

دانشکده مهندسی برق

**پایان‌نامه کارشناسی ارشد گرایش سیستم‌های الکترونیک دیجیتال**

طراحی و پیاده سازی یک پلتفرم MLOps به صورت ابری روی GPU

نگارش

**ابوالفضل یاریان**

استاد راهنما

**دکتر متین هاشمی**

... 1402

**برگه تصویب نامه**

تصویب نامه

به نام خدا

دانشگاه صنعتي شريف

دانشکده مهندسی برق

پایان نامه کارشناسي ارشد

این پایان­نامه به عنوان تحقق بخشی از شرایط دریافت درجه کارشناسي­ارشد است.

عنوان: طراحی و پیاده سازی پلتفرم MLOps به صورت ابری روی GPU

نگارش: ابوالفضل یاریان

کميته ممتحنین:

استاد راهنما: دکتر متین هاشمی امضاء.....................................

استاد راهنماي همکار:........................ امضاء.....................................

استاد مشاور:...................................... امضاء.....................................

تاريخ:......................................

برگه اظهارنامه

تقدیم به

این صفحه برای تقدیم پایان‌نامه از سوی دانشجو به افراد یا سازمان‌هایی که برای او مهم هستند در نظر گرفته شده است. اگر پایان‌نامه به کسی تقدیم نمی‌شود این صفحه را پاک کنید.

سپاسگزاری

این صفحه برای سپاسگزاری دانشجو از افراد یا سازمان‌ها در نظر گرفته شده است. اگر از کسی سپاسگزاری نمی‌شود این صفحه را پاک کنید.

# چکیده

در ابتدا هدف از پژوهش و روش پژوهش خود را بیان کنید. در پایان به نتایج و یافته‌های پژوهش اشاره کنید. تعداد کلمات چکیده بین 150 تا 250 کلمه بوده و در یک پاراگراف تنظیم می­شود. از بکارگیری مخفف ها و ذکر مآخذ در چکیده خودداری کنید.

کلیدواژه‌ها: کلیدواژه‌ها را این‌جا وارد کنید. (تعداد کلیدواژه ها 5 تا 7 کلمه)

فهرست مطالب

چکیده ‌ه

فهرست جدول‌ها ‌ح

فهرست تصویرها ‌ط

فهرست نمودارها ‌ي

فصل1 : عنوان فصل اول را اینجا وارد کنید 1

1-1 عنوان فرعی نخست را اینجا وارد کنید 1

1-2 عنوان فرعی دوم را اینجا وارد کنید 1

1-2-1 عنوان فرعی تر را اینجا وارد کنید 1

1-2-1-1 سرعنوان فرعی تر را اینجا وارد کنید 1

فصل2 عنوان فصل دوم را اینجا وارد کنید 2

2-1 عنوان فرعی نخست را اینجا وارد کنید 2

2-2 سرعنوان فرعی دوم را اینجا وارد کنید 2

2-2-1 سرعنوان فرعی تر را اینجا وارد کنید 2

2-2-1-1 سرعنوان فرعی تر را اینجا وارد کنید 2

فصل3 عنوان فصل سوم را اینجا وارد کنید 3

3-1 عنوان فرعی نخست را اینجا وارد کنید 3

3-2 عنوان فرعی دوم را اینجا وارد کنید 3

3-2-1 عنوان فرعی تر را اینجا وارد کنید 3

فصل4 عنوان فصل چهارم را اینجا وارد کنید 4

4-1 عنوان فرعی نخست را اینجا وارد کنید 4

4-2 عنوان فرعی دوم را اینجا وارد کنید 4

فصل5 عنوان فصل 5 را اینجا وارد کنید 6

5-1 عنوان فرعی نخست را اینجا وارد کنید 6

5-2 عنوان فرعی دوم را اینجا وارد کنید 6

منابع یا مراجع 7

پیوست1 8

# فهرست جدول‌ها

تصاویر و جداول بایستی دارای کاپشن اتوماتیک باشند. ((Insert Caption و تمام ارجاع به تصاویر و جداول بایستی به صورت Cross-reference باشند تا بتوان به طور اتوماتیک روزآمد کرد. برای انجام این فرایند در ورد 2007 روی گزینه References کلیک شود و بعد سعی کنید کاپشن جدول یا تصویر بسازید بعد جدول و تصویر را وارد نمایید. انجام این روند، فرایند روزآمدسازی و تهیه فهرست جداول و تصاویر را خیلی ساده می‏کند.

[جدول ‏4‑1- نمونه جدول 4](#_Toc524256067)

# فهرست تصویرها

[شکل ‏4‑1- نمونه تصویر 4](#_Toc524254629)

# فهرست نمودارها

[نمودار ‏4‑1- نمونه نمودار 5](#_Toc524254935)

# فصل اول – مقدمه

در این فصل تعریف مسئله MLOps، اجزا و مزایای آن، چالش های اصلی این حوزه و چهارچوب های کلی پایان نامه توضیح داده خواهد شد.

## تعریف مسئله، اجزا و مزایای آن

فرآیندهای پیاده سازی مدل بخشی پیچیده از چرخه عمر پروژه های یادگیری ماشین می باشد. استفاده از متدولوژی DevOps و ابزارهای یکپارچگی مداوم[[1]](#footnote-1) و ارائه مداوم[[2]](#footnote-2) در توسعه نرم افزار موفقیت هایی داشته است. یادگیری ماشین اغلب بخش کوچکی از سیستم های نرم افزاری است، اما هیچ گاه از این متدولوژی و ابزار CI/CD استفاده نکرده است. سیستم های یادگیری ماشین پیچیدگی و مشکلات منحصر به فردی را به پایپلاین CI/CD اضافه می کند استفاده از این متدولوژی در سیستم های یادگیری ماشین به عنوان MLOps شناخته می شود.

MLOps**، که مخفف عملیات یادگیری ماشین[[3]](#footnote-3) است، یک متدولوژی است که هدف آن بهبود کارآیی، قابلیت اطمینان و مقیاس پذیری جریان کارهای یادگیری ماشین در چرخه عمر آن ها می باشد. با افزایش انتشار مدل های یادگیری ماشین برای تصمیم گیری هایی مبتنی بر داده،** MLOps **به عنوان یک شاخصه حیاتی برای پر کردن شکاف بین توسعه[[4]](#footnote-4)، پیاده سازی[[5]](#footnote-5) و بهبود مداوم[[6]](#footnote-6) سیستم های یادگیری ماشین ظاهر می شود.** MLOps **شامل مجموعه ای از روش ها، ابزارها و رویکردهایی است که همکاری، اتوماسیون و نظارت را در چرخه عمر پروژه های یادگیری ماشین تسهیل می کند. این متدولوژی از عناصری چون** DevOps**، مهندسی داده و مهندسی نرم افزار استفاده می کند تا چالش های منحصربه فرد مرتبط با توسعه و راه اندازی مدل های یادگیری ماشین را برطرف کند.**

