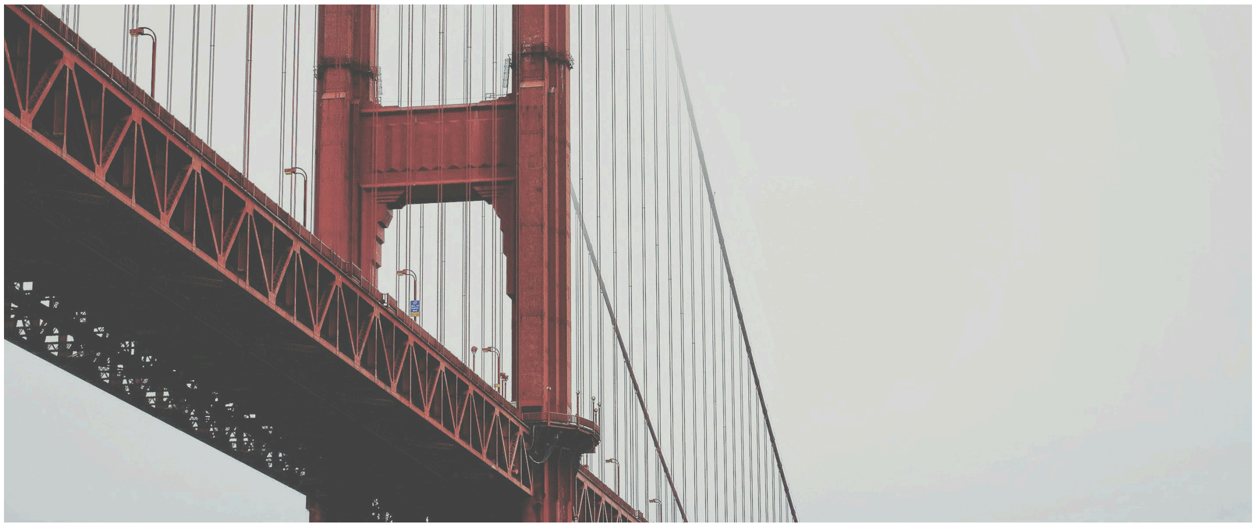

🚀 AutoML Platform - 8-Week Launch Roadmap

⚙️ WEEK 1: CLI ML Engine (Days 1-14)

Day 1-2: Project Setup + Base Framework

- [x] ~~Initialize monorepo structure~~
- [x] ~~Set up Python environment (Poetry) in `/apps/workers`~~
- [x] ~~Install core dependencies (scikit-learn, xgboost, tensorflow, pandas)~~
- [x] ~~Create base trainer interface~~
- [x] ~~Create evaluator interface~~
- [x] ~~Set up logging and error handling framework~~

Day 3-5: Implement 10 ML Algorithms

- [x] ~~`logistic_regression.py`~~
- [x] ~~`random_forest.py`~~
- [x] ~~`xgboost.py`~~
- [x] ~~`svm.py` - SVC/SVR~~

- [x] ~~`neural_network.py`~~
- [x] ~~`knn.py`~~
- [x] ~~`decision_tree.py`~~
- [x] ~~`naive_bayes.py`~~
- [x] ~~`linear_regression.py`~~
- [x] ~~`kmeans.py` (clustering)~~
- [x] ~~`pca.py` (dimensionality reduction)~~

Day 6-7: Test 5 Algorithms

- [x] ~~Load iris dataset (classification)~~
- [x] ~~Load housing data (regression)~~
- [x] ~~Test algorithms 1-5~~
- [x] ~~Verify predictions are sensible~~
- [x] ~~Check model serialization works~~

Day 8-9: CLI Interface

- [x] ~~Create CLI tool using Click~~
- [x] ~~Implement train command~~
- [x] ~~Implement preprocess command~~
- [x] ~~Add --dataset, --target, --task flags~~
- [x] ~~Add output handling (JSON, images)~~

