Symptom_Forgetfulness

April 26, 2025

[ ]:
```
! pip install biopython # Install biopython package to interact with PubMed
```

```
Collecting biopython
  Downloading biopython-1.85-cp311-cp311-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (13 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages
(from biopython) (2.0.2)
Downloading
biopython-1.85-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.3
MB)
                         3.3/3.3 MB
34.6 MB/s eta 0:00:00
Installing collected packages: biopython
Successfully installed biopython-1.85
```

[ ]:
```
# Import the Entrez module from biopython
from Bio import Entrez
import time

# Set the email for interact
Entrez.email = "annabian1122@gmail.com"

# Search for articles using a keyword
search_term = "Alzheimer's disease AND English[lang]"
handle = Entrez.esearch(db="pubmed", term=search_term, retmax=20000) # Search␣
 ↪PubMed with a limit of 20,000 results
record = Entrez.read(handle) # Read the search results
handle.close() # Close the handle after reading

# Get the list of PubMed IDs (PMIDs) from the search results
id_list = record["IdList"]
print(f"Number of articles found: {len(id_list)}")

# Fetch abstracts of the articles using the PubMed IDs
handle = Entrez.efetch(db="pubmed", id=",".join(id_list), rettype="abstract",␣
 ↪retmode="xml")
records = Entrez.read(handle) # Read the fetched data
handle.close() # Close the handle after reading
```

```
batch_size = 500

# Initialize lists to store abstracts and corresponding PMIDs
abstracts = []
pmids = []

import pandas as pd # Import pandas for data manipulation

for start in range(0, len(id_list), batch_size):
    end = min(start + batch_size, len(id_list))
    batch_ids = id_list[start:end]

    print(f"Fetching records {start+1} to {end}...")

    handle = Entrez.efetch(db="pubmed", id=",".join(batch_ids),␣
 ↪rettype="abstract", retmode="xml") # Fetch abstracts in XML format
    batch_records = Entrez.read(handle) # Read the fetched data
    handle.close() # Close the handle after reading

    # Loop through the fetched articles and extract abstract text and PMIDs
    for article in batch_records['PubmedArticle']:
        try:
            # Extract the abstract text and join it into a single string
            abstract_text =␣
 ↪article['MedlineCitation']['Article']['Abstract']['AbstractText']
            abstract_str = " ".join(abstract_text)
            # Extract the PubMed ID (PMID)
            pmid = article['MedlineCitation']['PMID']
            # Append abstract and PMID to the lists
            abstracts.append(abstract_str)
            pmids.append(pmid)
        except:
            # Skip articles without an abstract
            continue

    time.sleep(0.5)


# Save the extracted data into a DataFrame (ensure only articles with abstracts␣
 ↪are included)
df = pd.DataFrame({
    "PMID": pmids,
    "Abstract": abstracts
})

# Save the DataFrame to a CSV file
```

```
csv_path = "alzheimers_abstracts.csv"
df.to_csv(csv_path, index=False)

print(f"Saved {len(df)} abstracts to the CSV file: {csv_path}") # Print the␣
 ↪number of abstracts saved
```

```
Number of articles found: 9999
Fetching records 1 to 500…
Fetching records 501 to 1000…
Fetching records 1001 to 1500…
Fetching records 1501 to 2000…
Fetching records 2001 to 2500…
Fetching records 2501 to 3000…
Fetching records 3001 to 3500…
Fetching records 3501 to 4000…
Fetching records 4001 to 4500…
Fetching records 4501 to 5000…
Fetching records 5001 to 5500…
Fetching records 5501 to 6000…
Fetching records 6001 to 6500…
Fetching records 6501 to 7000…
Fetching records 7001 to 7500…
Fetching records 7501 to 8000…
Fetching records 8001 to 8500…
Fetching records 8501 to 9000…
Fetching records 9001 to 9500…
Fetching records 9501 to 9999…
Saved 9611 abstracts to the CSV file: alzheimers_abstracts.csv
```

[ ]:
```
print(f"len(id_list): {len(id_list)}")
print(f"len(abstracts): {len(abstracts)}")
```

```
len(id_list): 9999
len(abstracts): 9611
```

