Somatic Symptom Fatigue Project: Neural Network Analysis

Jessica Tanchone and Miao Yu

Invalid Date

Table of contents

1 Somatic Symptom Fatigue Project: Neural Network Analysis

2 Introduction

The primary objective of this project is to predict the presence or absence of fatigue (Symptom 12) based on a set of psychosocial and demographic predictors. Fatigue is a common somatic symptom that can have multiple underlying causes, including psychological stress, sleep disturbances, and other health-related factors. Identifying key predictors of fatigue can help researchers better understand its risk factors and potentially inform targeted interventions.

3 Literature review

Fatigue is often associated with significant reductions in quality of life and increased healthcare use. Prior research has identified multiple predictors of fatigue across clinical and community samples. In large population-based studies, fatigue has been significantly predicted by stressful life events, chronic illnesses, and psychological traits such as anxiety and neuroticism, with physical conditions explaining more variance than mental disorders alone (Creed, 2022). These findings underscore the multidimensional nature of fatigue etiology.

Symptom count has also been shown to be a reliable predictor of poor health outcomes and healthcare utilization, suggesting that modeling symptoms individually—rather than solely using aggregate diagnostic categories—may provide more nuanced insights (Tomenson et al., 2013). Clinical studies further highlight that fatigue, as a persistent somatic symptom (PSS), is influenced by both medical and psychosocial factors. For example, in cardiac populations, fatigue severity is shaped by gender, age, depressive and anxiety symptoms, and disease burden, reinforcing the need for a biopsychosocial perspective in understanding this symptom (Clifford et al., 2024).

Moreover, research from the SOMACROSS research unit has emphasized perceived stress as a modifiable risk factor for persistent somatic symptoms such as fatigue. Chronic stress exposure, particularly when perceived as uncontrollable, appears to exacerbate symptom experience and reporting (Löwe et al., 2022). Collectively, these studies demonstrate that fatigue is a complex, multifactorial symptom with overlapping biological, psychological, and social determinants.

3.0.1 Dataset

This analysis uses data from the Experience and Attitudes of Mental Health in International Students (EAMMi2) dataset published in the Journal of Open Psychology Data (Grahe et al., 2018). The dataset includes responses from international university students on a variety of psychological and demographic measures.

Sample: Over 900 participants from 31 universities in the UK, EU, and beyond. Variables: Mental health screening tools (e.g., PHQ-15, PHQ-9, GAD-7), help-seeking attitudes, demographic information, and psychosocial measures. Outcome variable: Symptom 12 (fatigue), from the Patient Health Questionnaire (PHQ-15), recoded into a binary variable indicating presence (1) or absence (0) of the symptom. This dataset is particularly valuable because it captures a broad range of psychosocial and demographic factors, making it suitable for predictive modeling of somatic symptoms such as fatigue.

3.1 Approach

Type of model: A feed-forward neural network is used for classification. The model is trained to predict a binary outcome: fatigue present (1) vs. fatigue absent (0).

Outcome variable: Symptom 12 (fatigue) is derived from the Patient Health Questionnaire (PHQ) dataset and is coded as a binary variable.

Predictors: Psychosocial (e.g., stress levels, emotional well-being) and demographic variables (e.g., age, sex, education) serve as predictors. These variables were selected based on prior evidence linking them to fatigue and related symptoms.

Evaluation Metrics: Model performance was assessed using accuracy, AUC, balanced accuracy, F1 score, and confusion matrix values to address class imbalance and prediction calibration

Model calibration and class imbalance were explicitly addressed during training through dropout regularization, early stopping, and metric selection. Results are interpreted using both performance metrics and SHAP values to identify the most influential predictors.

4 Data Preparation

```
# !pip install torch
# !pip install keras
```

```
# Core packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Display tools
from IPython.display import Markdown

# preprocessing and pipeline
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer
```

```
# models
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier

# feature selection
from sklearn.feature_selection import SelectKBest, f_classif

# metrics
from sklearn.metrics import (
    classification_report, roc_auc_score, confusion_matrix,
    RocCurveDisplay, accuracy_score, roc_curve, auc
)
from sklearn.metrics import accuracy_score, balanced_accuracy_score, f1_score, precision_score, recall_
from sklearn.impute import SimpleImputer

# neural networking
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
import os

import keras
from keras import layers, optimizers

import shap
```

```
!git clone https://J-Tanchone:github_pat_11BM6MAZY07YEkPYJuF6HO_1gidkLh7H4NfgdToNcnJf43
```

fatal: destination path 'somatic-symptom' already exists and is not an empty directory.

Table 1 shows a sample of the original dataset, which includes all recorded variables.

```
data = pd.read_excel("somatic-symptom/EAMMi2-Data1/EAMMi2-Data1.2.xlsx", sheet_name="EAM

data.sample(2)
```

	StartDate	EndDate	Status	Progress	Duration (in seconds)	Finished
1321	2016-11-28 12:45:27	2016-11-28 13:16:49	0	99	1882	0
2047	2016-10-25 17:10:45	2016-10-25 18:09:58	0	100	3552	1

This data has 328 columns and 3182 rows.

The EAMMi2 (Emerging Adulthood Measured at Multiple Institutions) dataset is a publicly available multi-site study involving over 3,000 undergraduate students across more than 30 institutions. It includes responses to a wide array of validated psychological and social questionnaires, such as the Patient Health Questionnaire (PHQ), Mindfulness scale, Perceived Stress Questionnaire, and others. The dataset also captures detailed demographic information, allowing for rich exploratory and predictive modeling of psychosocial constructs.

```
data.shape
```

(3182, 328)

```
codebook = pd.read_excel("somatic-symptom/EAMMi2-Data1/EAMMi2-Data1.2-Codebook.xlsx").set_i
```

The variables listed in the codebook Table 2 represent composite measures that I constructed from raw item-level responses within the EAMMi2 dataset. These variables were designed to capture a range of psychological and demographic characteristics theoretically relevant to the experience of somatic symptoms. Psychological predictors include mean scores from validated self-report instruments, such as identity development (IDEA-8), perceived stress, social support, mindfulness, self-efficacy, need to belong, and subjective well-being. Additional variables reflect interpersonal dimensions, including narcissistic traits (NPI-13), interpersonal exploitativeness, and transgressions. Social media use was assessed through three dimensions: maintaining connections, forming new connections, and information seeking.