Day 10-11: Preprocessors Implementation

- [x] ~~Missing value imputation~~
- [x] ~~Outlier handling~~
- [x] ~~Feature scaling~~
- [x] ~~One-hot encoding~~
- [x] ~~Label encoding~~

- [x] ~~Duplicate removal~~
- [x] ~~Data type conversion~~
- [x] ~~Datetime feature extraction~~

Day 12-13: Test All 10 Algorithms + Preprocessors

- [x] ~~Test remaining 5 algorithms~~
- [x] ~~Test all preprocessors~~
- [x] ~~Run full pipeline on 3 real-world datasets~~
- [x] ~~Fix any bugs found~~

Day 14: Buffer + Integration Testing

- [x] ~~Final integration testing~~
- [x] ~~CLI documentation~~
- [x] ~~All validation checkpoints passing~~

🔐 WEEK 2: Backend API (Days 15-28)

Day 15-16: FastAPI + Database Setup

- [x] ~~Set up FastAPI in `/apps/api/`~~
- [x] ~~Install dependencies (fastapi, uvicorn, sqlalchemy, alembic, psycopg2, redis)~~
- [x] ~~Create database models (User, Dataset, Workflow, Job, Model, CreditTransaction, FraudEvent, CostTracking)~~
- [x] ~~Set up Alembic migrations~~

Day 17-18: Database Deployment + Schemas

- [x] ~~Create Neon PostgreSQL database (dev + staging)~~
- [x] ~~Apply migrations to Neon~~
- [x] ~~Add database indexes for performance~~
- [x] ~~Set up connection pooling~~

- [x] ~~Configure database read replica (staging)~~

Day 19-20: JWT Authentication

- [x] ~~Implement password hashing (bcrypt)~~
- [x] ~~Implement JWT token generation (access + refresh)~~
- [x] ~~Implement token verification~~
- [x] ~~Create auth middleware~~
- [x] ~~Implement token refresh logic~~
- [x] ~~Set up Redis for token blacklist~~
- [x] ~~Create auth endpoints (register, login, refresh, logout, me)~~

Day 21-22: OAuth2 Integration

- [x] ~~Set up Google OAuth2~~
- [x] ~~Set up GitHub OAuth2~~
- [x] ~~Create OAuth callback endpoints~~
- [x] ~~Handle OAuth user creation/linking~~
- [x] ~~Test OAuth flow end-to-end~~

Day 23-24: Dataset Endpoints

- [x] ~~GET /api/datasets (paginated)~~
- [x] ~~POST /api/datasets/upload-url (presigned URL)~~
- [x] ~~POST /api/datasets/confirm~~
- [x] ~~GET /api/datasets/{id}~~
- [x] ~~DELETE /api/datasets/{id}~~
- [x] ~~GET /api/datasets/{id}/stats~~
- [x] ~~Set up Cloudflare R2 integration~~

Day 25-26: Job Endpoints + Celery

- [ ] Configure Celery with Redis broker
- [ ] Create job queues (high_priority, normal, overflow)
- [ ] Create job execution task
- [ ] POST /api/jobs (with credit deduction)
- [ ] GET /api/jobs (paginated, filtered)
- [ ] GET /api/jobs/{id}
- [ ] POST /api/jobs/{id}/cancel (with refund)
- [ ] POST /api/jobs/{id}/retry
- [ ] GET /api/jobs/{id}/logs

Day 27-28: Credit System + Testing

- [ ] GET /api/credits/balance
- [ ] GET /api/credits/transactions
- [ ] POST /api/credits/mock-purchase
- [ ] Implement immutable ledger logic
- [ ] Credit deduction with validation
- [ ] Refund logic with penalty multiplier
- [ ] Unit tests for credit logic
- [ ] Integration tests for credit endpoints

🎨 WEEK 3: Frontend Foundation (Days 29-42)

Day 29-30: Frontend Setup

- [ ] Create Vite + React project in `/apps/web/`
- [ ] Install dependencies (react-router, axios, zustand, tanstack query)
- [ ] Install shadcn/ui + Tailwind CSS
- [ ] Set up folder structure

