MySQL – zarządzania relacyjnymi bazami danych, magazynowanie danych, integracja z różnymi językami

PostgreSQL – rozszerza SQL, obsługa JSON, XML, analiza danych, obsługa złożonych zapytań

Microsoft SQL Server – magazynowanie danych, analiza, raportowanie i integracja danych, integracja z usługami Microsoft

MariaDB - Lepsza wydajność, bezpieczeństwo i skalowalność niż MySQL,

Apache Cassandra - Rozproszona baza danych NoSQL, duża ilość danych, skalowalność, duża tolerancja błędów.

MongoDB - analizy w czasie rzeczywistym, elastyczna, skalowalna,

Amazon Aurora – magazyny danych, zgodna z MySQL i PostgreSQL, zaprojektowana dla chumury

Google BigQuery - zapytania SQL, magazyn danych, integralność z usługami Google Cloud

Apache HBase - Analiza w czasie rzeczywistym, wysoka przepustowość zapisu, skalowalność i integracja z Hadoop.

IBM Db2 – zarządzanie relacyjną bazą danych, chmura, magazynowanie danych

Apache Hive – zarządzanie dużymi zestawami danych, może się łączyć z Hadoop (HDFS)

Google BigQuery - magazyn danych, integruje się z usługami Google AI

Snowflake – magazynowanie danych w chmurze

**Monitoring**

Logowanie, przetwarzanie, wizualizacja wyników – zapis w postaci logów i analiza w czasie rzeczywistym za pomocą Elasticsearch i Kiban

Metryki modelu – monitoring za pomocą Prometheus – dane mogą być agregowane i wysyłane do kolejki w formacie json (monitoring: dokładność, precyzja, czułość )

Monitorowanie wydajności różnych wersji modelu AI - Optimizely, Google Analytics za ich pomocą można wysyłać wyniki testów w formacie JSON do kolejki

System alertów – anomalie w wynikach, Grafana z wtyczką Alertmanager, może być przekazywane do kolejki w formacie json.

Model AI jest zintegrowany z zewnętrznymi usługami (np. API), dane zwrotne z tych usług również mogą być zbierane i wysyłane do kolejki.

Po przetworzeniu danych przez model AI, wyniki oraz metryki wydajności mogą być przesyłane do wybranego systemu monitorowania. Dane te mogą być agregowane, przetwarzane, a następnie w formacie JSON wysyłane do kolejki. RabbitMQ - może przesyłać te dane do dalszej analizy lub przechowywania.

Monitorowanie z istniejącym AI runtime, aby móc zbierać dane zwrotne i metryki.

Dane formatowane jako JSON i przesyłane do kolejki, z której będą mogły być dalej przetwarzane lub analizowane.

To podejście zapewni ciągłe monitorowanie i ocenę wydajności modelu AI, umożliwiając reagowanie na wszelkie nieprawidłowości i optymalizację modelu w czasie rzeczywistym.

**Inne**

Apache Hadoop - zestaw narzędzi do przetwarzania dużych zbiorów danych w rozproszonym środowisku. Składa się z Hadoop Distributed File System (HDFS) do przechowywania danych oraz MapReduce do przetwarzania danych.HDFS umożliwia przechowywanie ogromnych ilości danych w rozproszony sposób, a MapReduce pozwala na przetwarzanie tych danych lokalnie. Hadoop można integrować z innymi narzędziami, takimi jak Apache Hive, aby wykonywać zapytania SQL na danych. Idealne do przetwarzania dużych, złożonych zbiorów danych lokalnie.

Hive jest narzędziem, które umożliwia wykonywanie zapytań SQL na danych przechowywanych w HDFS. Jest to narzędzie stworzone na bazie Hadoop i jest często używane do analizy dużych zbiorów danych. Idealne do analizowania dużych zbiorów danych przy użyciu SQL.

Spark - narzędzie do szybkiego przetwarzania danych, które może działać na Hadoop HDFS, ale także na własnym lokalnym systemie plików. oferuje wsparcie dla operacji na dużych zbiorach danych, w tym analizy w pamięci, strumieniowania, oraz machine learning. Szybkie przetwarzanie i analiza dużych zbiorów danych, trenowanie modeli AI na danych tabelarycznych.

Elasticsearch - rozproszony silnik wyszukiwania i analizy, używany do indeksowania i wyszukiwania dużych zbiorów danych. Elasticsearch przechowuje dane w formie dokumentów JSON, ale może być używany do zarządzania danymi tabelarycznymi poprzez odpowiednie mapowania. Wyszukiwanie, analiza danych, monitorowanie danych w czasie rzeczywistym.

Druid - rozproszona baza danych zaprojektowana do bardzo szybkiego przetwarzania zapytań analitycznych na dużych ilościach danych. Druid łączy cechy baz danych kolumnowych oraz analityki w pamięci. pozwala na zarządzanie danymi w postaci tabel i oferuje bardzo szybkie odpowiedzi na zapytania analityczne. Analiza dużych ilości danych w czasie rzeczywistym, BI (business intelligence).

ClickHouse to rozproszona baza danych kolumnowych zaprojektowana do analizy dużych ilości danych w czasie rzeczywistym. Jest optymalizowana pod kątem zapytań analitycznych. umożliwia przechowywanie dużych zbiorów danych tabelarycznych, co pozwala na bardzo szybkie przetwarzanie zapytań analitycznych. Analiza danych w czasie rzeczywistym, BI, big data.

Presto - rozproszony silnik zapytań SQL typu open source, zapytania na dużych zestawach danych.

może wykonywać zapytania na danych w miejscu ich przechowywania, w HDFS, S3, Cassandra, relacyjnych bazach danych, bez przenoszenia danych.

