# MLOps & Monitoring Plan

---

### 4.11.1** مقدمه**

اگرچه در حال حاضر مبتنی بر قوانین ساده `nobatnou.ir` سیستم پیش‌بینی زمان انتظار در (rule-based) است، اما به تدریج به مدل‌های یادگیری ماشین (ML) ارتقاء خواهد یافت. این سند چارچوب **MLOps** (در محیط عملیاتی ML اداره مدل‌های) و **مانیتورینگ مستمر** را تعریف می‌کند تا اطمینان حاصل شود که مدل‌ها قابل اعتماد، قابل نگهداری و فاقد ریسک‌های عملیاتی هستند.

---

### 4.11.2** اهداف MLOps**

- **قابلیت اطمینان (Reliability):** مدل در محیط تولید همواره در دسترس و پایدار باشد.
- **قابلیت بازتولید (Reproducibility):** هر نسخه از مدل و نتایج آن قابل بازتولید باشد.
- **مقیاس‌پذیری (Scalability):** فرآیندهای آموزش و استنتاج با رشد داده‌ها مقیاس پذیرند.
- **مانیتورینگ (Monitoring):** به طور مستمر پایش شود drift عملکرد مدل، کیفیت داده و.
- **خودکارسازی (Automation):** کاهش مداخله دستی در چرخه حیات مدل.

---

### 4.11.3** چرخه حیات مدل (Model Lifecycle)**

#### **مراحل چرخه حیات مدل در نوبت نو:**

1. **جمع‌آوری داده (Data Collection)**

2. **آماده‌سازی داده (Data Preparation)**

3. **آموزش مدل (Model Training)**

4. **ارزیابی (Evaluation)**

5. **تست استقرار (Deployment Testing)**

6. **استقرار (Deployment)**

7. **مانیتورینگ و پایش (Monitoring & Observability)**

8. **بازآموزی (Retraining)**

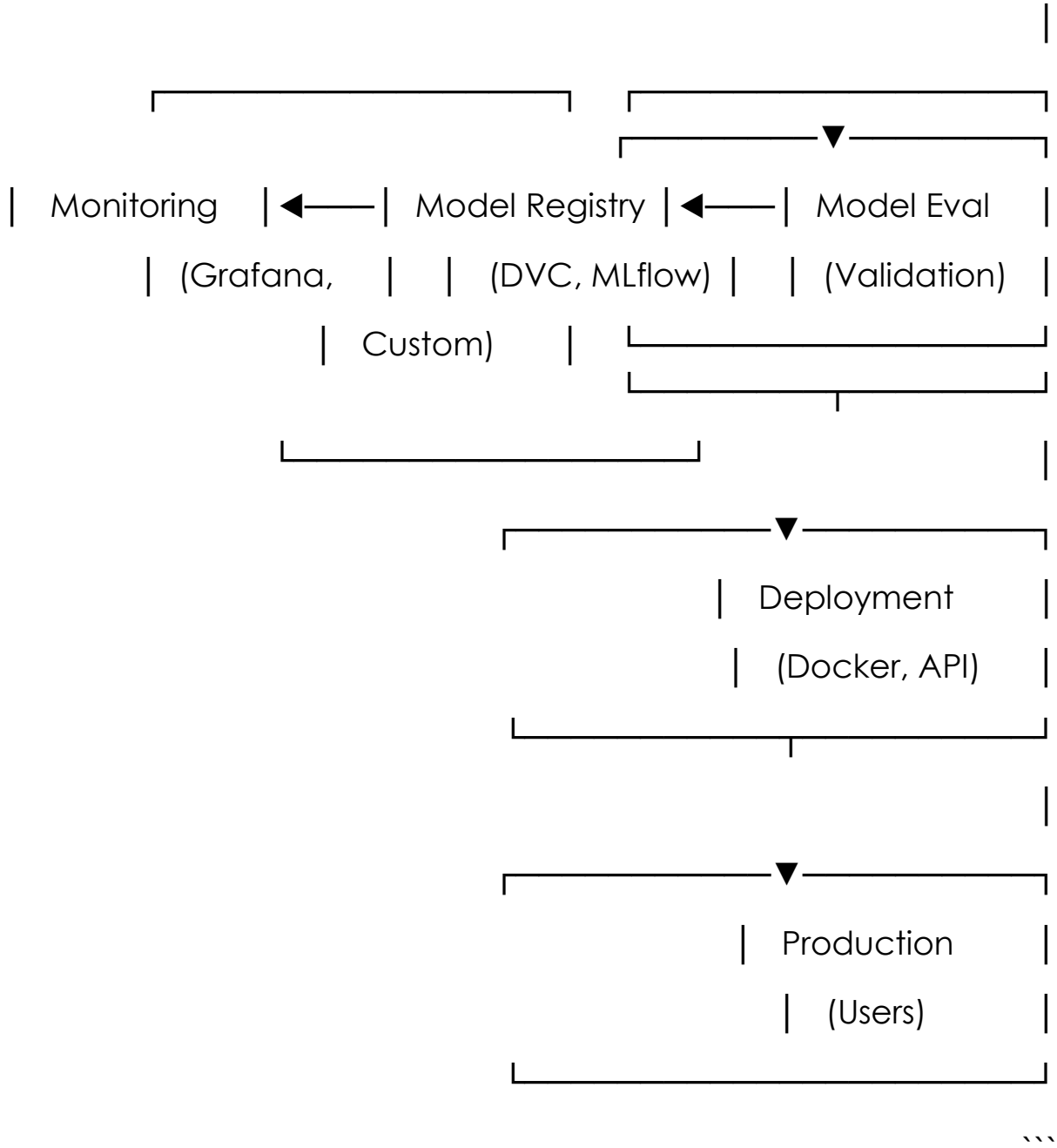
9. **نسخه‌بندی و بایگانی (Versioning & Archival)**