- [ ] Configure Axios base client
- [ ] Set up React Router
- [ ] Configure environment variables

Day 31-32: Auth UI

- [ ] LoginForm.jsx
- [ ] RegisterForm.jsx
- [ ] OAuthButtons.jsx
- [ ] ProtectedRoute.jsx
- [ ] Auth pages (LoginPage, RegisterPage, ForgotPassword)
- [ ] Zustand auth store with token management
- [ ] Axios interceptor for auto token refresh

Day 33-34: Dashboard Layout

- [ ] DashboardLayout.jsx (sidebar + navbar)
- [ ] Sidebar.jsx navigation
- [ ] Navbar.jsx with user menu
- [ ] StatCard.jsx component
- [ ] DashboardPage.jsx with stats and recent jobs
- [ ] Responsive mobile layout

Day 35-36: Dataset Management UI

- [ ] DatasetTable.jsx (with filters, sorting, pagination)
- [ ] UploadModal.jsx (drag-drop)
- [ ] DatasetDetails.jsx
- [ ] DatasetStats.jsx
- [ ] DatasetsPage.jsx
- [ ] DatasetDetailsPage.jsx

- [ ] Presigned URL upload flow with progress

Day 37-38: Job Management UI

- [ ] JobTable.jsx (status filters)
- [ ] JobDetails.jsx with node timeline
- [ ] JobStatusBadge.jsx
- [ ] LogsViewer.jsx
- [ ] JobsPage.jsx
- [ ] JobDetailsPage.jsx
- [ ] Cancel and retry functionality

Day 39-40: WebSocket Real-Time Updates

- [ ] Create useWebSocket.js hook
- [ ] Connect to WebSocket server
- [ ] Subscribe to job updates
- [ ] Handle connection errors + auto-reconnect
- [ ] Update React Query cache on events
- [ ] Add WebSocket endpoint in FastAPI
- [ ] Implement Redis Pub/Sub integration

Day 41-42: Testing + Polish

- [ ] Unit tests for auth store (Vitest)
- [ ] Component tests for forms
- [ ] Add loading skeletons
- [ ] Add error boundaries
- [ ] Add toast notifications
- [ ] Improve responsive design
- [ ] Fix accessibility issues

📚 WEEK 4: Workflow Canvas + Tutorial Pages (Days 43-56)

Day 43-45: React Flow Setup (Desktop)

- [ ] Install `reactflow` package
- [ ] Create workflow store (Zustand)
- [ ] WorkflowCanvas.jsx (main canvas)
- [ ] NodePalette.jsx (draggable palette)
- [ ] WorkflowToolbar.jsx (save, run, duplicate)
- [ ] Set up custom node types
- [ ] Configure edge types
- [ ] Add zoom, pan, minimap controls

Day 46-48: Custom Node Components (Desktop)

- [ ] DatasetNode.jsx
- [ ] PreprocessNode.jsx
- [ ] ModelNode.jsx
- [ ] EvaluationNode.jsx
- [ ] Configuration dialogs for each node
- [ ] Custom styling with Tailwind
- [ ] Input/output handles
- [ ] Delete functionality

Day 49-51: Mobile Building Block Builder

- [ ] Create BuildingBlockBuilder.jsx (mobile UI)
- [ ] Step-by-step form instead of canvas
- [ ] DatasetSelector component

- [ ] PreprocessingSelector component
- [ ] ModelSelector component
- [ ] MetricsSelector component
- [ ] Mobile-optimized styling
- [ ] Responsive transitions between steps

Day 52-53: Edge Validation + DAG Logic

- [ ] Detect cycles in DAG
- [ ] Validate node connections
- [ ] Prevent invalid connections
- [ ] Highlight invalid nodes/edges
- [ ] Create validation utility

Day 54: Workflow Persistence

- [ ] Create workflow CRUD endpoints
- [ ] Debounced autosave
- [ ] Save workflow to backend
- [ ] Load workflow from backend
- [ ] Duplicate workflow
- [ ] Delete workflow
- [ ] WorkflowsPage.jsx
- [ ] WorkflowEditorPage.jsx

