Symptom_Selection

April 26, 2025

```
[ ]: ! pip install seaborn
```

```
Requirement already satisfied: seaborn in
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-packages (from
seaborn) (2.2.5)
Requirement already satisfied: pandas>=1.2 in
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-packages (from
seaborn) (2.2.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-packages (from
seaborn) (3.10.1)
Requirement already satisfied: contourpy>=1.0.1 in
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (1.3.2)
Requirement already satisfied: cycler>=0.10 in
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (1.4.8)
Requirement already satisfied: packaging>=20.0 in
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (24.2)
Requirement already satisfied: pillow>=8 in
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
```

```
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-packages (from
pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-packages (from
pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: six>=1.5 in
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-packages (from
python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.17.0)
```

[ ]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪roc_auc_score, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
import os
```

[ ]:
```
## Prepare the symptom prediction dataset

# Load the Alzheimer's dataset
# Dataset source: https://www.kaggle.com/datasets/rabieelkharoua/
 ↪alzheimers-disease-dataset
data_path = "/Users/zhengfeibian/Desktop/5630final/MyOwnChooseDataSets/
 ↪alzheimers_disease_data.csv"

df = pd.read_csv(data_path)

# Preview the first few rows
df.head()
```

[ ]:
```
   PatientID  Age  Gender  Ethnicity  EducationLevel        BMI  Smoking  \
0       4751   73       0          0               2  22.927749        0
1       4752   89       0          0               0  26.827681        0
2       4753   73       0          3               1  17.795882        0
3       4754   74       1          0               1  33.800817        1
4       4755   89       0          0               0  20.716974        0

   AlcoholConsumption  PhysicalActivity  DietQuality  …  MemoryComplaints  \
0           13.297218          6.327112     1.347214  …                 0
1            4.542524          7.619885     0.518767  …                 0
2           19.555085          7.844988     1.826335  …                 0
3           12.209266          8.428001     7.435604  …                 0
4           18.454356          6.310461     0.795498  …                 0
```

```
   BehavioralProblems       ADL  Confusion  Disorientation  \
0                   0  1.725883          0               0
1                   0  2.592424          0               0
2                   0  7.119548          0               1
3                   1  6.481226          0               0
4                   0  0.014691          0               0

   PersonalityChanges  DifficultyCompletingTasks  Forgetfulness  Diagnosis  \
0                   0                          1              0          0
1                   0                          0              1          0
2                   0                          1              0          0
3                   0                          0              0          0
4                   1                          1              0          0

   DoctorInCharge
0       XXXConfid
1       XXXConfid
2       XXXConfid
3       XXXConfid
4       XXXConfid

[5 rows x 35 columns]
```

[ ]:
```
# Print all column names
print(list(df.columns))
```

```
['PatientID', 'Age', 'Gender', 'Ethnicity', 'EducationLevel', 'BMI', 'Smoking',
'AlcoholConsumption', 'PhysicalActivity', 'DietQuality', 'SleepQuality',
'FamilyHistoryAlzheimers', 'CardiovascularDisease', 'Diabetes', 'Depression',
'HeadInjury', 'Hypertension', 'SystolicBP', 'DiastolicBP', 'CholesterolTotal',
'CholesterolLDL', 'CholesterolHDL', 'CholesterolTriglycerides', 'MMSE',
'FunctionalAssessment', 'MemoryComplaints', 'BehavioralProblems', 'ADL',
'Confusion', 'Disorientation', 'PersonalityChanges',
'DifficultyCompletingTasks', 'Forgetfulness', 'Diagnosis', 'DoctorInCharge']
```

[ ]:
```
# Data Cleaning
# Drop irrelevant columns like Patient ID and Doctor in Charge
df_clean = df.drop(columns=["PatientID", "DoctorInCharge"])

# Define Target Symptoms
# We will predict multiple symptoms separately
target_variables = ["Forgetfulness", "Confusion", "Disorientation",␣
 ↪"PersonalityChanges"]

