بسم الله الرحمن الرحیم

دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

یادگیری ماشین پیشرفته

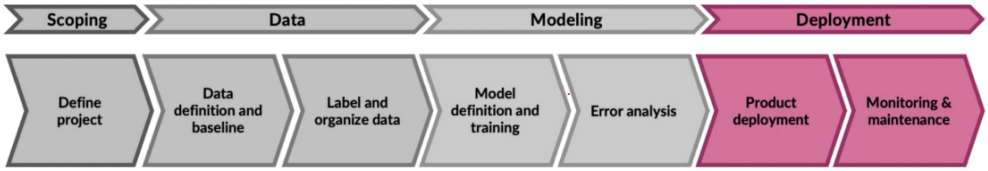
استاد محترم : دکتر زارع زاده
نیم‌سال دوم ۱۴۰۲-۱۴۰۱

گزارش پروژه

اعضای تیم : سید عارف جهانمیر یزدی و حسن صبور

استقرار مدل

در این فاز بعد از فاز قبل که روی داده و مدل بود تمرکز اصلی خود را برای بهبود مراحل اولیه و استقرار خوب مدل گذاشتیم :



شکل ۱ : مراحل انجام پروژه

(استقرار مدل)

استقرار یک مدل یادگیری ماشین یک مرحله مهم در فرآیند توسعه مدل است که نیازمند تصمیم‌گیری‌هایی درباره الگوی و استراتژی استقرار مدل می‌باشد. در اینجا من یک راهنمای کلی برای استقرار مدل را بر اساس مطالب فوق ارائه خواهم داد:

الگوی استقرار:

- استقرار در سرورهای فیزیکی یا مجازی:

  در این الگو، مدل روی سرورهای فیزیکی یا مجازی مستقر می‌شود. این روش برای پروژه‌هایی با نیازمندی‌های امنیتی یا اجرای مداوم مناسب است. از مزایای این الگو می‌توان به کنترل کامل بر سیستم و امکان تنظیمات دقیق اشاره کرد، اما نیازمندی‌های مربوط به مدیریت سخت‌افزار و نگهداری ممکن است چالش‌هایی را ایجاد کند.

- استقرار در ابر ($Cloud\ Deployment$): استفاده از سرویس‌های ابری مانند AWS، Azure یا Cloud Google برای استقرار مدل یک گزینه عالی است. این روش امکان انعطاف‌پذیری، مقیاس‌پذیری و مدیریت آسان را فراهم می‌کند. همچنین، شما تنها باید برای مصرف منابع پرداخت کنید. این روش معمولاً برای پروژه‌های کوچک تا متوسط و با منابع محدود پیشنهاد می‌شود.

استراتژی استقرار:

- استقرار مستقیم ($Direct\ Deployment$): در این روش، مدل به صورت مستقیم بر روی سرورها یا سرویس‌های ابری مستقر می‌شود و به واسطه وب سرویس‌ها یا هاAPI قابل دسترسی است. این روش برای مدل‌هایی که نیاز به پیش‌پردازش کمی دارند و یا به عبارت دیگر ورودی و خروجی ساده دارند مناسب است.

- استقرار به صورت کانتینرها ($Containerization$):

  استفاده از فناوری‌های مانند $Docker$ برای ایجاد کانتینرها به شما امکان می‌دهد تا مدل، وابستگی‌ها و پیکربندی‌های لازم را در یک محیط منزلتی ایجاد کنید. این روش مزیت‌هایی از جمله انعطاف‌پذیری و تکرارپذیری را دارد و می‌تواند برای مدل‌های پیچیده و با وابستگی‌های مختلف مناسب باشد.

معیارهای انتخاب:

- کارایی ($Performance$): انتخاب الگوی استقرار و استراتژی مناسبی باید با توجه به کارایی مدل انجام شود. برخی مدل‌ها به پردازش سنگین نیاز دارند و این ممکن است در تصمیم‌گیری‌های مربوط به استقرار تاثیر داشته باشد.

- امنیت ($Security$): انتخاب الگوی استقرار باید با نیازمندی‌های امنیتی پروژه همخوانی داشته باشد. مدل‌ها ممکن است اطلاعات حساسی را پردازش کنند و از این رو اهمیت ویژه‌ای دارد که اطلاعات محافظت شوند.

- مقیاس‌پذیری ($Scalability$): در صورتی که پیش‌بینی شود مدل به مرور زمان بیشترین بار کاری را تحمل کند، استقرار با استفاده از سرویس‌های ابری می‌تواند مقیاس‌پذیری را آسان‌تر کند.