Day 55-56: Tutorial Pages + Buffer

- [ ] Create `/pages/tutorials/` route
- [ ] Tutorial index page (all topics)
- [ ] ML Basics tutorial page
- [ ] Classification tutorial page

- [ ] Regression tutorial page
- [ ] Clustering tutorial page
- [ ] Feature Engineering tutorial page
- [ ] Model Evaluation tutorial page
- [ ] SEO optimization (meta tags, structured data)
- [ ] Mobile-optimized tutorial layout
- [ ] Testing + Polish all components

🔄 WEEK 5: Workflow Backend + Results (Days 57-70)

Day 57-59: Workflow Execution Logic

- [ ] Parse workflow DAG from JSON
- [ ] Execute nodes sequentially
- [ ] Update job status at each step
- [ ] Update node statuses
- [ ] Handle failures and refunds
- [ ] Publish progress events via Redis
- [ ] Download dataset from R2
- [ ] Run preprocessing
- [ ] Train models
- [ ] Run evaluation
- [ ] Save results to database

Day 60-61: Model Results Backend

- [ ] GET /api/models (filtered by job_id)
- [ ] GET /api/models/{id}
- [ ] GET /api/models/{id}/metrics

- [ ] GET /api/models/{id}/plots (presigned URLs)
- [ ] GET /api/jobs/{job_id}/models/compare
- [ ] Generate confusion matrix
- [ ] Generate ROC curve
- [ ] Generate precision-recall curve
- [ ] Generate feature importance
- [ ] Upload plots to R2

Day 62-63: Model Comparison UI

- [ ] ModelList.jsx
- [ ] ModelComparison.jsx
- [ ] MetricsCards.jsx
- [ ] ModelsPage.jsx
- [ ] ModelDetailsPage.jsx
- [ ] Model sorting and filtering

Day 64-65: Confusion Matrix + ROC Curve

- [ ] Install recharts
- [ ] ConfusionMatrix.jsx (heatmap)
- [ ] ROCCurve.jsx (line chart)
- [ ] Responsive design
- [ ] Color coding for matrices

Day 66-67: Feature Importance Visualization

- [ ] FeatureImportance.jsx (bar chart)
- [ ] Mobile-responsive charts
- [ ] Tooltip support
- [ ] Legend display

Day 68-69: Results Polish

- [ ] Add download button for plots
- [ ] Improve chart styling
- [ ] Add tooltips and legends
- [ ] Add filters (algorithm type, metric threshold)
- [ ] Integration testing (end-to-end)

Day 70: Buffer

- [ ] Fix outstanding bugs
- [ ] Improve error messages
- [ ] Add missing loading states
- [ ] Cross-browser testing

💳 WEEK 6: Stripe Integration (Days 71-84)

Day 71-73: Stripe Setup + Checkout

- [ ] Create Stripe account (test mode)
- [ ] Get API keys
- [ ] Install Stripe SDK (backend + frontend)
- [ ] POST /api/payments/create-checkout
- [ ] Configure checkout with credit packages
- [ ] Install `@stripe/stripe-js` and `@stripe/react-stripe-js`
- [ ] Create checkout flow
- [ ] Redirect to Stripe checkout
- [ ] Configure credit packages (100, 500, 1000, 5000)

Day 74-75: Stripe Webhook Handler

- [ ] POST /api/webhooks/stripe

- [ ] Verify webhook signature
- [ ] Handle checkout.session.completed event
- [ ] Issue credits to user
- [ ] Handle payment failures
- [ ] Handle refunds
- [ ] Add webhook to Stripe dashboard

Day 76-77: Purchase UI

- [ ] CreditBalance.jsx
- [ ] PurchaseModal.jsx
- [ ] PackageCard.jsx
- [ ] TransactionHistory.jsx
- [ ] CreditsPage.jsx
- [ ] Display balance prominently
- [ ] Show transaction history

