
- You can manage your ML system lifecycle by using feedback loops to collect data from production environment and retrain or update your models accordingly

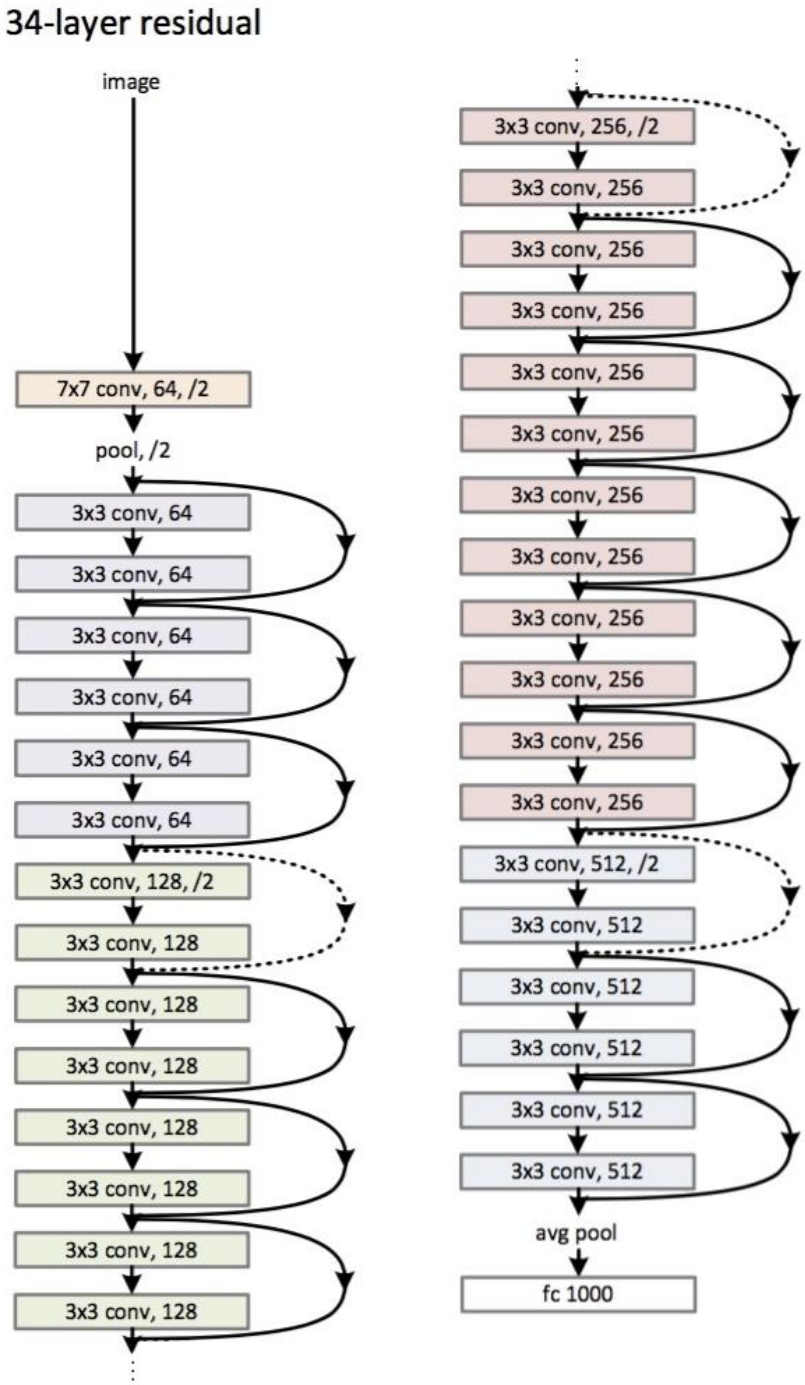
System Design: ML System Design Diagram



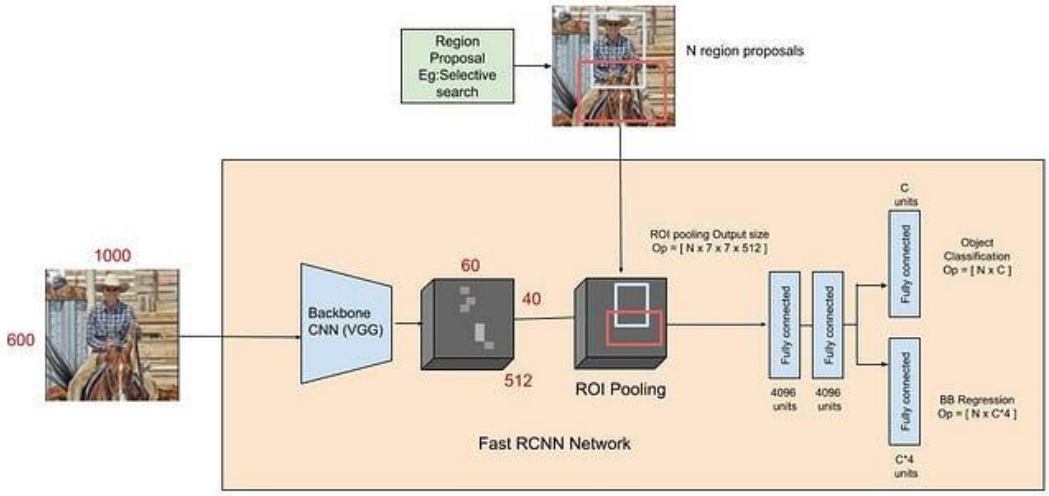
1. **Input:** Image/Video of a vehicle:
   Input from a cloud storage
2. **Preprocessing:** Resize, crop, normalize, etc.,
   I will be doing the data analysis and preprocessing and cleaning of data and data visualization