- هزینه ($Cost$): استراتژی استقرار باید با منابع مالی پروژه همخوانی داشته باشد. استفاده از سرویس‌های ابری ممکن است در مقایسه با سرورهای فیزیکی هزینه کمتری داشته باشد.

(معماری مدل)

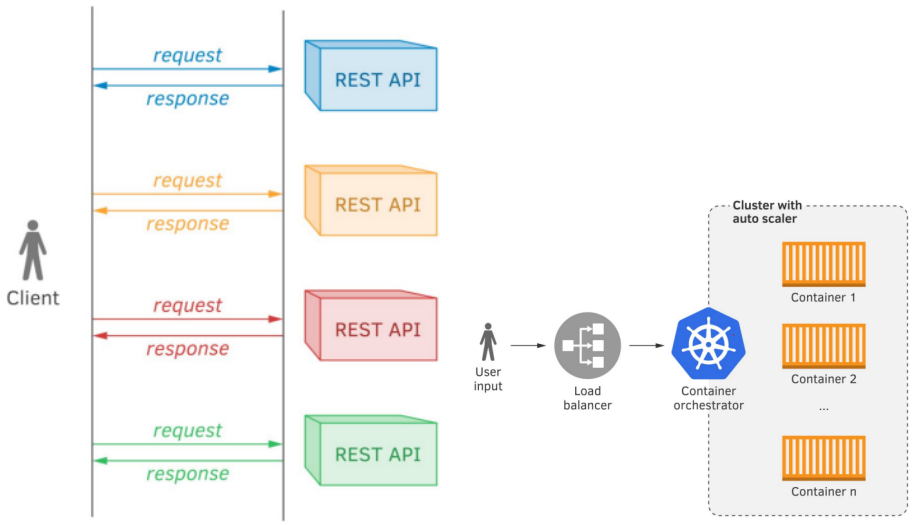
در معماری نرم‌افزاری، *dynamic deployment* به معنای انتقال و اجرای برنامه‌ها و سرویس‌ها بر روی یک سرور یا کانتینر به‌طور پویا و انعطاف‌پذیر است. در واقع، این مفهوم به ایجاد امکاناتی برای انتقال و اجرای برنامه‌ها به صورت خودکار و بدون توقف سیستم اشاره دارد.

یکی از اصول اساسی *dynamic deployment*، استفاده از مدیریت منابع خودکار و تخصیص بهینه منابع به برنامه‌هاست. به این ترتیب، وقتی بار کاری در سیستم افزایش یا کاهش می‌یابد، سیستم به‌طور خودکار تعداد منابع مورد نیاز را تغییر می‌دهد. این امر به بهبود کارایی و پایداری سیستم کمک می‌کند.

استفاده از کانتینرها در *dynamic deployment*بسیار متداول است. کانتینرها محیط‌های جداگانه‌ای هستند که شامل کد، وابستگی‌ها، کتابخانه‌ها، تنظیمات و هر آنچه برای اجرای یک برنامه لازم است می‌شوند. با استفاده از ابزارهای مانند Docker یا Kubernetes، می‌توان کانتینرها را بسیار سریع و در هر محیطی اجرا کرد، بدون نگرانی از تفاوت‌های محیطی.

با *dynamic deployment* در کانتینرها، می‌توانید برنامه‌ها را به صورت مداوم به‌روز رسانی کنید، تغییرات را به سرعت انجام دهید و حتی در صورت بروز مشکلات، با تغییر به نسخه‌های قبلی بازگشت کنید. این رویکرد به توسعه و مدیریت برنامه‌ها کمک می‌کند و به تیم‌ها امکان می‌دهد تا به‌سرعت واکنش نشان دهند و بازخوردهای کاربران را بهبود دهند.

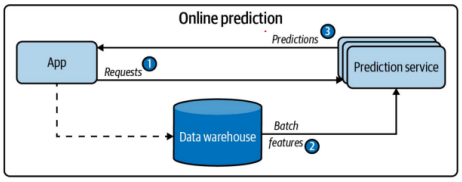
در کل، *dynamic deployment* با استفاده از کانتینرها و ابزارهای مدیریت محیط اجرایی، به توسعه‌دهندگان و مدیران سیستم امکان می‌دهد تا برنامه‌ها و سرویس‌ها را با انعطاف بیشتری اجرا و مدیریت کنند، تازه‌ترین تغییرات را به سرعت پیاده‌سازی کنند و به بهبود کارایی و پایداری سیستم دست یابند.