Apache Flink - możliwość obsługi dużych ilości danych w czasie rzeczywistym, co może być niezbędne do szkolenia i wnioskowania modeli AI.

analiza w czasie rzeczywistym i uczenia maszynowego na danych przesyłanych strumieniowo, lokalne zarządzanie danymi w aplikacjach AI.

Monitorinnng - Aby zaprojektować architekturę monitoringu dla systemu AI, który zasilany jest danymi z Oracle, przetwarza dane za pomocą kolejki RabbitMQ, a informacje są wysyłane do modelu AI w formie plików JSON.

Metryki do monitorowania - reklamacje rzeczywiście przepływają przez syste? od bazy, przez RabbitMQ, do modelu A?

- Liczba reklamacji, które wpłynęły do systemu w określonym czasie (z Oracle).

- Liczba reklamacji, które przeszły przez RabbitMQ (czy wiadomości nie zostały utracone).

- Liczba reklamacji przetworzonych przez model (czy każda reklamacja została przetworzona).

Alerting - Prometheus Alertmanager - Skonfigurowanie alertów w przypadku nietypowych zdarzeń, takich jak brak napływających danych. Zabbix - Monitorowanie stanu serwerów i aplikacji oraz konfiguracja powiadomień.

**Architektura monitoringu**

* Monitoring Oracle (z OEM lub custom SQL scripts)
* RabbitMQ Monitoring (RabbitMQ Management Plugin, Prometheus/Grafana)
* AI Runtime Monitoring (Log Analysis, ELK, Prometheus)
* JSON Integrity Checks (Validation Scripts, Monitoring)
* Model Monitoring (Accuracy, Time to Response)
* System Alerts (Alertmanager, Zabbix).

**6. Monitoring i ewaluacja modeli AI**

* **Prometheus + Grafana:** Prometheus może zbierać metryki z różnych systemów (w tym RabbitMQ i modeli AI), a Grafana służy do wizualizacji danych, co pozwala monitorować wydajność modeli w czasie rzeczywistym.
* **MLflow:** Narzędzie open-source do zarządzania procesem trenowania modeli, monitorowania metryk i zarządzania eksperymentami. Oferuje również rejestrowanie modeli i zarządzanie wersjami.
* **TensorBoard:** Narzędzie do monitorowania i wizualizacji trenowania modeli w TensorFlow. Pozwala na śledzenie metryk, takich jak dokładność, straty, i inne wskaźniki wydajności.
* **Neptune.ai:** Platforma do zarządzania eksperymentami w uczeniu maszynowym i AI. Pozwala na śledzenie i porównywanie różnych treningów modeli oraz wyników.
* **Weights & Biases:** Popularne narzędzie do zarządzania eksperymentami AI. Pozwala na śledzenie procesów trenowania modeli, zarządzanie hiperparametrami oraz monitorowanie wyników.

**Gdzie zapisywać wyniki z modelu?**

**Tworzenie nowej bazy danych na wyniki**

* Baza danych zapewnia trwałe przechowywanie wyników, co jest szczególnie przydatne, jeśli wyniki muszą być analizowane lub przetwarzane w przyszłości.
* Przy dużej ilości danych lepszym rozwiązaniem będzie przechowywanie wyników w dedykowanej bazie danych, ponieważ logi na dysku mogą szybko stać się nieefektywne. W zależności od skali, możesz zdecydować się na relacyjną bazę danych (RDBMS) lub rozwiązanie NoSQL.
* Użycie bazy danych umożliwia łatwe tworzenie zapytań SQL lub innych typów zapytań, aby analizować wyniki, monitorować trendy, sprawdzać poprawność oraz wyciągać wnioski na temat jakości modelu.
* Każdy wiersz może reprezentować jeden wynik, a kolumny mogą zawierać kluczowe dane, takie jak:

- Identyfikator przetworzonej reklamacji.

- Dane wejściowe.

- Wynik modelu.

- Czas przetwarzania.

- Ewentualne metryki modelu (np. pewność predykcji).

- Status reklamacji (czy została zaakceptowana, odrzucona, itp.).

* Możliwość łatwego wyszukiwania, filtrowania i analizy wyników.
* Trwałe przechowywanie wyników z dostępem do historycznych danych.
* Skalowalność – bazy danych lepiej radzą sobie z dużymi ilościami danych niż system plików.

**Logi zapisywane na dysku**

* Szybkie i łatwe w implementacji rozwiązanie.
* Dobrze nadaje się do systemów, które nie muszą przechowywać wyników w dłuższym okresie.
* Możliwość późniejszej agregacji i przetwarzania logów przy użyciu narzędzi takich jak ELK Stack (Elasticsearch, Logstash, Kibana) do analizy danych.
* Skalowalność może być problemem, jeśli dane zaczną się szybko gromadzić.
* Trudniejszy dostęp do historycznych danych i analiza w porównaniu do bazy danych.
* Potencjalne problemy z zarządzaniem dużymi plikami.

