PENGEMBANGAN TEKNOLOGI KONVERSI EKSPERIMEN PEMBELAJARAN MESIN UNTUK SISTEM YANG PRODUCTION-READY

Laporan Tugas Akhir I

Disusun sebagai syarat kelulusan tingkat sarjana

IF4091/Tugas Akhir I dan Seminar

Oleh

Matthew Kevin Amadeus

NIM : 13518035

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

DESEMBER 2021

PENGEMBANGAN TEKNOLOGI KONVERSI EKSPERIMEN PEMBELAJARAN MESIN UNTUK SISTEM YANG PRODUCTION-READY

Laporan Tugas Akhir I

Oleh

Matthew Kevin Amadeus

NIM : 13518035

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

Bandung, 13 Desember 2021

Mengetahui,

Pembimbing,

Achmad Imam Kistiantoro, S.T, M.Sc., Ph.D.

DAFTAR ISI

DAFTAR GAMBAR

The Latex typesetting markup language is specially suitable for documents that include mathematics. are rendered properly an easily once one gets used to the

BAB I

PENDAHULUAN

I.1 Latar Belakang

Dalam proses pengembangan sistem pembelajaran mesin, biasanya banyak tahapan yang perlu dilewati untuk memgembangkan sistem. Dari proses pengumpulan data hingga proses *delivery* dari sistem inferensi, hal-hal yang dilakukan berdasarkan kaidah-kaidah yang ada dalam MLOps. Terdapat beberapa proses yang memakan waktu dalam proses pembuatannya, salah satunya adalah bagaimana sistem tersebut dibangun dari eksperimen-eksperimen yang dilakukan.

Dalam makalah ini, penulis akan membahas terkait hal-hal apa saja yang bisa diterapkan dalam proses pengembangan sistem pembelajaran mesin sehingga proses tersebut dapat dikerjakan dengan lebih efisien. Khususnya, akan dibahas terkait teknologi yang bisa dikembangkan untuk melakukan konversi dari eksperimen-eksperimen yang dibuat menjadi satu sistem yang koheren dan *production-ready*.

I.2 Rumusan Masalah

Berikut ini adalah hal-hal yang menjadi rumusan masalah dalam tugas akhir ini:

1. Bagaimana alur kerja konversi eksperimen ke sistem yang *production-ready* dilakukan secara umum?
2. Bagaimana pekerjaan konversi eksperimen dapat dipercepat pada suatu bagian pekerjaan?
3. Bagaimana metode-metode otomatisasi dapat membantu proses konversi eksperimen?
4. Bagaimana *pipeline* pemrosesan dalam MLOps dapat membantu dalam otomatisasi proses konversi?

I.3 Tujuan

Tujuan utama makalah tugas akhir ini adalah untuk mempelajari dan mengembangan sebuah *proof of concept* untuk melakukan konversi dari eksperimen pembelajaran mesin ke sebuah sistem yang siap untuk digunakan. Dengan adanya makalah ini, diha-

rapkan proses konversi tersebut dapat membantu secara praktiknya agar lebih efisien.

I.4 Batasan Masalah

Berikut ini adalah batasan masalah yang penulis tetapkan dalam membuat makalah tugas akhir ini, yaitu:

1. Makalah ini akan membahas keseluruhan proses yang ada di MLOps secara umum saja dan hanya akan berfokus pada bagian konversi eksperimen menjadi sistem.
2. Makalah ini akan berfokus pada bagian-bagian yang menjadi bagian utama dalam proses yang ada di MLOps.
3. Makalah ini tidak akan membahas terkait metode *provisioning* infrastruktur dalam MLOps.

I.5 Metodologi

Berikut adalah tahapan atau metodologi yang dilakukan dalam proses penulisan makalah ini.

1. Melakukan penentuan masalah secara umum

   Pada tahapan awal, masalah terkait proses konversi dari eksperimen ke sistem akan ditentukan. Masalah yang ditemukan akan dipilah berdasarkan hal yang paling berpengaruh terhadap proses tersebut yang akan dicari solusinya.

2. Melakukan pencarian metode otomatisasi

   Penulis akan melakukan pencarian informasi terkait metode-metode *model serving* dalam sistem pembelajaran mesin dari masalah yang ada dalam pembuatan sistem. Melalui tahapan ini, berbagai metode otomatisasi akan dibandingkan untuk mendapatkan solusi yang optimal.

3. Melakukan pembuatan rancangan solusi

   Berdasarkan permasalahan yang ada, sebuah rancangan solusi yang dapat menyelesaikan beberapa poin masalah akan diajukan. Rancangan solusi dapat didasarkan dari sumber bacaan yang ditemui penulis atau dari ide penulis.