(*Batch vs Online Serving*)

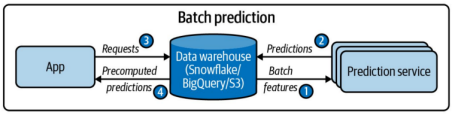
- پیش‌بینی آنلاین (*Online Prediction*):

پیش‌بینی آنلاین یا پیش‌بینی در زمان واقعی به معنای استفاده فوری از یک مدل یادگیری ماشین برای پیش‌بینی خروجی بر اساس ورودی است. در این الگو، وقتی که یک درخواست برای پیش‌بینی در سیستم شما وارد می‌شود، مدل به سرعت و بلافاصله ورودی را پردازش کرده و پاسخ را تولید می‌کند.



- پیش‌بینی دسته‌ای (*Batch Prediction*):

پیش‌بینی دسته‌ای به معنای پردازش تعداد زیادی از ورودی‌ها همزمان توسط مدل است. به عبارت دیگر، به جای پردازش تک تک ورودی‌ها در زمان واقعی، ورودی‌ها به صورت دسته‌ها یا دسته‌های بزرگ‌تری گروه‌بندی می‌شوند و سپس مدل بر روی هر دسته پیش‌بینی انجام می‌دهد. این الگو برای پردازش دسته‌های بزرگ از داده‌ها و پیش‌بینی بر روی مجموعه‌های بزرگ‌تر از ورودی‌ها مورد استفاده قرار می‌گیرد.



اتوماسیون

(توضیح کلی)

در این بخش از گزارش، می‌خواهم توضیح دقیق‌تری ارائه دهم درباره خط‌لوله استقرار خودکاری که برای مدل یادگیری ماشینی ایجاد کرده‌ام. این خط‌لوله شامل مراحل کانتینرسازی، CI/CD و اتوماسیون جریان کار یادگیری ماشینی است. همچنین، از ابزارها و فناوری‌هایی که در اتوماسیون فرآیند استقرار مدل استفاده کرده‌ام، نیز بحث خواهم کرد.

Containerization (کانتینرسازی):
برای ایجاد بستری یکپارچه و قابلیت انتقال به‌راحتی برای مدل یادگیری ماشینی، از تکنولوژی کانتینرسازی Docker استفاده کردیم. با ایجاد یک Dockerfile مرتبط با مدل و تنظیمات آن، محیط اجرایی مدل به صورت کامل و ایزوله را ایجاد کردیم. این کانتینر شامل مدل، وابستگی‌های لازم، کتابخانه‌ها و تنظیمات مورد نیاز برای اجرای صحیح مدل بوده است.

Continuous Integration/Continuous Deployment (*CI/CD*) (یکپارچه‌سازی پیوسته/استقرار پیوسته):
برای اتوماسیون فرآیند تست، ساخت و استقرار مدل، از رویکرد *CI/CD* بهره بردیم. این روند شامل مراحل زیر است:

- تست خودکار: ما یک سیستم تست خودکار برای مدل ایجاد کردیم که در هر تغییر در کد یا مدل، تست‌های واحد و انتگرال ما را اجرا می‌کند تا اطمینان حاصل شود که همه چیز به درستی کار می‌کند.
- ساخت خودکار تصویر *Docker*: در صورت گذراندن موفقیت‌آمیز تست‌ها، خودکار یک تصویر *Docker* جدید از مدل و تنظیمات آن ایجاد می‌شود.
- استقرار خودکار: تصویر *Docker* جدید به محیط‌های استقراری منتقل می‌شود و مدل به صورت خودکار در آن‌ها استقر می‌شود. این فرآیند از تغییر به تغییر به صورت خودکار انجام می‌شود.

اتوماسیون جریان کار یادگیری ماشینی: برای اتوماسیون جریان کار یادگیری ماشینی، از ابزارهای زیر استفاده کردیم:

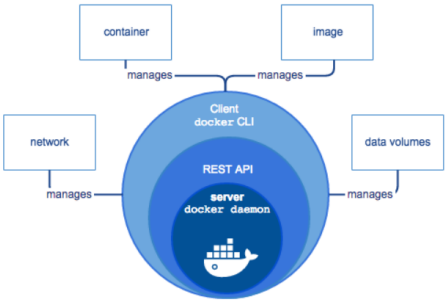
- *Jupyter Notebooks*: برای توسعه و آزمون مدل‌های یادگیری ماشینی، از دفترچه‌های *Jupyter* استفاده کردیم. این ابزار به ما امکان می‌دهد کد را در محیط تعاملی اجرا کرده و نتایج را به صورت تصویری مشاهده کنیم.
- *Git*: برای مدیریت نسخه‌های مختلف کد و مدل‌های یادگیری ماشینی، از سیستم کنترل نسخه *Git* استفاده کردیم. این ابزار به ما امکان می‌دهد تغییرات در کد را رصد کنیم و به‌طور همزمان با تیم به پروژه کار کنیم.
- *GitHub Actions*: برای تنظیم و اجرای خودکار فرآیند *CI/CD*، از سرویس *GitHub Actions* استفاده کردیم. این سرویس به ما اجازه می‌دهد تا از رویدادهای مختلف در مخزن *GitHub* بهره ببریم و تنظیمات مختلفی برای تست، ساخت و استقرار تعیین کنیم.

با استفاده از این ابزارها و روش‌ها، ما فرآیند کانتینرسازی، تست خودکار، ساخت خودکار تصویر Docker و استقرار خودکار مدل را ایجاد کرده‌ایم که باعث افزایش بهره‌وری، کاهش خطاهای انسانی و تسهیل در توسعه و استقرار مدل یادگیری ماشینی شده است.

(Docker)

استفاده از اتوماسیون و کانتینرها با $Docker$ در فرآیند استقرار مدل یادگیری ماشینی عملکرد موثری را به همراه دارد. $Docker$ یک پلتفرم معتبر برای کانتینرسازی است که به توسعه‌دهندگان امکان می‌دهد برنامه‌ها و توابع خود را به شکل کاملاً مستقل و در محیط‌های ایزوله اجرا کنند.
با استفاده از $Docker$، شما می‌توانید مدل یادگیری ماشینی، کتابخانه‌ها و وابستگی‌های مورد نیاز را به یک کانتینر ایزوله تبدیل کنید. این به شما امکان می‌دهد محیط اجرایی یکپارچه‌تری برای مدل خود داشته باشید و از مشکلات مرتبط با تفاوت‌های محیطی جلوگیری کنید.
فرآیند اتوماسیون با $Docker$ می‌تواند شامل موارد زیر باشد:

- تعریف $Docker file$: ابتدا باید یک فایل $Docker file$ ایجاد کنید که توصیف‌کننده‌ای از محیط مورد نیاز برای اجرای مدل شما باشد. این فایل شامل دستوراتی است که نشان می‌دهد چگونه محیط Docker باید تنظیم شود، از جمله تعیین پایگاه‌های تصویری اولیه و اضافی، نصب وابستگی‌ها و تنظیمات محیطی.
- ایجاد تصویر$Docker\ image$: با اجرای دستورات موجود در $Docker file$، تصویر $Docker$ مورد نیاز شما ایجاد می‌شود. این تصویر حاوی مدل یادگیری ماشینی، کتابخانه‌ها و تنظیمات مورد نیاز است.
- تست محلی: قبل از استقرار نهایی، می‌توانید با استفاده از تصویر ایجاد شده، مدل خود را به صورت محلی تست کنید و اطمینان حاصل کنید که همه چیز به درستی کار می‌کند.
- استقرار خودکار: می‌توانید سیستم CI/CD خود را تنظیم کنید تا با تغییرات در کد، تصویر $Docker$ را بسازد، آزمون‌ها را اجرا کند و در نهایت مدل را به محیط‌های استقراری منتقل کند.
- مدیریت محیط: با استفاده از $Docker$، محیط‌های مختلفی می‌توانید برای تست‌ها، استقرار و توسعه



ایجاد کنید و به سادگی بین آن‌ها جابجا شوید. با این رویکرد، توانستیم فرآیند استقرار مدل یادگیری ماشینی خود را اتوماتیک کنیم و از مزایای استفاده از کانتینرها با $Docker$ بهره‌برداری کنیم. البته استفاده از هموروش خیلی کمک کرد.