Demographic and attitudinal variables include disability identity, belief in the American dream, and the importance placed on marriage. I also recoded two categorical variables, parental marital status and sibling presence, to facilitate inclusion in statistical models. This curated and theoretically grounded set of predictors allows for a nuanced investigation of individual differences in somatic symptom reporting, particularly fatigue, in a young adult population.

```
# Recode sibling variable
data['sibling_c'] = data['sibling'].apply(lambda x: -0.5 if x == 1 else 0.5)

# Rename and process marriage variables
data = data.rename(columns={'marriage2': 'marriage_importance', 'marriage5': 'parental_marriage'})
data = pd.concat([data, pd.get_dummies(data['parental_marriage'].astype('category'),
                                       prefix='parental_marriage', drop_first=True)], axis=1)

# Compute scale means
scales = {
    'idea_m': [f'IDEA_{i}' for i in range(1, 9)],
    'moa_achievement_m': [f'moa1#2_{i}' for i in range(1, 11)] + [f'moa2#1_{i}' for i in range(1, 11)],
    'moa_importance_m': [f'moa2#1_{i}' for i in range(1, 11)] + [f'moa2#2_{i}' for i in range(1, 11)],
    'stress_m': [f'stress_{i}' for i in range(1, 11)],
    'support_m': [f'support_{i}' for i in range(1, 13)],
```

```
    'belong_m': [f'belong_{i}' for i in range(1, 11)],
    'mindful_m': [f'mindful_{i}' for i in range(1, 16)],
    'efficacy_m': [f'efficacy_{i}' for i in range(1, 11)],
    'npi_m': [f'NPI{i}' for i in range(1, 14)],
    'exploit_m': [f'exploit_{i}' for i in range(1, 4)],
    'disability_m': [f'Q10_{i}' for i in range(1, 16)] + ['Q11'] + [f'Q14_{i}' for i in range(1, 7)],
    'social_conn_m': [f'SocMedia_{i}' for i in range(1, 6)],
    'social_new_m': [f'SocMedia_{i}' for i in range(6, 10)],
    'social_info_m': [f'SocMedia_{i}' for i in range(10, 12)],
    'swb_m': [f'swb_{i}' for i in range(1, 7)],
    'transgres_m': [f'transgres_{i}' for i in range(1, 5)],
    'usdream_m': ['usdream_1', 'usdream_2']
}

for name, items in scales.items():
    valid_items = [item for item in items if item in data.columns]
    data[name] = data[valid_items].mean(axis=1, skipna=True)

# Select final columns
final_data = data[[col for col in data.columns if
                   col.endswith(('_m', '_c')) or
                   col in ['marriage_importance', 'parental_marriage'] or
                   col.startswith(('parental_marriage_', 'sex_'))]]
```

```
codebook_created = {
    'idea_m': "Mean score: Identity Exploration and Development Assessment (IDEA-8)",
    'moa_achievement_m': "Mean score: Markers of Adulthood - Achievement subscale",
    'moa_importance_m': "Mean score: Markers of Adulthood - Importance subscale",
    'stress_m': "Mean score: Perceived Stress Scale",
    'support_m': "Mean score: Perceived Social Support",
    'belong_m': "Mean score: Need to Belong Scale",
    'mindful_m': "Mean score: Mindfulness Scale",
    'efficacy_m': "Mean score: Efficacy/Competence Scale",
    'npi_m': "Mean score: Narcissistic Personality Inventory (NPI-13)",
    'exploit_m': "Mean score: Interpersonal Exploitativeness Scale",
    'disability_m': "Mean score: Disability Identity & Status items",
    'social_conn_m': "Mean score: Social Media Use - Maintaining Connections",
    'social_new_m': "Mean score: Social Media Use - Making New Connections",
    'social_info_m': "Mean score: Social Media Use - Information Seeking",
    'swb_m': "Mean score: Subjective Well-Being",
    'transgres_m': "Mean score: Interpersonal Transgressions",
    'usdream_m': "Mean score: American Dream (Importance and Belief in Achievability)",
    'sibling_c': "Recode: -0.5 = no siblings, +0.5 = at least one sibling",
    'marriage_importance': "Importance of getting married (1-5 scale)",
```

```
    'parental_marriage': "Parental marriage status (categorical)"

}


codebook_df = pd.DataFrame({
    'Variable': list(codebook_created.keys()),
    'Description': list(codebook_created.values())
})


Markdown(codebook_df.to_markdown())
```

Table 2: This is the codebook for all the variables I created.

	Variable	Description
0	idea_m	Mean score: Identity Exploration and Development Assessment (IDEA-8)
1	moa_achievement_m	Mean score: Markers of Adulthood - Achievement subscale
2	moa_importance_m	Mean score: Markers of Adulthood - Importance subscale
3	stress_m	Mean score: Perceived Stress Scale
4	support_m	Mean score: Perceived Social Support
5	belong_m	Mean score: Need to Belong Scale
6	mindful_m	Mean score: Mindfulness Scale
7	efficacy_m	Mean score: Efficacy/Competence Scale
8	npi_m	Mean score: Narcissistic Personality Inventory (NPI-13)
9	exploit_m	Mean score: Interpersonal Exploitativeness Scale
10	disability_m	Mean score: Disability Identity & Status items
11	social_conn_m	Mean score: Social Media Use - Maintaining Connections
12	social_new_m	Mean score: Social Media Use - Making New Connections
13	social_info_m	Mean score: Social Media Use - Information Seeking
14	swb_m	Mean score: Subjective Well-Being
15	transgres_m	Mean score: Interpersonal Transgressions
16	usdream_m	Mean score: American Dream (Importance and Belief in Achievability)
17	sibling_c	Recode: -0.5 = no siblings, +0.5 = at least one sibling
18	marriage_importance	Importance of getting married (1-5 scale)
19	parental_marriage	Parental marriage status (categorical)

In addition to the psychological scales that were processed into mean scores, the second codebook Table 3 describes the variables retained from the original EAMMi2 dataset without modification. These include sex (coded as 1 = male, 2 = female, 3 = other), edu (education level from high school to graduate degree), race (categorical coding for racial/ethnic identity), income (ordinal indicator of household income). The outcome variable physSx_12 (fatigue) was also retained in its original form before being recoded into a binary variable to indicate symptom presence (2 = symptom present). These variables were selected based on theoretical relevance to somatic symptom experiences and were not transformed beyond basic recoding or dummy encoding when necessary for modeling.

```
Markdown(
    codebook
    .loc[['sex', 'edu', 'race', 'income', 'physSx_12']]
    [['Question text', 'responses']]
    .to_markdown()
)
```

Table 3: This is the codebook of the variables retained without modification from the raw dataset.

Variable Name	Question text	responses
sex	What is your gender?	1-male, 2-female, 3-other
edu	Which of the following choices best describes your educational level/attainment?	1-high school, 9 completed graduate degree
race	What is your racial/ethnic group (check all the apply)? - Selected Choice	1-1White, 2-Black, 3-Hispanic, 4-Asian, 5 -Native American, 6-other
income	Please indicate your current household income in U.S. dollars	1-rather not say, 2 -under 20,000, 9-over 1 million
physSx_12	Feeling tired or having low energy	1-not bothered at all, 3 bothered a lot

The outcome variable physSx_12, which reflects the severity of a specific somatic symptom, shows a right-skewed distribution. The majority of responses fall in the moderate (2) and high (3) categories, with fewer individuals reporting mild symptoms (1). This class imbalance may influence model performance and will be considered during model evaluation.

For this project, we selected Item 12 (fatigue) from the PHQ somatic symptom inventory as the primary outcome of interest. This symptom was chosen due to its clinical relevance and sufficient variability in the sample, despite some imbalance in outcome distribution. A binary outcome variable was created to indicate whether the symptom was present (i.e., the participant reported experiencing fatigue “slightly” or “a lot,” score 2) or not present (“not at all,” score = 1). To account for the imbalance, classification models were adjusted using class weights. Separate logistic regression and random forest models were trained using mean scores from psychological scales and demographic variables as predictors to identify key risk factors associated with fatigue.

```
data["physSx_12"].value_counts()
```

physSx_12	count
3.0	1552
2.0	1220
1.0	405

4.1 Variables and Measures

4.1.1 Outcome Variable

The outcome variable were assessed using item 12 from the Patient Health Questionnaire (PHQ). Responses were recoded into a binary variable, where a score of 2 or higher was considered symptomatic (coded as 1), and a score of 1 indicated no significant fatigue symptoms (coded as 0). This binary formulation was chosen due to its moderate class balance and clinical relevance.