3. **Object Detection:**
   Using YOLO Detect if the image/video contains a car and draw bounding boxes around it. Here is the block diagram of the Yolo network



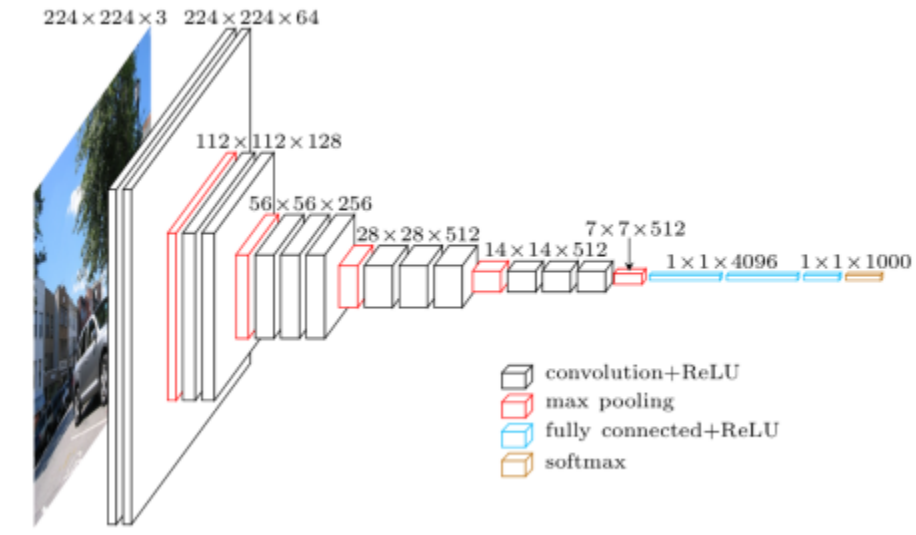
4. **Car recognition:**
   Identify the make and model of the car using ResNet, here is the block diagram for Resnet-34



5. **Damage Detection**:
   Detect if the car is damaged and draw bounding boxes around each damaged part using Faster R-CNN, here is an example block diagram for detection.