**اجزای کلیدی یک پلتفرم** MLOps **به صورت زیر می باشد:**

1. **یکپارچگی و ارائه مداوم (**CI/CD**) : این ابزار به منظور اتوماسیون فرآیندهای یکپارچه سازی تغییرات کدها، تست مدل های یادگیری ماشین و راه اندازی آن های در محیط های تولید استفاده می شود. این امر به معناست که سیستم های یادگیری ماشین به صورت سریع و قابل اعتماد توسعه، اعتبارسنجی و عرضه می شوند.**
2. **کنترل نسخه[[7]](#footnote-7): ابزارهای کنترل نسخه، مانند گیت، نقش حیاتی در این پلتفرم ها به دلیل امکان ردیابی تغییرات کدها و مجموعه داده ها دارند. این ابزار همکاری بین اعضای تیم را تسهیل می کنند، سابقه ای از نسخه های کد، داده را ایجاد می کنند و قابلیت تکرارپذیری را فراهم می کنند.**
3. **هماهنگی زیرساخت[[8]](#footnote-8): این ابزارها، مانند کوبرنتیز، به مقیاس پذیری کارآمد، تخصیص منابع و ظرفیت کانتینریزه کردن مدل های یادگیری ماشین کمک می کنند و راه اندازی قابل اعتماد و مقیاس پذیر را تضمین می نمایند.**
4. **نظارت و مدیریت مدل: نظارت بر عملکرد و رفتار مدل های یادگیری ماشین در محیط های تولیدی جزء جنبه های حیاتی این پلتفرم ها می باشند. این شامل پیگیری معیارهای کلیدی، تشخیص ناهنجاری و امکان مداخله پیشگیرانه برای حفظ دقت مدل است. مدیریت مدل شامل نسخه بندی و راه اندازی محیطی برای روزرسانی های مدل می باشد.**
5. **مدیریت داده: شامل پیش پردازش داده ها، نسخه بندی داده ها و تضمین کیفیت ویژگی های استخراج شده از داده ها می باشد.**
6. **یادگیری ماشین خودکار[[9]](#footnote-9): به خودکارسازی مراحل مختلف توسعه مدل های یادگیری ماشین از جمله پیش پردازش داده ها[[10]](#footnote-10)، مهندسی ویژگی[[11]](#footnote-11)، انتخاب مدل، تنظیم پارامتر ها[[12]](#footnote-12) و ارزیابی مدل اشاره دارد. هدف از یادگیری ماشین خودکار، ساده سازی و افزایش سرعت فرآیند ساخت مدل های یادگیری ماشین با استفاده از الگوریتم ها و استراتژی های بهینه سازی به منظور اتوماسیون بررسی و انتخاب بهترین ترکیب از تکنیک های پیش پردازش، معماری مدل و پارامتر ها می باشد. از این رو، به غیر از کارشناسان یادگیری ماشین، امکان ساخت مدل های یادگیری ماشین برای کسانی که در این حوزه متخصص نیستند، فراهم می کند.**

**از مزایای استفاده از پلتفرم های** MLOps **می توان به موارد زیر اشاده کرد:**

1. **افزایش بهره وری: در این پلتفرم ها، وظایف تکراری را اتوماسیون می شود که این امر به دانشمندان داده امکان می دهد تا بر روی وظایف اصلی یادگیری ماشین تمرکز کنند. این کار توسعه و راه اندازی را سریعتر و باعث بهبود بهره وری می شود.**
2. **قابلیت مقیاس پذیری و تکرارپذیری: این امر تضمین می کند که سیستم های یادگیری ماشین قادر به کار با مجموعه داده های بزرگ، تقاضای کاربران بیشتر و نیازهای محاسباتی پیچیده باشند.**
3. **بهبود قابلیت اعتماد مدل: این پلتفرم ها تست، اعتبارسنجی و مکانیزم های نظارت دربرمیگیرد و با نظارت بر عملکرد مدل و رفع مشکلات به موقع باعث جلوگیری از خطاها و انحراف داده ها[[13]](#footnote-13) می شود.**

**محصولات محبوب فعلی برای** MLOps **بیشتر متعلق به ارائه دهندگان ابری مختلف مانند آمازون[[14]](#footnote-14)، مایکروسافت[[15]](#footnote-15) و گوگل[[16]](#footnote-16) هستند. خدمات ارائه دهندگان ابری غالبا راه حل قابل قبولی برای شرکت هایی که روی سیستم های نرم افزاری نظارتی یا نرم افزاری هایی با نگرانی های مربوط به حریم خصوصی کاربر کار می کنند نیستند. آن ها به راه حل هایی نیاز دارند که که قابل اجرا روی ابرها و دستگاه های داخلی باشد. از سوی دیگر، به دلیل تحریم های اعمال شده، کاربران ایرانی امکان استفاده از این خدمات را ندارند. هم چنین، راه حل های منبع باز[[17]](#footnote-17) نیز که برای این منظور استفاده می شوند اغلب ناقص هستند و کار با آن دشوار می باشد.**

**هدف پایان نامه، طراحی و پیاده سازی یک پلتفرم** MLOps **برای سیستم های ابری محلی می باشد که نمی توانند به پلتفرم های معروف ارائه دهندگان ابری اتکا کنند. ما یک پلتفرم** MLOps **منبع باز و ساده بر پایه کوبرنتیز که دارای ابزارهای** CI/CD **و نظارت بر سیستم های یادگیری ماشین می باشد ارائه می دهیم. هم چنین ...**

## عنوان فرعی دوم را اینجا وارد کنید

متن را اینجا وارد کنید

### عنوان فرعی تر را اینجا وارد کنید

متن را اینجا وارد کنید

#### سر

#### عنوان فرعی تر را اینجا وارد کنید

# فصل دوم – مفاهیم پایه در MLOps

## مقدمه

...

## DevOps

**دِواپس (DevOps) - از اتحاد واژگان "development " و "operation" به معنای روش‌هایی است که تیم های توسعه و تیم operation را به همکاری موثر تر نزدیک میکند. ارزشهای روش دِواپس به فرایند توسعه نرم‌افزار اضافه می‌شود که با استفاده ازآن فرانید تولید نرم افزار چابک و خودکار میگردد. این خودکار سازی از منبع کد شروع می شود با استفاده از CI/CD pipeline تا مانیتورینگ محصول ادامه میابد .**

**در عمل، DevOps قصد دارد از ابزار و گردش کار برای خودکارسازی یک یا چند مورد استفاده کند مراحل DevOps :**

1. **مرحله کدنویسی: شامل توسعه، بررسی کد و ابزارهای کنترل نسخه است. مثلا، یک تیم تصمیم می گیرد از Git به عنوان ابزار کنترل نسخه و Github به عنوان یک مخزن راه دور استفاده کند. این تیم مجموعه‌ای از دستورالعمل‌های سبک کد اعمال‌شده خودکار و پوشش تست را تعریف می‌کند درصد، در مورد استفاده از استراتژی انشعاب توسعه تصمیم می گیرد و تمام تغییرات در شعبه اصلی را اعمال می کند تا توسط یک توسعه دهنده ارشد بررسی و پذیرفته شود.**
2. **مرحله استقرار: شامل ایجاد و ذخیره خودکار آثار است. مثلا، یک تیم تصمیم می گیرد یک image قابل اجرا از محصول خود ایجاد کند.**
3. **مرحله تست: شامل ابزارهای آزمایش مداوم است. تیم، محیطی را راه اندازی می کند که در آن یک مجموعه‌ای از آزمایش‌های مداوم به‌طور خودکار در برابر هر ویرایش کد اجرا می‌شود که بسیار مهم است. فرآیند تست برای هر مرحله از CI/CD می تواند متفاوت باشد.**
4. **مرحله انتشار: این مرحله شامل استراتژی انتشار است به طور مثال تیم می تواند تصمیم بگیرد که یک محصول به طور مستقیم منتشر شود یا ابتدا در یک محیط آزمایشی مورد ارزیابی قرار گیرد.**
5. **مرحله پیکربندی: شامل پیکربندی و مدیریت خودکار زیرساخت است.بهترین شیوه ها شامل اعلام زیرساخت تولید توسط کد است که به معنی مجموعه ای از اسکریپت ها برای بازتولید محیط در حال اجرا و زیرساخت نرم افزار شامل سیستم عامل تا پایگاه داده و سرویس های خاص و پیکربندی شبکه آنها است.**
6. **مرحله نظارت: از عملکرد محصول تا نظارت بر تجربه کاربر نهایی را شامل می شود. به عنوان مثال، می تواند مدت زمان پرس و جوهای پایگاه داده یا بارگذاری وب سایت یا تعداد آنها را پوشش دهد. همچنین می تواند عملکرد زیرساخت نرم افزار مانند سطح استفاده از CPU را مورد ارزیابی قرار دهد.**

**برای دستیابی به CI/CD، توسعه دهندگان "pipeline " را ایجاد می کنند که دارای manifest هایی برای ساخت خودکار، آزمایش، انتشار و پیکربندی انتشار نرم افزار است. رویدادی که مراحل را به صورت متوالی آغاز می کند. اگر هر مرحله ای با شکست مواجه شود، از پیشرفت بیشتر آن جلوگیری میکند و به توسعه دهندگان نرم افزار بازخورد میدهد.**