4.1.2 Psychological Predictors

All psychological variables were computed as mean scores across item-level responses, provided that a sufficient proportion of items (70%) were present. The following scales were included:

- IDEA-8: Identity exploration and development (8 items)
- Markers of Adulthood:
  - Achievement subscale (10 items)
  - Importance subscale (10 items)
- Perceived Stress Scale (10 items)
- Perceived Social Support (12 items)
- Need to Belong Scale (10 items)
- Mindfulness Scale (15 items)
- Self-Efficacy/Competence (10 items)
- Narcissistic Personality Inventory (NPI-13) (13 items)
- Interpersonal Exploitativeness (3 items)
- Disability Identity and Status (combination of 22 items from Q10, Q11, and Q14)
- Social Media Use:
  - Maintaining connections (5 items)
  - Making new connections (4 items)
  - Seeking information (2 items)
- Subjective Well-Being (6 items)
- Interpersonal Transgressions (4 items)
- Belief in and importance of the American Dream (2 items)

4.1.3 Demographic and Attitudinal Predictors

- Sex (male, female, other; categorical)
- Education level
- Race/Ethnicity
- Household income
- School attended
- Parental marriage status (recoded into categorical dummy variables)
- Siblings (recoded: -0.5 = no siblings, 0.5 = at least one sibling)
- Importance of marriage (single-item rating)

These variables represent a theoretically grounded and diverse set of predictors for modeling somatic symptom risk, with particular focus on psychological and social functioning within a young adult population. Table 5 shows the an experpt of the final data we are using.

```
# Binary outcome for fatigue:
# 1 = symptom present (score   2)
# 0 = symptom not present (score = 1)
data['fatigue_binary'] = (data['physSx_12'] >= 2).astype(int)
```

```
constructed_vars = [
    'idea_m', 'moa_achievement_m', 'moa_importance_m', 'stress_m', 'support_m',
    'belong_m', 'mindful_m', 'efficacy_m', 'npi_m', 'exploit_m', 'disability_m',
    'social_conn_m', 'social_new_m', 'social_info_m', 'swb_m', 'transgres_m',
    'usdream_m', 'sibling_c', 'marriage_importance', 'parental_marriage'
]
raw_demographics = ['sex', 'edu', 'race', 'income']

final_vars = constructed_vars + raw_demographics

final_data = data[final_vars].copy()

final_data.sample(2).transpose()
```

Table 5: This table desplays random 2 rows of the new subset I created and transpose for eacy to read

	2885	2066
idea_m	3.125	3.5
moa_achievement_m	2.5	2.7
moa_importance_m	2.631579	2.7
stress_m	2.7	3.0
support_m	4.833333	6.583333
belong_m	3.5	2.4
mindful_m	3.333333	3.8
efficacy_m	3.0	3.7

Table 5: This table desplays random 2 rows of the new subset I created and transpose for eacy to read

	2885	2066
npi_m	1.615385	1.538462
exploit_m	4.0	1.0
disability_m	2.772727	2.454545
social_conn_m	3.2	2.6
social_new_m	4.0	1.0
social_info_m	4.0	1.0
swb_m	4.666667	5.666667
transgres_m	2.5	1.25
usdream_m	3.5	3.0
sibling_c	0.5	0.5
marriage_importance	4.0	5.0
parental_marriage	1.0	1.0
sex	1.0	1.0
edu	2.0	2.0
race	1	1
income	7.0	1.0

Table 6 presents descriptive statistics for all study variables. Across variables, missing data were minimal (0–25 cases). Standard deviations indicate varying levels of dispersion, with the lowest variability observed for npi_m (SD = 0.12) and the highest for exploit_m (SD = 1.37). Categorical variables such as sex and sibling count showed appropriate coding ranges. Income and education variables were right-skewed, with maximum values of 9. Overall, distributions and missingness were acceptable for further analyses.

```
stats = final_data.describe().transpose().round(3)
stats['missing'] = final_data.isnull().sum()
stats = stats[['count', 'missing', 'mean', 'std', 'min', '50%', 'max']]
stats
```

Table 6: "Summary Statistics"

	count	missing	mean	std	min	50%	max
idea_m	3180.0	2	3.571	0.383	1.000	3.625	4.000
moa_achievement_m	3181.0	1	2.658	0.299	1.500	2.650	4.000
moa_importance_m	3180.0	2	2.778	0.296	1.650	2.800	4.000
stress_m	3177.0	5	3.266	0.408	1.000	3.300	5.000
support_m	3180.0	2	5.530	1.135	1.000	5.750	7.000
belong_m	3179.0	3	3.312	0.496	1.000	3.400	5.000
mindful_m	3177.0	5	3.710	0.843	1.133	3.733	6.000
efficacy_m	3180.0	2	3.125	0.450	1.000	3.100	4.000

Table 6: “Summary Statistics”

	count	missing	mean	std	min	50%	max
npi_m	3179.0	3	1.545	0.124	1.000	1.538	2.000
exploit_m	3179.0	3	2.387	1.374	1.000	2.000	7.000
disability_m	3180.0	2	2.266	0.431	1.167	2.111	4.045
social_conn_m	3179.0	3	3.095	0.862	1.000	3.200	5.000
social_new_m	3179.0	3	3.052	0.970	1.000	3.000	5.000
social_info_m	3179.0	3	3.386	1.042	1.000	3.500	5.000
swb_m	3179.0	3	4.471	1.323	1.000	4.667	7.000
transgres_m	3175.0	7	2.050	1.062	1.000	1.750	7.000
usdream_m	3173.0	9	3.503	1.033	1.000	3.500	5.000
sibling_c	3182.0	0	0.404	0.294	-0.500	0.500	0.500
marriage_importance	3172.0	10	3.630	1.109	1.000	4.000	5.000
parental_marriage	3172.0	10	1.500	0.869	1.000	1.000	4.000
sex	3178.0	4	1.768	0.461	1.000	2.000	3.000
edu	3174.0	8	2.630	1.655	1.000	2.000	9.000
income	3157.0	25	3.544	2.299	1.000	3.000	9.000

5 EDA

We now proceed with the exploratory data analysis (EDA) to examine variable relationships and detect potential outliers patterns that may inform subsequent modeling.

Figure 1 displays the distributions of all numeric variables in the dataset. Most variables showed approximately normal or slightly skewed distributions (e.g., mindful_m, efficacy_m, moa_achievement_m), while others were highly skewed (e.g., support_m, exploit_m, transgres_m). Categorical variables (e.g., sex, sibling_c, edu) showed clear discrete patterns, reflecting limited response options. Some variables (e.g., social_info_m, marriage_importance) exhibited multimodal or uneven distributions, suggesting clustering of responses. Overall, the figure indicates variability in distribution shapes across measures, which may inform modeling strategies.

```
# Histograms for numeric variables
numeric_vars = final_data.select_dtypes(include=['float64', 'int64']).columns
final_data[numeric_vars].hist(figsize=(16, 14), bins=20)
plt.suptitle("Distribution of Numeric Variables", fontsize=16)
plt.tight_layout()
plt.show()
```

Figure 1: Distribution of numeric variables across the final dataset.

Figure 2 presents a correlation matrix of numeric predictors, including the binary fatigue outcome. Most correlations were small to moderate, suggesting low multicollinearity among variables. Notably, moa_achievement_m and moa_importance_m were highly correlated (r = .85), indicating a strong relationship between these constructs. Moderate positive associations were observed between support_m and both swb_m (r = .47) and belong_m (r = .40). Correlations with fatigue_binary were generally weak, with the strongest negative correlation observed for support_m (r = −.17). These findings offer preliminary insights into variable associations and guide further analysis.

```
# heatmap
final_data['fatigue_binary'] = data['fatigue_binary']
selected_vars = list(numeric_vars) + ['fatigue_binary']

plt.figure(figsize=(24, 20))
```

```
sns.heatmap(final_data[selected_vars].corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Matrix (Numeric Predictors)", fontsize=14)
plt.show()
```

