FastAPI with ML Integration

Building an Intelligent RSS News Reader

Advanced Computer Programming

DR. MOHAMMED AL-HUBAISHI

What is FastAPI?

🚀 Modern, fast web framework for building APIs with Python 3.7+

⚡ High performance - comparable to NodeJS and Go

📝 Automatic interactive API documentation (Swagger UI)

✅ Built-in data validation using Pydantic

🔄 Async/await support for concurrent operations

🎯 Type hints for better IDE support and fewer bugs

Our Project: RSS Reader + ML

RSS Feed Parser - Fetches news from multiple sources

Machine Learning Integration - Analyzes content intelligently

Sentiment Analysis - Determines article tone (positive/negative)

Topic Classification - Categorizes into topics automatically

Aggregate Analytics - Overall feed insights and trends

AI Summarization - Generates concise article summaries

https://techcrunch.com/feed/

System Architecture

Layer 1: FastAPI Framework - HTTP requests and routing

Layer 2: RSS Parser (feedparser) - Fetches and parses feeds

Layer 3: ML Models - Transformers (HuggingFace)

- DistilBERT - Sentiment analysis
- BART - Summarization and classification

Layer 4: Response Models (Pydantic) - Data validation

Layer 5: JSON API - Returns structured insights

FastAPI Basics

Simple endpoints with automatic type validation

```
from fastapi import FastAPI

app = FastAPI(title="My API")

@app.get("/")
async def root():
    return {"message": "Hello World"}

@app.get("/items/{item_id}")
async def read_item(item_id: int):
    return {"item_id": item_id}
```

Data Validation with Pydantic

Type-safe models with automatic validation

```
from pydantic import BaseModel, HttpUrl

class RSSItem(BaseModel):
    title: str
    link: str
    published: Optional[str] = None
    summary: Optional[str] = None
    sentiment: Optional[Dict] = None
    topic: Optional[Dict] = None

# FastAPI automatically validates incoming data
```

Loading ML Models

Models loaded once at startup for efficiency

```
from transformers import pipeline

class MLModels:
    def __init__(self):
        self.sentiment_analyzer = pipeline(
            "sentiment-analysis",
            model="distilbert-base-uncased"
        )
        self.summarizer = pipeline(
            "summarization",
            model="facebook/bart-large-cnn"
        )

@app.on_event("startup")
async def load_models():
    global ml_models
    ml_models = MLModels()
```

Sentiment Analysis Endpoint

Real-time sentiment analysis on RSS articles

```
@app.get("/rss")
async def get_rss_feed(url: HttpUrl, analyze: bool = True):
    feed = feedparser.parse(str(url))
    items = []

    for entry in feed.entries:
        text = entry.get('summary', '')

        # Analyze sentiment
        sentiment = ml_models.analyze_sentiment(text)

        items.append({
            "title": entry.get('title'),
            "sentiment": sentiment  # POSITIVE/NEGATIVE
        })

    return {"items": items}
```

API Response Format

Structured JSON response with ML insights


```
{
 "feed_title": "TechCrunch",
 "items": [{
  "title": "New AI Breakthrough Announced",
  "link": "https://...",
  "sentiment": {
   "label": "POSITIVE",
   "score": 0.98
  },
  "topic": {
   "topic": "Technology",
   "confidence": 0.95
  }
 }],
 "analysis_summary": {
  "sentiment_distribution": {
   "positive": 8, "negative": 2
  },
  "top_topics": ["Technology", "AI"]
 }
}
```

Key Features & Benefits

🎯 Automatic Documentation - Interactive Swagger UI at /docs

⚡ Async Performance - Handle multiple requests concurrently

🔒 Type Safety - Catch errors before runtime

🤖 ML-Powered - Intelligent content analysis

📊 Analytics - Aggregate insights across articles

🔌 Flexible - Easy to add new ML models or features

🚀 Production Ready - Built-in validation and error handling

Available API Endpoints

GET /rss - Single feed with full ML analysis

Parameters: url, limit, analyze, summarize

GET /rss/simple - Fast fetch without ML analysis

Parameters: url, limit

POST /rss/multiple - Multiple feeds at once

Parameters: urls[], limit, analyze

GET /docs - Interactive API documentation

GET / - API information and endpoint list

Using the API

Simple to deploy and use

```
# Start the server
uvicorn main:app --reload

# Make requests
curl "http://localhost:8000/rss?\
  url=https://techcrunch.com/feed/&\
  limit=10&\
  analyze=true&\
  summarize=true"

# Result: News articles with:
# - Sentiment scores
# - Topic classifications
# - AI-generated summaries
# - Aggregate analytics
```

Performance & Optimization

🎯 Model Loading - Load once at startup (not per request)