**در این فرایند CI/CD هر تغییری در کد به طور مداوم آزمایش شده و چنانچه ثابت شود که که قابل انتشار است، پایگاه کد در حالت کار باقی می‌ماند.**

**CI/CD از انجام تعهدات منظم، کوچک و حتی ده ها نسخه در روز پشتیبانی می کند. بدون CI/CD خطر بزرگی وجود دارد که شاخه های کد با عمر طولانی نیاز ادغام با نرم افزار دارند. این اتفاق ممکن است تمام توان تیم را به خود مشغول کند و منجر به اتفاقی به اصطلاح "جهنم ادغام" شود که در آن تحویل برای مدت طولانی ایی متوقف میشود.**

**با وجود CI/CD قوی هیچ هزینه اضافی برای یکپارچه سازی و تحویل وجود ندارد و هنگام پیاده سازی ویژگی ها، تیم توسعه نرم افزار می تواند چابک بماند.**

**pipeline را می توان برای اجرا در سرویس های میزبانی شده نوشت، به عنوان مثال:GitHub Actions، CircleCI یا Travisهمچنین ابزارهایی برای اجرای CI/CD در ماشین های شخصی وجود دارد، به عنوان مثال، Jenkins یا ArgoCD. این ابزارها یک زبان پیکربندی برای اجرا فراهم می کنند.**

**همچنین امکان تقسیم pipeline به چندین زیر ممجوعه وجود دارد به طور مثال ممکن است ساخت و آزمایش در GitHub Actions و انتشار روی یک ماشین محلی انجام می شود.**

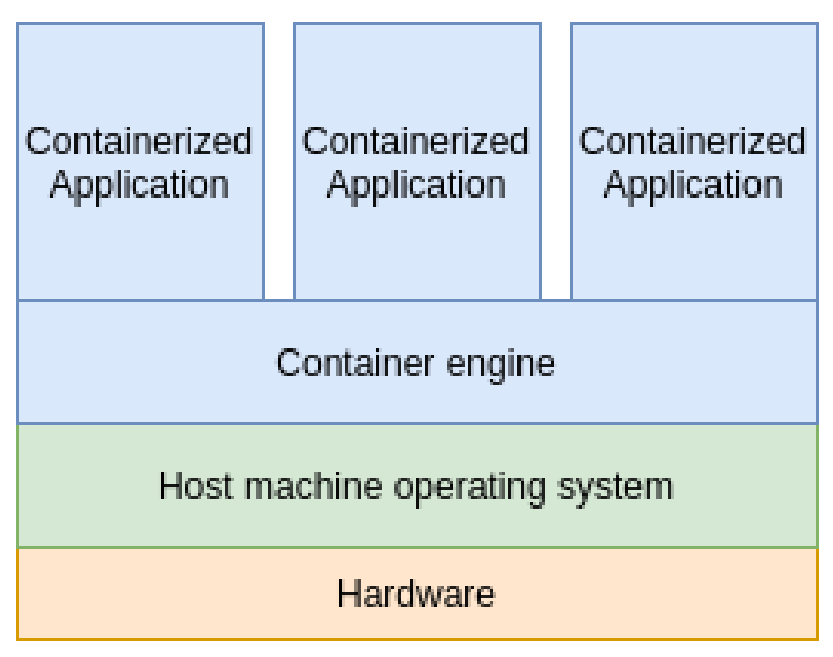
**عناصر کلیدی که باید در اجرای CI/CD در نظر گرفته شوند عبارتند از:**

1. **قابلیت بازگشت پذیری: برای هر نسخه منفرد، باید یک استراتژی برای بازگرداندن نسخه به نسخه قبلی وجود داشته باشد. یک راه حل آسان برای مواقع بروز خطا بازگشت به عقب است که می تواند از طریق همانpipeline CI/CD انجام شود اما بازگشت به عقب همیشه ساده نیست، حتی اگر بتوان سرویس در حال اجرا را به عقب بازگرداند امکان downtime وجود دارد .**
2. **قابلیت مشاهده رخداد ها و هشدار: هرrelease باید دارای اطلاعات شفافی از انچه تغییر کرده و همچنین کسی که این تغییر را ایجاد کرده است باشد و این باید برای تیم توسعه قابل مشاهده باشد . باید هشدار ها مناسبی از کد در فرایند خطوط تولید و همچنین احتمالات خطا در بخش releaseوجود داشته باشد چون اگر مشکلی در استقرار پیش بیاید، و هیچ مسیر حسابرسی واضحی از تغییرات وجود ندارد، دانستن اینکه کجا باید عقب نشینی کرد می تواند دشوار باشد.**
3. **ملاحظات امنیتی برای pipeline CI/CD عمدتاً به مدیریت افرادی است که می توانند استقرارها را مجاز کنند و همچنین نحوه مدیریت اسرار در pipeline مرتبط است . محدود کردن دسترسی توسعه دهندگان هم برای امنیت و هم برای مشاهده‌پذیری حیاتی است چون همه تغییرات تولید در pipeline CI/CD انجام می‌شود دادن دسترسی به ابزارهای CI/CD یک سیستم و اسرار آن می تواند سیستم را در صورت رخنه های امنیتی آسیب پذیر بگذارد.**
4. **زمان رسیدن محصول به محیط production: زمان تا تولید برای چابک ماندن یک شرکت ضروری است در حالی که تست ها و ساخت های طولانی مدت می تواند سایر نسخه ها را مسدود کند که منجر به بروز تعارضات زیاد در کدها، صرف انرژی زیاد برای تیم می شود.**

**3.4.1 – مجازی سازی و کانتینر سازی**

**مجازی سازی به معنای ایجاد یک لایه مجازی از سخت افزار، سیستم عامل یا ذخیره سازی است.برنامه‌ها را قادر می‌سازد تا سخت‌افزار و سیستم‌های زیربنایی خود را به‌صورت آگنوستیک اجرا کنند. در توسعه نرم افزار مدرن، مجازی سازی در سطح سیستم عامل اغلب از طریق کانتینرسازی انجام می شود. در این مدل ، هر برنامه یا سرویس توسط مجموعه ای از کانتینر های آن بر روی سیستم عامل یک سخت افزار اجرا میشود .**

**یک کانتینر از یک image ساخته می شود که الگوی دستورالعمل ساخت آن است .تمپلیت یک image می تواند شامل سیستم عامل و هر نرم افزار مورد نیاز برای اجرای خدمات یک سرویس باشد .یک image همچنین میتواند شامل کانفیگ های کانتینر مانند پورت های شبکه باشد .image ها میتوانند image های دیگر را به عنوان نقطه شروع کننده و ادامه دهنده از آن انتخاب کنند. یک Container در یک Container Runtime اجرا می شود، فرآیندی که چرخه زندگی یک کانتینر را مدیریت می کند.**

****

**معماری یک سیستم نرم افزاری متشکل از چندین جزء مستقل قابل استقرار ،میکروسرویس نامیده می شود، که در آن اجزاء به عنوان میکروسرویس ها نامیده می شوند .کانتینرها به عنوان واحدهای قابل استقرار برای میکروسرویس ها عمل می کنند.**

**3.4.2 : هماهنگ کردن کانتینر ها**

**برنامه‌ها و سیستم‌های نرم‌افزاری متشکل چندین کانتینر به هماهنگی نیاز دارند. ارکستراسیون به مدیریت lifecycle چندین کانتینر به هم وابسته مربوط میشود به طور مثال مسائل مربوط به مدیریت lifecycle یک کانتینر عبارتند از : چگونگی شروع و یا جایگزینی یک کانیتینر ،نحوه واکنش در مواقعی که یک کانتینتز به شکل غیر منتظره ای پایین می آید، نحوه اتصال یک کانتینر به یک ذخیره ساز ، نحوه فهمیدن ادرس شبکه ی دیگر کانتینر ها ، چگونگی تقسیم بار سرویس ها بین کپی های کانتینر .**