**Przechowywanie wyników w systemach Message Queue RabbitMQ**

* Jeśli wyniki muszą być przetwarzane asynchronicznie lub przez różne systemy, przechowywanie wyników w systemie kolejki, takim jak RabbitMQ czy Kafka, pozwala na ich dalszą dystrybucję.
* Przetwarzanie w czasie rzeczywistym - Kolejki wiadomości pozwalają na rozproszone przetwarzanie wyników i mogą być używane do przesyłania wyników do innych systemów w celu dalszej analizy.
* Można wysyłać wyniki z modelu do nowych kolejek, które będą odbierane przez inne komponenty systemu (np. do analizy lub dalszego przetwarzania).
* Umożliwia asynchroniczne przetwarzanie wyników przez wiele systemów.
* Dobre rozwiązanie dla systemów czasu rzeczywistego.
* Centralizacja logów i analiza (np. ELK Stack)
* Centralizacja logów z wynikami modelu AI w ELK Stack (Elasticsearch, Logstash, Kibana). To rozwiązanie umożliwia analizę wyników, tworzenie raportów i monitorowanie wyników modelu w czasie rzeczywistym.
* Logstash - agreguje logi z różnych systemów, w tym z wyników modelu AI.
* Elasticsearch - indeksuje logi i umożliwia szybkie wyszukiwanie wyników.
* Kibana - umożliwia wizualizację wyników oraz tworzenie raportów i dashboardów.
* monitorowanie w czasie rzeczywistym, łatwe przeszukiwanie i analiza wyników
* Jeśli wyniki są generowane w dużych ilościach i muszą być przechowywane przez dłuższy czas -baza danych najlepszym rozwiązaniem.
* skomplikowane analizy wyników - użycie bazy danych lub narzędzi do analizy logów (np. ELK Stack).
* Jeśli wyniki muszą być przetwarzane przez inne systemy (np. BI, analityka) – kolejki wiadomości lub bazy danych.
* Dostęp do historycznych danych - Baza danych pozwala łatwiej przeszukiwać i analizować dane historyczne niż logi na dysku (trudne do zarządzania, gdy rosną w objętości).

Kiedy potrzebna baza danych:

- Trwałe przechowywanie wyników - RabbitMQ działa jako system kolejek wiadomości i nie jest przeznaczony do długoterminowego przechowywania danych. Jeśli potrzebujesz przechowywać wyniki modelu AI na stałe, np. do audytu, analizy lub raportowania, warto rozważyć zapisanie tych wyników do bazy danych (np. PostgreSQL, MongoDB).

Analiza wyników - baza danych umożliwi szybkie wyszukiwanie, filtrowanie i przetwarzanie tych danych historycznych.

Baza danych może służyć jako bezpieczny magazyn dla wyników na wypadek problemów z RabbitMQ (np. awarie, utrata danych).

Kiedy nie potrzebna baza danych

Krótkoterminowe przetwarzanie - Jeśli wyniki są przetwarzane niemal natychmiast i nie potrzeba przechowywać ich przez dłuższy czas

Przetwarzanie danych w czasie rzeczywistym, nie wymaga długoterminowego przechowywania

implementacja dysków:

przechowywanie danych przez pewien czas (monitoring, analiza),

źle skalowalne przy dużych zbiorach danych

krótkoterminowe zapisywanie wyników – zapis w plikach json lub csv i analiza w partiach

przeszukiwanie i filtrowanie na poziomie plików może być uciążliwe

JSON - najpopularniejszy format do przesyłania wyników, RabbitMQ dobrze obsługuje dane tekstowe, a JSON jest łatwy do odczytania i parsowania przez różne komponenty. Możesz zapisywać wyniki modelu AI w formacie JSON (zawierającym kluczowe dane, takie jak identyfikator, wynik predykcji, czas przetwarzania itp.) i wysyłać je do RabbitMQ jako wiadomości.

Monitoring i alerty - monitorowanie RabbitMQ - RabbitMQ Management Plugin lub Prometheus + Grafana

Logi na dyskach - Mogą być stosowane jako dodatkowe rozwiązanie do krótkoterminowego przechowywania wyników, ale baza danych będzie bardziej efektywna w przypadku dużych ilości danych i konieczności ich przechowywania, analizy, raportowania.

Użycie formatu JSON i kolejek RabbitMQ, z monitorowaniem i alertami, aby zapewnić prawidłowy przepływ wyników modelu AI

+---------------------------------------+

| Oracle Database (dane) |

+---------------------------------------+

|

v

+---------------------------------------+

| Kolejka RabbitMQ | <-- Monitoring wiadomości w kolejce

+---------------------------------------+

|

v

+---------------------------------------+

| AI Runtime/Model | <-- Monitorowanie wyników modelu

+---------------------------------------+

|

v

+---------------------------------------+

| Kolejka RabbitMQ (wyniki) | <-- Monitoring wyników modelu

+---------------------------------------+

|

v

+---------------------------------------+

| Konsumenci wyników | <-- Ewaluacja wyników

+---------------------------------------+

|

v

+---------------------------------------+

| Baza danych na wyniki (opcjonalna) | <-- Przechowywanie wyników do analizy

+---------------------------------------+

**Narzędzia do monitorowania modelu AI:**

TensorBoard może służyć do monitorowania metryk treningu i predykcji modelu. Można również używać do monitorowania wydajności modelu w środowisku produkcyjnym.

Prometheus + Grafana - Wydajność modelu AI (czas odpowiedzi, liczba przetworzonych wiadomości, błędy) może być monitorowana w czasie rzeczywistym za pomocą metryk zbieranych przez Prometheus i wizualizowanych w Grafana.

MLflow / Neptune.ai - Te narzędzia pozwalają monitorować metryki trenowania i wdrażania modeli AI, a także śledzić ewaluację i automatyzację eksperymentów.

**Przykład metryk dla modelu AI:**

* Czas odpowiedzi modelu
* Dokładność
* Ilość przetworzonych wiadomości

**Monitoring wyników z RabbitMQ (po stronie konsumenta)**

Wyniki generowane przez model AI są wysyłane z powrotem do RabbitMQ, gdzie są odbierane przez konsumentów (np. do bazy danych lub innych systemów). Monitoring tego etapu jest istotny, aby upewnić się, że wyniki są prawidłowo przetwarzane i dostarczane.

Każdy wynik modelu AI powinien być walidowany, aby upewnić się, że został poprawnie wygenerowany (np. walidacja struktury JSON, sprawdzanie kluczowych pól).

Liczba przetworzonych wyników - RabbitMQ powinni być monitorowani, aby sprawdzić, ile wyników zostało prawidłowo przetworzonych, ile wiadomości nie zostało przetworzonych z powodu błędów, Mierzenie czasu, jaki zajmuje konsumentom przetwarzanie wyników modelu.