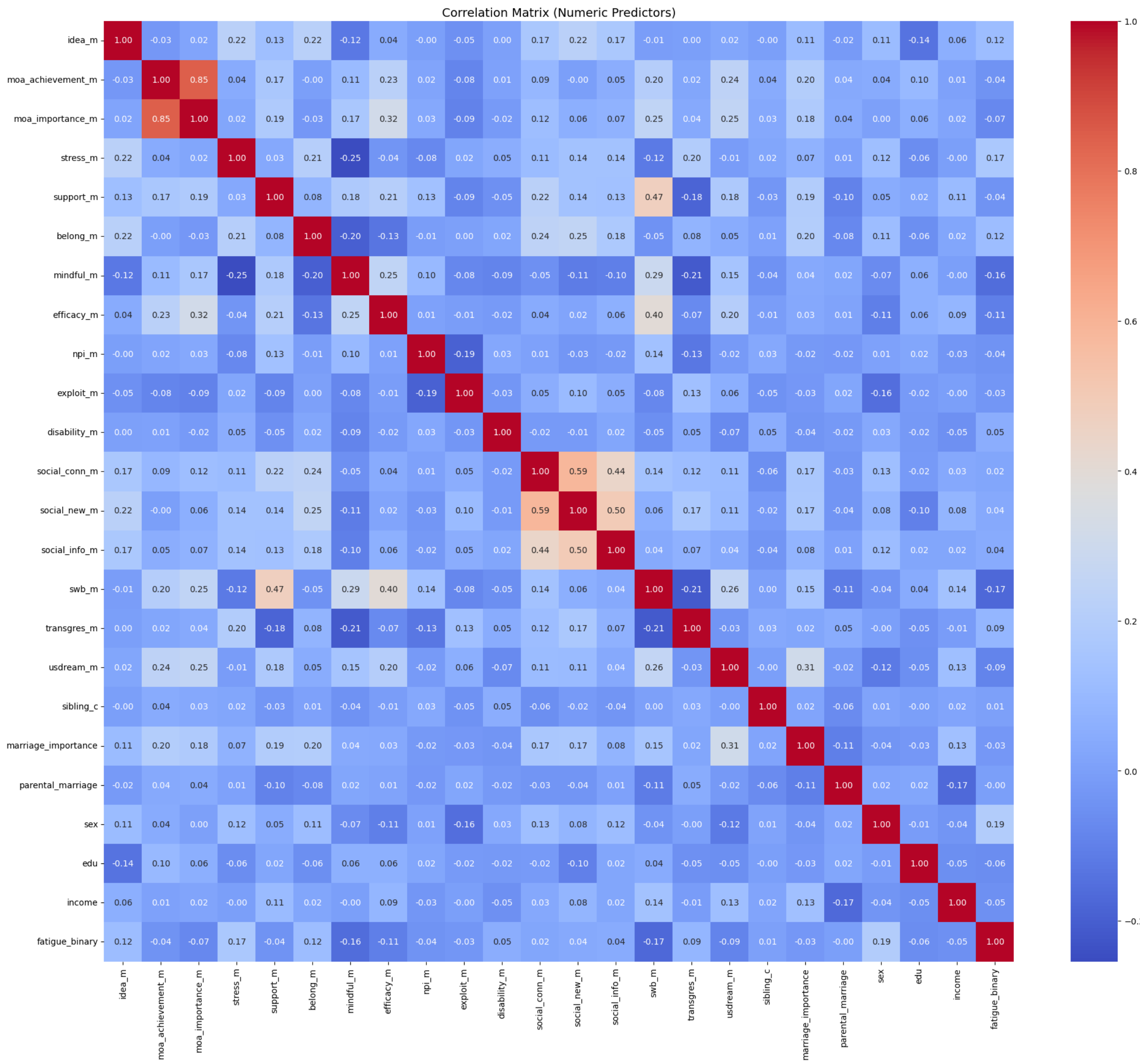

Figure 2: Correlation heatmap showing pairwise Pearson correlations. Stronger correlations appear in darker shades.

#Train-Test Split The dataset was split into training and test sets using a stratified sampling approach to preserve the class distribution of the binary fatigue outcome. A total of 3,132 complete cases were included, with 70% allocated to training (n = 2,192) and 30% to testing (n = 940). The target variable (fatigue_binary) was imbalanced, with 87.3% of participants coded as "1" and 12.7% coded as "0". This stratified split ensures both sets maintain the original class proportions, supporting more reliable model evaluation.

```
X = final_data.drop(columns=['fatigue_binary'], errors='ignore')  # safer drop
y = data['fatigue_binary']


model_data = X.join(y.rename('fatigue_binary')).dropna()
```

```
# Separate X and y
X_clean = model_data.drop(columns='fatigue_binary')
y_clean = model_data['fatigue_binary']

# Split
X_train, X_test, y_train, y_test = train_test_split(
    X_clean, y_clean,
    test_size=0.3,
    random_state=45,
    stratify=y_clean
)
print("Train shape:", X_train.shape)
print("Test shape:", X_test.shape)
print(y_clean.value_counts(normalize=True))
```

```
Train shape: (2192, 24)
Test shape: (940, 24)
fatigue_binary
1    0.872605
0    0.127395
Name: proportion, dtype: float64
```

5.1 Data Normalization

The data were preprocessed to prepare for model training. Non-numeric columns were identified and converted to numeric format, with commas replaced by decimal points. Missing values in the race variable were imputed using the most frequent category (mode). Following data cleaning, all predictor variables were standardized using StandardScaler, which was fit on the training set and applied to both training and test sets. This normalization step ensures that features are on a comparable scale, which is important for many machine learning algorithms.

```
# Identify non-numeric columns before scaling
non_numeric_cols = X_train.select_dtypes(exclude=['float64', 'int64']).columns

# Convert identified columns to numeric, replacing commas with periods
for col in non_numeric_cols:
    X_train[col] = X_train[col].astype(str).str.replace(',', '.', regex=False)
    X_test[col] = X_test[col].astype(str).str.replace(',', '.', regex=False)
```

```
X_train[non_numeric_cols] = X_train[non_numeric_cols].apply(pd.to_numeric, errors='coerce')
X_test[non_numeric_cols] = X_test[non_numeric_cols].apply(pd.to_numeric, errors='coerce')

# Impute missing values in the 'race' column with the mode (most frequent value)
# Fit imputer on training data and transform both train and test
imputer_race = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
X_train['race'] = imputer_race.fit_transform(X_train[['race']])
X_test['race'] = imputer_race.transform(X_test[['race']])


# Step 1: Initialize the scaler
scaler = StandardScaler()

# Step 2: Fit the scaler on the training data and transform both train and test
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

6 Model Evaluation

6.1 Tensorflow

Model performance was first evaluated using a neural network built with TensorFlow. Accuracy and area under the ROC curve (AUC) served as the primary evaluation metrics. A confusion matrix and classification report offered further insight into precision, recall, and F1 scores. Visual diagnostics—such as loss and accuracy over training epochs, a receiver operating characteristic (ROC) curve, and a histogram of predicted probabilities—were used to assess model fit, class discrimination, and prediction confidence.

```
# Build a simple model
model = Sequential([
    Dense(32, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    Dropout(0.3),
    Dense(16, activation='relu'),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])

# Check for and handle potential NaN values in X_train_scaled
if np.isnan(X_train_scaled).any():
    print("NaN values found in X_train_scaled. Imputing with the mean of the scaled training data.")
    imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
```

```
    X_train_scaled_imputed = imputer.fit_transform(X_train_scaled)
else:
    X_train_scaled_imputed = X_train_scaled


# Train
history = model.fit(
    X_train_scaled_imputed, y_train,
    validation_split=0.2,
    epochs=30,
    batch_size=32,
    verbose=1
)
```