4. Melakukan implementasi dari proposal solusi yang telah dirancang

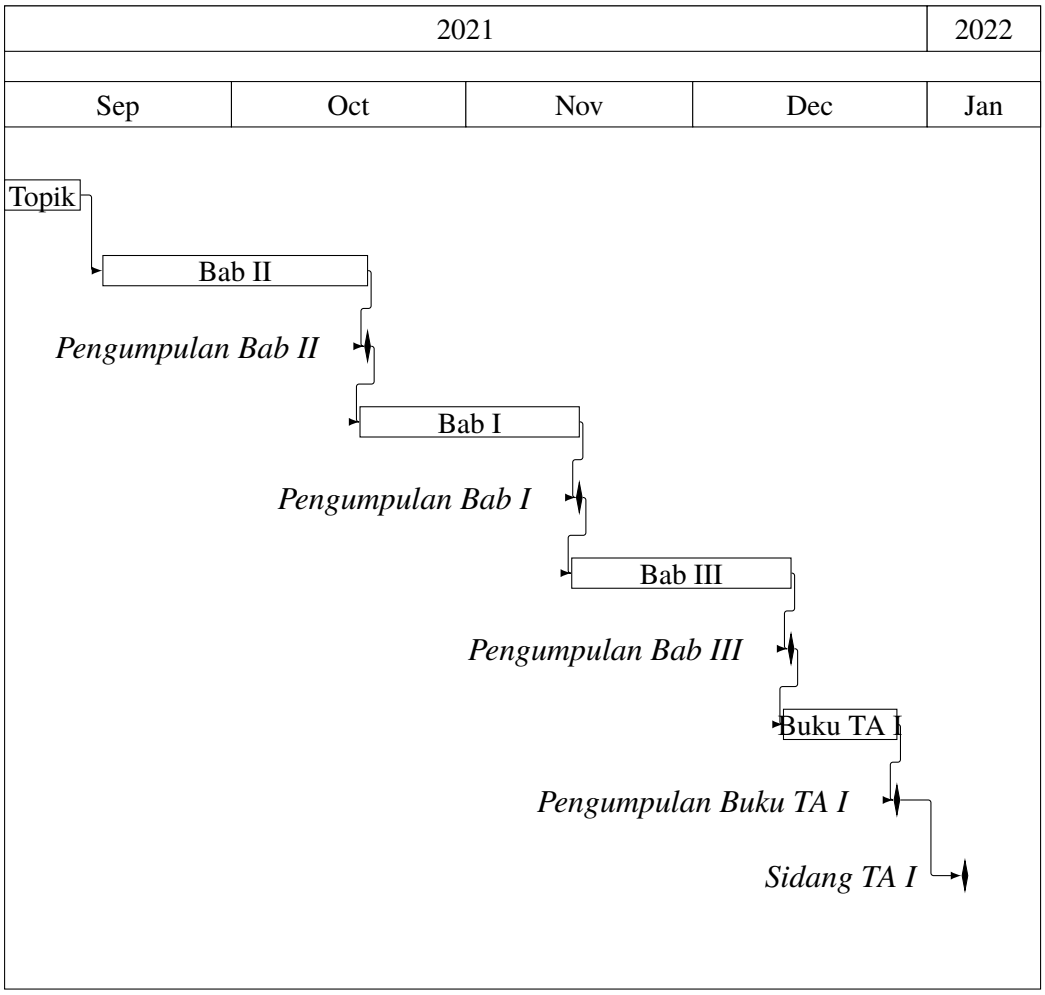
Rancangan solusi yang sudah diajukan akan diimplementasikan. Selain melakukan implementasi, dalam tahapan ini juga akan dilakukan pengembangan beberapa alternatif solusi secara teknis. Hal tersebut bertujuan untuk mencari implementasi yang baik dari aspek pengembangan dan dalam penyelesaian solusinya. Implementasi akan dilakukan secara iteratif bergantung pada hasil pengujian.

5. Melakukan pengujian terhadap implementasi proposal solusi terhadap contoh yang ada di dunia nyata

   Berhubungan dengan tahapan sebelumnya, rancangan solusi yang diimplementasikan akan dilakukan pengujian pada beberapa studi kasus. Studi kasus yang diambil merupakan kasus yang umum ditemui dalam pembangunan sistem pembelajaran mesin.

I.6 Jadwal Pelaksanaan

Berikut ini adalah jadwal pelaksanaan untuk pengerjaan tugas akhir ini.



Gambar I.6.1: Jadwal Pelaksanaan Tugas Akhir

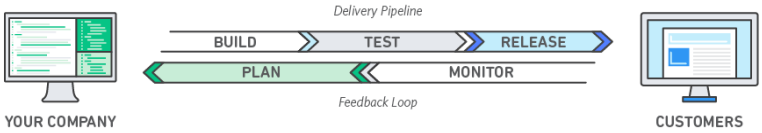
BAB II

STUDI LITERATUR

Pada bab ini, akan dibahas terkait literatur-literatur yang menjadi dasar dari pengembangan sistem konversi ini. Literatur yang dibahas juga akan membahas terkait permasalahan umum dalam MLOps dan apa saja hal-hal yang berpotensi untuk dikembangkan.

II.1 DevOps

II.1.1 Definisi DevOps



Gambar II.1.1: Mekanisme DevOps (Sumber: Amazon Web Services 2021)

DevOps adalah serangkaian filosofi, *practices*, dan kakas yang bisa meningkatkan produktivitas perusahaan, baik secara internal maupun untuk mempercepat proses *delivery* layanan untuk konsumen (Amazon Web Services 2021). Dengan *delivery* yang cepat, tentunya akan bisa berkompetisi dengan lebih efektif di pasar. Terdapat lima bagian utama dari filosofi DevOps, yaitu build, test, release, plan, dan monitor. Hal yang menjadi tujuan utama untuk DevOps adalah untuk mempercepat dalam proses-proses yang memakan waktu dengan menggunakan otomatisasi yang bisa dilakukan terhadap sistem.

II.1.2 *Practices* dalam DevOps

Terdapat beberapa hal yang menjadi *best practices* dalam DevOps. Dalam bagian ini, penulis akan membahas sedikit terkait masing-masing practices yang ada di dalam DevOps yang relevan dengan poin-poin yang ada pada makalah ini.

1. Continuous Integration

   Continuous Integration adalah filosofi yang diterapkan dengan tujuan memastikan integritas sistem. Biasanya, continuous integration dicapai lewat menerapkan *build* dan *test* yang otomatis. Continuous integration akan sangat menunjang

TDD (*Test Driven Development*) karena selalu dilakukan secara otomatis setiap kali ada perubahan yang dilakukan kepada sistem. Dengan begitu, kualitas sistem, pencarian bug, dan penyelesaiannya bisa diselesaikan dengan lebih cepat.