Narzędzia do monitorowania wyników:

* Elasticsearch + Kibana - wysyłanie logów wyników do Elasticsearch, aby łatwo analizować i przeszukiwać je za pomocą Kibana. To pozwala na łatwe monitorowanie trendów i błędów w przetwarzaniu wyników.
* Prometheus + Grafana - Zbieranie i wizualizacja metryk wyników w czasie rzeczywistym, np. liczba przetworzonych wiadomości, liczba błędnych wyników.
* Logi RabbitMQ - Sprawdzanie logów RabbitMQ w celu monitorowania stanu wiadomości i sprawdzania, czy wyniki są poprawnie dostarczane.

Ewaluacja modelu AI obejmuje ocenę jego skuteczności na podstawie różnych metryk, które są monitorowane w środowisku produkcyjnym. Ewaluacja pozwala na ciągłą optymalizację modelu i zapobieganie degradacji jego wydajności.

Jakie metryki ewaluować?

* Dokładność predykcji - Dokładność modelu powinna być regularnie oceniana na rzeczywistych danych produkcyjnych.
* Drift danych - Monitorowanie zmiany rozkładu danych wejściowych w czasie

**MLflow vs Neptune.ai**

MLflow - Jest bardziej kompleksowym narzędziem do monitorowania i zarządzania eksperymentami, a także wdrażania modeli AI w różnych środowiskach. Jest open-source i łatwy do rozszerzenia, ale wymaga dodatkowej konfiguracji (np. backendu do metryk)

Neptune.ai - Jest bardziej zaawansowanym narzędziem do zarządzania eksperymentami, z naciskiem na wizualizację wyników i zarządzanie modelami. Jego siłą jest łatwość użycia i gotowe rozwiązania do monitorowania modeli w czasie rzeczywistym.

**Rodzaje monitorowania danych:**

* Monitorowanie jakości danych (Data Quality Monitoring)
* Brakujące dane (Missing Data)- Sprawdzanie, czy dane wejściowe nie mają brakujących wartości, które mogą wpłynąć na wydajność modelu.
* Błędy w danych (Data Errors) - Monitorowanie wystąpień nieprawidłowych wartości (np. ujemne wartości dla zmiennych, które powinny być dodatnie).
* Outliery - Wykrywanie wartości odstających, które mogą zniekształcać wyniki modelu.
* Monitorowanie spójności danych (Data Consistency Monitoring)
* Sprawdzanie, czy dane wejściowe zachowują spójność z danymi treningowymi. Należy monitorować struktury danych, ich typy i zakresy wartości.
* Monitorowanie rozkładu danych (Data Distribution Monitoring)
* Śledzenie rozkładu cech danych w czasie, aby upewnić się, że dane wejściowe nie zmieniają się w sposób, który wpłynie na model. To kluczowe dla wykrywania dryfu danych.
* Monitorowanie danych strumieniowych (Streaming Data Monitoring)
* Jeśli dane napływają do systemu w czasie rzeczywistym (np. przez RabbitMQ), należy monitorować ich przepływ, aby sprawdzić, czy model otrzymuje odpowiednią ilość danych, a także wykrywać opóźnienia i brakujące dane.
* Walidacja danych (Data Validation)
* Sprawdzanie, czy dane są zgodne ze schematami, które model wymaga. Może to obejmować walidację typów danych, ich zakresów wartości oraz struktury.

**Narzędzia do monitorowania dancyh:**

* Great Expectations - Narzędzie do walidacji i monitorowania jakości danych w systemach produkcyjnych.
* Deequ (AWS) - Narzędzie do monitorowania jakości danych i wykrywania problemów w dużych zbiorach danych.
* DataRobot MLOps - Narzędzie do monitorowania i zarządzania jakością danych oraz monitorowania modeli.

Śledzenie jakości wyników predykcji na podstawie danych rzeczywistych. Jeśli jest możliwość porównania wyników modelu z rzeczywistymi wynikami (np. w systemach, gdzie prawdziwe wyniki są znane po czasie), można monitorować, jak dokładnie model przewiduje wartości.

Niektóre modele AI mogą przewidywać wyniki z różnymi poziomami pewności. Monitorowanie niepewności wyników pomaga ocenić, kiedy model przewiduje wyniki z wysoką niepewnością i może wymagać dodatkowej analizy.

Detekcja anomalii wyników

Monitorowanie nietypowych wyników, które mogą wskazywać na problemy z modelem lub danymi. Na przykład, jeśli model AI nagle zaczyna przewidywać wyniki zbyt jednostronne (np. wszystkie pozytywne klasy), może to wskazywać na problem.

Monitorowanie decyzji biznesowych opartych na wynikach modelu

wyniki modelu są wykorzystywane do podejmowania decyzji biznesowych, monitorowanie wpływu tych decyzji (np. sprzedaż, zadowolenie klientów, wskaźniki konwersji) pomaga ocenić skuteczność modeli AI w rzeczywistym środowisku.

Monitorowanie modeli AI - obejmuje śledzenie metryk wydajności, opóźnień, dryfu modelu oraz zasobów systemowych. Jest to kluczowe dla zapewnienia, że model działa wydajnie i nie występują problemy po wdrożeniu.

Monitorowanie danych - koncentruje się na jakości, spójności i rozkładzie danych wejściowych, co jest kluczowe dla zapewnienia, że model przetwarza odpowiednie dane.

Monitorowanie wyników - pozwala na ocenę, czy model dostarcza poprawne i trafne predykcje, a także jak wyniki te wpływają na decyzje biznesowe lub inne procesy.

