Football Jackpot Probability Engine - Implementation Guide

Complete Setup & Deployment Instructions

This guide walks you through setting up, running, and deploying both the backend and frontend components of the Football Jackpot Probability Engine.

📋 Table of Contents

Prerequisites

Required Software

- **Python 3.11+** (for backend)
- **PostgreSQL 15+** (for database)
- **Redis 7+** (for task queue)
- **Node.js 18+** and npm (for frontend)
- **Git** (for version control)

Optional Tools

- **Docker & Docker Compose** (recommended for easier setup)
- **AWS CLI** or **Vercel CLI** (for deployment)
- **Postman** or **Insomnia** (for API testing)

Backend Setup

Step 1: Install Dependencies

```bash
cd jackpot-backend

# Create virtual environment
python3 -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate

# Install Python packages
pip install -r requirements.txt --break-system-packages
```

Step 2: Configure Environment

Create `.env` file in backend directory:

```env
# Database
DATABASE_URL=postgresql://user:password@localhost:5432/jackpot_db

# Redis
REDIS_URL=redis://localhost:6379/0

# API Keys
API_FOOTBALL_KEY=your_api_football_key_here

# Security
SECRET_KEY=your-secret-key-generate-random-string
ALLOWED_ORIGINS=http://localhost:3000

# Model Configuration
MODEL_VERSION=v2.3.1
DATA_VERSION=snapshot_20251220
```

Step 3: Initialize Database

```bash
# Create database
createdb jackpot_db

# Run migrations (if using Alembic)
alembic upgrade head

# Or run initialization script
python scripts/init_db.py
```

Step 4: Download Initial Data

```bash
# Option 1: Download from Football-Data.co.uk (free)
python scripts/download_historical_data.py \
  --leagues "EPL,LaLiga,Bundesliga" \
  --seasons "2020-21,2021-22,2022-23,2023-24,2024-25" \
  --source csv

# Option 2: Use API-Football (requires API key)
python scripts/download_historical_data.py \
  --leagues "39,140,78" \
  --seasons "2020,2021,2022,2023,2024" \
  --source api
```

Step 5: Train Initial Model

```bash
```

```
# Train Dixon-Coles model
python scripts/train_model.py \
  --model dixon-coles \
  --leagues all \
  --decay-factor 0.0065

# This will output:
# - Team strength parameters → /models/team_strengths.pkl
# - Model metadata → /models/model_v2.3.1.json
# - Validation metrics → /models/validation_report.json
```

Step 6: Start Backend Services

bash

```
# Terminal 1: Start API server
uvicorn app.main:app --reload --host 0.0.0.0 --port 8000

# Terminal 2: Start Celery worker (for background tasks)
celery -A app.tasks worker --loglevel=info

# Terminal 3: Start Celery beat (for scheduled tasks)
celery -A app.tasks beat --loglevel=info
```

Backend should now be available at `http://localhost:8000`

Frontend Setup

Step 1: Install Dependencies

bash

```
cd jackpot-frontend

# Install npm packages
npm install
```

Step 2: Configure Environment

Create `.env` file in frontend directory:

```
env

VITE_API_BASE_URL=http://localhost:8000/api/v1
VITE_ENV=development
```

Step 3: Start Development Server

```
bash

npm run dev
```

Frontend should now be available at `http://localhost:3000`

Running Locally

Complete System Startup

Option 1: Manual (3 terminals for backend + 1 for frontend)

```
bash

# Terminal 1: Backend API
cd jackpot-backend
source venv/bin/activate
uvicorn app.main:app --reload --port 8000

# Terminal 2: Celery Worker
cd jackpot-backend
source venv/bin/activate
celery -A app.tasks worker --loglevel=info

# Terminal 3: Celery Beat
cd jackpot-backend
source venv/bin/activate
celery -A app.tasks beat --loglevel=info

# Terminal 4: Frontend
cd jackpot-frontend
npm run dev
```

Option 2: Docker Compose (recommended)