Day 78-79: Transaction History

- [ ] Add filters (date range, transaction type)
- [ ] Add pagination
- [ ] Add export to CSV
- [ ] Color-code transaction types
- [ ] Add credit notifications (toast)
- [ ] Low credit warnings
- [ ] Insufficient credit errors

Day 80-81: Payment Testing

- [ ] Test successful payment
- [ ] Test declined payment

- [ ] Test webhook failure scenarios
- [ ] Test refund flow
- [ ] Test all credit packages

Day 82-84: Edge Cases + Refunds

- [ ] Implement refund logic
- [ ] Calculate penalty multiplier
- [ ] Update cancellation_count_30d
- [ ] Handle duplicate webhook events
- [ ] Handle user balance edge cases
- [ ] Transaction safety testing
- [ ] Production readiness check

📱 WEEK 7: AdMob Integration (Days 85-102)

Day 85-87: AdMob Frontend Integration

- [ ] Create AdMob account
- [ ] Create ad unit
- [ ] Get AdMob app ID and ad unit ID
- [ ] Install AdMob SDK
- [ ] AdRewardButton.jsx
- [ ] AdRewardModal.jsx
- [ ] Handle ad callbacks (loaded, watched, failed)

Day 88-90: Server-Side Ad Verification

- [ ] POST /api/credits/claim-ad-reward
- [ ] Verify with AdMob server
- [ ] Extract reward amount

- [ ] Handle verification failures
- [ ] Fallback if verification unavailable

Day 91-93: IP Intelligence Integration

- [ ] Create IPQualityScore account
- [ ] Get API key
- [ ] Extract IP from request
- [ ] Call IPQualityScore API
- [ ] Calculate fraud score
- [ ] Detect VPN, datacenter, proxy
- [ ] Store results in database

Day 94-96: Device Fingerprinting

- [ ] Create FingerprintJS account
- [ ] Install frontend SDK
- [ ] Generate device fingerprint on client
- [ ] Store in memory
- [ ] Send with ad reward request
- [ ] Check if fingerprint is blocked
- [ ] Update fraud score based on device checks

Day 97-98: Velocity Checks

- [ ] Track ad claims per user per hour/day
- [ ] Track ad claims per IP per hour/day
- [ ] Track ad claims per device per hour/day
- [ ] Enforce 10 ads per day limit
- [ ] Enforce 30-second cooldown
- [ ] Display time until next ad available

- [ ] Return 429 on rate limit

Day 99-100: Fraud Scoring Algorithm

- [ ] Combine all fraud signals
- [ ] Calculate final fraud score (0.0 to 1.0)
- [ ] Determine action (allow, flag, block)
- [ ] Log to fraud_events table
- [ ] Set action thresholds (0.7 = blocked, 0.5 = flagged)
- [ ] Update user fraud flags

Day 101-102: Fraud Testing + Polish

- [ ] Test with VPN (should be flagged/blocked)
- [ ] Test rapid ad clicking
- [ ] Test same device, different accounts
- [ ] Test normal user behavior
- [ ] Show daily ad limit remaining
- [ ] Show cooldown timer
- [ ] Test bot detection

🚀 WEEK 8: Deploy + Polish + Launch (Days 103-120)

Day 103-105: Railway Deployment

- [ ] Create Railway project
- [ ] Configure environment variables
- [ ] Deploy FastAPI app
- [ ] Deploy Celery worker
- [ ] Configure Redis addon
- [ ] Configure PostgreSQL connection

- [ ] Test API endpoints
- [ ] Build production bundle
- [ ] Deploy frontend to Railway (or Vercel)
- [ ] Configure frontend environment variables
- [ ] Test in production

Day 106-107: Cost Tracking + Admin Panel

- [ ] Record estimated cost before job
- [ ] Record actual cost after job
- [ ] Store node breakdown
- [ ] Admin login (separate credentials)
- [ ] User management (view, suspend, ban)
- [ ] GET /api/admin/users
- [ ] POST /api/admin/users/{id}/suspend
- [ ] POST /api/admin/users/{id}/ban
- [ ] GET /api/admin/fraud-events
- [ ] GET /api/admin/cost-monitoring