**MLflow nie posiada wbudowanej funkcjonalności alertów** w sposób podobny do narzędzi takich jak **Prometheus** czy **Grafana**. Jednak możesz ustawić alerty na podstawie wyników monitorowanych przez MLflow, korzystając z integracji z zewnętrznymi systemami, które mają możliwości alertowania.

**Grafana vs tableau**

Tableau – Zalety

* Zaawansowane raportowanie
* Interaktywne dashboardy
* Bogata integracja

Tableau – Wady

* Brak natywnych alertów - nie oferuje wbudowanej obsługi alertów w czasie rzeczywistym

Grafana – Zalety

* Real-time monitoring
* Natywne alerty

Grafana – Wady

* Ograniczone raportowanie - Grafana nie jest zaprojektowana jako narzędzie do raportowania biznesowego

Tableau - może być z powodzeniem użyte do monitorowania wyników modelu AI, zwłaszcza w zakresie tworzenia interaktywnych raportów i analizowania danych historycznych. Jednak

Grafana - lepiej sprawdzi się do monitorowania w czasie rzeczywistym i ustawiania alertów.

Narzędzia podobne do Tableau:

Power BI - Idealny do interaktywnych dashboardów i automatycznych raportów z dobrym stosunkiem jakości do ceny.

Qlik Sense - Zaawansowane narzędzie analityczne z elastyczną integracją i możliwością tworzenia zaawansowanych raportów.

Looker - Narzędzie Google Cloud o wysokiej elastyczności w integracji z danymi chmurowymi.

Sisense - Skupione na analizie dużych zbiorów danych i automatyzacji raportów.

Zoho Analytics - Bardziej dostępne cenowo narzędzie z dobrymi funkcjami do raportowania i wizualizacji.

Domo – Narzędzie oferujące generowanie raportów w czasie rzeczywistym z silną integracją z bazami danych.

Metabase – Open-source'owe narzędzie do prostych wizualizacji i raportów.

Redash – Open-source'owe narzędzie skoncentrowane na prostocie i szybkim tworzeniu raportów z baz danych.

Grafana może być z powodzeniem stosowana do ciągłego monitorowania wyników modeli AI oraz generowania raportów na podstawie danych historycznych.

Grafana - może być zintegrowana z bazą danych Hive (często używaną w ekosystemach Hadoop do zarządzania dużymi zbiorami danych) oraz Oracle do monitorowania wyników modelu AI i generowania długoterminowych raportów.

Grafana + Presto/ Trino - Grafana może łączyć się z Presto, aby pobierać dane z Hive i wizualizować je na dashboardach.

Oracle to relacyjna baza danych o szerokim zastosowaniu w biznesie, a Grafana wspiera bezpośrednią integrację z bazami Oracle.

Grafana posiada wtyczkę do połączeń z bazami danych Oracle. Można skonfigurować Grafanę do bezpośredniego połączenia z bazą Oracle, a następnie tworzyć zapytania SQL do pobierania i wizualizacji danych historycznych.

Aby monitorować wyniki modelu AI w perspektywie długoterminowej, Grafana może łączyć się z bazą danych (Hive lub Oracle) i wykonywać zapytania, które pobierają dane historyczne.

Tworzenie dashboardów - które będą wyświetlały dane z długiego okresu (np. półroczne, roczne).

Generowanie raporty - z danych zapisanych w bazie, które można eksportować i dystrybuować jako raporty kwartalne, półroczne czy roczne.

Ustawiać alerty - na podstawie danych historycznych – np. jeśli wskaźniki jakości modelu spadają poniżej określonego progu w dłuższym okresie.

Dashboard z danymi historycznymi - wyniki modelu AI za ostatni rok (np. dokładność, precyzję, liczbę TP/FP) na podstawie danych przechowywanych w Hive lub Oracle.

Automatyczne raporty - W Grafana można skonfigurować automatyczne generowanie raportów w formie PDF na podstawie danych z Hive lub Oracle, które będą wysyłane w określonym harmonogramie (np. co miesiąc, pół roku, rok).

Grafana może zostać zintegrowana zarówno z Hive (za pośrednictwem Presto/Trino), jak i z Oracle co pozwala na monitorowanie wyników modelu AI oraz generowanie długoterminowych raportów, takich jak półroczne czy roczne analizy wyników.

W przypadku Hive integracja z Presto/Trino umożliwia wykonywanie zapytań SQL, które pobierają dane z rozproszonych systemów, a Grafana może te dane wizualizować i analizować.

W przypadku Oracle Grafana bezpośrednio łączy się z bazą, co umożliwia tworzenie raportów na podstawie danych przechowywanych w długim okresie.

**Notatki ze spotkania:**

Z ifd do hive d-1 jest zrzut danych, d-1 gubi informacje d-2 odzyska informację, d-1 jest o 3 w nocy realizowane jest więc nie powinno być tej bezwładności.

Mogłoby być bez zapisu żadnego, job na bdbatch – przepisanie wartości z plato do ifd, swoje złożenie po stronie hive, odpytując. Plato to kązdy dzień ma zrzuty, jeśli biznesowo wystarczy to może być warstwa persystencji.

Jaka warstwa najmniejobciążająca – zapis do bazy – warto zmierzyć? Czy to jest znarzące że coś spowolni. Gdyby spowalniało to można na rabbicie i osobna kolejka, respond que jest , idf odczytuje z que do jednego exhange moż na 2 que.

Dodatkowa usługa? Tiket zutylizowany można ponownie? – bez sensu? W kolejce czasem możemy coś tracić? Uniknąć ryzyka stracenia – dodawanie warstwy persystenci to trzeba przerobić rabbita, na ten moment czasy są bezpieczne?