⚡ Async Operations - Non-blocking I/O for RSS fetching

🔄 Caching Strategy - Cache feed results for faster responses

📏 Text Truncation - Limit input to 512 tokens for speed

⚙️ Optional Analysis - Toggle ML features per request

💾 Batch Processing - Analyze multiple articles together

🚀 GPU Support - Use CUDA for faster ML inference

Summary & Next Steps

✓ FastAPI provides modern, fast API development

 Easy integration with ML models for intelligent features

Built-in documentation and validation save development time

Production-ready with async support and error handling

Next Steps:

- Add more ML models (entity extraction, language detection)
- Implement caching with Redis
- Add user authentication and saved feeds
- Deploy to cloud (AWS, GCP, Azure)

Machine Learning for Precision Medicine: Optimizing Treatment Selection and Reducing Diagnostic Errors

https://youtu.be/Bk3apQKMf3g

Mohanad Alayedi ; Mohammed Al-Hubaishi All Authors

https://ieeexplore.ieee.org/abstract/document/11101823

Abstract:

In recent years, integrating machine learning (ML) techniques into healthcare settings has demonstrated significant potential for improving patient results. A particularly promising application lies in enhancing disease diagnosis and treatment selection. Despite advancements in medical science, diagnostic errors and suboptimal treatment decisions persist as significant healthcare challenges, often leading to poor patient outcomes and increased costs. This research explores how ML algorithms can enhance diagnostic precision and assist physicians in selecting optimal treatment strategies tailored to individual patients.Our objective is to

TABLE I
AN OVERVIEW OF VARIOUS DISEASES, INCLUDING THEIR SYMPTOMS, CAUSES, AND TREATMENTS. EACH DISEASE IS ACCOMPANIED BY RELEVANT CITATIONS.

Disease	Symptoms	Diagnosis Method	Causes	Category	Treatment
Diabetes [19]	Increased thirst, frequent urination [20]	Blood test [21]	Insulin resistance [21]	Metabolic Disease [2]	Metformin, Insulin therapy [20]
Influenza [22]	Fever, cough, sore throat, muscle aches [19]	Influenza test [23], [23]	Influenza virus [24]	Infectious Disease [11]	Antiviral medications [25]
Hypertension [25]	High blood pressure, headache [26]	Blood pressure test [26]	Genetics, lifestyle [2]	Cardiovascular [3]	Beta blockers, ACE inhibitors [27]
Malaria [3]	Fever, chills, flu-like symptoms [1]	Blood test [21]	Plasmodium parasite [10]	Infectious Disease [11]	Antimalarial medication [25]
Tuberculosis [8]	Cough, fever, weight loss [11]	Chest X-ray, sputum test [2]	Mycobacterium tuberculosis [11]	Infectious Disease [11]	Antibiotics [26]
Asthma [4]	Wheezing, cough, shortness of breath [28]	Spirometry [19]	Genetics, allergies [5]	Respiratory Disease [5]	Inhalers, corticosteroids [22]
Chronic Kidney Disease [1]	Fatigue, swelling, decreased urine output [10]	Blood test, urine test [11]	Diabetes, hypertension [19]	Renal Disease [4]	Dialysis, kidney transplant [29]
Osteoporosis [26]	Back pain, fractures, loss of height [30]	Bone density test [27]	Genetics, lifestyle [2]	Musculoskeletal Disease [20]	Bisphosphonates, calcium supplements [16]
Alzheimer's Disease [27]	Memory loss, confusion, difficulty with daily activities [21]	Cognitive test, brain scan [8]	Genetics, lifestyle [2]	Neurological Disease [26]	Cholinesterase inhibitors, memantine [10]
Parkinson's Disease [21]	Tremors, rigidity, difficulty with movement [22]	Physical examination, brain scan [17]	Genetics, lifestyle [2]	Neurological Disease [26]	Dopamine agonists, levodopa [2]
Rheumatoid Arthritis [16]	Joint pain, swelling, stiffness [5]	Blood test, joint examination [10]	Genetics, autoimmune [31]	Musculoskeletal Disease [20]	DMARDs, biologics, corticosteroids [28]
Lupus [31]	Skin rash, joint pain, fever [31]	Blood test, skin biopsy [4]	Genetics, autoimmune [31]	Musculoskeletal Disease [20]	Corticosteroids, immunosuppressants [26]
Heart Failure [11]	Shortness of breath, fatigue, swelling [20]	Echocardiogram, blood test [20]	Coronary artery disease, hypertension [29]	Cardiovascular [3]	ACE inhibitors, beta blockers, diuretics [8]
Pneumonia [10]	Cough, fever, difficulty breathing [8]	Chest X-ray, blood test [20]	Bacteria, viruses [26]	Infectious Disease [11]	Antibiotics, antiviral medications [31]

```python
# Numeric data
data = {
    "Disease": [12, 5, 19, 3, 8, 4, 1, 7, 18, 21, 16, 14, 11, 10, 15, 24, 17, 23, 13, 6, 9, 2, 22, 20],
    "Symptoms": [6, 12, 7, 1, 19, 23, 10, 24, 13, 22, 5, 14, 20, 11, 16, 17, 9, 2, 4, 3, 21, 15, 18, 8],
    "Diagnosis_Method": [13, 15, 7, 13, 2, 12, 11, 18, 8, 17, 10, 4, 20, 6, 9, 14, 13, 10, 16, 1, 3, 5, 5, 19],
    "Causes": [13, 4, 2, 10, 11, 5, 12, 2, 2, 2, 14, 14, 9, 6, 14, 7, 7, 1, 14, 14, 8, 8, 14, 3],
    "Category": [2, 11, 3, 11, 11, 5, 4, 6, 7, 7, 6, 6, 3, 11, 7, 5, 9, 6, 10, 1, 8, 3, 9, 5],
    "Treatment": [20, 13, 18, 19, 6, 22, 9, 16, 10, 2, 23, 7, 8, 14, 24, 11, 15, 12, 21, 3, 17, 1, 4, 5]
}
```