Day 108-109: End-to-End Testing

- [ ] Free user journey (ads → credits → job → results)
- [ ] Paying user journey (purchase → job → results)
- [ ] Admin journey (review fraud, suspend user, cost monitoring)
- [ ] All critical flows working

Day 110-111: Performance Optimization

- [ ] Add database indexes
- [ ] Optimize slow queries
- [ ] Add Redis caching

- [ ] Configure connection pooling
- [ ] Run Lighthouse audit
- [ ] Optimize bundle size (code splitting)
- [ ] Lazy load images
- [ ] Minify assets
- [ ] Configure CDN (Cloudflare)
- [ ] Target Lighthouse score > 90
- [ ] Load testing (50-100 concurrent users)

Day 112-113: Security Audit

- [ ] Run OWASP ZAP scan
- [ ] Run npm audit
- [ ] Review authentication logic
- [ ] Review authorization logic
- [ ] Verify HTTPS-only
- [ ] Check CORS configuration
- [ ] Add security headers (CSP, HSTS, X-Frame-Options)
- [ ] Disable debug mode in production
- [ ] Rotate API keys/secrets
- [ ] Enable database SSL

Day 114-116: Final Polish

- [ ] Review all error messages
- [ ] Add helpful tooltips
- [ ] Improve loading states
- [ ] Add empty states with CTA
- [ ] Fix visual bugs

- [ ] Test mobile devices
- [ ] Test different browsers
- [ ] Improve accessibility (WCAG 2.1 AA)
- [ ] Better error messages
- [ ] Onboarding tooltips
- [ ] Fix all reported issues

Day 117-118: Documentation + User Guide

- [ ] Getting started guide
- [ ] How to upload datasets
- [ ] How to create workflows
- [ ] How to interpret results
- [ ] How to purchase credits
- [ ] How to earn credits via ads
- [ ] FAQ section
- [ ] Troubleshooting guide
- [ ] API documentation (Swagger)
- [ ] Rate limits documentation
- [ ] Error codes documentation
- [ ] Landing page copy
- [ ] Pricing page

Day 119-120: Final Checks + LAUNCH

- [ ] Verify all services running
- [ ] Test payment flow
- [ ] Test ad reward flow
- [ ] Verify monitoring and alerts

- [ ] Check database backups
- [ ] Test disaster recovery
- [ ] Smoke test all critical flows
- [ ] Set up social media accounts
- [ ] Write launch blog post
- [ ] Switch Stripe to production mode
- [ ] Switch AdMob to production mode
- [ ] Announce on Twitter
- [ ] Announce on LinkedIn
- [ ] Submit to Product Hunt
- [ ] Monitor for issues post-launch

🎯 Critical Checkpoints

Week 4 Checkpoint (Day 56)

- [ ] Workflow canvas working smoothly
- [ ] Mobile building blocks working
- [ ] Tutorial pages complete
- [ ] All validation passing

Week 6 Checkpoint (Day 84)

- [ ] Stripe payments fully functional
- [ ] All payment tests passing
- [ ] Webhook handler robust

Week 8 Checkpoint (Day 115)

- [ ] All critical features working
- [ ] No critical bugs

- [ ] System stable
- [ ] Performance acceptable
- [ ] **GO LIVE DECISION**

📊 Success Criteria

Minimum for Launch:

- [ ] Users can register and login
- [ ] Users can upload datasets
- [ ] Users can create workflows (desktop canvas OR mobile builder)
- [ ] Users can run jobs and see progress
- [ ] Users can view model results and comparisons
- [ ] Users can purchase credits (Stripe)
- [ ] Users can earn credits via ads (AdMob)
- [ ] System deployed and accessible
- [ ] Basic fraud protection active
- [ ] No critical security issues
- [ ] Tutorial pages indexed by search engines

Nice to Have (not required):

- [ ] Both desktop canvas AND mobile builder (can launch with one)
- [ ] All 10 algorithms (can launch with 5-7)
- [ ] Perfect UI polish
- [ ] Advanced fraud detection