Inserty do hive 5/5 kolumn? Raczej nie jest to szybko, nie trzeba robić szybko, jak coś do hive to wszytsko całego yarna, persystencja co jakiś czas. Jak się zrestartuje, hdfs też można co chwilę i raz dziennie scalanie w bazę jsonów – do tego raczej redis, co chwila persystencja. Może lepiej baza prostgresowa – z automatu wszytsko jest – tu wszystko jest , można wrzucić na klaster i zrobić analizę

Hdfs – proces mało wydajny

Najmniejszy z bazay zrzutów na dsa, na hive i możliwe że będzie wystarczające? Plis z bazy plato i ifd? Można joiny, ta sama robota codziennie, lepiej job co będzie codziennie robił, Tablo codziennie odpytuje i dojeżdża bazę, można job odpalić i sprawdza dane.

36 tysięcy e pół roku, chcemy porównać i 36 rekordów z bazą plato robić złożenie – czy wydajnościowo? Łączyć negatywy z ifd i plato i codziennie przetworzyć jakąś grupę danych? I wszytsko na hive jest to spoko (zakładając że partycje są spoko zrobione to spoko) duże pole do optymalizacji,

Musimy pobrać listę idików i pobrać + join. Musi być szybko?

Na ten moment model danych i zobaczyć jak się kręci, jak 10 minut i barchowo to robić. Wypluwać mailem i potem tablo się podłączy. Tablo może do csv i hive można podpiąć

Grafana – jest w planach i dopiero będzie monitoring, będzie moduł do monitoringu modelu. W tej konfiguracji nie za bardzo i jako szablonowe dashboardy.

Rabbit + grafana czy są logi. Rabbit udostępnia prometheusa i tam sięga grafana i tam prometheus przechowuje historię metryk. Logowanie pracy debugowania to też zrzucamy do prometcheusa? To tutaj trzeba dostarczyć nfsa żeby nie uciekły po jakimś czasie. Nfs ma być zamówiony. Prometheus jest obok i nie zapewnia

**Data Quality in:**

**Analiza jakości danych (data quality) oraz wykrywanie dryfu danych (data drift) są kluczowymi elementami, które mają wpływ na poprawność działania modeli AI. W twoim przypadku, gdzie znasz nazwy kolumn na wejściu modelu, musisz zaplanować system walidacji, który będzie wykrywał, kiedy dane wejściowe różnią się od oczekiwanych wzorców.**

**Poniżej krok po kroku opiszę, jak podejść do tego zadania:**

**### 1. \*\*Określenie oczekiwanych wartości dla każdej kolumny\*\***

**Dla każdej kolumny musisz zdefiniować zakresy akceptowalnych wartości, typ danych oraz ewentualne wzorce lub zależności pomiędzy kolumnami. W szczególności będziesz definiować:**

**- \*\*Typ danych\*\*: czy to są liczby całkowite, zmiennoprzecinkowe, ciągi znaków itp.**

**- \*\*Zakres wartości\*\*: jakie wartości są akceptowalne (np. wartości numeryczne od 0 do 100).**

**- \*\*Kategoria/wartości nominalne\*\*: jeżeli masz kolumny z wartościami kategorycznymi, musisz określić zestaw możliwych kategorii (np. „Mężczyzna” / „Kobieta”).**

**- \*\*Braki danych (Missing values)\*\*: zdefiniuj, jakie kolumny mogą mieć braki danych oraz w jaki sposób je obsługiwać (np. imputacja, uzupełnianie wartościami domyślnymi, odrzucanie wierszy).**

**\*\*Przykład dla każdej kolumny\*\*:**

**1. Kolumna: \*\*wiek\*\* (age)**

**- Typ danych: `integer`**

**- Zakres: od 0 do 120**

**- Akceptowalne braki danych: \*\*Nie\*\***

**2. Kolumna: \*\*płeć\*\* (gender)**

**- Typ danych: `string`**

**- Zakres: `["Mężczyzna", "Kobieta"]`**

**- Akceptowalne braki danych: \*\*Tak\*\*, wypełnienie wartością domyślną „Nieznany”**

**3. Kolumna: \*\*dochód\*\* (income)**

**- Typ danych: `float`**

**- Zakres: od 0 do 1,000,000 (w przypadku, gdyby ktoś podał błędną wartość np. 10 milionów, wysyłasz alert)**

**- Akceptowalne braki danych: \*\*Tak\*\*, imputacja medianą**

**### 2. \*\*Definiowanie kryteriów jakości danych\*\***

**Po określeniu, jakie wartości są akceptowalne dla każdej kolumny, musisz wdrożyć mechanizm, który będzie w stanie sprawdzać jakość danych na wejściu do modelu. Warto zwrócić uwagę na kilka głównych kryteriów:**

**- \*\*Braki danych\*\*: sprawdzaj, czy kolumny, które nie mogą zawierać braków, są kompletne.**

**- \*\*Typy danych\*\*: walidacja, czy dane mają odpowiedni typ (np. liczby tam, gdzie powinny być liczby).**

**- \*\*Zakresy wartości\*\*: walidacja, czy wartości mieszczą się w akceptowalnych zakresach.**

**- \*\*Spójność wartości kategorycznych\*\*: dla kolumn kategorycznych sprawdzaj, czy wartości należą do zestawu oczekiwanych kategorii.**

**- \*\*Spójność zależności między kolumnami\*\*: sprawdzaj, czy istnieją zależności między kolumnami, np. jeśli ktoś ma dochód = 0, nie powinien mieć w kolumnie „praca” wartości innej niż „bezrobotny”.**

**### 3. \*\*Wykrywanie dryfu danych (Data Drift)\*\***

**Dryf danych to sytuacja, w której rozkład danych wejściowych ulega zmianie w stosunku do danych, na których model został wytrenowany. To może powodować obniżenie skuteczności modelu. W kontekście monitorowania dryfu danych warto:**