2. Continuous Delivery

   Continuous Delivery adalah suatu filosofi dengan tujuan untuk memastikan *delivery* dari sistem. Dalam konteks ini, *delivery* yang dimaksud adalah terkait melakukan *build* dan *deployment* ke *production* secara otomatis. Continuous delivery juga sangat erat hubungannya dengan continuous integration, yang terkadang bisa juga dilakukan dalam *pipeline* yang sama. Pengaturan *deployment* juga fleksibel, tidak harus melulu untuk *production* saja, walaupun tujuan awalnya untuk melakukan *delivery* dengan cepat.

3. Microservices

   Microservices adalah sebuah aristektur untuk sistem, di mana sebuah sistem besar dibuat atas dasar sistem-sistem kecil yang membentuk satu kesatuan besar. Secara umum, microservices dapat memberi dampak positif bagi tim yang besar, karena pengembangan bisa dilakukan secara independen antar sistem. Walau begitu, dari sudut pandang deployment tentunya tidak trivial. Meskipun demikian, microservices akan mendukung *scalability* dari sistem ke depannya, dan dengan filosofi DevOps lainnya aristektur ini sangat baik untuk diterapkan dalam lingkungan pengembangan dengan jumlah orang yang banyak.

4. Infrastructure as Code

   Infrastructure as Code adalah suatu filosofi di mana infrastruktur yang dibuat harus dapat disimpan sebagai program, dengan tujuan memudahkan replikasi dan manajemen infrastruktur sistem. Seiring meningkatnya penggunaan sistem berbasis *cloud*, penggunaan Infrastructure as Code juga meningkat karena alasan yang disebut sebelumnya. Hal-hal yang bisa dimanajemen lewat infrastructure as code meliputi konfigurasi infrastruktur dan policy atau aturan terhadap infrastruktur. Dengan kakas-kakas yang ada sekarang, konfigurasi yang ada bisa direalisasikan menjadi infrastruktur yang sesungguhnya.

5. Monitoring dan Logging

   Dalam DevOps, Monitoring dan Logging terhadap sistem menjadi suatu hal yang patut diperhatikan. Dengan sistem yang terus berkembang, sistem perlu diawasi untuk memastikan kelancaran dari sistem. Sistem dapat diawasi melalui kinerja maupun dan melakukan pengawasan terhadap pencatatan rekam historis dari penggunaan sistem.

II.2 MLOps

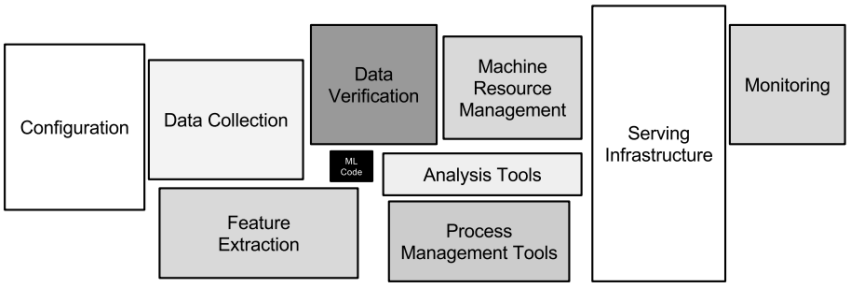
II.2.1 Definisi MLOps

MLOps adalah penerapan yang lebih spesifik dari DevOps. Seperti yang telah dibahas pada bagian sebelumnya, DevOps adalah suatu metodologi dan prinsip yang diterapkan untuk membuat proses pengembangan sistem menjadi lebih cepat dan efisien. MLOps adalah metodologi yang menerapkan DevOps untuk proses-proses yang ada pada pengembangan model machine learning.

Dalam MLOps, terdapat proses-proses tertentu yang spesifik untuk domain pengembangan sistem pembelajaran mesin (Rick Merrit 2021). MLOps ini merupakan bidang yang masih baru karena baru-baru ini teknologi kecerdasan buatan mulai diterapkan dalam perusahaan yang mulai merambah ke layanan dengan kecerdasan buatan. Tujuan dari MLOps adalah untuk melakukan manajemen dari eksperimen dalam proses pengembangan sistem pembelajaran mesin. Saat ini, metodologi MLOps juga sudah diterapkan pada berbagai perusahaan besar di dunia, seperti Facebook, Google, AirBnB, dan Uber.

II.2.2 Alasan MLOps Diperlukan

Hal ini patut menjadi pertanyaan; apakah MLOps benar-benar diperlukan dalam pengembangan sistem pembelajaran mesin? Seiring berjalannya waktu, kebutuhan terhadap pembelajaran mesin semakin meluas. Berbagai eksperimentasi, pemrosesan data, dan pemodelan dilakukan dengan cepat dan dengan terus menerus dikembangkan versi yang lebih baru. Bisa jadi, model yang dibuat sudah bisa berjalan dalam eksperimen. Hal yang menjadi pertanyaan sekarang adalah bagaimana cara memberikan layanan dari model tersebut kepada khalayak umum.



Gambar II.2.1: Gambaran umum komponenen sistem pembelajaran mesin (Sumber: Sculley dkk. 2015)

Ternyata, model ML hanya menjadi sebagian kecil saja dari suatu sistem (Sculley dkk. 2015). Bagian yang mendefinisikan model tersebut pada akhirnya hanya menjadi sebagian kecil dari sistem, dan merupakan yang paling kecil dibandingkan bagian lainnya (lihat Gambar II.2.1). Hal-hal lain seperti bagaimana melakukan ekstraksi fitur, analisis terhadap data, metode pemberian layanan, dan pengawasan terhadap data menjadi bagian yang lebih utama dalam sebuah sistem.