---

### 4.11.4 معماری MLOps (MLOps Architecture)

**ساختار پیشنهادی (بر اساس مقیاس فعلی):**

```
┌──────────────────┐    ┌──────────────────┐    ┌──────────────────┐
│   Data Sources   │──▶│   Feature Store  │──▶│   Training Pipe  │
│   (DB, Logs)     │    │   (MySQL + CSV)  │    │   (Python)       │
└──────────────────┘    └──────────────────┘    └──────────────────┘
                                                        │
```




```

**:اجزای کلیدی** ####

مثلاً) ذخیره ویژگی‌های از پیش محاسبه‌شده **:Feature Store** -
.`avg_wait_time_last_week`)

.**DVC** یا **MLflow** ثبت و نسخه‌بندی مدل‌ها با استفاده از **:Model Registry** -
```

- **Training Pipeline:** اسکریپت‌های خودکار آموزش با **GitHub Actions**.

- **Serving API:** سرویس مدل با **FastAPI** درون **Docker**.

- **Monitoring Dashboard:** نمایش متریک‌های مدل و داده.

---

### 4.11.5 **مانیتورینگ مدل (Model Monitoring)**

#### الف) مانیتورینگ عملکرد (Performance Monitoring)**

**متریک‌های کلیدی تحت پایش:**

| متریک | هدف | آستانه هشدار | ابزار اندازه‌گیری |

| --- | --- | --- | --- |

| **MAE (Mean Absolute Error)** | دقت پیش‌بینی | < ۲۰ دقیقه | محاسبه روزانه بر اساس داده‌های واقعی |

| درصد پیش‌بینی‌های درون محدوده** | قابلیت اطمینان | < ۵۰٪ (محدوده ±۲۰ دقیقه) | محاسبه ** هر ۶ ساعت |

| **زمان پاسخگویی مدل (Latency)** | API کارایی | < ۱ ثانیه (p95) | APM (New Relic/Sentry) |

| لاگ‌های** | در دسترس بودن | < ۵٪ | Nginx نرخ درخواست‌های ناموفق** |

**هشدارها:**

- اگر MAE (ایمیل به تیم داده) **هشدار سطح ۲** ← از ۲۰ دقیقه عبور کرد.

.از ۳۰ دقیقه عبور کرد ← **هشدار سطح ۱** (تماس با مدیر فنی) MAE اگر -

**(انحراف) Drift ب) مانیتورینگ** ####

| نحوه تشخیص | فرکانس | آستانه هشدار | Drift نوع |

| ---------- | ---------- | ---------- | ---------- |

| **Data Drift** | مقایسه توزیع ویژگی‌های ورودی (مانند `hour_of_day`) بین هفته جاری و گذشته | هفتگی | KL Divergence > 0.2 |

| **Concept Drift** | مقایسه MAE افزایش | هفتگی | هفته جاری با میانگین ۴ هفته گذشته | MAE > ۱۰٪ |

| **Label Drift** | تغییر توزیع زمان واقعی انتظار (`actual_wait`) | ماهانه | Kolmogorov-Smirnov test p-value < 0.05 |

**:ابزار تشخیص**

- کتابخانه‌های Python: `alibi-detect`, `evidently`.

- Dashboard: **Grafana** با داده‌های محاسبه‌شده.