6. **Damage classification:**
   Classify each damaged part based on its severity (minor, moderate, major) using VGG, Here is a block diagram for VGG-16

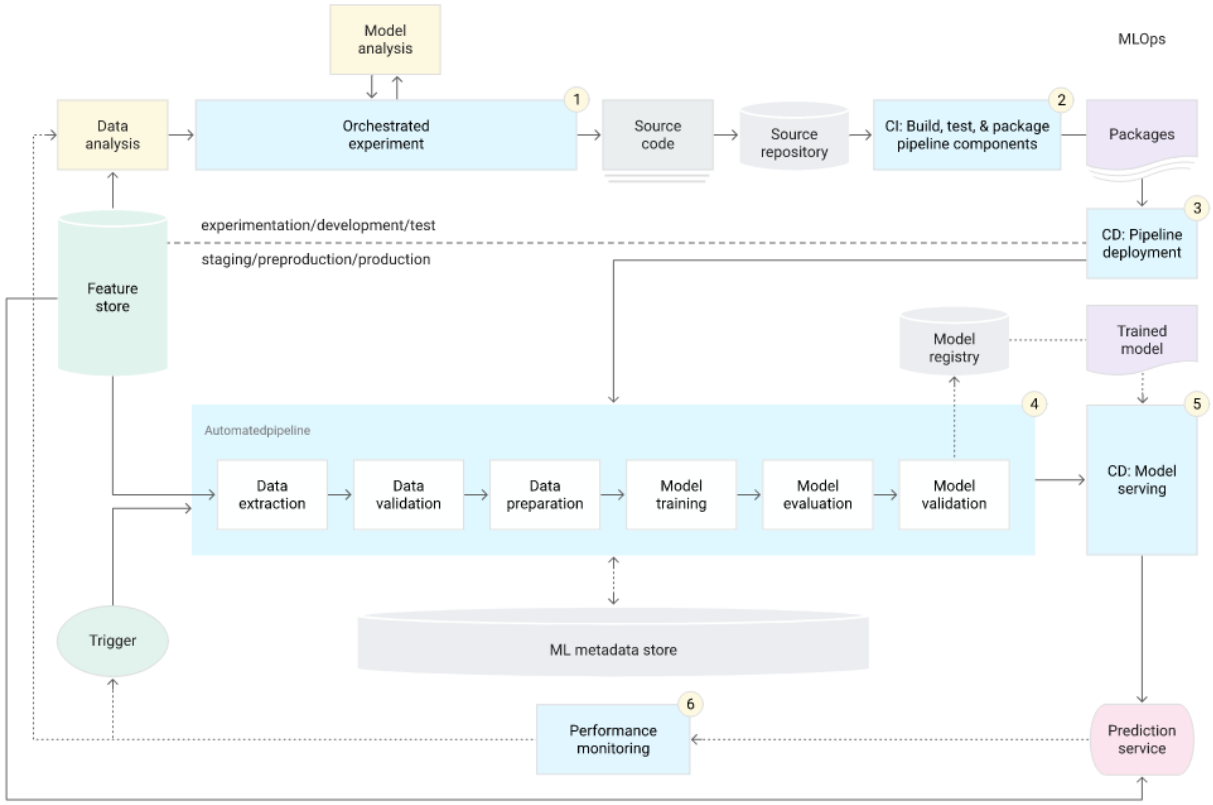


7. **Output:**
   Annotated image/video with damage information and repair estimation.

   Here is an example

The automated CI/CD system lets your data scientists rapidly explore new ideas around feature engineering, model architecture, and hyperparameters. They can implement these ideas and automatically build, test, and deploy the new pipeline components to the target environment.

The following diagram shows the implementation of the ML pipeline using CI/CD, which has the characteristics of the automated ML pipelines setup plus the automated CI/CD routines.



This MLOps setup includes the following components:

- Source control
- Test and build services
- Deployment services
- Model registry
- Feature store
- ML metadata store

- ML pipeline orchestrator

The data analysis step is still a manual process for data scientists before the pipeline starts a new iteration of the experiment. The model analysis step is also a manual process.