[ ]:
```
import pandas as pd

# Read the cleaned and saved abstracts data
df = pd.read_csv("/content/alzheimers_abstracts.csv")

# Define a custom list of symptom-related keywords
symptom_keywords = [
    "memory loss", "forgetfulness", "cognitive decline", "confusion",
    "disorientation", "language impairment", "attention deficit", "behavioral␣
 ↪change",
    "mild cognitive impairment", "cognitive symptoms"
]
```

```
# Add a new column 'MentionsSymptom': mark 1 if any keyword is found in the␣
 ↪abstract, else 0
df["MentionsSymptom"] = df["Abstract"].apply(
    lambda text: int(any(kw.lower() in text.lower() for kw in symptom_keywords))
)

# Save the updated DataFrame as a new CSV file
df.to_csv("alzheimers_abstracts__symptoms.csv", index=False)
print("Successfully completed keyword tagging and saved the new file.")
```

```
Successfully completed keyword tagging and saved the new file.
```

[ ]:
```
! pip install spacy

! python -m spacy download en_core_web_sm
```

```
Requirement already satisfied: spacy in /usr/local/lib/python3.11/dist-packages
(3.8.5)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in
/usr/local/lib/python3.11/dist-packages (from spacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in
/usr/local/lib/python3.11/dist-packages (from spacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in
/usr/local/lib/python3.11/dist-packages (from spacy) (1.0.12)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in
/usr/local/lib/python3.11/dist-packages (from spacy) (2.0.11)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in
/usr/local/lib/python3.11/dist-packages (from spacy) (3.0.9)
Requirement already satisfied: thinc<8.4.0,>=8.3.4 in
/usr/local/lib/python3.11/dist-packages (from spacy) (8.3.6)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in
/usr/local/lib/python3.11/dist-packages (from spacy) (1.1.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in
/usr/local/lib/python3.11/dist-packages (from spacy) (2.5.1)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in
/usr/local/lib/python3.11/dist-packages (from spacy) (2.0.10)
Requirement already satisfied: weasel<0.5.0,>=0.1.0 in
/usr/local/lib/python3.11/dist-packages (from spacy) (0.4.1)
Requirement already satisfied: typer<1.0.0,>=0.3.0 in
/usr/local/lib/python3.11/dist-packages (from spacy) (0.15.2)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in
/usr/local/lib/python3.11/dist-packages (from spacy) (4.67.1)
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.11/dist-
packages (from spacy) (2.0.2)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in
/usr/local/lib/python3.11/dist-packages (from spacy) (2.32.3)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in
/usr/local/lib/python3.11/dist-packages (from spacy) (2.11.3)
```

```
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages
(from spacy) (3.1.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-
packages (from spacy) (75.2.0)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from spacy) (24.2)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in
/usr/local/lib/python3.11/dist-packages (from spacy) (3.5.0)
Requirement already satisfied: language-data>=1.2 in
/usr/local/lib/python3.11/dist-packages (from langcodes<4.0.0,>=3.2.0->spacy)
(1.3.0)
Requirement already satisfied: annotated-types>=0.6.0 in
/usr/local/lib/python3.11/dist-packages (from
pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (0.7.0)
Requirement already satisfied: pydantic-core==2.33.1 in
/usr/local/lib/python3.11/dist-packages (from
pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (2.33.1)
Requirement already satisfied: typing-extensions>=4.12.2 in
/usr/local/lib/python3.11/dist-packages (from
pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (4.13.2)
Requirement already satisfied: typing-inspection>=0.4.0 in
/usr/local/lib/python3.11/dist-packages (from
pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (0.4.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0->spacy)
(3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests<3.0.0,>=2.13.0->spacy) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0->spacy)
(2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0->spacy)
(2025.1.31)
Requirement already satisfied: blis<1.4.0,>=1.3.0 in
/usr/local/lib/python3.11/dist-packages (from thinc<8.4.0,>=8.3.4->spacy)
(1.3.0)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in
/usr/local/lib/python3.11/dist-packages (from thinc<8.4.0,>=8.3.4->spacy)
(0.1.5)
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-
packages (from typer<1.0.0,>=0.3.0->spacy) (8.1.8)
Requirement already satisfied: shellingham>=1.3.0 in
/usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->spacy)
(1.5.4)
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-
packages (from typer<1.0.0,>=0.3.0->spacy) (13.9.4)
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in
```

```
/usr/local/lib/python3.11/dist-packages (from weasel<0.5.0,>=0.1.0->spacy)
(0.21.0)
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in
/usr/local/lib/python3.11/dist-packages (from weasel<0.5.0,>=0.1.0->spacy)
(7.1.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.11/dist-packages (from jinja2->spacy) (3.0.2)
Requirement already satisfied: marisa-trie>=1.1.0 in
/usr/local/lib/python3.11/dist-packages (from language-
data>=1.2->langcodes<4.0.0,>=3.2.0->spacy) (1.2.1)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.11/dist-packages (from
rich>=10.11.0->typer<1.0.0,>=0.3.0->spacy) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.11/dist-packages (from
rich>=10.11.0->typer<1.0.0,>=0.3.0->spacy) (2.18.0)
Requirement already satisfied: wrapt in /usr/local/lib/python3.11/dist-packages
(from smart-open<8.0.0,>=5.2.1->weasel<0.5.0,>=0.1.0->spacy) (1.17.2)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-
packages (from markdown-it-py>=2.2.0->rich>=10.11.0->typer<1.0.0,>=0.3.0->spacy)
(0.1.2)
Collecting en-core-web-sm==3.8.0
  Downloading https://github.com/explosion/spacy-
models/releases/download/en_core_web_sm-3.8.0/en_core_web_sm-3.8.0-py3-none-
any.whl (12.8 MB)
                           12.8/12.8 MB
123.6 MB/s eta 0:00:00
 Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
 Restart to reload dependencies
If you are in a Jupyter or Colab notebook, you may need to restart Python in
order to load all the package's dependencies. You can do this by selecting the
'Restart kernel' or 'Restart runtime' option.
```