Oleh karena itu, di luar dari proses pengembangan dan eksperimentasi dari sebuah model, bagaimana sebuah model dapat diberikan lewat layanan juga perlu menjadi hal yang diperhatikan. Pada nyatanya, tidak semudah itu untuk memberikan laanan

II.2.3 Contoh Kakas MLOps

Di masa lampau, kebutuhan untuk sistem berbasis teknologi pembelajaran mesin tidak setinggi di masa kini. Perusahaan-perusahaan raksasa seperti Google membuat suatu *framework* internal untuk kebutuhan mereka saat itu (Rick Merrit 2021). Seiring meningkatnya keperluan untuk MLOps dalam dunia industri, kakas-kakas MLOps mulai bermunculan.

Kubeflow adalah salah satu kakas yang dikembangkan dengan tujuan untuk mempermudah proses *deployment* pada sistem ML dengan mudah (The Kubeflow Authors 2021). Seperti namanya, Kubeflow dibuat untuk melakukan *deployment* di atas Kubernetes. Kubeflow memiliki fitur yang lengkap, seperti adanya TensorFlow dan kakas-kakas ML lainnya, kakas-kakas untuk melakukan eksperimen, menyimpan *hyperparameter*, dan melakukan *model versioning*, serta beberapa aplikasi yang digunakan untuk melakukan monitoring di Kubernetes juga, seperti Prometheus. Dengan dibangunnya-

Kubeflow di atas Kubernetes, hal itu membuat melakukan *deployment* pada *platform-platform* berbasis *cloud* menjadi mudah.

Dalam Kubeflow, terdapat suatu sistem yang menggunakan pipeline; serupa dengan CI/CD, namun dengan konteks pengunaan yang berbeda. Pipeline pada Kubeflow dapat membentuk sebuah graf yang masing-masing langkah atau *node*-nya mendefinisikan parameter masukan dan keluaran yang terdapat pada tiap langkahnya. Tiap langkah merepresentasikan suatu proses terhadap data yang dilakukan lewat Docker yang membungkus suatu kode Python. Pendekatan pipeline ini secara umum fleksibel dan bisa menjadi potensi pengembangan untuk MLOps.

II.2.4 Alur Kerja MLOps

Terdapat tiga tahapan utama dalam alur kerja MLOps (Dr. Larysa Visengeriyeva, Anja Kammer, Isabel Bär, Alexander Kniesz, dan Michael Plöd 2021). Alur kerja tersebut adalah sebagai berikut:

1. Data Engineering
2. Model Engineering
3. Code Engineering

Dalam makalah ini, akan dibahas secara lebih mendetail pada tahap code engineering. Untuk tahapan lainnya, akan dibahas beberapa hal yang relevan terhadap tahap code engineering.

II.2.4.1 Data Engineering

Tahapan awal dalam bidang ilmu data secara umum seharusnya meliputi pengumpulan data dan persiapan data untuk digunakan dalam analisis. Secara lebih detailnya, tahapan ini dibagi menjadi beberapa bagian:

1. Data Ingestion, yaitu mengumpulkan data dari berbagai sumber melalui framework yang ada.
2. Data Exploration and Validation, yaitu melakukan EDA (*Exploratory Data Analysis*) yang meliputi *profiling*
3. Data Cleaning, yaitu melakukan proses lanjut pada data untuk memperbaiki ke-

salahan yang ada pada dataset.

4. Data Labelling, yaitu melakukan labelling kepada dataset yang ada.
5. Data Splitting, yaitu memisahkan dataset menjadi beberapa bagian yang umumnya adalah untuk *training*, *testing*, dan *validation*.

II.2.4.2 Model Engineering

Tahapan selanjutnya setelah melakukan pemrosesan terhadap data adalah melakukan pemodelan terhadap dataset yang sudah disiapkan. Untuk mendapatkan model yang optimal, biasanya tahap ini dibagi menjadi beberapa tahapan yang lebih kecil lagi, yaitu:

1. Model Training, yaitu sebuah proses untuk mengaplikasikan algoritma machine learning terhadap training dataset yang ada.
2. Model Evaluation, yaitu melakukan evaluasi terhadap model yang dibuat dengan validation dataset atau untuk memastikan apakah model sesuai standar yang diharapkan
3. Model Testing, yaitu melakukan testing terhadap test dataset atau data yang sengaja disembunyikan untuk menguji kasus-kasus tertentu.
4. Model Packaging, yaitu melakukan penyimpanan model yang sudah dibuat dan diuji dalam suatu format file yang bisa digunakan kembali tanpa perlu melakukan training.

Lewat framework-framework machine learning seperti Kubeflow, biasanya akan dilakukan versioning terhadap model. Hal ini dilakukan untuk melakukan eksperimen dan melakukan pencatatan terhadap hyperparameter yang unik untuk masing-masing model. Nantinya, model dengan parameter dan hasil terbaik akan dipilih dan digunakan.

Tahapan ini cukup berpengaruh terhadap tahapan selanjutnya. Dalam tahapan ini, akan dibahas terkait model packaging secara lebih mendalam. Bila layanan akan diberikan kepada pelanggan secara masal, biasanya teknologi web akan dipilih untuk memberikan layanan tersebut. Jadi, akan dibuat sebuah API agar model tersebut bisa dimanfaatkan untuk layanan lain dengan tujuan akhir untuk pelanggan. Detail lengkapnya akan dibahas pada bagian berikutnya.

II.2.4.3 Code Engineering

Tahapan terakhir dalam MLOps adalah untuk melakukan code engineering. Model yang sudah siap digunakan akan dihubungkan dengan program atau sistem yang ada atau akan ada. Tahap ini dibagi dalam beberapa tahapan, yaitu:

1. Model Serving, yaitu terkait metode bagaimana model yang dibuat di-*deploy* agar bisa diakses pelanggan.
2. Model Performance Monitoring, yaitu terkait metode melakukan monitoring terhadap kinerja model bila diberikan data yang sesungguhnya
3. Model Performance Logging, yaitu metode pilihan untuk menyimpan log dari request yang diberikan.

