**Machine Learning Model Details**

**Primary Model: Random Forest Regressor**

The Mental Fatigue Detector uses a Random Forest Regressor as its primary prediction model. This choice was made for several key reasons:

**Model Configuration**

train\_with\_real\_data.py

Copy

rf\_model = RandomForestRegressor(

    n\_estimators=100,

    max\_depth=10,

    min\_samples\_split=5,

    min\_samples\_leaf=2,

    random\_state=42,

    n\_jobs=-1

)



**Training Data**

* **410 real samples**: The model is trained on genuine behavioral data rather than synthetic data
* **Data sources**:
  + Keyboard data (100 samples)
  + Mouse data (22 sessions expanded to ~110 samples)
  + Facial data (200 samples selected from 2,900 facial images)

**Feature Set (14 behavioral indicators)**

1. typing\_speed
2. error\_rate
3. pause\_frequency
4. key\_press\_duration
5. movement\_speed
6. click\_frequency
7. eye\_blink\_rate
8. eye\_closure\_duration
9. speech\_rate
10. pitch\_variation
11. volume
12. clarity
13. hour\_of\_day
14. day\_of\_week

**Feature Importance**

The top 5 most predictive features:

1. eye\_closure\_duration: 43.65%
2. click\_frequency: 26.61%
3. eye\_blink\_rate: 9.72%
4. movement\_speed: 3.79%
5. typing\_speed: 2.40%

**Performance Metrics**

* **Training accuracy**: 89% (R² score of 0.890)
* **Test accuracy**: 76.6% (R² score of 0.766)
* **Training MSE(Mean Squared Error)**: 0.0084
* **Test MSE**: 0.0170

**Advantages of Random Forest**

1. **Interpretability**: Provides feature importance analysis
2. **Robustness**: Less prone to overfitting than neural networks
3. **Efficiency**: Works well with the available dataset size
4. **Handles mixed data types**: Can process various behavioral indicators

**Fallback Model: Neural Network (4-layer deep network)**

A neural network serves as a fallback when the Random Forest model has low confidence:

**Architecture**

* **Input layer**: 14 nodes (one for each feature)
* **Hidden layer 1**: 64 neurons with ReLU activation and BatchNormalization
* **Dropout layer**: 30% dropout for regularization
* **Hidden layer 2**: 32 neurons with ReLU activation and BatchNormalization
* **Dropout layer**: 20% dropout
* **Hidden layer 3**: 16 neurons with ReLU activation
* **Output layer**: 1 neuron with linear activation (regression output)

**When It's Used**

The system falls back to this neural network when:

1. The Random Forest model has low confidence (below a threshold)
2. There are data quality issues with certain modalities
3. The system detects unusual patterns that might be outside the Random Forest's training distribution

**Training Process**

Both models are trained using the same dataset but with different validation approaches:

* Random Forest: Standard train-test split (80/20)
* Neural Network: K-fold cross-validation for better generalization

**Model Integration in the Application**

The models are integrated into the application through the

fatigue\_predictor.py

module:

1. **Data preprocessing**: Raw behavioral data is normalized using StandardScaler
2. **Feature extraction**: The system extracts the 14 behavioral indicators
3. **Prediction**: The Random Forest model predicts the fatigue score
4. **Confidence calculation**: The system calculates confidence based on data quality
5. **Fallback mechanism**: If confidence is low, the Neural Network is used
6. **Result combination**: The final prediction combines individual scores with confidence weights

This comprehensive ML approach ensures reliable fatigue detection across different user behaviors and data quality scenarios.

**Important Files in the Mental Fatigue Detector Project**

**Core Application Files**

1. **fatique/views.py**
   * Contains the main view functions for rendering pages and handling user interactions
   * Includes the dashboard view, data collection endpoints, and analysis functions
2. **fatique/models.py**
   * Defines database models for storing user profiles, behavioral data, and analysis results
   * Contains UserProfile, BehavioralData, KeyboardMetrics, MouseMetrics, FacialMetrics, VoiceMetrics, FatigueAnalysis, ProductivityRecommendation, and ProductivitySession models
3. **fatique/urls.py**
   * Maps URL patterns to view functions
   * Includes routes for the dashboard, tracking, fatigue analysis, and recommendations
4. **fatique/admin.py**
   * Configures the Django admin interface for managing application data
   * Registers models with custom admin classes for better organization

**Machine Learning Components**

1. **fatique/ml\_models/fatigue\_detector.py**
   * Implements the FatigueDetector class that handles prediction of fatigue levels
   * Contains methods for feature extraction, model loading, and prediction
2. **train\_with\_real\_data.py**
   * Script for training the Random Forest model with real datasets
   * Includes data preparation, model training, evaluation, and saving
3. **fatique/ml\_models/recommendation\_engine.py**
   * Generates personalized productivity recommendations based on fatigue analysis
   * Contains logic for selecting appropriate recommendations by fatigue level
4. **fatique/ml\_models/data\_preprocessor.py**
   * Handles preprocessing of raw behavioral data
   * Implements feature extraction and normalization using StandardScaler

**API Components**

1. **fatique/api/views.py**
   * Contains API endpoints for the REST interface
   * Includes viewsets for user profiles, behavioral data, and fatigue analysis
2. **fatique/api/serializers.py**
   * Defines serializers for converting model instances to JSON
   * Contains serializers for all models used in the API
3. **fatique/api/urls.py**
   * Configures URL routing for API endpoints
   * Sets up the DefaultRouter for REST framework viewsets

**Data Collection and Integration**

1. **fatique/data\_collection/data\_collector.py**
   * Implements the DataCollector class for gathering behavioral data
   * Contains methods for collecting keyboard, mouse, and facial data
2. **fatique/datasets/data\_integrator.py**
   * Handles integration of different data sources
   * Used by the management command for dataset integration
3. **fatique/management/commands/integrate\_datasets.py**
   * Django management command for integrating facial, mouse, and keyboard datasets
   * Used for initial setup and data preparation

**Frontend Templates**

1. **templates/index.html**
   * Main landing page template
   * Introduces the application and its features
2. **templates/dashboard/index.html**
   * Dashboard template with visualizations and analytics
   * Contains Chart.js integration for fatigue trends and component analysis
3. **templates/data\_collection.html**
   * Template for the data collection flow
   * Includes JavaScript for capturing keyboard, mouse, and facial data

**Configuration and Project Setup**

1. **mental\_fatique/settings.py**
   * Django settings file with configuration for the entire project
   * Includes database settings, installed apps, middleware, and security settings
2. **mental\_fatique/urls.py**
   * Main URL configuration for the project
   * Includes routes for the admin interface, API, and main application
3. **requirements.txt**
   * Lists all Python dependencies required for the project
   * Includes Django, scikit-learn, TensorFlow, OpenCV, and other libraries

**Static Assets and Resources**

1. **static/js/data\_collection.js**
   * JavaScript for the data collection process
   * Implements event listeners for keyboard and mouse tracking
2. **static/js/dashboard.js**
   * JavaScript for the dashboard functionality
   * Handles chart initialization and data updates
3. **static/css/main.css**
   * Main stylesheet for the application
   * Defines the visual style and responsive design

**Trained Models and Data**

1. **ml\_models/trained\_models/fatigue\_model\_real\_data.joblib**
   * Saved Random Forest model trained on real data
   * Used for production predictions
2. **ml\_models/trained\_models/feature\_scaler\_real\_data.joblib**
   * Saved StandardScaler for feature normalization
   * Ensures consistent scaling between training and prediction