**با توجه به مطالب فوق در یک محیط production بزرگ که وضعیت سرویس ها و منابع به طور مداوم تغییر میکند هماهنگ سازی کانتینرها به شکل دستی تقریبا غیر ممکن است . خوشبختانه چندین پروژه نرم افزاری وجود دارد که ارکستراسیون خودکار را ارائه می دهد**

**3.4.3- kubernetess**

**Kubernetes یک استاندارد صنعتی برای ارکستراسیون کانتینرها در فضای ابری است Kubernetes یک پلتفرم منبع باز برای اجرای خدمات کانتینری به عنوان یک خوشه، چندین مزیت و ویژگی را برای اجرای برنامه های قابل اعتماد و قابل حمل ارائه می دهد.**

**Kubernetes کشف سرویس ، تعادل بار و DNS برای کانتینرها مهیا میکند و به طور خودکار ترافیک را بین چندین نسخه از یک کانتینر توزیع می کند و همچنین میتواند به طور اتوماتیک بر اساس بار cpu ماشین ها سرویس ها را تکرار کند**

**Kubernetes می‌تواند به‌طور خودکار نسخه‌های جدید سرویس‌ها را با زمان خاموشی صفر بازگرداند. نسخه‌های جدید سرویس را راه‌اندازی می‌کند، منتظر می‌ماند تا نمونه‌سازی شود یا پاسخ دهد.هنگامی که نسخه جدیدی از یک سرویس آماده میشود، به طور خودکار تمام ترافیک را به نسخه جدید تغییر می دهد و نسخه قدیمی را حذف می کند. توسعه دهنده‌هامیتوانندبرای Kubernetesمشخص کنند هر سرویس چقدر CPU و حافظه استفاده کند .**

**در زمینه یادگیری ماشین ما می‌توانیم از انتخابگرهای گره و برچسب‌های گره برای تعریف بارهای کاری خاص استفاده کنیم به طور مثال برای کار training سنگین یک مدل ماشین لرنینگ به یک دستگاه با کارت گرافیک قدرتمند سپرده شود که این کار توسط Kubernetes به خوبی انجام میشود .**

**Kubernetes یک هماهنگ کننده با ویژگی self-healing است، به این معنی که اگر کانتینری به طور غیر منتظره پایین بیاید و از کار بیفتد یا نتواند به درخواست های مانیتورنگی کاربر پاسخ دهد Kubernetes به طور خودکار آن را جایگزین می کند.همه منابع در یک خوشه Kubernetes به صورت بارهای کاری کانتینری در حال اجرا در pod ها هستند .غلاف ها برای اجرای کانتینرها به یک زمان اجرا کانتینر سازگار با OCI متکی است. یک pod مجموعه ای از کانتینرهای در حال اجرا بر روی یک خوشه و کوچکترین واحد قابل استقرار یک Kubernetes است. منابع بار کاری استاندارد در یک خوشه Kubernetes عبارتند از Deployment، Replica Set، StatefulSet، DaemonSet و Jobs .**

**در kubernetess امکان نوشتن کنترلر های سفارشی که مستقیما با این منابع کار میکنند از طریق Kubernetes API وجود دارند .به عنوان مثال، یک کنترلر سفارشی می تواند یک Cronjob باشد که مجموعه ای از jobs ها را در فواصل زمانی تعریف شده کنترل می کند.**

## MLOps

**MLOps عمل به کارگیری DevOps در یک پروژه نرم افزاری شامل یادگیری ماشین است یادگیری ماشین نقش ها و عناصر جدیدی را در فرآیند توسعه نرم افزار سنتی مانند داده ها، دانشمندان داده، مهندسان داده، مهندسان یادگیری ماشین ، مدل ها و مقررات آنها،pipeline های آموزش مدل و مانیتورینگ مدل را به ارمغان می آورد.این اضافات می تواند پیچیدگی یک سیستم تحویل مداوم را به میزان قابل توجهی افزایش دهد.در سیستم های یادگیری ماشین، pipeline CI/CD برای تسریع تحویل و بهبود تکرارپذیری استفاده می شود Valohai، یک استارتاپ فنلاندی، 330 مورد نظرسنجی را در حوزه یادگیری ماشینی، انجام داد موضوع نظر سنجی موضوعاتی بود که این شرکت ها در سه ماه قبل از نظر سنجی بر روی آن کار میکردند و موانع و مشکلات بر سر راه آن ها چه بوده است .پاسخ اکثر پاسخ دهندگان این بود که آنها در حال تلاش برای اتوماتیک کردن مراحل deploy تا production هستند همچنین میخواهند با استفاده از ابزار های موجود مراحل re-training و مانیتورینگ مدل را به صورت اتوماتیک انجام دهند.هنگام طبقه‌بندی پاسخ‌ها بین افرادی که در مراحل توسعه یادگیری ماشین هستند، یعنی در حال حاضر با مسائل مربوط به جمع‌آوری یا کشف چگونگی استفاده از داده ها، یادگیری فناوری های یادگیری ماشینی یا اثبات ارزش آنها، می بینیم که چالش های درک شده بیشتر به سمت مسائلی که مفاهیم MLOps سعی در حل آنها دارد.**

**مراحلی که می تواند کار دستی باشد یا توسط pipeline تحویل خودکار انجام شود. یک فرآیند یادگیری ماشین شامل موارد زیر است:**

**1. الزامات مدل**

**2. پاکسازی داده (ETL)**

**3. برچسب گذاری داده(ETL)**

**4. مهندسی ویژگی**

**5. آموزش مدل**

**6. ارزیابی مدل**

**7. استقرار مدل**

**8. نظارت بر مدل**

**مرحله الزامات مدل شامل جمع آوری اطلاعات و ایجاد درک نسبت به داده برای پیشبرد مراحل بعدی آموزش مدل می باشد.**

### استخراج-تبدیل-بار[[18]](#footnote-18)

**اولین چیزی که ساخت سیستم یادگیری ماشین به آن نیاز دارد یک روش ETL است.**

**مرحله استخراج: استخراج داده ها از چندین منبع داده بر روی یک production database، data store یا data lake است . داده های اصلی می توانند در هر اندازه ، فرمت یا موقعیت جغرافیایی باشند.**

**مرحله تبدیل: داده ها پاکسازی و برچسب گذاری می شوند، ناسازگاری، اطلاعات تکراری، متناقض یا گم شده حذف می شوند و داده ها به یک فرم خاص در می آیند .در نهایت فرم ها از منابع مختلف تجمیع میشوند و فرم نهایی شکل میگیرد.**

**مرحله بار: داده های تبدیل شده سپس در دیتابیس های محیط production و یا data store ها ذخیره میشوند تا pipeline های یادگیری یا دیگر کاربران برای تحلیل داده از آنها استفاده کنند.روش ETL می تواند به اندازه استخراج ردیف ها از یک CSV منفرد بر روی یک پایگاه داده کم عمق باشد و یا در مواردی میتواند متشکل از صدها عملیات تبدیل بر روی داده های استخراج شده در سطح بسیار وسیع ، وبارگذاری داده در اندازه ترابایت بر روی دریاچه داده باشد.**