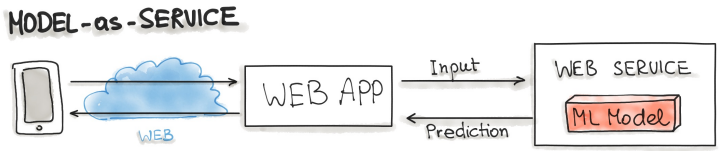
Hal yang menjadi perhatian utama dalam tahap ini adalah model serving.Banyak pilihan dan *pattern* yang bisa digunakan untuk melakukan serving, serta pilihan teknologi yang digunakan untuk *deployment*. Model Serving akan dibahas pada bagian berikutnya secara lebih mendalam.

II.3 Model Serving

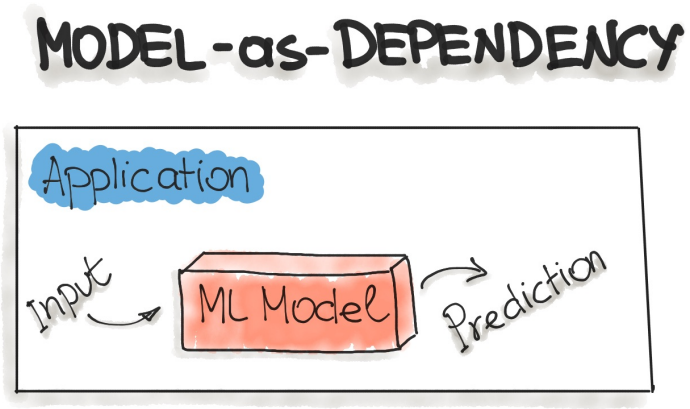
Dalam makalah ini akan dibahas secara lebih mendalam terkait Model Serving. Hal-hal yang akan dibahas meliputi bagaimana metode-metode yang ada dalam melakukan serving dan teknologi apa yang digunakan untuk melakukan *deployment* model tersebut.

Terdapat lima jenis *serving pattern* yang umum digunakan untuk membuat layanan dari model (Dr. Larysa Visengeriyeva, Anja Kammer, Isabel Bär, Alexander Kniesz, dan Michael Plöd 2021):

1. Model-as-Service
2. Model-as-Dependency
3. Precompute Serving
4. Model-on-Demand
5. Hybrid Serving



Gambar II.3.1: Ilustrasi Model-as-Service (Sumber: Aurélien 2019)



Gambar II.3.2: Ilustrasi Model-as-Dependency (Sumber: Aurélien 2019)

II.3.1 Model-as-Service

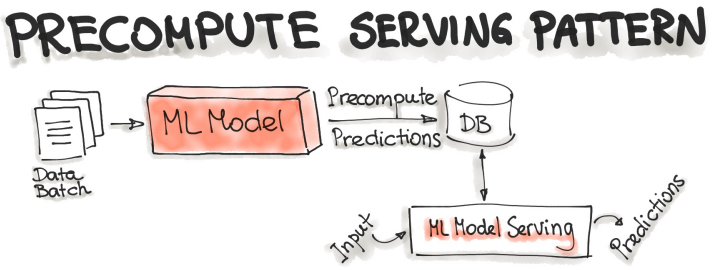
Serving pattern ini adalah metode yang paling sederhana. Pada dasarnya, model yang sudah ada akan dibungkus dengan sebuah interface agar dapat dilakukan RPC terhadap model tersebut (lihat Gambar II.3.1). Teknologi yang umum digunakan biasanya seperti REST API dan gRPC, namun metode-metode RPC lainnya bisa juga digunakan.

II.3.2 Model-as-Dependency

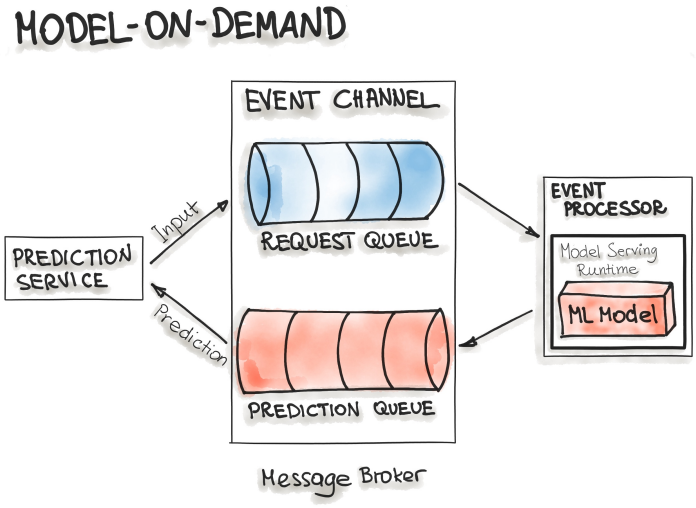
Serving pattern serupa dengan Model-as-Service, tetapi perbedaan utamanya adalah di mana model tersebut menjadi dependency dari layanan yang dibuat secara internal, bukan melakukan *deployment* secara terpisah untuk model tersebut (lihat Gambar II.3.2). *Pattern* ini akan digunakan apabila model akan digunakan sebagai satu bagian dari layanan yang lebih besar.