[ ]:
```python
import pandas as pd
import spacy
from spacy.pipeline import EntityRuler

# Load the English spaCy model
nlp = spacy.load("en_core_web_sm")

# Add an EntityRuler to insert custom entity recognition rules before the␣
 ↪built-in NER
ruler = nlp.add_pipe("entity_ruler", before="ner")

# Define a list of symptom keywords
symptom_keywords = [
```

```
    "memory loss", "forgetfulness", "cognitive decline", "confusion",
    "disorientation", "language impairment", "attention deficit"
]

# Build patterns for the EntityRuler
patterns = [{"label": "SYMPTOM", "pattern": kw} for kw in symptom_keywords]
ruler.add_patterns(patterns)

# Test entity recognition on a single sample sentence
doc = nlp("The patient experienced memory loss and confusion over time.")
print("\nTesting entity recognition on a sample sentence:")
for ent in doc.ents:
    print(f" - {ent.text} ({ent.label_})")

# Read the abstracts dataset
df = pd.read_csv("/content/alzheimers_abstracts__symptoms.csv")

# Perform entity recognition on the first 5 abstracts
print("\nBatch processing entity recognition results for abstracts: ")
for i in range(5):
    text = df.loc[i, "Abstract"]
    print(f"\nAbstract #{i+1}:\n{text}")

    doc = nlp(text)

    print("Recognized Entities:")
    for ent in doc.ents:
        print(f" - {ent.text} ({ent.label_})")
```

```
Testing entity recognition on a sample sentence:
 - memory loss (SYMPTOM)
 - confusion (SYMPTOM)

Batch processing entity recognition results for abstracts:

Abstract #1:
Whether or not neuropsychiatric symptoms (NPS) in advance of dementia are
associated with Alzheimer disease (AD) and/or other neurodegenerative dementias
remains to be determined. The mild behavioural impairment (MBI) construct
selects persons with NPS that are later-life emergent and persistent to identify
a high-risk group for cognitive decline and incident dementia. Here, in older
adults without dementia at baseline, we examined whether postmortem AD and other
neurodegenerative pathologies were associated with MBI in the five years before
death. National Alzheimer's Coordinating Center study autopsy participants
(n=1016, 82.6 years, 48.7% female, 60% normal cognition) were included in the
analyses. Using the Neuropsychiatric Inventory-Questionnaire, MBI+ status was
operationalized as NPS persistence at >2/3 of pre-dementia study visits;
```