**#### a) \*\*Analiza rozkładów danych\*\***

**Dla każdej kolumny określ rozkład danych, z jakim model został wytrenowany. Może to być:**

**- \*\*Średnia i odchylenie standardowe\*\* dla danych numerycznych.**

**- \*\*Rozkład częstości kategorii\*\* dla danych kategorycznych.**

**Po załadowaniu nowych danych porównaj je z rozkładem wyjściowym:**

**- Dla danych numerycznych możesz porównywać statystyki (średnia, odchylenie standardowe, percentyle itp.).**

**- Dla danych kategorycznych możesz porównywać rozkład częstości występowania każdej kategorii.**

**#### b) \*\*Testy statystyczne\*\***

**Możesz stosować testy statystyczne, takie jak \*\*test KS (Kolmogorov-Smirnov)\*\*, \*\*test chi-kwadrat\*\* (dla danych kategorycznych), czy \*\*test t\*\* (dla różnic w średnich), aby sprawdzić, czy rozkład nowych danych znacząco różni się od tego, na którym model został wytrenowany.**

**#### c) \*\*Alerty w przypadku dryfu\*\***

**Gdy wykryjesz dryf danych, powinieneś wysłać alert lub ostrzeżenie, które poinformuje, że dane wejściowe różnią się od oczekiwanych, co może wpłynąć na jakość predykcji modelu.**

**#### Przykład:**

**Jeśli trenujesz model na danych dotyczących wieku, w którym średnia wartość wynosi 35 lat, a nagle na wejściu zaczynają pojawiać się wartości skupione wokół 50 lat, może to świadczyć o dryfie danych. Konieczne jest więc wysłanie alertu, a może nawet ponowne trenowanie modelu na nowych danych.**

**### 4. \*\*Wykorzystanie narzędzi do monitorowania danych i dryfu\*\***

**Do implementacji tych kroków możesz użyć narzędzi takich jak:**

**- \*\*Pydantic\*\* (do walidacji danych wejściowych w Pythonie).**

**- \*\*Great Expectations\*\* (framework do walidacji danych, który pozwala na tworzenie reguł dotyczących jakości danych).**

**- \*\*Evidently AI\*\* (narzędzie do monitorowania dryfu danych i wydajności modelu).**

**- \*\*Scikit-learn\*\* – pozwala na implementację prostych testów statystycznych do monitorowania dryfu.**

**### 5. \*\*Automatyczne działania w przypadku problemów z danymi\*\***

**- \*\*Odrzucanie danych\*\*: Możesz odrzucać dane, które nie spełniają kryteriów jakości.**

**- \*\*Logowanie błędów\*\*: Loguj każde naruszenie reguł jakości danych, aby móc analizować, skąd pochodzą błędne dane.**

**- \*\*Ponowne trenowanie modelu\*\*: Jeśli dryf danych jest znaczący, może być konieczne ponowne wytrenowanie modelu na nowych danych, aby poprawić jego skuteczność.**

**### 6. \*\*Podłączenie do modelu AI\*\***

**Na końcu wszystkie**

## **Data Quality in:**

**Pydantic vs json schema:**

Zalety

Pydantic ma wbudowaną integrację z JSON Schema

Pydantic automatycznie waliduje dane przesyłane w formacie JSON( sprawdza typy, zakresy, zgodność z polami).

Pydantic umożliwia automatyczną konwersję danych JSON na natywne typy Pythonowe.

Pydantic jest zoptymalizowany pod kątem wydajności, co jest istotne, gdy przetwarzamy duże ilości danych JSON w czasie rzeczywistym lub w aplikacjach o wysokiej przepustowości.

Pydantic wspiera typy adnotacji Pythonowych (`type annotations).

Obsługa walidacji niestandardowej. Walidacja złożonych struktur danych JSON np.: zagnieżdżone listy, słowniki, obiekty.

Pydantic jest zintegrowany z FastAPI (pozwala na automatyczną walidację danych JSON przy tworzeniu API i komunikacji z modelami AI).

Czytelne komunikaty o błędach, gdy walidacja się nie powiedzie.

Pydantic ułatwia serializację danych z Pythonowych obiektów z powrotem do formatu JSON (przydatne przy zwracaniu wyników modeli AI jako odpowiedzi API)

**Wady:**

Pydantic jest zoptymalizowany i szybki, ale przy bardzo dużych i złożonych strukturach danych może być mało wydajny.

Brak wsparcia dla walidacji na poziomie strumienia JSON – może być konieczność wczytania danych wcześniej.

Pydantic ma ograniczoną obsługa niestandardowych typów JSON (wymagane tworzenie obsługitych formatów). `null`, brakujące klucze, wymagają dodatkowej logiki. Gdy dynamiczne/zmienne klucze walidacja może być utrudniona.

Pydantic nie obsługuje strumieniowych danych JSON (weryfikowania danych w czasie rzeczywistym). Minus przy aplikacjach wymagających niskich opóźnień.

**Pydantic – walidacja:**

**typy danych**

**Ograniczenia dla liczb**

**Walidacja dla ciągów znaków** - wyrażenia regularne, **minimalna/maksymalna długość**

walidacja list, tuple, setów z możliwością określania minimalnej/maksymalnej liczby elementów oraz typu elementów

walidacja specjalnych typów danych: adresy email, adresy URL, daty

**Pola opcjonalne** - które mogą przyjmować wartość None (Optional, Union), Union[T1, T2] – pole może przyjmować T1, lub T2.

**Funkcje walidujące** (walidatory - @validator). Komunikaty o błędach – można dodać swoje.

Wartości domyślne dla pól wymaganych

Komunikaty o błędach: zawiera zestaw wbudowanych komunikatów o błędach pokrywa najczęstsze przypadki walidacji danych (typy danych, zakresy wartości, długości stringów, formaty itp.), komunikaty są szczegółowe. Można dostosować własne komunikaty, definiując własne walidatory. Automatycznie zbiera i raportuje wszystkie błędy walidacyjne w jednym raporcie.