II.3.3 Precompute Serving

Sesuai namanya, model akan digunakan untuk melakukan prekomputasi tanpa langsung menggunakan modelnya dalam sistem (lihat Gambar II.3.3). Hasil dari prekomputasi



Gambar II.3.3: Ilustrasi Precompute Serving Pattern (Sumber: Aurélien 2019)

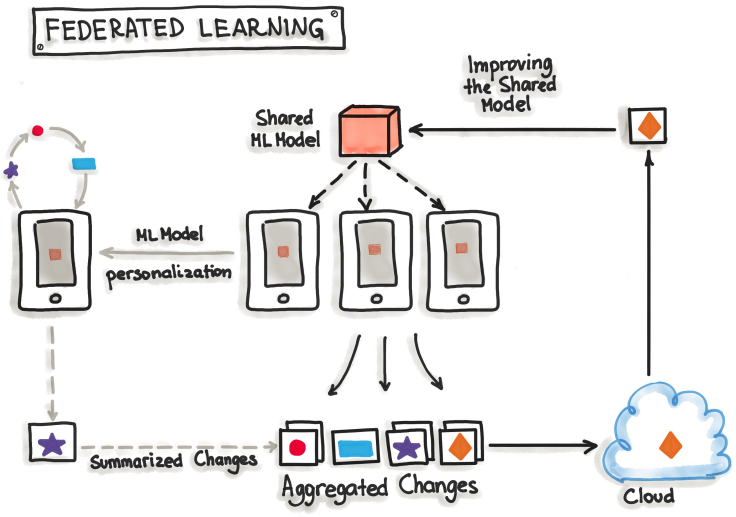


Gambar II.3.4: Ilustrasi Model-on-Demand (Sumber: Aurélien 2019)

yang dilakukan biasanya akan disimpan pada suatu basis data, yang nantinya akan digunakan oleh layanan tertentu. Sehingga, model tidak perlu dijalankan terus menerus; hanya perlu dijalankan bila diperlukan atau secara berkala saja.

II.3.4 Model-on-Demand

Pada dasarnya, *pattern* ini memiliki karakteristik yang serupa dengan Model-as-Service dengan perbedaan dalam infrastruktur pembentuk sistem. Perbedaan utamanya adalah dalam Model-as-Service, request dijalankan secara sinkron. Dalam *pattern* Model-on-Demand, terdapat penggunaan *broker* sehingga request terhadap model bisa dibuat asinkron (lihat Gambar II.3.4). Waktu prediksi juga bisa diatur lewat broker atau model yang digunakan.



Gambar II.3.5: Ilustrasi Federated Learning (Sumber: Aurélien 2019)

II.3.5 Hybrid Serving

Hybrid Serving, atau biasa dikenal dengan istilah Federated Learning adalah teknik yang memanfaatkan beberapa perangkat untuk proses *training*-nya. Tidak hanya melakukan *training* secara terpusat, model yang sudah jadi akan didistribusikan pada perangkat-perangkat berbeda, dan dari sana data baru akan dikumpulkan (lihat Gambar II.3.5). Data yang dikumpulkan akan dikirim ke sebuah central store untuk membuat model yang lebih mutakhir.

II.4 Otomatisasi

Dalam bagian ini, akan dibahas terkait prinsip dan konsep yang secara umum diterapkan dalam melakukan otomatisasi sistem. Selain itu, akan dibahas terkait aspek apa saja dari sebuah sistem yang diotomatisasi. Hal yang akan dibahas pada bagian ini adalah hal-hal yang sudah terbukti efektif untuk diotomatisasi.

II.4.1 Alasan Melakukan Otomatisasi

Otomatisasi di masa ini menjadi hal yang diusahakan oleh tim pengembang sistem dan perangkat lunak, terutama pada tim yang besar. Walaupun begitu, terkadang otomatisasi menjadi hal yang dikesampingkan dalam proses pengembangan, yang seharusnya dapat mengingkatkan produktivitas pengembangan.

Pada sub bagian ini, penulis akan membahas terkait beberapa alasan untuk melakukan otomatisasi (Beyer dkk. 2016), yaitu sebagai berikut:

- Menetapkan konsistensi dalam melakukan pekerjaan yang repetitif
- Memusatkan titik kesalahan dalam pengembangan sistem
- Mempercepat proses perbaikan sistem
- Memperkecil waktu aksi untuk melakukan suatu kegiatan terkait infrastruktur
- Menghemat waktu secara umum

Dalam otomatisasi, tentunya konsistensi bisa ditetapkan secara internal dengan asumsi faktor eksternal di luar dari kode sistem tidak ada masalah. Otomatisasi yang seharusnya dilakukan hanya melakukan satu hal saja secara konsisten. Bila terjadi masalah di luar fungsionalitas internal sistem, kemungkinan besar masalah tersebut berasal dari faktor eksternal.

Dengan melakukan otomatisasi juga, secara umum tim-tim menggunakan satu platform. Tentu tujuannya untuk mempermudah tim dalam melakukan otomatisasi, sehingga semuanya serba terpusat pada platform tersebut. Hal ini membawa dampak positif juga, karena semua masalah bisa ditelusuri dari platform tersebut.

Perbaikan sistem menjadi hal umum yang dilakukan di setiap proses pengembangan. Oleh karena itu, otomatisasi tentunya dapat membantu proses perbaikan agar lebih efisien, sehingga waktu ynag tersisa bisa digunakan untuk tugas atau kegiatan lain yang lebih produktif untuk pengembangan sistem.

Seperti dituliskan sebelumnya, mengurangi kegiatan yang repetitif dengan otomatisasi bisa menambah waktu untuk pengembangan sistem. Dalam kata lain, aksi-aksi tertentu juga dapat dilakukan lebih cepat lewat proses otomatisasi ini. Tentu saja, salah satu tujuan utamanya untuk menghemat waktu dari proses pengembangan ini. Walaupun mungkin tidak begitu terasa di awal dan mungkin malah menambah *overhead* dalam proses awal pengembangan, semakin lama nilai yang didapatkan dari otomatisasi semakin besar.