(Google MLOps: CI/CD Piplelines, 2023)[1]

System Design: Reasoning for the tools used

I chose **Google Cloud Platform (GCP)** as my cloud service provider because it offers various services and tools for MLOps such as Cloud Storage, Cloud Functions, Cloud AI Platform Pipelines (Kubeflow), Cloud AI Platform Prediction (TensorFlow Serving), etc.

I chose **Google Cloud Storage** as my data storage solution because it allows me to store large amounts of structured or unstructured data in buckets with high availability and durability

I chose **Google Cloud Functions** as my serverless computing solution because it allows me to run code snippets triggered by events such as uploading images to Cloud Storage or sending requests to Cloud AI Platform Prediction

I chose **Google Cloud AI Platform Pipelines** as my CI/CD pipeline solution because it allows me to create end-to-end workflows using Kubeflow components such as Dataflow (Apache Beam), TensorFlow Extended (TFX), etc.

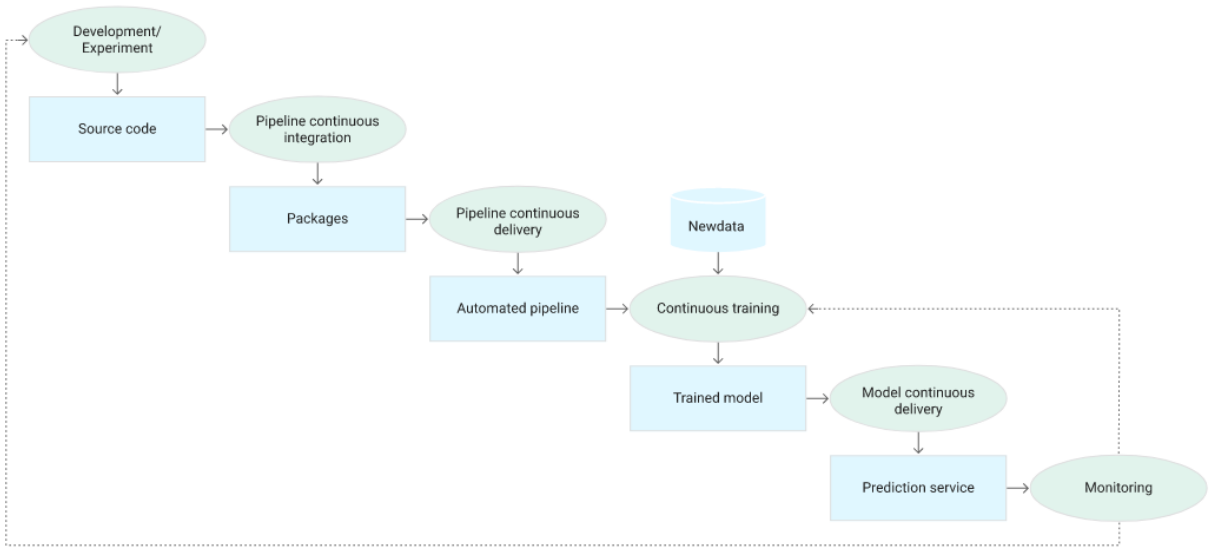
I chose **Google Cloud AI Platform Prediction** as my inference service solution because it allows me to deploy trained models using TensorFlow Serving with autoscaling and load balancing features

Workflow of solution

The steps that should be taken to build such a system end-to-end are:

1. Collect images of used cars with different types of damages such as scratches or dents from various sources such as customers or insurance agents
2. Upload images to **Google Cloud Storage** buckets with appropriate labels such as “scratch”, “dent”, “none”, etc.
3. Use **Google Cloud Functions** to trigger Dataflow jobs that preprocess images such as resizing, cropping, augmenting
4. Use **Google Cloud Functions** to trigger TFX(TensorFlow Extended) pipelines that train object detection models using Mask R-CNN or YOLO algorithms on preprocessed images
5. Use TFX pipelines to evaluate trained models using metrics such as accuracy and latency
6. Use TFX pipelines to deploy best performing models to Cloud AI Platform Prediction
7. Use **Google Cloud Functions** to trigger inference requests from customers or insurance agents who upload images of used cars they want to sell or buy
8. Use **Google Cloud AI Platform Prediction** to return results of object detection models such as bounding boxes or segmentation masks for car damages
9. Use feedback loops from
    a. Experiments to learn and iterate
    b. Human Annotators who can check and correct the output of the ML model, to improve accuracy and quality of the data and model over time
    c. Customers or Policyholders who can provide ratings and reviews on their experience from the service