### اکتشاف[[19]](#footnote-19)

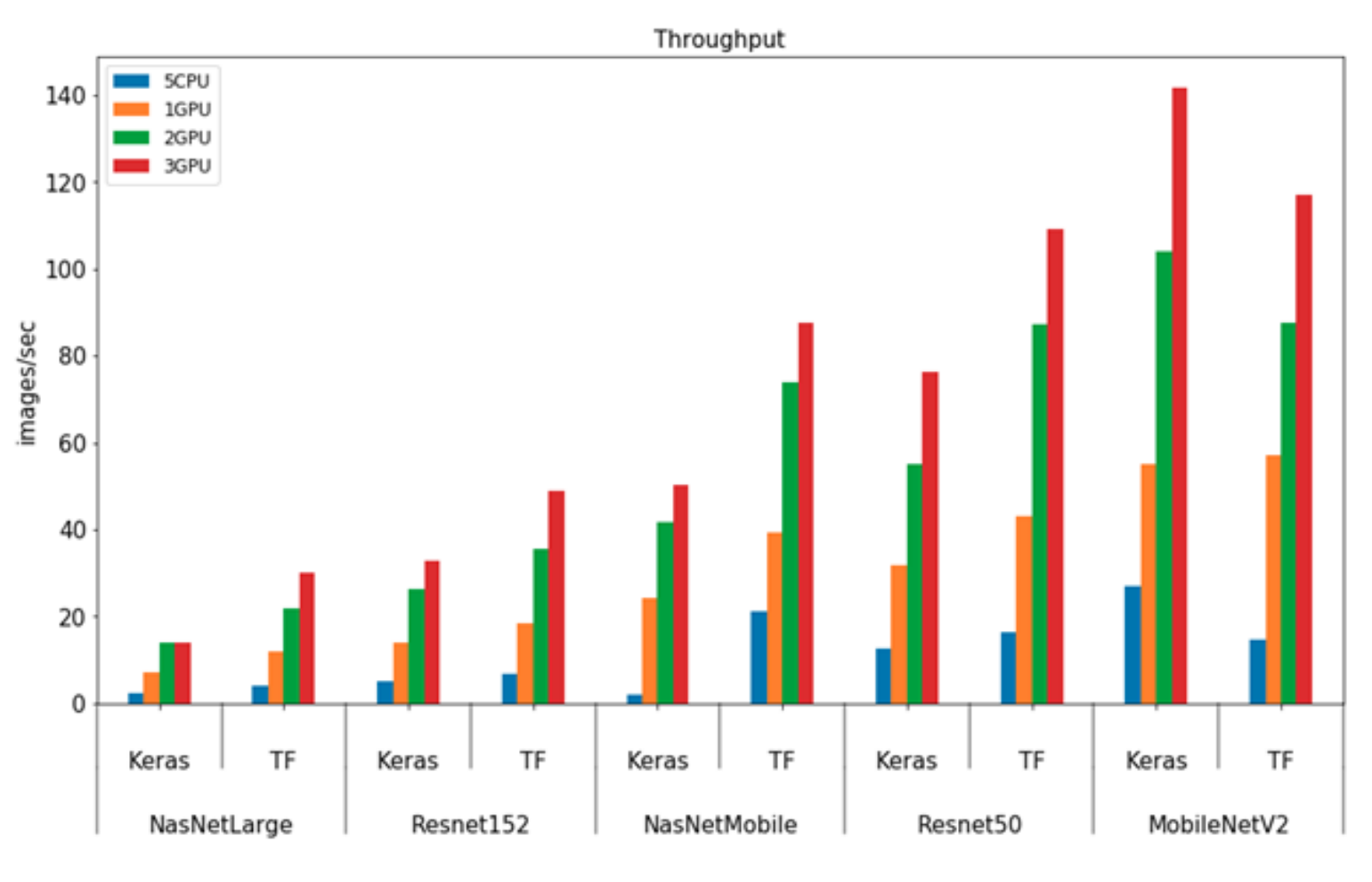
**مراحل اکتشاف بخشی از pipeline آموزش مدل و یا deployment در نظر گرفته نمی شود، اما اطلاعات ارزشمندی را در مورد اینکه چه چیزی و چگونه باید ساخته شود به تیم میدهد ، بنابراین این مرحله بخشی از فرایند MLOps نیست و اکتشاف داده فقط برای درک بهتری از داده ها انجام میگیرد چرا که این درک سبب میشود مدل ها و الگوریتم های بهتری برای کار بر روی داده ها انتخاب شود .در کاوش مدل، چندین مدل مختلف با داده ها آموزش داده می شوند تا مشخص شود کدام یک کاربردی تر به نظر می رسد.**

### خط لوله[[20]](#footnote-20) آموزش مدل

**آموزش مدل و بهینه سازی هایپرپارامترها پس از انتخاب مدل شروع می شود. مدل انتخاب شده با هایپر پارامتر های مختلف اموزش داده میشود و با استفاده از مجموعه از پارامتر ها ارزیابی میگردد. استراتژی های متعددی برای بهینه سازی هایپرپارامتر وجود دارد و این به پیاده سازی بستگی دارد که از چه چیزی استفاده کنیم.در مرحله ارزیابی، مدل‌های آموزش دیده با مجموعه از داده های دیده نشده مورد آزمایش قرار می‌گیرند که نتایج خوبی در انتخاب داشته باشند.**

**توصیه می شود تست عملکرد مدل با موارد لبه شناخته شده، ورودی های مخالف و سایر ورودی های انتخاب شده باشد تا سوگیری یا تبعیض ناخواسته در مدل ظاهر شود. این تست ها باید به عنوان یک معیار برای تمام تکرارهای ممکن مدل های آموزش دیده انجام شود تا یک معیار قابل مقایسه بین مدل های پیشنهادی به وجود آید ،مدلی با نتایج تست بهتر به عنوان production model انتخاب می شود.**

**مدل نهایی در مکانی ذخیره می شود، سرویس هایی که از آن استفاده می کنند می توانند به آن دسترسی داشته باشند. مدل ذخیره شده می تواند روی دیسک سرور، ذخیره سازی داده از راه دور یا داخل برنامه های مشتری باشد و یا فرآیندهای آموزش مدل هایی که از نظر محاسباتی بسیار سنگین هستند نیاز به دسترسی به منابع سخت افزاری GPU دارند. شکل زیر تفاوت های عملکردی فرآیندهای آموزشی مدل های مختلف را نشان می دهد ، مطابق آن می بینیم که استفاده از GPU ها عملکرد را به طور چشمگیری افزایش می دهد.**

****

### خط لوله[[21]](#footnote-21) سرویس مدل

**مدل ارائه شده یک رابط برای کاربران نهایی یا سرویس‌ها فراهم می‌کند تا داده‌ها را به آن وارد کنند. خروجی مدل به یک مکان مناسب ، معمولاً به وارد کننده داده یا به سرویس یا مدل دیگری برای پردازش بیشتر هدایت می شود. Interface server را می توان به عنوان یک وب سرور یا یک برنامه به مشتری ارائه کرد، مانند یک برنامه موبایل. Interface server به کد برنامه نوشته شده و یک زیرساخت قوی برای موفقیت نیاز دارد. زیرساخت باید مقیاس پذیر و self-healing برای بروز ماندن باشد . مدل به تنهایی نیازمند مانیتور شدن است تا بتواند bias ، drift و حملات خصمانه را رفع کند . وقتی که خطایی در مدل رخ میدهد و این خطا شناسایی میشود اولین عکس العمل باید swift کردن باشد تا از خطاهای بیشتر جلوگیری کند .**

**Retrain کردن مدل میتواند یک راه معمول برای برای رفع کردن bias و drift باشد که البته برای این آموزش دوباره به داده های جدید نیاز داریم . موارد گفته شده در این بخش که از جمله موارد حیاتی است بدون داشتن automation و version control به سختی قابل پیاده سازی است و فرایند تحویل نرم افزار را بسیار کند میکند .**

### عناصر MLOps

**MLOps ابر مجموعه ای از DevOps است. افزودن یادگیری ماشین به یک سیستم نرم افزاری پیچیدگی سیستم و همچنین pipeline CI/CD آن را گسترش می دهد و عناصر بیشتری باید در کنترل نسخه باشند .در این چرخه یک فرایند آموزشی حتی ممکن است چندین روز به طول انجامد همچنین منابع سخت افزاری مورد نیاز برا ی هماهنگ ساز ها[[22]](#footnote-22) معمولا ویژگی های خاصی باید داشته باشند.**

**CI/CD همیشه باید به آخرین آدرسی که محصول production در آن ذخیره شده است دسترسی داشته باشد. داده های ذخیره شده ممکن است به صورت روزانه و حتی بلادرنگ به روز شوند ، این دیتا ها معمولا در بستر اینترنت از محل ذخیره سازی به pipeline طراحی شده ارسال میشوند ، این اتفاق ممکن است به دلایل فاصله فیزیکی و حجم دیتا و ویژگی های اینترنت فرایند گفته شده را طولانی تر کند.**