# Set Output Directory
output_dir = "/Users/zhengfeibian/Desktop/5630final/MyOwnChooseDataSets"
```

```python
# For each target variable, prepare training and testing sets
for target_variable in target_variables:
    print(f"\nPreparing data for target variable: {target_variable}")

    # Split features (X) and label (y)
    X = df_clean.drop(columns=target_variables)  # Features (drop all targets)
    y = df_clean[target_variable]                # Current label

    # Split into train and test sets (80/20) with stratification
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42, stratify=y
    )

    # Merge features and labels
    train_df = X_train.copy()
    train_df[target_variable] = y_train

    test_df = X_test.copy()
    test_df[target_variable] = y_test

    # Save train and test sets
    train_path = os.path.join(output_dir, f"train_{target_variable.lower()}.
 ↪csv")
    test_path = os.path.join(output_dir, f"test_{target_variable.lower()}.csv")

    train_df.to_csv(train_path, index=False)
    test_df.to_csv(test_path, index=False)

    print(f"Saved train set: {train_path}")
    print(f"Saved test set: {test_path}")
```

```
Preparing data for target variable: Forgetfulness
Saved train set: /Users/zhengfeibian/Desktop/5630final/MyOwnChooseDataSets/train
_forgetfulness.csv
Saved test set:
/Users/zhengfeibian/Desktop/5630final/MyOwnChooseDataSets/test_forgetfulness.csv

Preparing data for target variable: Confusion
Saved train set:
/Users/zhengfeibian/Desktop/5630final/MyOwnChooseDataSets/train_confusion.csv
Saved test set:
/Users/zhengfeibian/Desktop/5630final/MyOwnChooseDataSets/test_confusion.csv

Preparing data for target variable: Disorientation
Saved train set: /Users/zhengfeibian/Desktop/5630final/MyOwnChooseDataSets/train
_disorientation.csv
Saved test set: /Users/zhengfeibian/Desktop/5630final/MyOwnChooseDataSets/test_d
```

```
isorientation.csv

Preparing data for target variable: PersonalityChanges
Saved train set: /Users/zhengfeibian/Desktop/5630final/MyOwnChooseDataSets/train
_personalitychanges.csv
Saved test set: /Users/zhengfeibian/Desktop/5630final/MyOwnChooseDataSets/test_p
ersonalitychanges.csv
```

[ ]:
```
# Symptom occurrence statistics
# Count number of patients showing each symptom
symptom_counts = df_clean[["Forgetfulness", "Confusion", "Disorientation",␣
↪"PersonalityChanges"]].sum()
print(symptom_counts)
```

```
Forgetfulness         648
Confusion             441
Disorientation        340
PersonalityChanges    324
dtype: int64
```

[ ]:
```
from sklearn.preprocessing import StandardScaler
```

[ ]:
```
# Train Logistic Regression Models
output_dir = "/Users/zhengfeibian/Desktop/5630final/MyOwnChooseDataSets"

# To store logistic regression results
results = []

for target_variable in target_variables:
    print(f"\nTraining Logistic Regression for target: {target_variable}")

    # Load train and test sets
    train_path = f"{output_dir}/train_{target_variable.lower()}.csv"
    test_path = f"{output_dir}/test_{target_variable.lower()}.csv"

    train_df = pd.read_csv(train_path)
    test_df = pd.read_csv(test_path)

    # Separate features and label
    X_train = train_df.drop(columns=[target_variable])
    y_train = train_df[target_variable]
    X_test = test_df.drop(columns=[target_variable])
    y_test = test_df[target_variable]

    # Standardize features
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
    # Build and train Logistic Regression model
    lr_model = LogisticRegression(class_weight='balanced', max_iter=1000,␣
  ↪random_state=42)
    lr_model.fit(X_train_scaled, y_train)