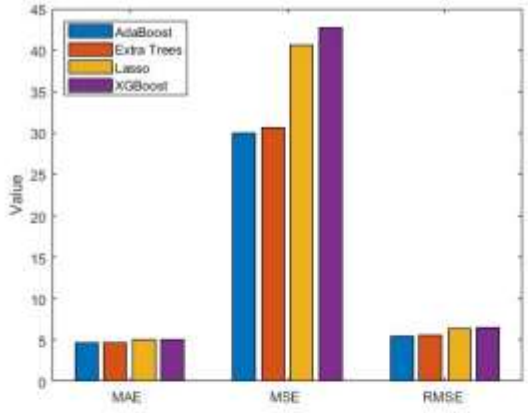

Fig. 1. Performance comparison of four commonly used ML classifiers.

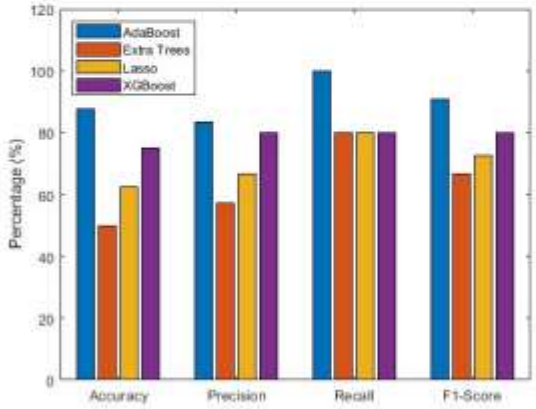

Fig. 2. Performance evaluation of four frequently employed ML classifiers.

analysis to assess how well a model explains the variability of the dependent outcome based on the predictors [18]. It can be calculated as shown below in Equation 8.

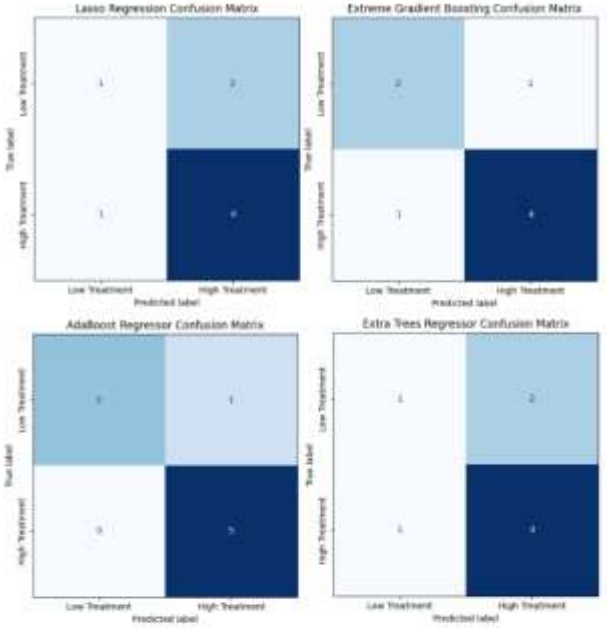

Fig. 3. Confusion matrices for four different models.

Extra Trees Regressor also demonstrated solid performance, with slightly higher errors than AdaBoost but a similarly positive R^2 score. This suggests that while it's not the best, it still performs reliably in this context.

On the other hand, Lasso Regression and XGBoost struggled to generalize well. Both models recorded higher error values and negative R^2 scores, meaning their predictions were less reliable and often worse than simply predicting the mean. This performance gap is evident in Figure 1, where Lasso and XGBoost exhibit the highest MSE and RMSE values. Meanwhile, classification-based performance of accuracy, recall, precision, and F1-score is shown in Figure 2. Again, AdaBoost stands out with near-perfect recall and the highest overall scores, reinforcing its strength not only in regression but also when applied in a classification context.

- `GET /` - API documentation
- `GET /health` - Health check
- `GET /model/accuracy` - Quick accuracy check
- `GET /model/metrics` - Detailed performance metrics
- `POST /predict` - Single prediction with confidence scores
- `POST /predict/batch` - Batch predictions
- `POST /model/retrain` - Retrain the model

Thank You!

Questions?