**در حالی که نرم افزارهای سنتی نیاز به واحد، یکپارچه سازی، امنیت و تست سیستم انتها به انتها دارند، یک سیستم یادگیری ماشینی به همه اینها به اضافه داده ها و آزمایش مدل نیاز دارد. آزمایش ها به عنوان گام هایی در pipeline تعریف میشود و اگر هر آزمونی با معیارهای تعریف شده شکست بخورد،pipeline متوقف میشود و بنابراین مدل هرگز منتشر نمی شود. Pipeline همچنین می تواند شامل مراحلی باشد که انتخاب مدل براساس معیارهای تست و ارزیابی انجام می شود.از آنجایی که آموزش ممکن است از ساعت ها تا چند روز طول بکشد، training pipeline باید قادر به بازگرداندن پیشرفت از یک نقطه با حالت قبلی آن در صورت خطا یا تعلیق آن باشد، همچنین pipeline باید قادر به train کردن چندین مدل به صورت موازی و مستقل باشد.**

**هنگامی که یک ورژن از مدل ساخته میشود ، ورژن ساخته شده به یک ارزیابی و تست در خط لوله نیاز دارد تا پس از آن به عنوان یک محصول در محیط production ارائه شود .یک مدل به وسیله یک tag number ، metadata از ها یپرپارامتر ها ، دیتاست آن ، ورژن تست و آموزش آن شناسایی می شود و به وسیله این اطلاعات یک مدل میتواند باز تولید گردد که مزیت بسیار خوبی برای pipeline ایجاد میکند.**

### نظارت بر استقرار و تولید

**وقتی یک مدل آموزش داده شده است – یا در نرم افزارهای سنتی، نسخه جدیدی از کد ایجاد می شود معمول است که آن را در یک stageقبل از شروع به تولید مستقر کنید. بعد از گذراندن محیط stage، مصنوعات جدید ممکن است به طور کامل با استقرارهای مختلف مستقر شوند به عنوان مثال به عنوان یک انتشار A/B ، که در آن ترافیک داده بین نسخه قدیمی و نسخه جدید تقسیم می شود و سهم نسخه جدیدتر به طور مدوام افزایش می یابد. این استراتژی باعث می شود نسخه جدیدتر با بخشی از کاربران نهایی تست و ارزیابی شود و در صورت بروز اشکال از خسارت های بزرگ تر جلوگیری شود.**

**Pipeline به طور خودکار مدل های ارائه شده با استفاده از ابزار های مختلف مانیتور میکند تا عملکرد محصول را ارزیابی کند و خطاهای احتمالی را تشخیص دهد و پس از آن با اتخاد تصمیم درست مانند آموزش مدل با داده های بروز مدل جدیدی را برای ارائه آماده کند.**

**از اقدامات نظارتی مختلف برای اطمینان از بازخورد سریع بر هشدارهای سیستم، باگ ها و خرابی ها استفاده میشود سیستم مانیتورینگ نرم افزاری است که به طور فعال داده ها را از سرویس تولید می خواند یا دریافت می کند. به عنوان مثال، یک سرویس ممکن است همه خطاهای نرم افزاری در یک سیستم مانیتورینگ ارسال کند یا یک سیستم مانیتورینگ می تواند معیارهای مانیتورنیگ را مانند کاربران هم زمان بخواند . یک سیستم مانیتورینگ معمولا به thresholds های تعریف شده توسط کاربر با ارسال هشدار یا حتی تغییر متغییر های محیط production واکنش نشان میدهد . سیاست سیستم مانیتورینگ می تواند متفاوت باشد به عنوان مثال، هنگامی که drift تشخیص داده می شود، مشخص نیست که آیا مدل فعلی باید اجازه داده شود آنلاین بماند یا خیر. در برخی حالت ها مدل باید بلافاصله متوقف شود ویک مدل جدید جایگرین شود و در مقابل ممکن است drift رخ داده بحرانی نباشد و مدل قبلی بتواند آنلاین بماند و فقط به یک آموزش مجدد نیاز داشته باشد البته خط مشی اقدام باید برای هر نسخه قابل تغییر باشد، به عنوان مثال، هنگام تشخیص نرخ خطای بالا، عمل می‌تواند بازگشت به نسخه قبلی باشد، اما چنانچه نسخه قدیمی تر دارای یک خطای بسیار بزرگ باشد عملکرد ساده میتواند ارسال هشدار برای توسعه دهندگان برنامه باشد که البته در بسیاری از حالت این پاسخ موثر نیست .**

### سرعنوان فرعی تر را اینجا وارد کنید

متن را اینجا وارد کنید

#### سرعنوان فرعی تر را اینجا وارد کنید

# مروری بر ارائه دهنگان سرویس های MLOps

## مقدمه

در این فصل به مرور و بررسی سرویس های سه شرکت بزرگ آمازون، مایکروسافت و گوگل در زمینه MLOps در سه بخش مجزا می‌پردازیم و در نهایت آن ها را مقایسه می‌کنیم

## سرویس AWS SageMaker

یک سرویس ابری قدرتمند برای توسعه و استقرار مدل های یادگیری ماشین و هوش مصنوعی است. بااستفاده از آن، کاربران می توانند به راحتی الگوریتم های یادگیری ماشین را آموزش داده و آن ها را در مقیاس بزرگ پیاده سازی کنند. این سرویس به کاربران امکان مدیریت داده ها، تجزیه و تحلیل آن ها، آموزش مدل ها یادگیری ماشین، ارزیابی مدل ها و توسعه و پیاده سازی مدل ها یادگیری ماشین به منظور استفاده از آن ها را می دهد. هم چنین می توان از این سرویس به منظور توسعه برنامه های نرم افزاری مرتبط با یادگیری ماشین استفاده نمود. این سرویس با ارائه یک زیرساخت امن و قابل اعتماد ابزاری مناسب برای پروژه های یادگیری ماشین و هوش مصنوعی در محیط ابری است.

**این سرویس برای هرکدام از ب ...**

### یادگیری ماشین خودکار

سرویس SageMaker ابزار زیر را به منظور خودکارسازی پروسه یادگیری ماشین و استفاده از راه حل های کم کد-بدون کد[[23]](#footnote-23) ارائه می دهد:

#### SageMaker Autopilot

یک ابزار قدرتمند است که با خودکارسازی فرآیند ساخت و استقرار مدل های یادگیری ماشین، جریان کاری[[24]](#footnote-24) یادگیری ماشین را ساده تر و سریع تر می‌کند. از جمله وظایفی که این ابزار انجام می دهد به شرح زیر می باشد:


      Overview of Amazon SageMaker Autopilot AutoML process.
    

1. **تجزیه، تحلیل و پیش پردازش داده: Autopilot ابتدا داده ها را تحلیل کرده و پس از اعمال پیش پردازش هایی نظیر نرمال کردن[[25]](#footnote-25)، حذف و مدیریت مقادیر گمشده[[26]](#footnote-26)، انتخاب ویژگی های مناسب، داده ها را برای آموزش مدل آماده می کند.**
2. **انتخاب مدل: الگوریتم های مختلف را بررسی کرده و با استفاده از تکنیک های اعتبارسنجی متقابل[[27]](#footnote-27)، کیفیت پیش بینی هر الگوریتم را براساس معیارهای هدف از پیش تعیین شده ارزیابی می کند.**
3. **بهینه سازی هاپرپارامترها: به صورت کاملا خودکار پیکربندی های بهینه برای هایپرپارامترها جستجو می کند. این پیکربندی ها بسیار مهم بوده و بر عملکرد مدل تاثیر بسزایی دارد.**
4. **آموزش و ارزیابی مدل: به صورت خودکار چندین مدل را آموزش می دهد و ارزیابی می کند. در این بخش ابتدا داده را به دو مجموعه ی آموزش و اعتبارسنجی تقسیم می کند، سپس مدل های انتخاب شده را با استفاده از داد های آموزش آموزش می دهد و عملکرد آن ها را با استفاده از داده های اعتبارسنجی ارزیابی می کند و آن ها را براساس عملکردشان رتبه بندی می کند تا بهترین مدل را شناسایی کند.**
5. **استقرار مدل: پس از شناسایی بهترین مدل، Autopilot امکان استقرار خودکار مدل را ارائه می دهد. در پایان با استفاده از Restful API می توان با ارسال داده به مدل استقرار یافته، پیش بینی ها یا استنتاج ها را از مدل دریافت کرد.**

