train-models

December 31, 2025

[2]:
```
# Importing our libraries
import pandas as pd
import numpy as np
import pickle
import os
from sklearn.model_selection import train_test_split
from sklearn.multioutput import MultiOutputClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
import xgboost as xgb
```

[3]:
```
# 1. SETUP
os.makedirs("models", exist_ok=True)
```

[4]:
```
# 2. LOAD DATA (Prioritize the Cleaned "No Sleep" File)
if os.path.exists("final_dataset_no_sleep.csv"):
    print(" Loading pre-cleaned dataset: 'final_dataset_no_sleep.csv'")
    df = pd.read_csv("final_dataset_no_sleep.csv")

    # Since the file is already cleaned, we just take all Q columns
    feature_cols = [c for c in df.columns if "Q" in c]

else:
    # Fallback: Load original/corrected and clean in-memory
    filename = "corrected_mental_health_dataset.csv"
    if not os.path.exists(filename):
        filename = "synthetic_college_mental_health_dataset_v3_with_severity.
 ↪csv"
        print(f" Clean file not found. Using '{filename}' and cleaning␣
 ↪in-memory.")
    else:
        print(f" Clean file not found. Using '{filename}' and cleaning␣
 ↪in-memory.")

    df = pd.read_csv(filename)
```

```
    # A. Apply Clinical Hierarchy (Safety Check)
    overlap_mask = (df['ASD_risk'] == 1) & (df['SPCD_risk'] == 1)
    df.loc[overlap_mask, 'SPCD_risk'] = 0

    # B. REMOVE SLEEP QUESTIONS (Madam's Requirement)
    all_q_cols = [c for c in df.columns if "Q" in c]
    # Filter out any column containing "CONF"
    feature_cols = [c for c in all_q_cols if "CONF" not in c]

print(f"Training on {len(feature_cols)} features (27 Expected).")

risk_cols = ["ADHD_risk", "ASD_risk", "SPCD_risk", "DEP_risk", "ANX_risk"]

X = df[feature_cols]
y = df[risk_cols]
```

```
 Loading pre-cleaned dataset: 'final_dataset_no_sleep.csv'
Training on 27 features (27 Expected).
```

[5]:
```
# 3. SPLIT DATA
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Save Test Data for the Report Generator
pickle.dump((X_test, y_test), open("models/test_data.pkl", "wb"))
print("Saved 'models/test_data.pkl' for graph generation.")
```

```
Saved 'models/test_data.pkl' for graph generation.
```

[6]:
```
# 4. TRAIN ALL 4 RISK MODELS
models = {
    "RF": MultiOutputClassifier(RandomForestClassifier(n_estimators=100,␣
 ↪random_state=42)),
    # Removed use_label_encoder=False to fix UserWarning
    "XGB": MultiOutputClassifier(xgb.XGBClassifier(eval_metric='logloss',␣
 ↪random_state=42)),
    "SVM": MultiOutputClassifier(SVC(kernel='linear', probability=True,␣
 ↪random_state=42)),
    "KNN": MultiOutputClassifier(KNeighborsClassifier(n_neighbors=5))
}

for name, model in models.items():
    print(f"Training {name} Risk Model...")
    model.fit(X_train, y_train)
    pickle.dump(model, open(f"models/{name}_risk.pkl", "wb"))
```

```
Training RF Risk Model…
```

```
Training XGB Risk Model…
Training SVM Risk Model…
Training KNN Risk Model…
```

[7]:
```python
# 5. TRAIN SEVERITY MODELS (Random Forest)
SEVERITY_MAPPING = {'Low': 0, 'Medium': 1, 'High': 2}
severity_map_cols = {
    "ADHD": "ADHD_severity", "ASD": "ASD_severity", "SPCD": "SPCD_severity",
    "DEP": "DEP_severity", "ANX": "ANX_severity"
}

print("\nTraining Severity Models (RF)...")
for disorder, sev_col in severity_map_cols.items():
    risk_col = f"{disorder}_risk"

    # Train only on rows where the disorder exists
    sev_df = df[df[risk_col] == 1].copy()

    X_sev = sev_df[feature_cols] # 27 cols
    y_sev = sev_df[sev_col].map(SEVERITY_MAPPING).fillna(0)

    rf_sev = RandomForestClassifier(n_estimators=100, max_depth=8,␣
 ↪random_state=42)
    rf_sev.fit(X_sev, y_sev)
    pickle.dump(rf_sev, open(f"models/rf_{disorder}_sev.pkl", "wb"))

print("\n  All models retrained successfully!")
```

```
Training Severity Models (RF)…

  All models retrained successfully!
```