PainCare AI Model

Advanced AI-powered endometriosis pain management and prediction system with explainable AI capabilities.

Overview

PainCare AI is a sophisticated machine learning system designed specifically for endometriosis patients, providing real-time pain predictions, personalized treatment recommendations, and explainable AI insights. The system integrates with Firebase for real-time data synchronization and provides a production-ready REST API for mobile applications.

Key Capabilities

- **Pain Level Prediction**: ML-powered forecasting using Random Forest algorithms
- **Treatment Recommendations**: Personalized suggestions via K-Means clustering
- **Symptom Pattern Analysis**: Advanced temporal analysis with Gradient Boosting
- **Explainable AI (XAI)**: SHAP and LIME-based model explanations
- **Real-time Integration**: Firebase-powered live data synchronization
- **Evidence-based Insights**: Integration with medical research databases
- **Production-ready API**: FastAPI with async endpoints and comprehensive error handling

Architecture

System Architecture

```
Mobile App REST API AI Engine
(React N.) (FastAPI) (ML Models)

Firebase
(Real-time DB)
```

Model Pipeline

```
Raw Data Feature Engineering ML Models XAI Layer API Response

Symptoms 37+ Features 3 Algorithms SHAP/LIME JSON
Diagnostics Temporal RF, KMeans, Feature Predictions
User Data Patterns GradBoost Importance Explanations
```

Machine Learning Models

1. Pain Prediction Model

- **Algorithm**: Random Forest Classifier
- **Features**: 37+ engineered features including temporal patterns, symptom combinations
- **Accuracy**: 85%+ on test data
- **Use Case**: Predicts pain levels 1-7 days ahead

2. Treatment Recommendation Engine

- **Algorithm**: K-Means Clustering + Collaborative Filtering
- **Silhouette Score**: 0.25
- **Features**: Treatment history, symptom patterns, user preferences
- **Use Case**: Personalized treatment suggestions

3. Symptom Analysis Model

- **Algorithm**: Gradient Boosting Regressor
- **R^2 Score**: 0.63
- **Features**: Temporal symptom data, external factors
- **Use Case**: Pattern recognition and trend analysis

4. Explainable AI (XAI) Layer

- **SHAP Values**: Feature importance for individual predictions
- **LIME**: Local model explanations
- **Feature Importance**: Global model insights
- **Use Case**: Transparent AI decision-making

Features & Data Processing

Core Features (37+)

PYTHON:

```
# Symptom Features
- pain_level, sleep_hours, energy_level, mood
- stress_level, physical_activity, medication_taken

# Temporal Features
- day_of_week, hour_of_day, days_since_period
- symptom_trend_3d, symptom_trend_7d

# External Factors
- weather_pressure, temperature, humidity
- medication_effectiveness, treatment_adherence

# Engineered Features
- pain_severity_score, quality_of_life_index
- symptom_pattern_stability, medication_response_rate
```

Data Augmentation

- **Synthetic Data Generation**: SMOTE for balanced datasets
- **Temporal Augmentation**: Time-series data enhancement
- **Feature Scaling**: StandardScaler for numerical features
- **Categorical Encoding**: LabelEncoder for categorical data

Quick Start

Prerequisites

BASH:

```
Python 3.8+
Firebase Account
8GB RAM (recommended)
```

Installation

BASH:

```
# Clone repository
git clone <repository-url>
cd PainCare_Model

# Create virtual environment
python -m venv .venv
source .venv/bin/activate # Windows: .venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt

# Configure environment
cp .env.example .env
# Edit .env with your Firebase credentials
```

Development Setup

BASH:

```
# Start development server
python run_server.py

# Or with uvicorn directly
uvicorn src.api.main:app --host 0.0.0.0 --port 8000 --reload

# API will be available at: http://localhost:8000
# Interactive docs: http://localhost:8000/docs
```

Production Deployment

1. Docker Deployment

Create `Dockerfile`:

DOCKERFILE:

```
FROM python:3.11-slim
```

```
WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 8000

CMD ["uvicorn", "src.api.main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Create `docker-compose.yml`:

YAML:

```
version: '3.8'
services:
paincare-ai:
build: .
ports:
- "8000:8000"
environment:
- FIREBASE_SERVICE_ACCOUNT_PATH=/app/firebase-service-account.json
- API_HOST=0.0.0.0
- API_PORT=8000
volumes:
- ./firebase-service-account.json:/app/firebase-service-account.json:ro
restart: unless-stopped
```

Deploy:

BASH:

```
docker-compose up -d
```

2. Cloud Deployment (AWS/GCP/Azure)

AWS Elastic Beanstalk

BASH:

```
# Install EB CLI
pip install awsebcli

# Initialize and deploy
eb init paincare-ai
eb create production
```

```
eb deploy
```

Google Cloud Run

BASH:

```
# Build and deploy
gcloud builds submit --tag gcr.io/YOUR_PROJECT/paincare-ai
gcloud run deploy --image gcr.io/YOUR_PROJECT/paincare-ai --platform managed
```

Azure Container Instances

BASH:

```
# Create resource group
az group create --name paincare-ai --location eastus

# Deploy container
az container create \
--resource-group paincare-ai \
--name paincare-ai-api \
--image your-registry/paincare-ai:latest \
--dns-name-label paincare-ai \
--ports 8000
```

3. Kubernetes Deployment

Create `k8s-deployment.yaml`:

YAML:

```
apiVersion: apps/v1
kind: Deployment
metadata:
name: paincare-ai
spec:
replicas: 3
selector:
matchLabels:
app: paincare-ai
template:
metadata:
labels:
app: paincare-ai
spec:
containers:
```

```
- name: paincare-ai
image: your-registry/paincare-ai:latest
ports:
- containerPort: 8000
env:
- name: API_HOST
value: "0.0.0.0"
resources:
requests:
memory: "512Mi"
cpu: "250m"
limits:
memory: "1Gi"
cpu: "500m"
---
apiVersion: v1
kind: Service
metadata:
name: paincare-ai-service
spec:
selector:
app: paincare-ai
ports:
- protocol: TCP
port: 80
targetPort: 8000
type: LoadBalancer
```

Deploy:

BASH:

```
kubectl apply -f k8s-deployment.yaml
```

API Reference

Endpoints

Health Check

HTTP:

```
GET /health
```

Pain Prediction

HTTP:

```
POST /predict/pain
Content-Type: application/json

{
"symptoms": {
"pain_level": 6,
"sleep_hours": 5,
"stress_level": 8,
"energy_level": 3,
"mood": 4
},
"include_explanation": true
}
```

Treatment Recommendations

HTTP:

```
POST /recommend/treatment
Content-Type: application/json

{
"symptoms": {...},
"medical_history": {...},
"preferences": {...}
}
```

Model Status

HTTP:

```
GET /model/status
```

XAI Explanations

HTTP:

```
POST /explain/{prediction_id}
```

Response Format

JSON:

```
{
"success": true,
"data": {
"prediction": 6.2,
"confidence": 0.87,
"explanation": {
"top_features": ["stress_level", "sleep_hours"],
"shap_values": {...}
}
},
"timestamp": "2025-09-02T10:30:00Z",
"model_version": "1.0.0"
}
```

Security & Authentication

API Security

PYTHON:

```
# JWT Authentication
from fastapi.security import HTTPBearer
security = HTTPBearer()

# Rate Limiting
from slowapi import Limiter
limiter = Limiter(key_func=get_remote_address)

# CORS Configuration
app.add_middleware(
CORSMiddleware,
allow_origins=["https://yourapp.com"],
allow_credentials=True,
allow_methods=["GET", "POST"],
allow_headers=["*"],
)
```

Environment Variables

ENV:

```
# Firebase
FIREBASE_SERVICE_ACCOUNT_PATH=/path/to/service-account.json

# API Security
SECRET_KEY=your-super-secret-key-here
ALGORITHM=HS256
ACCESS_TOKEN_EXPIRE_MINUTES=30

# Production Settings
DEBUG_MODE=False
LOG_LEVEL=INFO
```

Monitoring & Observability

Health Checks

PYTHON:

```
@app.get("/health")
async def health_check():
return {
"status": "healthy",
"model_loaded": ai_model.is_trained,
"firebase_connected": firebase_service.is_connected(),
"timestamp": datetime.now().isoformat()
}
```

Logging

PYTHON:

```
import logging

logging.basicConfig(
level=logging.INFO,
format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
handlers=[
logging.FileHandler('paincare_ai.log'),
logging.StreamHandler()
]
)
```

Metrics Collection

- **Prometheus**: Custom metrics for model performance
- **Grafana**: Dashboards for monitoring
- **Sentry**: Error tracking and performance monitoring

Testing

Unit Tests

BASH:

```
# Run all tests
pytest tests/

# Run with coverage
pytest tests/ --cov=src --cov-report=html
```

Load Testing

BASH:

```
# Install locust
pip install locust

# Run load tests
locust -f tests/load_test.py --host http://localhost:8000
```

Model Validation

PYTHON:

```
# Cross-validation
scores = cross_val_score(model, X, y, cv=5)

# Performance metrics
from sklearn.metrics import classification_report
print(classification_report(y_true, y_pred))
```

Performance Optimization

Model Optimization

- **Model Pruning**: Remove unnecessary features
- **Quantization**: Reduce model size for deployment
- **Caching**: Redis for frequent predictions
- **Batch Processing**: Handle multiple predictions

API Optimization

PYTHON:

```
# Async endpoints
@app.post("/predict/pain")
async def predict_pain(request: PainPredictionRequest):
result = await ai_model.predict_async(request.symptoms)
return result

# Response caching
from fastapi_cache import FastAPICache
from fastapi_cache.backends.redis import RedisBackend

@cache(expire=300) # 5 minutes
async def get_cached_prediction():
return await model.predict(data)
```

CI/CD Pipeline

GitHub Actions

YAML:

```
name: CI/CD Pipeline

on:
push:
```

```
branches: [ main ]
pull_request:
branches: [ main ]

jobs:
test:
runs-on: ubuntu-latest
steps:
- uses: actions/checkout@v2
- name: Set up Python
uses: actions/setup-python@v2
with:
python-version: 3.11
- name: Install dependencies
run: pip install -r requirements.txt
- name: Run tests
run: pytest tests/

deploy:
needs: test
runs-on: ubuntu-latest
if: github.ref == 'refs/heads/main'
steps:
- name: Deploy to production
run: |
# Your deployment script here
docker build -t paincare-ai .
docker push your-registry/paincare-ai:latest
```

Model Management

Model Versioning

PYTHON:

```
# Save model with version
joblib.dump(model, f'models/paincare_v{VERSION}.joblib')

# Model registry
class ModelRegistry:
def __init__(self):
self.models = {}

def register_model(self, name: str, version: str, model):
self.models[f"{name}_v{version}"] = {
'model': model,
'timestamp': datetime.now(),
'metrics': self.evaluate_model(model)
}
```

A/B Testing

PYTHON:

```
@app.post("/predict/ab_test")
async def ab_test_prediction(request: PredictionRequest):
# Route 50% to new model, 50% to current
if hash(request.user_id) % 2 == 0:
return await new_model.predict(request)
else:
return await current_model.predict(request)
```

Documentation

API Documentation

- **Swagger UI**: http://localhost:8000/docs
- **ReDoc**: http://localhost:8000/redoc

Model Documentation

PYTHON:

```
# Model cards for transparency
model_card = {
"model_details": {
"name": "PainCare Pain Predictor",
"version": "1.0.0",
"type": "Random Forest Classifier"
},
"intended_use": {
"primary_uses": "Endometriosis pain prediction",
"primary_users": "Healthcare providers, patients"
},
"metrics": {
"accuracy": 0.85,
"precision": 0.83,
"recall": 0.87
}
}
```

Contributing

1. **Fork the repository**
2. **Create feature branch**: `git checkout -b feature/amazing-feature`
3. **Commit changes**: `git commit -m 'Add amazing feature'`
4. **Push to branch**: `git push origin feature/amazing-feature`
5. **Open Pull Request**

Development Guidelines

- Follow PEP 8 style guide
- Add tests for new features
- Update documentation
- Use type hints
- Add docstrings for all functions

License

This project is licensed under the MIT License - see the LICENSE file for details.

Support

Troubleshooting

- **Model not loading**: Check Firebase credentials
- **Slow predictions**: Enable model caching
- **Memory issues**: Reduce batch size

Contact

- **Email**: support@paincare.ai
- **Issues**: GitHub Issues
- **Documentation**: Wiki

Built with for endometriosis patients worldwide