```
Epoch 1/30
55/55          2s 7ms/step - accuracy: 0.8141 - loss: 0.4645 - val_accuracy: 0.8770 -
val_loss: 0.3866
Epoch 2/30
55/55          0s 4ms/step - accuracy: 0.8470 - loss: 0.4395 - val_accuracy: 0.8770 -
val_loss: 0.3808
Epoch 3/30
55/55          0s 4ms/step - accuracy: 0.8734 - loss: 0.3967 - val_accuracy: 0.8770 -
val_loss: 0.3770
Epoch 4/30
55/55          0s 4ms/step - accuracy: 0.8632 - loss: 0.3847 - val_accuracy: 0.8770 -
val_loss: 0.3760
Epoch 5/30
55/55          0s 4ms/step - accuracy: 0.8728 - loss: 0.3688 - val_accuracy: 0.8770 -
val_loss: 0.3748
Epoch 6/30
55/55          0s 4ms/step - accuracy: 0.8659 - loss: 0.3749 - val_accuracy: 0.8770 -
val_loss: 0.3731
Epoch 7/30
55/55          0s 4ms/step - accuracy: 0.8588 - loss: 0.3677 - val_accuracy: 0.8770 -
val_loss: 0.3729
Epoch 8/30
55/55          0s 4ms/step - accuracy: 0.8758 - loss: 0.3606 - val_accuracy: 0.8770 -
val_loss: 0.3714
Epoch 9/30
55/55          0s 4ms/step - accuracy: 0.8750 - loss: 0.3484 - val_accuracy: 0.8747 -
val_loss: 0.3725
Epoch 10/30
55/55          0s 4ms/step - accuracy: 0.8620 - loss: 0.3483 - val_accuracy: 0.8747 -
val_loss: 0.3735
```

Epoch 11/30
55/55 0s 6ms/step - accuracy: 0.8775 - loss: 0.3376 - val_accuracy: 0.8747 - val_loss: 0.3738
Epoch 12/30
55/55 0s 8ms/step - accuracy: 0.8597 - loss: 0.3586 - val_accuracy: 0.8747 - val_loss: 0.3740
Epoch 13/30
55/55 0s 6ms/step - accuracy: 0.8736 - loss: 0.3522 - val_accuracy: 0.8747 - val_loss: 0.3740
Epoch 14/30
55/55 1s 7ms/step - accuracy: 0.8636 - loss: 0.3321 - val_accuracy: 0.8747 - val_loss: 0.3758
Epoch 15/30
55/55 1s 7ms/step - accuracy: 0.8579 - loss: 0.3616 - val_accuracy: 0.8747 - val_loss: 0.3767
Epoch 16/30
55/55 0s 4ms/step - accuracy: 0.8616 - loss: 0.3454 - val_accuracy: 0.8747 - val_loss: 0.3782
Epoch 17/30
55/55 0s 4ms/step - accuracy: 0.8824 - loss: 0.3256 - val_accuracy: 0.8747 - val_loss: 0.3774
Epoch 18/30
55/55 0s 4ms/step - accuracy: 0.8602 - loss: 0.3458 - val_accuracy: 0.8747 - val_loss: 0.3777
Epoch 19/30
55/55 0s 4ms/step - accuracy: 0.8638 - loss: 0.3348 - val_accuracy: 0.8747 - val_loss: 0.3777
Epoch 20/30
55/55 0s 4ms/step - accuracy: 0.8847 - loss: 0.3187 - val_accuracy: 0.8747 - val_loss: 0.3764
Epoch 21/30
55/55 0s 4ms/step - accuracy: 0.8624 - loss: 0.3393 - val_accuracy: 0.8747 - val_loss: 0.3783
Epoch 22/30
55/55 0s 4ms/step - accuracy: 0.8671 - loss: 0.3401 - val_accuracy: 0.8747 - val_loss: 0.3783
Epoch 23/30
55/55 0s 4ms/step - accuracy: 0.8730 - loss: 0.3249 - val_accuracy: 0.8747 - val_loss: 0.3783
Epoch 24/30
55/55 0s 4ms/step - accuracy: 0.8643 - loss: 0.3592 - val_accuracy: 0.8747 - val_loss: 0.3773
Epoch 25/30
55/55 0s 4ms/step - accuracy: 0.8827 - loss: 0.3120 - val_accuracy: 0.8747 - val_loss: 0.3766

Epoch 26/30
55/55 0s 4ms/step - accuracy: 0.8773 - loss: 0.3029 - val_accuracy: 0.8724 - val_loss: 0.3769
Epoch 27/30
55/55 0s 4ms/step - accuracy: 0.8759 - loss: 0.3109 - val_accuracy: 0.8724 - val_loss: 0.3775
Epoch 28/30
55/55 0s 4ms/step - accuracy: 0.8778 - loss: 0.3123 - val_accuracy: 0.8724 - val_loss: 0.3781
Epoch 29/30
55/55 0s 4ms/step - accuracy: 0.8625 - loss: 0.3247 - val_accuracy: 0.8724 - val_loss: 0.3802
Epoch 30/30
55/55 0s 4ms/step - accuracy: 0.8729 - loss: 0.3202 - val_accuracy: 0.8724 - val_loss: 0.3783

Figure 3 shows the training and validation loss (left) and accuracy (right) across 30 training epochs for the neural network model. Training loss steadily decreased and stabilized, while validation loss plateaued early, indicating good convergence without overfitting. Accuracy for both training and validation sets increased rapidly and remained stable around 87%, suggesting the model generalized well to unseen data. These patterns reflect effective learning and a well-tuned model.

```
# Plot training and validation loss
plt.figure(figsize=(10, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train')
plt.plot(history.history['val_loss'], label='Validation')
plt.title('Loss Over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

# Plot training and validation accuracy
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Train')
plt.plot(history.history['val_accuracy'], label='Validation')
plt.title('Accuracy Over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```

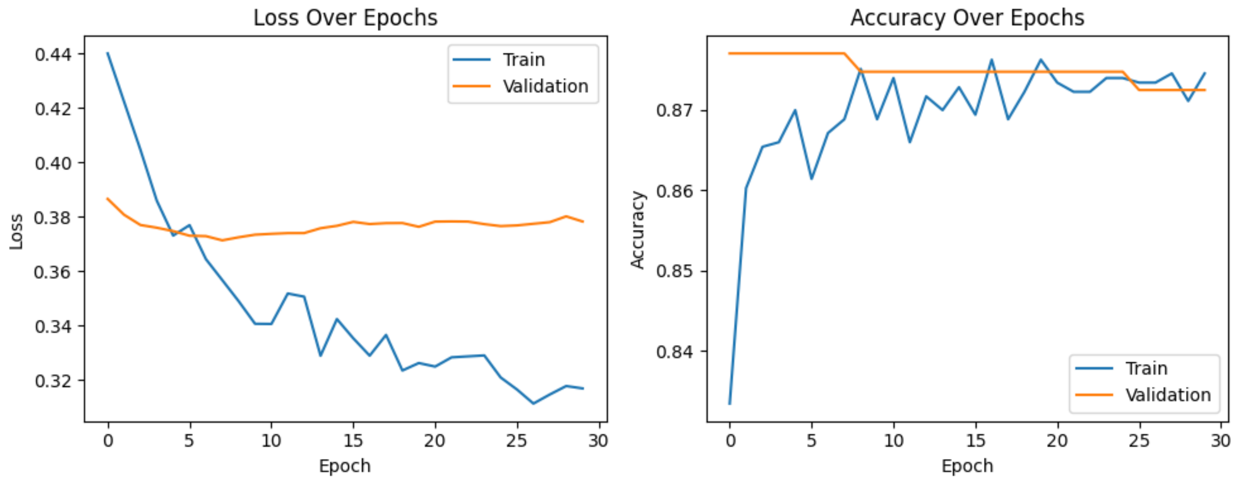

Figure 3: Training and Validation Loss and Accuracy Across Epochs for Neural Network Model

As shown in Table 7, the neural network model achieved a test accuracy of 87.2% and an area under the curve (AUC) of 0.73, indicating good overall performance. However, the classification report revealed a critical limitation: the model failed to identify any instances of the minority class (label = 0), resulting in a precision, recall, and F1-score of 0.00 for that class. In contrast, the model demonstrated strong performance on the majority class (label = 1), with a recall of 1.00 and an F1-score of 0.93. These results suggest that while the model performs well in aggregate, it is highly imbalanced in its predictions and unable to detect the minority class, likely due to class imbalance in the training data.

```
if np.isnan(X_test_scaled).any():
    imputer = SimpleImputer(strategy='mean')
    imputer.fit(X_train_scaled)
    X_test_scaled_imputed = imputer.transform(X_test_scaled)
else:
    X_test_scaled_imputed = X_test_scaled

# Predict
y_pred_prob = model.predict(X_test_scaled_imputed).flatten()
y_pred_class = (y_pred_prob >= 0.5).astype(int)

# Calculate metrics
metrics = {
    "Accuracy": accuracy_score(y_test, y_pred_class),
    "Balanced Accuracy": balanced_accuracy_score(y_test, y_pred_class),
    "AUC": roc_auc_score(y_test, y_pred_prob),
    "F1 Score": f1_score(y_test, y_pred_class),
    "Precision": precision_score(y_test, y_pred_class),
    "Recall": recall_score(y_test, y_pred_class),
```

```
}

# Create a metrics table
metrics_df = pd.DataFrame({
    "Metric": list(metrics.keys()),
    "Value": [round(v, 3) for v in metrics.values()]
})

metrics_df
```

30/30 0s 3ms/step

Table 7: Performance metrics for the neural network model predicting fatigue.