**(Infrastructure Monitoring) ج) مانیتورینگ زیرساخت** ####

- **منابع سرور:** CPU ،RAM ،I/O (با **Netdata**).

- **وضعیت کانتینر:** سرویس مدل Docker، سلامت API endpoint.

- **دیتابیس ویژگی‌ها:** اتصالات، حجم داده.

---

### 4.11.6 **نسخه‌بندی و مدیریت مدل (Model Versioning)**

#### **سیاست نسخه‌بندی:**

- نسخه‌ها به صورت `MAJOR.MINOR.PATCH` مدیریت می‌شوند.

- `MAJOR`: تغییرات اساسی در معماری مدل یا ویژگی‌ها.

- `MINOR`: بهبود دقت یا افزودن ویژگی جدید.

- `PATCH`: رفع باگ یا بهینه‌سازی جزئی.

#### **ذخیره‌سازی:**

| ابزار | محل ذخیره | مؤلفه |
|---|---|---|
| Git | GitHub (با tagging) | **کد آموزش** |
| MLflow Tracking | MLflow در Metadata + مدل آموزش‌دیده | **مدل** |
| DVC | Object Storage (مثلاً MinIO) + DVC | **دیتاست‌ها** |
| Git | در مخزن کد YAML هایپرپارامترها | فایل** |

#### **نمونه ساختار فایل‌ها:**

```
models/
├── v1.0.0/
│   ├── model.pkl
```

```
|   ├── metadata.json
|   └── train_params.yaml
├── v1.1.0/
|   └── ...
└── latest -> v1.1.0
```

---

### 4.11.7** فرآیند بازآموزی (Retraining Pipeline)**

#### محرک‌های بازآموزی (Retraining Triggers):**

۱. **زمان‌بندی شده (Scheduled):** هر ۲ هفته یکبار.

۲. **بر اساس عملکرد (Performance-based):** اگر MAE > ۲۰ برای ۳ روز متوالی دقیقه.

۳. **بر اساس Drift (Drift-based):** اگر Data Drift از آستانه عبور کند.

۴. **بر اساس حجم داده (Data Volume-based):** پس از جمع‌آوری ۵۰۰ نمونه جدید.

#### مراحل خودکار بازآموزی:**

1. **جمع‌آوری داده جدید** از ۲ هفته اخیر.

2. **اجرای آموزش** با اسکریپت `train.py`.

3. **ارزیابی مدل جدید** روی داده‌های اعتبارسنجی.

.(مجازی A/B testing) **مقایسه با مدل فعلی** .4

.Model Registry اگر بهتر بود:** ثبت در** .5

.staging تست استقرار** در محیط** .6

.انتخاب برای استقرار** توسط مدیر مدل** .7

.**Airflow** یا **GitHub Actions** خودکارسازی:** با**

---

### **4.11.8 تست قبل از انتشار (Pre-deployment Testing)**

**:مراحل تست** ####

| مرحله | هدف | روش |

|---|---|---|

| **Unit Testing** | تست توابع آماده‌سازی داده و آموزش | `pytest` روی کد Python |

| **Integration Testing** | تست end-to-end pipeline (داده → مدل → اجرای pipeline پیش‌بینی) | روی داده‌های ساختگی |

| **A/B Testing** | مقایسه مدل جدید با مدل فعلی روی داده‌های تاریخی | محاسبه متریک‌ها برای هر دو |

| **Canary Deployment** | تست مدل جدید روی بخش کوچکی از ترافیک واقعی (۱٪) | load balancer با routing |

| **Stress Testing** | تست بار بر روی سرویس مدل API | `locust` یا `k6` |

#### معیارهای پذیرش** (Acceptance Criteria):**

- مدل جدید باید **MAE** (یا حداکثر ۵٪ بدتر) **کم‌تر** از مدل فعلی داشته باشد.

- **نباید حذف شده باشد** هیچ ویژگی حیاتی.

- **باید زیر ۵۰۰ میلی‌ثانیه باشد** زمان استنتاج.

- **همه باید عبور** کنند تست‌های یکپارچگی.

---

### استقرار و Rollback (Deployment & Rollback)** 4.11.9**

#### استراتژی استقرار**:**

- **Blue-Green Deployment:**

- محیط **آبی (Blue)**: مدل فعلی (v1.0.0).

- محیط **سبز (Green)**: مدل جدید (v1.1.0).

- پس از تست، ترافیک به‌تدریج به سبز منتقل می‌شود.

#### روش Rollback**:**

۱. **شناسایی مشکل:** از طریق هشدارها یا گزارش کاربران.