II.4.2 Aspek Otomatisasi

Dalam bagian ini tidak akan dibahas secara menyeluruh terkait aspek apa saja yang dapat diotomatisasi, karena pada dasarnya apapun bisa diotomatisasi dalam sebuah sistem, namun dampak yang diberikan bisa jadi belum tentu positif. Oleh karena itu, pada bagian ini aspek-aspek yang sudah terbukti.

Beberapa hal yang bisa dilakukan seputar otomatisasi sistem adalah sebagai berikut (Beyer dkk. 2016):

- Pembuatan user account baru
- Melakukan turnup dan turndown pada sebuah layanan di cluster
- Melakukan rilis versi baru
- Menginstall software dengan versi yang lebih baru

Tentunya, masih banyak hal yang bisa diotomatisasi. Daftar tersebut hanya daftar singkat dari apa yang bisa diotomatisasi dari sistem secara umum. Perlu kita ingat bahwa otomatisasi yang dilakukan belum tentu efektif, sehingga kita perlu mengujinya secara langsung untuk melihat apakah *feasible* untuk melakukan hal tersebut. Masalah-masalah yang ada dalam sebuah sistem dapat menjadi pedoman untuk membuat otomatisasi pada sistem lewat teknologi-teknologi yang ada.

II.5 Potensi Otomatisasi dalam MLOps

Seperti yang telah dibahas pada bagian sebelumnya, MLOps terdiri dari tiga tahapan utama, yaitu Data Engineering, Model Engineering, dan Code Engineering. Dalam praktiknya, bagian yang merupakan *toil* dalam ketiga proses ini berada paling banyak dalam tahap data engineering. Preprocessing dan eksplorasi data umumnya perlu butuh waktu yang lama karena melibatkan beberapa domain dan beberapa faktor lainnya. Sehingga, perlu metodologi tertentu untuk membantu dalam proses preprocessing data baik secara infrastruktur dan metode pemrosesannya.

Selain tahap tersebut, dalam tahap lain juga terdapat potensi untuk melakukan otomatisasi. Di masa ini, banyak kakas-kakas yang tersedia untuk membantu dalam proses MLOps. Contohnya, dalam Kubeflow terdapat fitur melakukan pipelining untuk melakukan proses-proses tertentu dalam langkah yang berbeda-beda.

BAB III

ANALISIS DAN PERANCANGAN

Pada bab ini, akan dibahas terkait literatur-literatur yang menjadi dasar dari pengembangan sistem konversi ini. Literatur yang dibahas juga akan membahas terkait permasalahan umum dalam MLOps dan apa saja hal-hal yang berpotensi untuk dikembangkan.

III.1 Analisis Masalah

Saat ini, konversi dari eksperimen ke sistem yang *production-ready* masih dilakukan secara manual. Dari ekperimen tersebut, biasanya terdapat proses-proses untuk melakukan pemrosesan data. Pemrosesan data bisa dilakukan secara langsung sebelum dimasukkan ke dalam model, maupun dilakukan prekomputasi untuk mempercepat proses inferensi dengan mengabaikan faktor masukan data. Dalam percobaan-percobaan yang memanfaatkan teknologi seperti Kubeflow, dapat digunakan sistem pipeline yang tersedia untuk melakukan eksperimentasi.

Dengan penggunaan pipeline dalam eksperimen pembelajaran mesin, proses eksperimen bisa dilakukan dengan lebih cepat. Hal tersebut karena pipeline didesain agar bisa digunakan kembali di beberapa proses yang serupa. Proses yang dimaksud bisa berupa proses untuk melakukan preproses pada data, atau bisa untuk melakukan proses pelatihan model. Pembuatannya yang membutuhkan script Python secara umum membuat proses pada pipeline ini bersifat atomik serta fleksibel untuk digunakan dalam pelatihan model yang berbeda.

Namun, penggunaan pipeline ini biasanya hanya digunakan untuk eksperimen saja. Pipeline yang dibuat untuk eksperimen ini biasanya tidak digunakan lagi dalam proses pembuatan sistem inferensi, karena hasil eksperimen biasanya akan disimpan dalam sebuah file yang hanya perlu dibungkus lewat suatu metode komunikasi, seperti yang telah dibahas pada bagian sebelumnya. Sedangkan, dengan penggunaan pipeline ini sebenarnya mungkin saja dapat membantu agar proses pembuatan sistem lebih mulus.

Dalam sebuah eksperimen yang sukses, model biasanya hanya hanya perlu untuk dibungkus dalam sebuah metode komunikasi sesuai dengan kebutuhan arsitektur, misal-

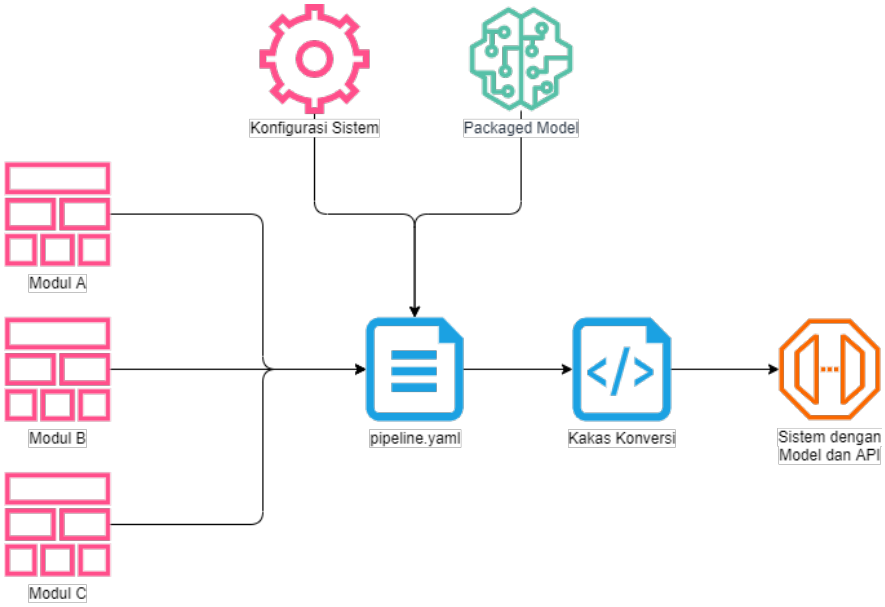
nya dengan gRPC atau REST. Perlu adanya perjanjian antar sistem untuk memastikan objek yang dikirim dan diterima sesuai dengan pihak yang membutuhkan sistem. Misalnya, seperti dalam gRPC yang menggunakan Protobuf terdapat konvensi perjanjian prosedur dan struktur data yang terlibat, atau JSON Schema dan Swagger dalam REST. Hal ini juga perlu menjadi perhatian dalam pembuatan sistem pembelajaran mesin.