	Metric	Value
0	Accuracy	0.871
1	Balanced Accuracy	0.499
2	AUC	0.746
3	F1 Score	0.931
4	Precision	0.872
5	Recall	0.999

Figure 4 presents the receiver operating characteristic (ROC) curve for the neural network model. The area under the curve (AUC) was 0.74, indicating fair discriminatory ability in distinguishing between participants with and without fatigue. The curve lies consistently above the diagonal reference line, suggesting the model performs better than random chance. However, the moderate AUC also reflects limitations in sensitivity, especially given the model's failure to detect the minority class in the classification results.

```
fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f"AUC = {roc_auc:.2f}")
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

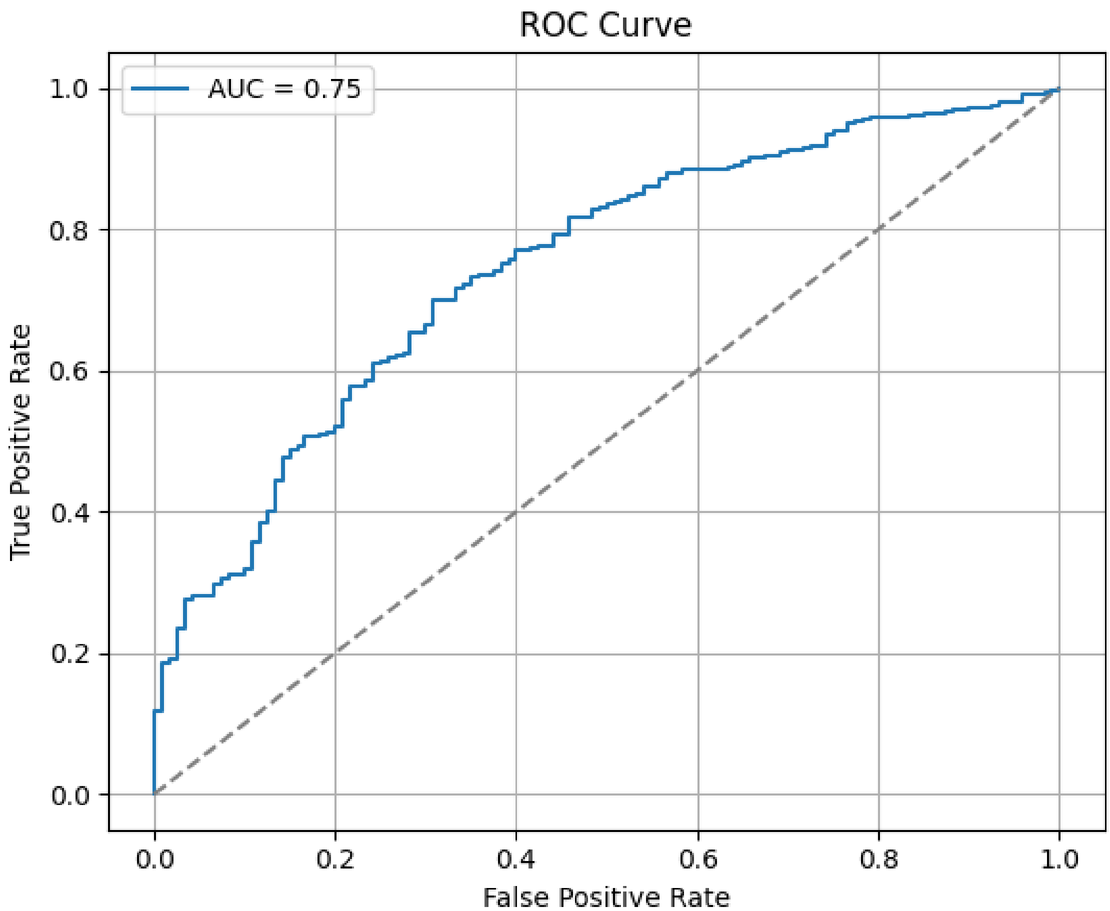

Figure 4: Receiver Operating Characteristic (ROC) Curve for Neural Network Model

Figure 5 displays the distribution of predicted probabilities for fatigue (class 1) in the test set. The histogram shows that most predictions were concentrated at the higher end of the probability range, particularly above 0.85. This indicates that the model was highly confident in predicting the majority class. The absence of low-probability predictions suggests poor discrimination for identifying the minority class (no fatigue), consistent with earlier findings from the confusion matrix and classification report. This skewed probability distribution further highlights the model's bias toward the dominant class.

```
plt.figure(figsize=(6, 4))
plt.hist(y_pred_prob, bins=30, edgecolor='k', alpha=0.7)
plt.title("Predicted Probabilities (Test Set)")
plt.xlabel("Predicted Probability of Class 1 (Fatigue)")
plt.ylabel("Count")
plt.tight_layout()
plt.show()
```

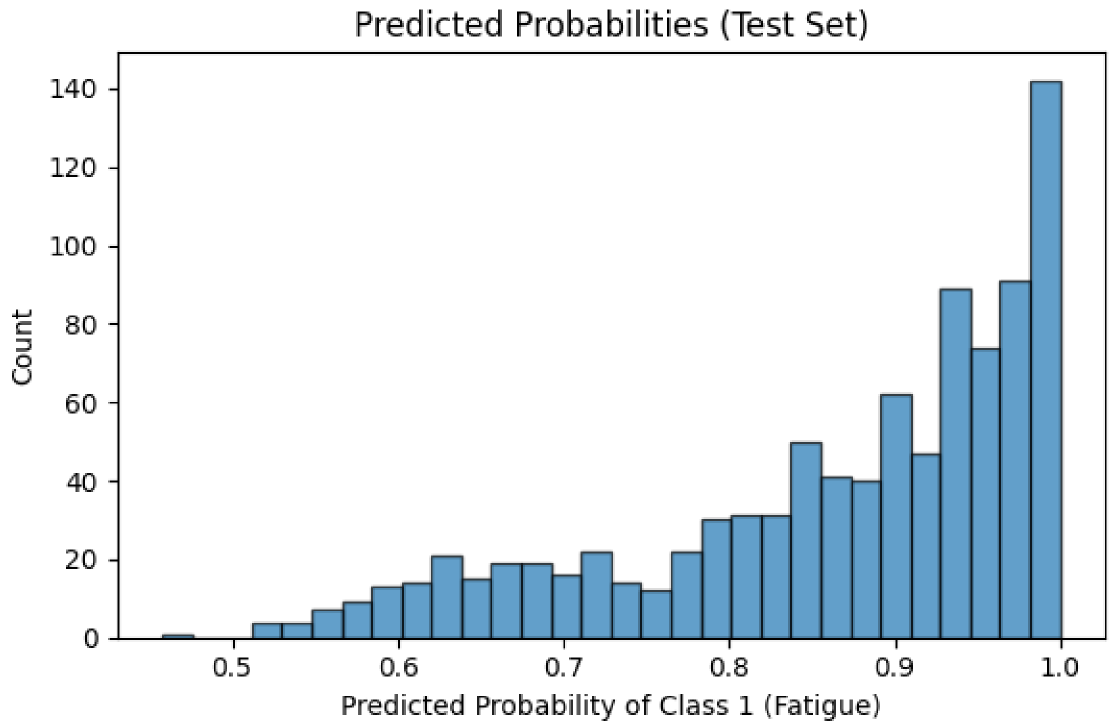

Figure 5: Histogram of Predicted Probabilities for Fatigue Class (Test Set)

6.2 Keras 3 + PyTorch

The second neural network model was built using the Keras Sequential API and consisted of an input layer, two hidden layers (with 64 and 32 units, respectively), ReLU activations, and dropout regularization (rate = 0.3) to reduce overfitting. The output layer used a sigmoid activation to support binary classification. The model was compiled with the Adam optimizer (learning rate = 0.001) and binary cross-entropy loss. It was trained for 20 epochs with a batch size of 32 and 20% of the training data reserved for validation. The final test accuracy achieved was 87.2%, indicating strong overall predictive performance on the fatigue classification task.