otherwise, status was non-MBI NPS. The presence of AD, Lewy body disease (LBD), and TDP-43 neuropathological changes were determined using published guidelines. Adjusted multinomial logistic regressions modeled pathology-NPS status associations. Adjusted Cox proportional hazards regressions modeled hazard for AD-dementia at each NPS status level, including interaction terms with cognitive status and each co-pathology. AD+ individuals (51.4%) were 88.4% more likely to be MBI+ 5 years prior than AD- individuals (odds ratio (OR):1.88, 95% confidence interval (CI):1.29-2.75, p<0.01); however, the likelihood of having non-MBI NPS was not different (OR:1.22, CI:0.90-1.66, p=0.20). No significant associations were seen for LBD pathology, even among AD+ participants. There were no significant differences in the levels of LBD or TDP-43 in those with MBI compared to no MBI. Among MBI progressors to dementia (n=106), 33.0% were solely AD+, 18.9% were mixed AD+/LBD+, and 11.3% had all three pathologies. For all those with MBI (including dementia non-progressors), of persons with LBD, 83.4% were comorbid with AD. In the survival analysis, MBI+ individuals had a 2.03-fold greater progression rate to AD-dementia than noNPS (CI: 1.60-2.57, p<0.01). Progression rate was higher in MCI, but the effect of MBI on progression was greater in NC (HR:3.05, CI:1.37-6.80, p<0.01) vs. MCI (HR:1.93, CI:1.51-2.47, p<0.01). Limbic LBD appeared to also moderate the association between MBI and incident AD (Limbic LBD+ HR: 4.64, CI: 2.05-10.50, p<0.001; Limbic LBD- HR: 1.87, CI: 1.46-2.40, p<0.001). Antecedent MBI was strongly associated with AD pathology but not with other neurodegenerative dementias. Inclusion of MBI in research and clinical frameworks for dementia may aid in identification of early stages of neurodegenerative disease, which may be helpful for selecting patients for treatment with AD disease-modifying drugs.
Recognized Entities:
- NPS (ORG)
- NPS (ORG)
- cognitive decline (SYMPTOM)
- the five years (DATE)
- National Alzheimer's (ORG)
- Coordinating Center (ORG)
- n=1016 (GPE)
- 82.6 years (DATE)
- 48.7% (PERCENT)
- 60% (PERCENT)
- the Neuropsychiatric Inventory-Questionnaire (FAC)
- NPS (ORG)
- 2/3 (CARDINAL)
- LBD (ORG)
- NPS (ORG)
- NPS (ORG)
- 51.4% (PERCENT)
- 88.4% (PERCENT)
- years (DATE)
- 95% (PERCENT)
- NPS (ORG)
- LBD (ORG)

- LBD (ORG)
- 33.0% (PERCENT)
- 18.9% (PERCENT)
- 11.3% (PERCENT)
- three (CARDINAL)
- LBD (ORG)
- 83.4% (PERCENT)
- 2.03-fold (CARDINAL)
- noNPS (PERSON)
- CI (ORG)
- 1.60-2.57 (CARDINAL)
- MCI (ORG)
- NC (GPE)
- HR:3.05 (ORG)
- CI:1.37-6.80 (ORG)
- MCI (ORG)
- Limbic LBD (ORG)
- 4.64 (CARDINAL)
- CI (ORG)
- 2.05 (CARDINAL)
- p<0.001 (ORG)
- Limbic LBD- (ORG)
- 1.87 (CARDINAL)
- CI (ORG)
- 1.46-2.40 (CARDINAL)

Abstract #2:
Neurodegenerative diseases, such as Alzheimer's and Parkinson's Disease, pose a significant healthcare burden to the aging population. Structural MRI brain parameters and accelerometry data from wearable devices have been proven to be useful predictors for these diseases but have been separately examined in the prior literature. This study aims to determine whether a combination of accelerometry data and MRI brain parameters may improve the detection and prognostication of Alzheimer's and Parkinson's disease, compared with MRI brain parameters alone. A cohort of 19,793 participants free of neurodegenerative disease at the time of imaging and accelerometry data capture from the UK Biobank with longitudinal follow-up was derived to test this hypothesis. Relevant structural MRI brain parameters, accelerometry data collected from wearable devices, standard polygenic risk scores and lifestyle information were obtained. Subsequent development of neurodegenerative diseases among participants was recorded (mean follow-up time of 5.9 years), with positive cases defined as those diagnosed at least one year after imaging. A machine learning algorithm (XGBoost) was employed to create prediction models for the development of neurodegenerative disease. A prediction model consisting of all factors, including structural MRI brain parameters, accelerometry data, PRS, and lifestyle information, achieved the highest AUC value (0.819) out of all tested models. A model that excluded MRI brain parameters achieved the lowest AUC value (0.688). Feature importance analyses revealed 18 out of 20 most important

features were structural MRI brain parameters, while 2 were derived from accelerometry data. Our study demonstrates the potential utility of combining structural MRI brain parameters with accelerometry data from wearable devices to predict the incidence of neurodegenerative diseases. Future prospective studies across different populations should be conducted to confirm these study results and look for differences in predictive ability for various types of neurodegenerative diseases.
Recognized Entities:
- 19,793 (CARDINAL)
- UK (GPE)
- 5.9 years (DATE)
- at least one year (DATE)
- XGBoost (ORG)
- PRS (ORG)
- 0.819 (CARDINAL)
- 18 (CARDINAL)
- 20 (CARDINAL)
- 2 (CARDINAL)