۲. **تصمیم Rollback:** مدیر مدل تأیید می‌کند.

۳. **بازگشت به محیط آبی (مدل قبلی) از طریق** تغییر مسیر ترافیک: load balancer.

۴. **بررسی علت:** تحلیل لاگ‌ها و داده‌ها برای شناسایی ریشه مشکل.

.کمتر از ۱۵ دقیقه از شناسایی تا بازیابی **:Rollback زمان هدف**

---

### **4.11.10 مسئولیت‌ها و نقش‌ها (Roles & Responsibilities)**

| نقش | مسئولیت‌ها | فرد/تیم |

|---|---|---|

| مدیر مدل** (Model Owner)** | تأیید انتشار، نظارت بر عملکرد، تصمیم‌گیری Rollback | مدیر فنی |

| ML مهندس** (ML Engineer)** | توسعه pipeline، آموزش مدل، نگهداری زیرساخت | تیم داده | MLOps |

| مهندس نرم‌افزار** (Software Engineer)** | توسعه API یکپارچه‌سازی مدل با سرویس‌های همراه | تیم توسعه |

| تحلیلگر داده** (Data Analyst)** | تحلیل Drift | گزارش‌دهی عملکرد، پیشنهاد بازآموزی | تیم داده |

| پشتیبان عملیاتی** (Ops)** | مانیتورینگ زیرساخت، هشدارها، پشتیبان‌گیری | تیم فنی |

---

### **4.11.11 ابزارهای پیشنهادی (Tool Stack)**

| دسته | ابزار | توضیح |

| ------ | -------- | -------- |

| کد و کنترل نسخه | Git, GitHub | **مدیریت کد و نسخه** |

| ردیابی آزمایش‌ها، نسخه‌بندی مدل و داده | MLflow, DVC | **مدیریت مدل** |

| خودکارسازی آموزش و استقرار | GitHub Actions, Apache Airflow | **پایپ‌لاین** |

| نمایش متریک‌ها و هشدار | Grafana, Prometheus, Custom Scripts | **مانیتورینگ** |

| و کانتینری‌سازی API | FastAPI, Docker, Nginx | **سرویس‌دهی مدل** |

| Drift تست کد، بار و | pytest, locust, evidently | **تست** |

---

### **4.11.12 برنامه اجرایی (Implementation Roadmap)**

#### **فاز ۱: پایه MLOps (۳ ماه)**

- راه‌اندازی Model Registry با MLflow.

- ایجاد مانیتورینگ ساده (MAE روزانه).

- راه‌اندازی pipeline دستی آموزش.

#### **فاز ۲: خودکارسازی (۶ ماه)**

- خودکارسازی بازآموزی با GitHub Actions.

- راه‌اندازی مانیتورینگ Drift.

- اجرای Canary Deployment.

**فاز ۳: تکمیل (۱۲ ماه)** ####

- استقرار کامل Blue-Green Deployment.

- مانیتورینگ Real-time با Grafana.

- یکپارچه‌سازی Feature Store.

---

### **4.11.13 هزینه‌های MLOps (MLOps Costs)**

| مؤلفه | هزینه ماهانه تخمینی | توضیح |

|---|---|---|

| **ابزارهای مانیتورینگ (Grafana Cloud)** | ۱۰ دلار | برای مانیتورینگ پیشرفته |

| **ذخیره‌سازی مدل‌ها و داده‌ها** | ۵ دلار** | Object Storage |

| **سرورهای اضافی (Training/Staging)** | ۲۰ دلار | ۲ سرور مجازی |

| **نیروی انسانی (نیمه‌وقت ML مهندس)** | ۵۰۰ دلار | حقوق و دستمزد |

| **مجموع** | **۵۳۵ دلار** | برای فاز اول** |

**توجه:** هزینه‌ها با مقیاس سیستم افزایش می‌یابد.

---

### **4.11.۱۴ **نتیجه‌گیری

به گونه‌ای طراحی شده است که **مدل‌ها را از `nobatnou.ir` و مانیتورینگ MLOps سیستم بمب ساعتی به دارایی پایدار** تبدیل کند. با ترکیب **خودکارسازی، مانیتورینگ چندلایه و فرآیندهای کنترل‌شده انتشار**، اطمینان حاصل می‌شود که مدل‌ها نه‌تنها دقیق، بلکه **قابل اعتماد، قابل نگهداری و واکنش‌گرا** به تغییرات هستند. این سند به عنوان نقشه راه تیم فنی برای حرکت از مدل ساده rule-based به سیستم ML صنعتی عمل می‌کند.