```
model = keras.Sequential([
    layers.Input(shape=(X_train.shape[1],)),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(32, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(1, activation='sigmoid')
])

model.compile(
    optimizer=optimizers.Adam(learning_rate=0.001),
    loss='binary_crossentropy',
```

```
    metrics=['accuracy']
)

model.fit(
    X_train, y_train,
    validation_split=0.2,
    epochs=20,
    batch_size=32
)
```

```
Epoch 1/20
55/55            2s 7ms/step - accuracy: 0.7538 - loss: 0.6467 - val_accuracy: 0.8770 -
val_loss: 0.3789
Epoch 2/20
55/55            0s 4ms/step - accuracy: 0.8486 - loss: 0.4519 - val_accuracy: 0.8770 -
val_loss: 0.3690
Epoch 3/20
55/55            0s 4ms/step - accuracy: 0.8687 - loss: 0.4070 - val_accuracy: 0.8770 -
val_loss: 0.3790
Epoch 4/20
55/55            0s 4ms/step - accuracy: 0.8611 - loss: 0.4037 - val_accuracy: 0.8770 -
val_loss: 0.3700
Epoch 5/20
55/55            0s 4ms/step - accuracy: 0.8783 - loss: 0.3625 - val_accuracy: 0.8770 -
val_loss: 0.3782
Epoch 6/20
55/55            0s 4ms/step - accuracy: 0.8723 - loss: 0.3641 - val_accuracy: 0.8770 -
val_loss: 0.3606
Epoch 7/20
55/55            0s 4ms/step - accuracy: 0.8707 - loss: 0.3922 - val_accuracy: 0.8770 -
val_loss: 0.3708
Epoch 8/20
55/55            0s 4ms/step - accuracy: 0.8738 - loss: 0.3702 - val_accuracy: 0.8770 -
val_loss: 0.3616
Epoch 9/20
55/55            0s 4ms/step - accuracy: 0.8662 - loss: 0.3824 - val_accuracy: 0.8770 -
val_loss: 0.3742
Epoch 10/20
55/55            0s 4ms/step - accuracy: 0.8788 - loss: 0.3503 - val_accuracy: 0.8770 -
val_loss: 0.3639
Epoch 11/20
55/55            0s 7ms/step - accuracy: 0.8705 - loss: 0.3635 - val_accuracy: 0.8770 -
val_loss: 0.3779
Epoch 12/20
```

```
55/55                1s 6ms/step - accuracy: 0.8767 - loss: 0.3415 - val_accuracy: 0.8770 -
val_loss: 0.3645
Epoch 13/20
55/55                1s 7ms/step - accuracy: 0.8673 - loss: 0.3720 - val_accuracy: 0.8770 -
val_loss: 0.3638
Epoch 14/20
55/55                0s 8ms/step - accuracy: 0.8642 - loss: 0.3557 - val_accuracy: 0.8770 -
val_loss: 0.3663
Epoch 15/20
55/55                0s 4ms/step - accuracy: 0.8771 - loss: 0.3527 - val_accuracy: 0.8770 -
val_loss: 0.3759
Epoch 16/20
55/55                0s 4ms/step - accuracy: 0.8829 - loss: 0.3294 - val_accuracy: 0.8770 -
val_loss: 0.3665
Epoch 17/20
55/55                0s 4ms/step - accuracy: 0.8602 - loss: 0.3584 - val_accuracy: 0.8770 -
val_loss: 0.3726
Epoch 18/20
55/55                0s 4ms/step - accuracy: 0.8609 - loss: 0.3604 - val_accuracy: 0.8770 -
val_loss: 0.3708
Epoch 19/20
55/55                0s 4ms/step - accuracy: 0.8677 - loss: 0.3618 - val_accuracy: 0.8770 -
val_loss: 0.3649
Epoch 20/20
55/55                0s 4ms/step - accuracy: 0.8819 - loss: 0.3285 - val_accuracy: 0.8770 -
val_loss: 0.3694
```

```
<keras.src.callbacks.history.History at 0x79194ae5d650>
```

```python
# Predictions (probabilities → binary)
y_pred_proba = model.predict(X_test)
y_pred = (y_pred_proba > 0.5).astype(int)

# Accuracy
acc = accuracy_score(y_test, y_pred)
print("Model Accuracy:", acc)
```

```
30/30                0s 3ms/step
Model Accuracy: 0.8723404255319149
```

The evaluation of the Keras neural network model was conducted on the test set, with missing values imputed using the mean of the training data where necessary. Predicted probabilities were converted to binary class labels using a 0.5 threshold. The model achieved a test accuracy of 87.2% and an AUC of 0.74, indicating strong overall predictive performance and acceptable class discrimination. These results are consistent with earlier evaluation metrics, further confirming the model's effectiveness in identifying fatigue cases.

```
# Impute missing values in the test set using the imputer fitted on the training data
if np.isnan(X_test).any().any():
    imputer = SimpleImputer(strategy='mean')
    imputer.fit(X_train) # Fit on training data
    X_test_imputed = imputer.transform(X_test)
else:
    X_test_imputed = X_test.values # Convert to numpy array if no imputation needed


# Predictions (probabilities → binary)
y_pred_proba = model.predict(X_test_imputed).flatten()
y_pred = (y_pred_proba > 0.5).astype(int)

#Metrics
acc = accuracy_score(y_test, y_pred)
balanced_acc = balanced_accuracy_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_pred_proba)
f1 = f1_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)


# Store results in a dictionary
results_dict = {
    'Metric': ['Accuracy', 'Balanced Accuracy', 'AUC', 'F1 Score', 'Precision', 'Recall'],
    'Value': [acc, balanced_acc, auc, f1, precision, recall]
}

# Create DataFrame and round values
results_df = pd.DataFrame(results_dict)
results_df['Value'] = results_df['Value'].round(3)


results_df
```

30/30 0s 2ms/step

	Metric	Value
0	Accuracy	0.872
1	Balanced Accuracy	0.500
2	AUC	0.735
3	F1 Score	0.932
4	Precision	0.872
5	Recall	1.000

```
# Convert X_test to numpy if it's a dataframe
X_test_np = X_test.values

# Use SHAP DeepExplainer (works for Keras/TensorFlow and PyTorch)
explainer = shap.DeepExplainer(model, X_train.values[:100])  # use a subset as background
shap_values = explainer.shap_values(X_test_np)
```

```
# Calculate mean absolute SHAP values per feature
# Access the SHAP values for the positive class (class 1)
if isinstance(shap_values, list):
    shap_abs = np.abs(shap_values[0]).mean(axis=0)
else:
    shap_abs = np.abs(shap_values).mean(axis=0)

# Ensure shap_abs is 1-dimensional
shap_abs = shap_abs.flatten()

shap_importance = pd.DataFrame({
    'Feature': X_test.columns,
    'MeanAbsSHAP': shap_abs
}).sort_values(by='MeanAbsSHAP', ascending=True)  # ascending=True so smallest at top for horizont
```

?@fig-SHAP presents the SHAP-based feature importance for the Keras neural network model. The results indicate that swb_m (subjective well-being) was the most influential predictor of fatigue, followed by sex, usdream_m, and idea_m. Other variables, such as race, social_conn_m, and exploit_m, contributed moderately, while features like npi_m, sibling_c, and parental_marriage had minimal impact. The SHAP values reflect the average contribution of each feature to the model's predictions, highlighting the relative dominance of well-being and demographic factors in predicting fatigue.