The following diagram shows the stages of the ML CI/CD automation pipeline:



The pipeline consists of the following stages:

1. **Development and experimentation**: You iteratively try out new ML algorithms and new modeling where the experiment steps are orchestrated. The output of this stage is the source code of the ML pipeline steps that are then pushed to a source repository.

2. **Pipeline continuous integration**: You build source code and run various tests. The outputs of this stage are pipeline components (packages, executables, and artifacts) to be deployed in a later stage.

3. **Pipeline continuous delivery**: You deploy the artifacts produced by the CI stage to the target environment. The output of this stage is a deployed pipeline with the new implementation of the model.

4. **Automated triggering**: The pipeline is automatically executed in production based on a schedule or in response to a trigger. The output of this stage is a trained model that is pushed to the model registry.

5. **Model continuous delivery**: You serve the trained model as a prediction service for the predictions. The output of this stage is a deployed model prediction service.

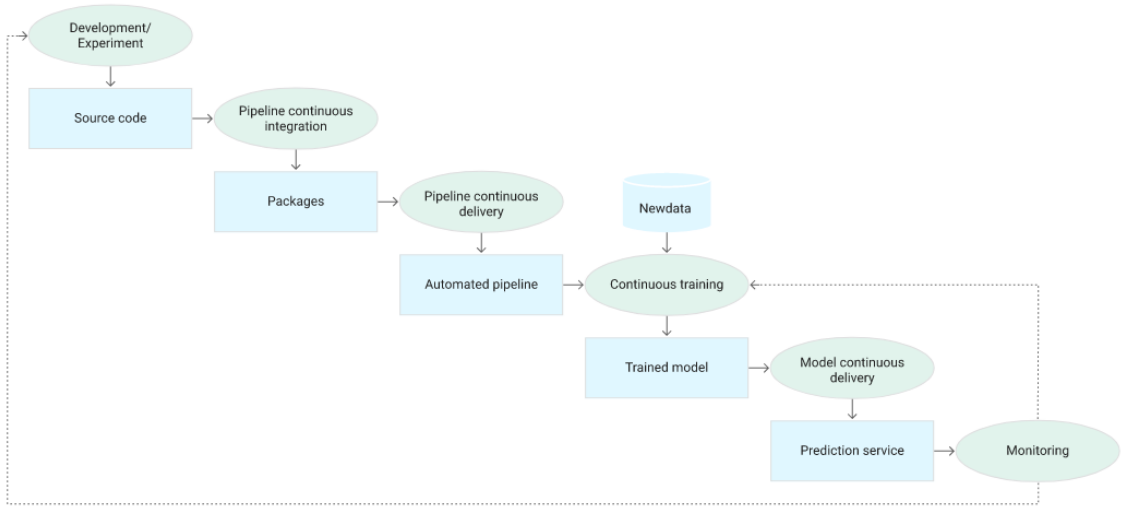
6. **Monitoring**: You collect statistics on the model performance based on live data. The output of this stage is a trigger to execute the pipeline or to execute a new experiment cycle.

(Google MLOps: CI/CD Piplelines, 2023)

Workflow Action

- If there is drift detected say the accuracy or latency of the model degrades to more than 5% than the results obtained from the trained model, then we trigger and experiment
- If there is new data available, then we start retraining the model with new data.

This is explained clearly in the figure below:



References:

MLOps: Continuous delivery and automation pipelines in machine learning | Cloud Architecture Center | Google Cloud