Create `docker-compose.yml`:

```yaml
```

```yaml
version: '3.8'

services:
  postgres:
    image: postgres:15
    environment:
      POSTGRES_DB: jackpot_db
      POSTGRES_USER: jackpot_user
      POSTGRES_PASSWORD: secure_password
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data

  redis:
    image: redis:7-alpine
    ports:
      - "6379:6379"

  backend:
    build: ./jackpot-backend
    ports:
      - "8000:8000"
    environment:
      DATABASE_URL: postgresql://jackpot_user:secure_password@postgres:5432/jackpot_db
      REDIS_URL: redis://redis:6379/0
    depends_on:
      - postgres
      - redis

  celery-worker:
    build: ./jackpot-backend
    command: celery -A app.tasks worker --loglevel=info
    environment:
      DATABASE_URL: postgresql://jackpot_user:secure_password@postgres:5432/jackpot_db
      REDIS_URL: redis://redis:6379/0
    depends_on:
      - postgres
      - redis

  frontend:
    build: ./jackpot-frontend
    ports:
```

```
    - "3000:80"
  depends_on:
    - backend

volumes:
  postgres_data:
```

Start everything:

```bash
docker-compose up
```

Testing the System

1. Verify Backend Health

```bash
curl http://localhost:8000/health
# Expected: {"status": "healthy"}

curl http://localhost:8000/api/v1/model/status
# Expected: Model version and metrics
```

2. Test Prediction Endpoint

```bash
```

```
curl -X POST http://localhost:8000/api/v1/predictions \
  -H "Content-Type: application/json" \
  -d '{
    "fixtures": [
      {
        "id": "1",
        "homeTeam": "Arsenal",
        "awayTeam": "Chelsea",
        "odds": {"home": 2.10, "draw": 3.40, "away": 3.20}
      }
    ],
    "createdAt": "2025-12-28T14:00:00Z"
  }'
```

3. Test Frontend

1. Open `http://localhost:3000`
2. Navigate to "Jackpot Input"
3. Add a fixture manually
4. Click "Generate Predictions"
5. Verify probabilities are displayed correctly

4. End-to-End Test

Scenario: Create a 5-match jackpot, compare probability sets, export results

1. **Input Section**: Enter 5 fixtures with odds
2. **Generate**: Click "Generate Predictions"
3. **Output Section**: Verify probabilities sum to 100%
4. **Comparison**: Toggle Sets A, B, C - verify differences
5. **Calibration**: Check reliability curves render
6. **Export**: Download CSV - verify file contents

Production Deployment

Backend Deployment (AWS Example)

Step 1: Prepare for Production

```
bash

# Update requirements.txt with production packages
pip install gunicorn psycopg2-binary

# Create production .env
DATABASE_URL=postgresql://...  # AWS RDS
REDIS_URL=redis://...          # AWS ElastiCache
SECRET_KEY=...                 # Generate secure key
ALLOWED_ORIGINS=https://yourdomain.com
```

Step 2: Deploy to AWS Elastic Beanstalk

```
bash

# Initialize EB
eb init -p python-3.11 jackpot-backend

# Create environment
eb create jackpot-production \
  --database.engine postgres \
  --database.version 15 \
  --envvars DATABASE_URL=...,REDIS_URL=...

# Deploy
eb deploy
```

Step 3: Setup Celery with AWS ECS

Create ECS task definitions for:

- Celery worker
- Celery beat

Configure Auto Scaling based on queue depth.

Frontend Deployment (Vercel Example)

Step 1: Connect to Vercel

```bash
# Install Vercel CLI
npm i -g vercel

# Login
vercel login

# Deploy
cd jackpot-frontend
vercel --prod
```

Step 2: Configure Environment

In Vercel dashboard:

- Set `VITE_API_BASE_URL` to your backend URL
- Enable automatic deployments from `main` branch

Alternative: Static Hosting (S3 + CloudFront)

```bash
# Build production bundle
npm run build

# Upload to S3
aws s3 sync dist/ s3://your-bucket-name/ \
  --acl public-read \
  --cache-control max-age=31536000

# Create CloudFront distribution
# (Use AWS Console or CloudFormation)
```