```
plt.figure(figsize=(8,6))
plt.barh(shap_importance['Feature'], shap_importance['MeanAbsSHAP'], color='skyblue')
plt.xlabel("Mean Absolute SHAP Value (Feature Importance)")
plt.title("Feature Importance (SHAP)")
plt.show()
```

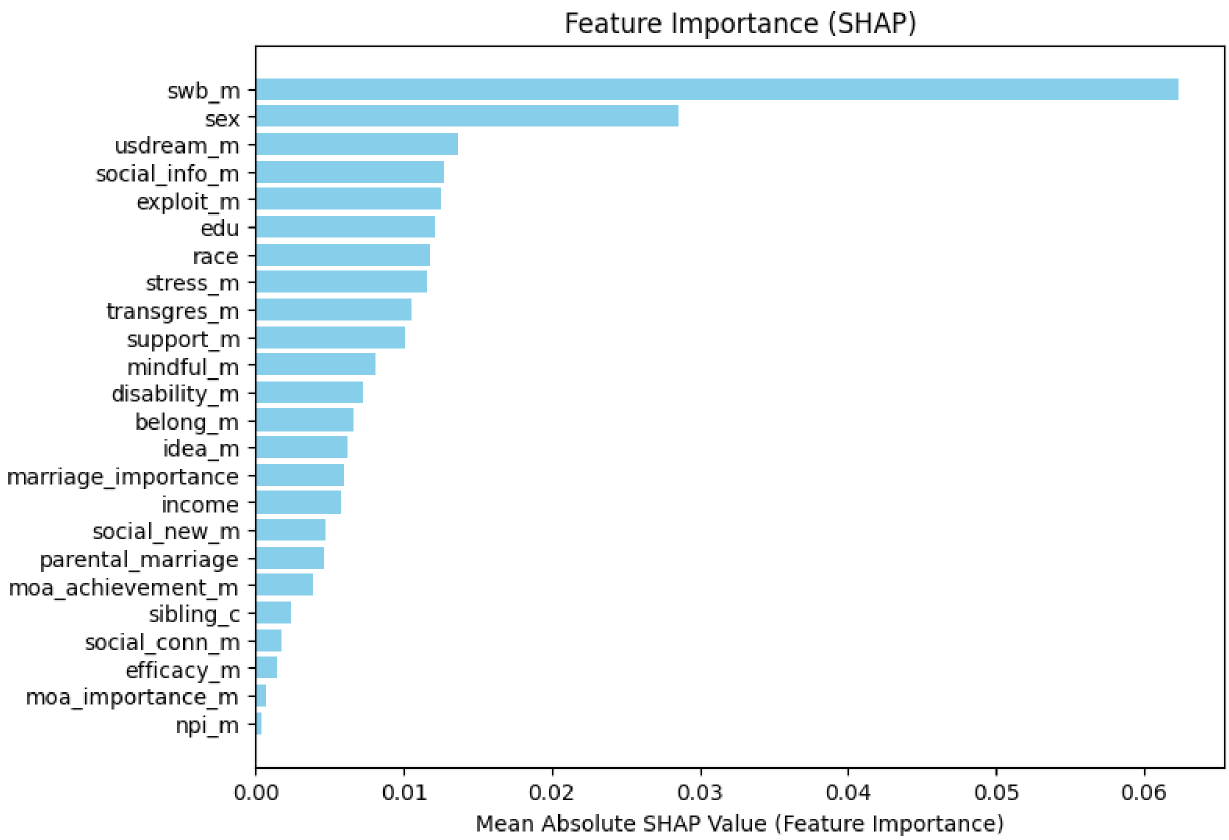

Figure 6: Feature Importance Based on SHAP Values for the Keras Neural Network Model

7 Discussion

This project aimed to predict the presence of fatigue (Symptom 12) using psychosocial and demographic predictors through multiple supervised machine learning approaches, including neural networks. The models achieved moderate predictive accuracy, but early neural network models showed highly skewed probability distributions, with most predictions near 1.0. This suggests overconfidence and poor probability calibration, likely influenced by class imbalance. A refined neural network that incorporated dropout and early stopping displayed a more balanced probability distribution and better calibration, improving generalization.

Two neural network architectures were implemented and compared. The initial model included fewer hidden units and relied on a simpler configuration, while the refined model introduced more layers, increased units per layer, and incorporated dropout regularization to prevent overfitting. Although both models achieved similar test accuracy (87.2%) and AUC (~0.74), the refined model demonstrated more stable validation loss curves and a less skewed probability distribution. This suggests that architectural adjustments such as dropout may improve generalization and probability calibration, even when overall accuracy remains unchanged.

These findings highlight the importance of evaluating calibration and using metrics beyond raw accuracy, such as balanced accuracy and F1 score, when class imbalance is present. Future work should explore probability calibration techniques and richer data sources to enhance prediction quality. Clinically, better-calibrated models could support early identification of individuals at risk for fatigue, allowing for more targeted interventions.

8 Conclusion

The models showed moderate accuracy in predicting fatigue, with the refined neural network providing the best-calibrated results. Class imbalance and overconfident predictions limited overall performance. Future work should focus on improving calibration, addressing imbalance, and testing on independent datasets to strengthen reliability.

9 Reference

Clifford, C., Löwe, B., & Kohlmann, S. (2024). Characteristics and predictors of persistent somatic symptoms in patients with cardiac disease. Scientific Reports, 14(1), 25517. https://doi.org/10.1038/s41598-024-76554-z

Creed, F. (2022). The predictors of somatic symptoms in a population sample: The lifelines cohort study. Psychosomatic Medicine, 84(9), 1056–1066. https://doi.org/10.1097/PSY.0000000000001101

Grahe, J. E., Chalk, H. M., Cramblet Alvarez, L. D., Faas, C. S., Hermann, A. D., & McFall, J. P. (2018). Emerging adulthood measured at multiple institutions 2: The data. Journal of Open Psychology Data, 6(1), 4. https://doi.org/10.5334/jopd.38

Löwe, B., Andresen, V., Van Den Bergh, O., Huber, T. B., Von Dem Knesebeck, O., Lohse, A. W., Nestoriuc, Y., Schneider, G., Schneider, S. W., Schramm, C., Ständer, S., Vettorazzi, E., Zapf, A., Shedden-Mora, M., & Toussaint, A. (2022). Persistent SOMAtic symptoms ACROSS diseases—from risk factors to modification: Scientific framework and overarching protocol of the interdisciplinary SOMACROSS research unit (RU 5211). BMJ Open, 12(1), e057596. https://doi.org/10.1136/bmjopen-2021-057596

Tomenson, B., Essau, C., Jacobi, F., Ladwig, K. H., Leiknes, K. A., Lieb, R., Meinlschmidt, G., McBeth, J., Rosmalen, J., Rief, W., & Sumathipala, A. (2013). Total somatic symptom score as a predictor of health outcome in somatic symptom disorders. British Journal of Psychiatry, 203(5), 373–380. https://doi.org/10.1192/bjp.bp.112.114405