    # Predict
    y_pred = lr_model.predict(X_test_scaled)
    y_prob = lr_model.predict_proba(X_test_scaled)[:, 1]

    # Evaluate
    acc = accuracy_score(y_test, y_pred)
    auc = roc_auc_score(y_test, y_prob)
    report = classification_report(y_test, y_pred, digits=4)
    cm = confusion_matrix(y_test, y_pred)

    print(f"Accuracy: {acc:.4f}")
    print(f"ROC-AUC: {auc:.4f}")
    print(f"Classification Report:\n{report}")
    print(f"Confusion Matrix:\n{cm}")

    # Save performance
    results.append({
        "Target": target_variable,
        "Accuracy": acc,
        "ROC_AUC": auc
    })
```

```
Training Logistic Regression for target: Forgetfulness
Accuracy: 0.4977
ROC-AUC: 0.4721
Classification Report:
              precision    recall  f1-score   support

           0     0.6736    0.5433    0.6015       300
           1     0.2713    0.3923    0.3208       130

    accuracy                         0.4977       430
   macro avg     0.4724    0.4678    0.4611       430
weighted avg     0.5519    0.4977    0.5166       430

Confusion Matrix:
[[163 137]
 [ 79  51]]

Training Logistic Regression for target: Confusion
Accuracy: 0.4628
```

```
ROC-AUC: 0.4671
Classification Report:
              precision    recall  f1-score   support

           0     0.7789    0.4532    0.5730       342
           1     0.1905    0.5000    0.2759        88

    accuracy                         0.4628       430
   macro avg     0.4847    0.4766    0.4244       430
weighted avg     0.6585    0.4628    0.5122       430

Confusion Matrix:
[[155 187]
 [ 44  44]]

Training Logistic Regression for target: Disorientation
Accuracy: 0.5326
ROC-AUC: 0.5375
Classification Report:
              precision    recall  f1-score   support

           0     0.8676    0.5249    0.6540       362
           1     0.1848    0.5735    0.2796        68

    accuracy                         0.5326       430
   macro avg     0.5262    0.5492    0.4668       430
weighted avg     0.7596    0.5326    0.5948       430

Confusion Matrix:
[[190 172]
 [ 29  39]]

Training Logistic Regression for target: PersonalityChanges
Accuracy: 0.5349
ROC-AUC: 0.4395
Classification Report:
              precision    recall  f1-score   support

           0     0.8367    0.5616    0.6721       365
           1     0.1351    0.3846    0.2000        65

    accuracy                         0.5349       430
   macro avg     0.4859    0.4731    0.4361       430
weighted avg     0.7307    0.5349    0.6008       430

Confusion Matrix:
[[205 160]
 [ 40  25]]
[ ]: # Summary of Logistic Regression Results
     import pandas as pd

     lr_results_df = pd.DataFrame(results)
     print("\nLogistic Regression Results Summary:")
     print(lr_results_df)
```

```
Logistic Regression Results Summary:
               Target  Accuracy   ROC_AUC
0       Forgetfulness  0.497674  0.472051
1           Confusion  0.462791  0.467072
2      Disorientation  0.532558  0.537496
3   PersonalityChanges  0.534884  0.439452
```

```
[ ]: # Train Random Forest Models
     output_dir = "/Users/zhengfeibian/Desktop/5630final/MyOwnChooseDataSets"

     # To store random forest results
     rf_results = []

     for target_variable in target_variables:
         print(f"\nTraining Random Forest for target: {target_variable}")

         # Load train and test sets
         train_path = f"{output_dir}/train_{target_variable.lower()}.csv"
         test_path = f"{output_dir}/test_{target_variable.lower()}.csv"

         train_df = pd.read_csv(train_path)
         test_df = pd.read_csv(test_path)

         # Separate features and label
         X_train = train_df.drop(columns=[target_variable])
         y_train = train_df[target_variable]
         X_test = test_df.drop(columns=[target_variable])
         y_test = test_df[target_variable]