Inne do walidacji danych: **Pydantic**, Marshmallow, Cerberus, Voluptuous

Dryf danych – narzędzia:

Oto najpopularniejsze narzędzia do monitorowania dryfu danych, które mogą współpracować z **FastAPI** i **Pydantic** oraz obsługują dane przesyłane w formacie **JSON**:

1. **Evidently AI** – Proste narzędzie z bogatymi raportami wizualnymi.
2. **Deepchecks** – Kompleksowe rozwiązanie do monitorowania jakości danych i dryfu.
3. **Alibi Detect** – Specjalistyczne narzędzie do wykrywania dryfu i anomalii.
4. **WhyLabs AI Observatory** – Monitorowanie modeli AI i danych w czasie rzeczywistym.

Kroki monitorowania dryfu danych

#### 1. \*\*Zdefiniowanie danych bazowych (baseline)\*\*

Na początku monitorowania dryfu danych konieczne jest ustalenie bazy odniesienia (tzw. baseline). To zestaw danych, który odzwierciedla dane historyczne, na których model był trenowany. Jest to punkt odniesienia, względem którego będziemy porównywać nowe dane.

- \*\*Dane bazowe\*\*: Dane historyczne, na których model był trenowany lub wstępnie testowany.

- \*\*Metryki bazowe\*\*: Obliczenie statystyk takich jak średnia, mediana, odchylenie standardowe dla danych liczbowych lub rozkład procentowy dla danych kategorycznych.

#### 2. \*\*Zbieranie nowych danych\*\*

Regularnie zbieraj nowe dane wejściowe, które napływają do systemu. Mogą to być dane przesyłane przez API lub dane zapisane w bazach danych. Te dane będą porównywane z danymi bazowymi.

- \*\*Dane bieżące\*\*: Nowe dane wejściowe przesyłane w czasie rzeczywistym lub okresowo (np. co miesiąc).

#### 3. \*\*Porównywanie nowych danych z bazą odniesienia\*\*

Aby wykryć dryf, porównujesz dane bazowe z nowymi danymi. Porównania mogą obejmować różne typy analiz, w zależności od rodzaju danych i modelu.

- \*\*Metryki porównawcze\*\*:

- \*\*Dla danych liczbowych\*\*: Średnia, odchylenie standardowe, percentyle, wykrywanie wartości odstających.

- \*\*Dla danych kategorycznych\*\*: Rozkład procentowy poszczególnych kategorii.

#### 4. \*\*Wykrywanie dryfu\*\*

Wykorzystując narzędzia takie jak \*\*Evidently AI\*\*, \*\*Alibi Detect\*\* czy \*\*Deepchecks\*\*, możesz przeprowadzać testy statystyczne porównujące rozkłady danych bazowych i nowych danych. Istnieje kilka technik wykrywania dryfu:

- \*\*Statystyczne testy dryfu\*\*:

- \*\*Kolmogorov-Smirnov test (KS test)\*\*: Test statystyczny do porównania rozkładów danych liczbowych.

- \*\*Chi-kwadrat test\*\*: Porównuje rozkłady danych kategorycznych.

- \*\*Klasyfikatory dryfu\*\*:

- Klasyfikatory dryfu to modele, które uczą się rozróżniać, czy dane pochodzą z bazy odniesienia, czy z nowych danych. Jeśli klasyfikator dobrze rozróżnia dane, oznacza to, że dryf występuje.

#### 5. \*\*Generowanie raportów dryfu\*\*

Po przeprowadzeniu porównań możesz wygenerować raport dryfu, który informuje o tym, które cechy zmieniły swoje rozkłady, oraz o skali tych zmian. W zależności od narzędzi, raport może być w formie wizualnej (np. wykresy) lub w formacie JSON/HTML, który można zintegrować z systemem monitorowania.

- \*\*Raport dryfu\*\*: Raport wizualny lub JSON/HTML, zawierający informacje o zmianach w danych.

#### 6. \*\*Ustawienie progów dryfu\*\*

Określ próg dryfu, po przekroczeniu którego uznajesz, że zmiana w danych jest na tyle istotna, że wymaga interwencji. Może to oznaczać konieczność przetrenowania modelu lub dostosowania parametrów systemu.

- \*\*Próg dryfu\*\*: Przykładowo, można ustalić, że zmiana średniej w danej cechy o więcej niż 10% w porównaniu do bazy odniesienia będzie uznawana za dryf.

#### 7. \*\*Reagowanie na wykryty dryf\*\*

Jeśli wykryto dryf, musisz zareagować:

- \*\*Przetrenowanie modelu\*\*: Jeśli dryf danych jest znaczący, może być konieczne przetrenowanie modelu AI na nowych danych, aby przywrócić jego skuteczność.

- \*\*Aktualizacja bazy odniesienia\*\*: Możesz zaktualizować dane bazowe, aby odzwierciedlały bieżące warunki.

### Przykładowe dane do monitorowania dryfu

- \*\*Dane liczbowe\*\* (np. `wiek`, `przychód`, `ocena`):

- Statystyki: Średnia, mediana, odchylenie standardowe, maksymalna i minimalna wartość.

- Analiza: Zmiana w rozkładzie danych liczbowych.

- \*\*Dane kategoryczne\*\* (np. `płeć`, `kraj`, `kategoria produktu`):

- Statystyki: Rozkład procentowy dla każdej kategorii.

- Analiza: Porównanie rozkładów kategorycznych w danych bazowych i bieżących.

- \*\*Wyniki modelu\*\* (np. `predykcje`, `klasyfikacje`):

- Statystyki: Rozkład wyników, zmiany w dokładności lub wskaźnikach wydajności.

- Analiza: Dryf wyników modelu może wskazywać na potrzebę przetrenowania modelu.