III.2 Solusi Umum

Seperti yang disinggung sebelumnya, konversi eksperimen masih relatif manual dan belum memanfaatkan pipeline yang ada. Sehingga, terdapat potensi pengembangan untuk membuat suatu kakas yang bisa mengubah eksperimen menjadi sistem yang siap digunakan dari sisi pengembangan perangkat lunak Berdasarkan eksperimen yang dilakukan lewat pipeline, secara teori memungkinkan untuk mengubah komponen-komponen yang ada dalam pipeline tersebut menjadi beberapa bagian dalam suatu sistem. Solusi yang dibuat dapat memperhatikan juga struktur data dan prosedur sehingga masukan, keluaran, dan prosedur yang ada dalam sistem bisa terjamin.

Dalam sistem pembelajaran mesin, masukan data diberikan secara mentah tanpa pemrosesan apapun, tetapi terkadang terdapat data yang berbeda sehinga perlu dilakukan pemrosesan yang berbeda. Pipeline yang sudah ada dalam membuat suatu alur pemrosesan data di dalam sistem dapat dimanfaatkan. Bila diimplementasikan secara modular, untuk beberapa hal mungkin juga digunakan bahasa yang lebih cepat, sehingga hanya proses inferensi model saja dan beberapa hal saja yang sulit untuk dipindahkan ke bahasa lain.

Sebagai contoh, dalam pemrosesan data tabular umumnya terdapat beberapa tahapan dalam melakukan pemrosesan terhadap data. Tahapan *preprocessing* yang dimaksud misalnya bisa berupa normalisasi terhadap fitur, pembentukan fitur baru, seleksi fitur, dan transformasi fitur. Seperti pada transformasi fitur menggunakan metode seperti *Principal Component Analysis* (PCA) atau *Linear Discriminant Analysis* (LDA), tentunya akan memerlukan proses konversi dari data mentah menjadi data dengan dimensi yang berbeda.



Gambar III.3.1: Rancangan Solusi Konversi Eksperimen

III.3 Rancangan Solusi

Solusi yang saya tawarkan adalah membuat prototipe kakas yang dapat menganalisis suatu eksperimen dan mengubahnya menjadi suatu sistem yang koheren. Definisi dari perjanjian data yang menjadi masukan dan keluaran akan dibuat oleh sistem secara otomatis berdasarkan eksperimen. Alur pemrosesan yang didefinisikan dapat dibuat lewat *markup* file seperti JSON atau YAML yang formatnya sudah ditentukan untuk kakas tersebut. Alur didefinisikan lewat pemanggilan modul-modul yang diimplementasikan lebih dulu, dan nantinya dapat memanggil sebuah model yang biasanya sudah siap dan disimpan dalam sebuah file.

Modul dapat diimplementasikan dalam bahasa apapun, selama semua modul yang digunakan menggunakan bahasa yang sama. Dalam pendekatan yang umum, digunakan Docker sebagai bantuan untuk melakukan pemrosesan terhadap data. Dalam sebuah sistem yang koheren hal ini menjadi kurang baik, karena akan memerlukan komputasi yang lebih banyak dan akan membuang sumber daya.

Hasil akhir dari kakas ini adalah sebuah sistem yang siap digunakan untuk production. Pemilihan interface untuk model ini ditentukan lewat file markup yang telah dibuat. Metode komunikasi menggunakan gRPC dan REST akan diimplementasikan sebagai contoh metode yang umum digunakan.

DAFTAR PUSTAKA

Amazon Web Services (2021). URL: https://aws.amazon.com/devops/what-is-devops/. (diakses: 30.10.2021).

Aurélien, Géron (Okt. 2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. English. Paperback. O'Reilly Media, hal. 856. ISBN: 978-1492032649. URL: https://lead.to/amazon/com/?op=bt&la=en&cu=usd&key=1492032646.

Beyer, B. dkk. (2016). *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media, Incorporated. ISBN: 9781491929124. URL: https://books.google.co.id/books?id=81UrjwEACAAJ.

Dr. Larysa Visengeriyeva, Anja Kammer, Isabel Bär, Alexander Kniesz, dan Michael Plöd (2021). URL: https://ml-ops.org/. (diakses: 06.11.2021).

Rick Merrit (2021). URL: https://blogs.nvidia.com/blog/2020/09/03/what-is-mlops/. (diakses: 30.10.2021).

Sculley, D. dkk. (2015). "Hidden Technical Debt in Machine Learning Systems". Pada: *Advances in Neural Information Processing Systems*. Disunting C. Cortes dkk. Vol. 28. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf.

The Kubeflow Authors (2021). URL: https://www.kubeflow.org/docs/. (diakses: 18.11.2021).