         # Random Forest does not require feature scaling
         rf_model = RandomForestClassifier(
             n_estimators=100,
             max_depth=None,
             class_weight='balanced',
             random_state=42
         )
         rf_model.fit(X_train, y_train)

         # Predict
         y_pred = rf_model.predict(X_test)
```

```python
    y_prob = rf_model.predict_proba(X_test)[:, 1]

    # Evaluate
    acc = accuracy_score(y_test, y_pred)
    auc = roc_auc_score(y_test, y_prob)
    report = classification_report(y_test, y_pred, digits=4)
    cm = confusion_matrix(y_test, y_pred)

    print(f"Accuracy: {acc:.4f}")
    print(f"ROC-AUC: {auc:.4f}")
    print(f"Classification Report:\n{report}")
    print(f"Confusion Matrix:\n{cm}")

    # Save performance
    rf_results.append({
        "Target": target_variable,
        "Accuracy": acc,
        "ROC_AUC": auc
    })
```

```
Training Random Forest for target: Forgetfulness
Accuracy: 0.6977
ROC-AUC: 0.5057
Classification Report:
              precision    recall  f1-score   support

           0     0.6986    0.9967    0.8214       300
           1     0.5000    0.0077    0.0152       130

    accuracy                         0.6977       430
   macro avg     0.5993    0.5022    0.4183       430
weighted avg     0.6386    0.6977    0.5777       430

Confusion Matrix:
[[299   1]
 [129   1]]

Training Random Forest for target: Confusion
```

```
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
```

```
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
Accuracy: 0.7953
ROC-AUC: 0.4596
Classification Report:
              precision    recall  f1-score   support

           0     0.7953    1.0000    0.8860       342
           1     0.0000    0.0000    0.0000        88

    accuracy                         0.7953       430
   macro avg     0.3977    0.5000    0.4430       430
weighted avg     0.6326    0.7953    0.7047       430

Confusion Matrix:
[[342   0]
 [ 88   0]]

Training Random Forest for target: Disorientation
```

```
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
Accuracy: 0.8419
ROC-AUC: 0.4852
Classification Report:
              precision    recall  f1-score   support

           0     0.8419    1.0000    0.9141       362
           1     0.0000    0.0000    0.0000        68
```

```
    accuracy                         0.8419       430
   macro avg     0.4209    0.5000    0.4571       430
weighted avg     0.7087    0.8419    0.7696       430

Confusion Matrix:
[[362   0]
 [ 68   0]]

Training Random Forest for target: PersonalityChanges
Accuracy: 0.8488
ROC-AUC: 0.5329
Classification Report:
              precision    recall  f1-score   support

           0     0.8488    1.0000    0.9182       365
           1     0.0000    0.0000    0.0000        65

    accuracy                         0.8488       430
   macro avg     0.4244    0.5000    0.4591       430
weighted avg     0.7205    0.8488    0.7794       430

Confusion Matrix:
[[365   0]
 [ 65   0]]
```

```
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/zhengfeibian/anaconda3/envs/happy/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Among the Alzheimer's disease symptoms analyzed, Forgetfulness shows the highest prevalence, suggesting it should be prioritized when designing keyword extraction strategies. Disorientation and Personality Changes, although less prevalent, demonstrated higher predictability, indicating their clear pattern within patient characteristics."

[ ]:
```python
# Summary of Random Forest Results
import pandas as pd

rf_results_df = pd.DataFrame(rf_results)
print("\nRandom Forest Results Summary:")
print(rf_results_df)
```

```
Random Forest Results Summary:
              Target  Accuracy   ROC_AUC
0      Forgetfulness  0.697674  0.505731
1          Confusion  0.795349  0.459629
2     Disorientation  0.841860  0.485172
3  PersonalityChanges  0.848837  0.532919
```