Monitoring & Maintenance

Application Monitoring

```bash
# Install monitoring tools
pip install prometheus-client
npm install @sentry/react

# Configure in production
SENTRY_DSN=...
PROMETHEUS_PORT=9090
```

Database Backups

```bash
# Automated PostgreSQL backups (cron job)
0 2 * * * pg_dump jackpot_db > /backups/jackpot_$(date +\%Y\%m\%d).sql
```

Model Retraining Schedule

```python
# Celery Beat schedule (app/tasks.py)
@app.task
def weekly_model_retrain():
    """Retrain model every Sunday at 2 AM"""
    train_model(data_version='latest')

# Add to beat schedule:
app.conf.beat_schedule = {
    'retrain-weekly': {
        'task': 'app.tasks.weekly_model_retrain',
        'schedule': crontab(hour=2, minute=0, day_of_week=0),
    },
}
```

Troubleshooting

Common Issues

1. Backend won't start

```bash
# Check PostgreSQL connection
psql -U jackpot_user -d jackpot_db -h localhost

# Check Redis
redis-cli ping
# Should return: PONG

# Check Python dependencies
pip list | grep -E "fastapi|uvicorn|sqlalchemy"
```

2. Frontend can't connect to backend

```bash
# Verify CORS settings in backend
# In app/main.py:
app.add_middleware(
    CORSMiddleware,
    allow_origins=["http://localhost:3000"],
    allow_methods=["*"],
    allow_headers=["*"],
)

# Check browser console for CORS errors
```

3. Predictions return NaN or invalid probabilities

```bash
```

```bash
# Check model file exists
ls -la models/team_strengths.pkl

# Retrain model
python scripts/train_model.py --force

# Check validation metrics
cat models/validation_report.json
```

4. Celery tasks not processing

```bash
# Check Celery worker logs
celery -A app.tasks worker --loglevel=debug

# Inspect Redis queue
redis-cli
> LLEN celery

# Purge stuck tasks
celery -A app.tasks purge
```

Performance Optimization

Backend

```python
# Add caching to prediction endpoint
from functools import lru_cache

@lru_cache(maxsize=1000)
def cached_prediction(fixture_hash):
    return generate_prediction(fixtures)
```

Frontend

```javascript
```

```
// Lazy load sections
const CalibrationDashboard = React.lazy(() =>
  import('./components/sections/CalibrationDashboard')
);

// Memoize expensive calculations
const overallProbability = useMemo(() =>
  calculateJackpotWinProbability(probabilities),
  [probabilities]
);
```

Database

```
sql

-- Add indexes for common queries
CREATE INDEX idx_matches_league_season ON matches(league_id, season);
CREATE INDEX idx_matches_date ON matches(match_date DESC);
```

Security Checklist

- [ ] Change all default passwords
- [ ] Enable HTTPS (SSL certificates)
- [ ] Set up firewall rules (only expose 80/443)
- [ ] Implement rate limiting
- [ ] Add API authentication (JWT tokens)
- [ ] Enable database encryption at rest
- [ ] Set up DDoS protection (Cloudflare)
- [ ] Regular security updates (dependabot)
- [ ] Implement CSP headers
- [ ] Set up logging and monitoring

Next Steps

1. **Day 1-2**: Complete local setup and testing
2. **Day 3-4**: Deploy backend to staging environment

3. **Day 5**: Deploy frontend to Vercel/S3
4. **Week 2**: Load test with 1000+ concurrent users
5. **Week 3**: Set up monitoring and alerts
6. **Week 4**: User acceptance testing
7. **Month 2**: Production launch

Support & Resources

Documentation

- Backend API: `/docs` endpoint (Swagger UI)
- Frontend: `jackpot-frontend/README.md`
- Architecture: `jackpot_system_architecture.md`

Data Sources

- Football-Data.co.uk: https://www.football-data.co.uk
- API-Football: https://www.api-football.com/documentation

Community

- Stack Overflow: Tag `dixon-coles` or `sports-betting`
- GitHub Discussions: Enable for Q&A

Remember: This is a production-grade system. Test thoroughly in staging before deploying to production. Monitor closely for the first month after launch.