نظارت و مانیتورینگ

(پاسخ)

تنظیم نظارت بر برنامه: برای تضمین کیفیت، عملکرد و قابلیت اعتماد برنامه یادگیری ماشینی استقرار یافته، نظارت مداوم بر معیارهای مختلف ضروری است. به منظور انجام این کار، ما یک سیستم جامع نظارت راه‌اندازی کردیم.
معیارهای نظارتی:
معیارهایی که ما نظارت می‌کنیم، شامل امور زیر است:

- زمان پاسخ ($Response\ Time$): زمانی که مدل برای پاسخ به درخواست‌ها نیاز دارد. این معیار نشان‌دهنده عملکرد سریع واکنش به درخواست‌های کاربران است.
- نرخ خطا ($Error\ Rate$): تعداد درخواست‌هایی که توسط مدل با خطا پاسخ داده می‌شوند. نرخ خطا می‌تواند نشان‌دهنده کیفیت عملکرد مدل باشد.
- استفاده از منابع ($Resource\ Utilization$): میزان منابع سیستم مانند CPU، حافظه و پهنای باند شبکه که توسط مدل استفاده می‌شود. این معیار نشان‌دهنده بهینه‌سازی و کارایی مصرف منابع است.

حلقه‌های بازخوردی:
برای اطمینان از کارآیی مداوم برنامه، حلقه‌های بازخوردی اجرایی‌سازی کردیم. این حلقه‌ها به صورت اتوماتیک عملکرد برنامه را نظارت می‌کنند و در صورت انحراف از معیارهای تعیین شده، واکنش مناسب انجام می‌دهند:

- هشدارها و اعلان‌ها ($Alerts\ and\ Notifications$): هنگامی که مقادیر معیارهای نظارتی از حد تعیین شده عبور می‌کنند، سیستم هشدارها و اعلان‌ها را فعال می‌کند. این اعلان‌ها به افراد مسئول یا تیم فنی ارسال می‌شوند تا به سرعت واکنش نشان دهند.
- خودترمیمی ($Self-Healing$): برای جلوگیری از از کارافتادگی و بهبود روند خودترمیمی، ما اقدام به پیاده‌سازی خودترمیمی در محیط‌های استقراری کردیم. اگر نظارت نشان دهد که معیارها از حد تعیین شده فراتر رفته‌اند، سیستم به‌طور اتوماتیک تلاش می‌کند مشکل را حل کرده و مدل را به وضعیت عادی بازگرداند.

چالش‌ها و تبادلات

(چالش‌ها و تبادلات)

در مرحله استقرار پروژه، با چالش‌ها و تبادلاتی روبه‌رو شدیم که تأثیر مستقیمی بر فرآیند مدیریت و استقرار مدل‌های یادگیری ماشینی داشت. همچنین، در مواردی با نقاط کور و بن‌بست‌ها نیز مواجه شدیم که با تلاش و همکاری تیم، به راه‌حل‌هایی برای آن‌ها دست پیدا کردیم. در ادامه به بیان دقیق‌تر چند چالش و تبادل و همچنین نحوه مقابله با آن‌ها می‌پردازم.

چالش: مدیریت و نظارت بر میزان منابع $Docker$ تبادلات: یکی دیگر از چالش‌هایی که با آن مواجه شدیم، مدیریت بهینه منابع مورد استفاده توسط $Docker\ container$ بود. تنظیمات منابع برای کانتینرها باید به گونه‌ای انجام می‌شد که بهره‌وری بالا و همزمان اجتناب از بیش‌مصرفی منابع حاصل شود.

راه‌حل: برای حل این چالش، از راه‌حل‌های زیر استفاده کردیم:

تنظیمات منابع دقیق: با تست‌های متعدد، تنظیمات منابع مورد نیاز برای هر کانتینر را تعیین کردیم تا بهره‌وری و پایداری در هنگام اجرا حفظ شود.

مانیتورینگ پیشرفته: از ابزارهای مانیتورینگ پیشرفته برای نظارت بر مصرف منابع و عملکرد کانتینرها استفاده کردیم. این ابزارها به ما امکان می‌دادند در زمان واقعی مشاهده کنیم که کدام کانتینرها به منابع زیادی نیاز دارند.

بن‌بست: تعاملات پیچیده بین مدیریت مدل‌ها و تصاویر $Docker$ با $mlflow$ چالش: در یک مرحله، تصاویر $Docker$ و $mlflow$ با یکدیگر هماهنگ نبودند و تعدادی از نسخه‌های مدل‌ها در mlflow از طریق Docker image قابل دسترسی نبود.

راه‌حل: با تحقیق و مشاوره با تیم‌های متخصص، مشکل ناهماهنگی را پیدا کردیم و اصلاحات لازم را انجام دادیم تا تصاویر Docker و mlflow همخوانی داشته باشند. همچنین، نقطه بن‌بست را به تیم توسعه گزارش دادیم تا مشکل را بهبود دهند.