**لازم به ذکر است که این ابزار قابلیت ساخت مدل های یادگیری ماشین روی داده های حجیم را نیز دارد.**

**از Autopilot می توان بسته به سطح راحتی و نیازها به روش زیر استفاده نمود:**

1. **حالت کاملا خودکار با رابط کاربری Amazon SageMaker Studio: این رویکرد نیاز به حداقل کدنویسی یا دانش فنی دارد. از این روش تنها به منظور رگرسیون یا طبقه بندی در داده های جدولی استفاده می شود.**
2. **حالت راهنمایی انسانی با متغیرهای خاص: در این حالت، می توانید متغیرهای خاصی را به experiment خود اضافه کنید که میزانی از راهنمایی انسانی را ممکن می سازد. این قابلیت امکانات سفارشی سازی و کنترل بیشتری را در experiment یادگیری ماشین فراهم می کند.**
3. **پیاده سازی مد با استفاده از کیت های توسعه نرم افزاری[[28]](#footnote-28): با استفاده از SDK ها نیز می توان از Autopilot استفاده نمود. این رویکرد امکانات بیشتری از نظر انعطاف پذیری و سفارشی سازی را فراهم می کند.**

**از Autopilot می توان برای موارد زیر استفاده نمود:**

1. **داده های جدولی: داده هایی به فرمت CSV و Parquet که هر ستون آن بیانگر یک ویژگی و هر سطر آن بیانگر یک مشاهده می باشد. داده ها نیز به صورت عددی، متن و زمان می باشد. بر روی این داده ها می توان رگرسیون و طبقه بندی دو تا چندکلاسه را استفاده نمود.**
2. **داده های متنی: داده هایی به صورت CSV و Parquet می باشد. برروی این داده ها تنها می توان طبقه بندی متن را اعمال کرد.**
3. **داده های تصویری: به فرمت PNG یا JPRG می باشند که می توان طبقه بندی تصویر را روی آن ها اعمال کرد.**

**هم چنین Autopilot گزارش هایی شامل تجزیه و تحلیل اهمیت ویژگی ها، معیارهای عملکرد مدل، ماتریس اشتباهات[[29]](#footnote-29) و منحنی هایی نظیر ROC، دقت-بازخوانی[[30]](#footnote-30) و ... هستند.**

#### SageMaker JumpStart

**این ابزار مدل های منبع باز و از قبل آموزش دیده را برای انواع مسائل رایج فراهم می کند. به کمک این ابزار علاوه بر استفاده از این مدل ها می توان آن ها را برای مسئله شخصی کاربر آموزش داد. هم چنین این ابزار قالب های از پیش آماده شده ای را برای استفاده از این مدل ها و آموزش آن ها ارائه می دهد( مثل notebook).**

### برچسب گذاری داده ها

به منظور آموزش مدل های یادگیری ماشین ما نیاز به برچسب داده ها داریم. به این منظور می توان از ابزار **SageMaker Ground Truth** برای برچسب گذاری داده ها استفاده کرد. این داده ها توسط افرادی که با شرکت آمازون قرار دارد دارند انجام می شود.

### آماده سازی و پیش پردازش داده ها

**با استفاده از SageMaker Data Wrangler، می توان داده ها را به آسانی فراخوانی کرد و پس از اعمال پیش پردازش بر آن ها ویژگی ها مهم را استخراج نمود. تمام این روند به صورت راه حل های کم کد-بدون کد می باشد.**

### پردازش داده ها

**پیش از آموزش مدل در فرآیند آموزش مدل یادگیری ماشین، داده ها ارزیابی و پیش پردازش خواهند شد. هم چنین پس از آموزش مدل نیاز داریم که مدل آموزش داده شده را به منظور آگاهی از عملکرد مدل ارزیابی نماییم. ابزار SageMaker Processing برای وظایف مختلفی نظیر پردازش داده، استخراج ویژگی، اعتبارسنجی داده و هم چنین ارزیابی مدل و تفسیر مدل استفاده می شود.**

### استخراج و ذخیره سازی ویژگی ها

فرآیند توسعه یادگیری ماشین معمولا با استخراج سیگنال های داده (ویژگی[[31]](#footnote-31) ها) از داده ها برای آموزش مدل های یادگیری ماشین آغاز می شود. با استفاده از ابزار SageMaker Feature Store کاربر امکان استخراج، اشتراک گذاری و مدیریت ویژگی ها را دارا خاوهد بود. این ابزار با کاهش کارهای تکراری پیش پردازش داده و انتخاب داده ها خام به منظور تبدیل آن ها به ویژگی برای آموزش مدل های یادگیری ماشین، فرآیند استخراج و مدیریت را تسریع می دهد. هم چنین از آنجایی که از منطق یکسانی هم برای داده های آموزش و هم داده های پیش بینی استفاده می شود، انحراف ارائه آموزش[[32]](#footnote-32) را کاهش می دهد.

**با استفاده از این ابزار کاربر می تواند برای ویژگی های خود از پایگاه داده های آنلاین یا آفلاین استفاده نماید. از پایگاه داده آنلاین برای استنتاج به صورت بلادرنگ با تاخیر کم استفاده می شود در حالی که از پایگاه داده آفلاین به منظور استنتاج دسته ای[[33]](#footnote-33) از داده ها که نیاز به تاخیر کم نیست استفاده می شود. از این پایگاه داده اصولا به منظور آموزش مدل استفاده می شود.لازم به ذکر است که در پایگاه داده های آفلاین اصولا آخرین ویژگی ها حفظ خواهند شد و همیشه ویژگی در حال تغییر می باشند.**

### آموزش مدل

پس از استخراج ویژگی ها، حال نیاز به آموزش مدل ها به وسیله این ویژگی ها داریم. با استفاده از ابزار SageMaker Training کاربر می تواند مدل دلخواه خود را آموزش دهد. هم چنین می توان با استفاده از محیط SageMaker Notebook مدل و معماری دلخواه خود را طراحی کرد و بر روی زیرساخت ابری شرکت آمازون آموزش دهد.

### استقرار مدل

با استفاده از SageMaker Endpoint کاربر می تواند مدل آموزش دیده خود را به عنوان یک نقطه پایانی[[34]](#footnote-34) مستقر کند. نقطه پایانی یک سرویس وب است که به شما این امکان را می دهد که با استفاده از آن در بستر اینترنت، پیش بینی های لحظه ای انجام دهید. در این قسمت کاربر می تواند میزان منابع مورد نیاز برای استقرار مدل خود را تعیین نماید.

### نظارت بر کیفیت داده ها و مدل

**با استفاده از SageMaker Model Monitor می توان کیفیت مدل های یادگیری ماشین را در محیط تولید ارزیابی و نظارت نمود. به وسیله این ابزار در صورتی که اتفاقی برای مدل رخ دهد یا به عبارتی دقت و کیفیت آن کاهش یابد، هشداری مبنی برای آن به کاربر داده می شود. برای جلوگیری از این کاهش دقت باید به صورت متناوب مدل خود را دوباره آموزش دهیم تا بتواند به خوبی به داده های جدید اگاهی یابد. هم چنین از این ابزار برای نظارت بر کیفیت داده به منظور آگاهی از رانش[[35]](#footnote-35) داده ها استفاده می گردد.**

# عنوان فصل چهارم را اینجا وارد کنید

## عنوان فرعی نخست را اینجا وارد کنید

## عنوان فرعی دوم را اینجا وارد کنید

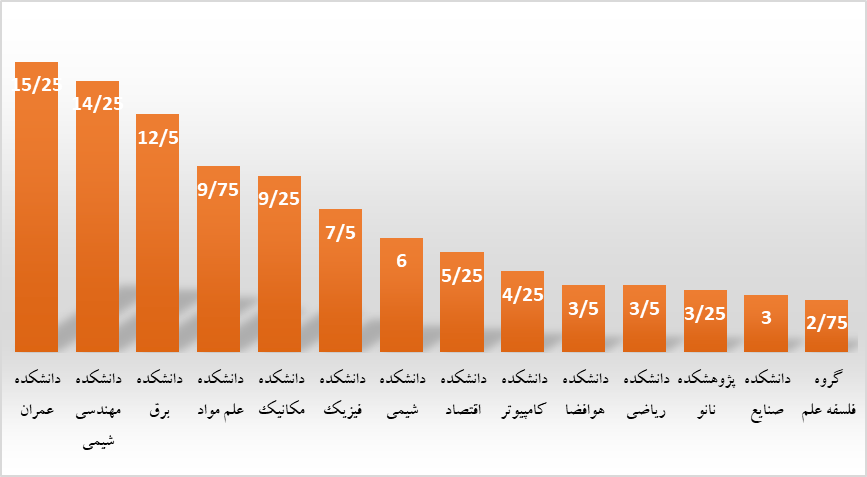
متن را در اینجا وارد کنید (شیوه ارجاع شکل یا جدول در متن جدول ‏4‑ 1) (شکل ‏4‑1)...

جدول ‏4‑1- نمونه جدول

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **رديف** | **ملاک‌ها** | | **ضريب آلفا** | |
| **تحليل پرسشنامه اصلی** | **تحليل پرسشنامه فرعی** |
| 1 | **تمايل به استفاده** | | 7505/0 | 7108/0 |
| 2 | **رضايت‌مندي** | | 7965/0 | 7430/0 |
|  | 2-1 | **مفيدبودن** | 6860/0 | 9216/0 |
| 2-2 | کارآمدي | 8163/0 | 6849/0 |



شکل ‏4‑1- نمونه تصویر

نمودار ‏4‑1- نمونه نمودار

# عنوان فصل 5 را اینجا وارد کنید

## عنوان فرعی نخست را اینجا وارد کنید

متن را اینجا وارد کنید

## عنوان فرعی دوم را اینجا وارد کنید

متن را اینجا وارد کنید

# منابع یا مراجع

(دو شیوه ماخذنویسی وجود دارد به این ترتیب که 1) می­توانید مراجع خود را به ترتیب الفبایی نام خانوادگی نویسنده تنظیم نمایید، در این صورت مراجع فارسی در ابتدا و سپس مراجع لاتین می آید، 2) به ترتیب عددی و براساس حضور مراجع در متن تنظیم نمایید. بدین معنی که اولین مراجعی که در متن استفاده کردید عدد یک و ادامه پیدا می کند. ولی ترجیح با تنظیم الفیایی می باشد.

نام خانوادگی، نام (سال). عنوان و مشخصات کتابشناختی منبع.

نام خانوادگی، نام (سال). عنوان و مشخصات کتابشناختی منبع.)

برای مثال (به ترتیب الفبایی):

* کریمی، مهدی (1387). بررسی تولیدات علمی نویسندگان دانشگاه های ایران همراه با شبکه های تالیف مشترک. فصلنامه علمی و پژوهشی مدیریت. 11 (34): 59-67.
* نیکزاد، مهسا (1389). بررسی تطبیقی شبکه های هم تالیفی در مقالات ایرانی رشته های کتابداری و اطلاع رسانی، روانشناسی، مدیریت و اقتصاد در پایگاه ISI بین سال های 2000 تا 2009. پایان نامه کارشناسی ارشد. دانشگاه آزاد اسلامی، واحد علوم و تحقیقات تهران.
* ولایتی، خالید (1387). بررسی میزان همکاریهای علمی بین ایران و کشورهای همچوار طی سال های 1990 تا 2007. پایان نامه کارشناسی ارشد. دانشگاه تهران.
* Abramo, G. et al (2009). Research collaboration and productivity: is there correlation? High Education, 57:155–171.
* Braun, T. & Glanzel, W (2001), Publication and cooperation patterns of the authors of neuroscience journals, *Scientometrics*, 51 : 499–510
* Chakrabarti, D. & Faloutsos, C. (2006), Graph mining: Laws, generators, and algorithms, ACM Computing Surveys, 38 :1–69.
* Cho, Cheng-Chung, Hu, Ming-Wen, & Liu, Meng-Chun (2010). Improvements in productivity based on co-authorship: a case study of published articles in China. Scientometrics 85: 463-470.
* Durden, G., & Perri, T. J. (1995). Co-authorship and publication efficiency. Atlantic Economic Journal, 23(1):69–76.
* Englebrecht, T. D. et al (2008). An assessment of patterns of co-authorship accountants within premier journals: evidence from 1979-2004. Advances in International Accounting, 24: 172-181.



Abstract

The abstract is the most important section of the thesis because many readers limit most of their reading to abstracts, saving in-depth reading for specific theses. It should give the reader a "preview" of what's to come. The abstract should emphasize new and important aspects of the study or observations. The purpose of the abstract is to allow researchers to decide whether or not to read the whole thesis. The abstract is what researchers read first to decide if the thesis is important, interesting, and it allows them to assess the relevance of a thesis to their own research, without having to read the entire thesis. Thus, it is crucial that the abstract both summarize succinctly the key findings of the thesis and clearly articulate what is novel and important about the work. It should be able to stand alone as a representation of the research without any footnotes. Any information included in the abstract must also be included in the body of the thesis. Although the abstract comes first in the thesis, it is usually easiest to write the abstract last after completing the other sections of the thesis. It is one of the most difficult sections to write. The abstract should generally not contain non-standard acronyms or abbreviations and should not include citations: Briefly, the abstract, introduces topic; mentions techniques used without going into experimental detail; summarizes most important results; is most appropriately written last.

**Keywords (5 to 7 keywords):**

****

**Sharif University of Technology**

**(****Faculty Name)**

Thesis Title

By:

Student Name

Supervisor:

Dr.

Advisor:

Dr.

Month, Year

1. Continuous Integration (CI) [↑](#footnote-ref-1)
2. Continuous Delivery(CD) [↑](#footnote-ref-2)
3. Machine learning and Operation [↑](#footnote-ref-3)
4. Development [↑](#footnote-ref-4)
5. Deployment [↑](#footnote-ref-5)
6. Continuous improvement [↑](#footnote-ref-6)
7. Version Control [↑](#footnote-ref-7)
8. Infrastructure Orchestration [↑](#footnote-ref-8)
9. AutoML [↑](#footnote-ref-9)
10. Data pre-processing [↑](#footnote-ref-10)
11. Feature engineering [↑](#footnote-ref-11)
12. Hyperparameter tuning [↑](#footnote-ref-12)
13. Data Drift [↑](#footnote-ref-13)
14. AWS SageMaker [↑](#footnote-ref-14)
15. Azure Machine learning [↑](#footnote-ref-15)
16. GCP Vertex AI [↑](#footnote-ref-16)
17. Open-Source [↑](#footnote-ref-17)
18. Extract-transform-load (ETL) [↑](#footnote-ref-18)
19. Exploration [↑](#footnote-ref-19)
20. Pipeline [↑](#footnote-ref-20)
21. Pipeline [↑](#footnote-ref-21)
22. Orchestrators [↑](#footnote-ref-22)
23. No-Code Low-Code [↑](#footnote-ref-23)
24. Workflow [↑](#footnote-ref-24)
25. Normalizaion [↑](#footnote-ref-25)
26. Missing values [↑](#footnote-ref-26)
27. cross-validation [↑](#footnote-ref-27)
28. SDK [↑](#footnote-ref-28)
29. Confusion Matrix [↑](#footnote-ref-29)
30. Precision-recall Curve [↑](#footnote-ref-30)
31. Features [↑](#footnote-ref-31)
32. Training-serving skew [↑](#footnote-ref-32)
33. Batch [↑](#footnote-ref-33)
34. Endpoint [↑](#footnote-ref-34)
35. Drift [↑](#footnote-ref-35)