Abstract #3:
Recent reports suggest dysregulation of the N6-methyladenosine (m6A) RNA modification may contribute to the pathology of neurodegenerative diseases. Herein, we show the m6A methyltransferase complex including METTL3-the catalytic component of the nuclear-localized complex-is robustly upregulated in human microglia and astrocytes exposed to Syn_f and Mn. Subcellular localization studies reveal METTL3 was predominantly cytoplasmic following Mn insult but remained nuclear following Syn_f stimulation in activated microglia. Functional analysis revealed METTL3 and downstream m6A readers, including YTHDF2 and IGF2BP1-3, may regulate the proinflammatory secretome of activated microglia. Notably, methyltransferase activity and m6A abundance were significantly increased following Mn and Syn_f treatment. METTL3 in Mn and Syn_fin vivo models of neuroinflammation, along with human postmortem tissues from Alzheimer's disease (AD), Parkinson's disease (PD), and dementia with Lewy bodies (DLB) patients, was significantly upregulated. This was further confirmed by single-cell RNA sequencing (scRNA-seq) analysis. Overall, we demonstrate the m6A writer METTL3 may function as a major regulator of chronic neuroinflammation in synucleinopathies.
Recognized Entities:
- N6-methyladenosine (DATE)
- Herein (PERSON)
- METTL3 (PRODUCT)
- Syn (ORG)
- METTL3 (PRODUCT)
- Syn (ORG)
- METTL3 (PRODUCT)
- YTHDF2 (ORG)
- IGF2BP1-3 (DATE)
- Mn (ORG)

- Syn (ORG)
- Syn (ORG)
- Lewy (ORG)
- DLB (ORG)
- RNA (ORG)
- METTL3 (PRODUCT)

Abstract #4:
Aging is a slow and irreversible biological process leading to decreased cell and tissue functions with higher risks of multiple age-related diseases, including neurodegenerative diseases. It is widely accepted that aging represents the leading risk factor for neurodegeneration. The pathogenesis of these diseases involves complex interactions of genetic mutations, environmental factors, oxidative stress, neuroinflammation, and mitochondrial dysfunction, which complicate treatment with traditional mono-targeted therapies. Network pharmacology can help identify potential gene or protein targets related to neurodegenerative diseases. Integrating advanced molecular profiling technologies and computer-aided drug design further enhances the potential of network pharmacology, enabling the identification of biomarkers and therapeutic targets, thus paving the way for precision medicine in neurodegenerative diseases. This review article delves into the application of network pharmacology in understanding and treating neurodegenerative disorders such as Alzheimer's disease, Parkinson's disease, amyotrophic lateral sclerosis, Huntington's disease, and spinal muscular atrophy. Overall, this article emphasizes the importance of addressing aging as a central factor in developing effective disease-modifying therapies, highlighting how network pharmacology can unravel the complex biological networks associated with aging and pave the way for personalized medical strategies.
Recognized Entities:
- Huntington (ORG)

Abstract #5:
Alzheimer's disease (AD) is an age-related neurodegenerative disorder characterized by insidious and gradual onset. Identifying biomarkers associated with the early stages of AD is crucial for delaying its progression. In this study, we aimed to identify AD-related biomarkers in blood and urine by integrating genetic correlation analysis, shared genetic loci identification, and causal inference using linkage disequilibrium score regression (LDSC), conjunction false discovery rate (conjFDR), generalized summary data-based Mendelian randomization (GSMR) and two-sample Mendelian randomization (MR). To enhance robustness and minimize sample bias, we cross-validated findings using different AD GWAS datasets. Across multiple AD GWASs, we consistently observed nominally significant genetic correlations: AD was positively correlated with albumin (ALB) and negatively correlated with cystatin C (CYS) and urea (BUN). MR analysis further suggested that genetic predisposition to higher level of ALB and lower level of non-albumin protein (NAP) can represent risk factors for AD. In reverse MR analysis, a higher genetic risk for AD can predispose individuals to higher levels of ratio of aspartate aminotransferase to alanine

```
aminotransferase (AST2ALT) and estimated glomerular filtration rate (EGFR), as
well as lower level of creatinine (CRE). Overall, this study provides insights
into the genetic correlations and causal relationships between AD and several
biomarkers, offering potential candidates for AD diagnosis and management.
Recognized Entities:
 - LDSC (PERSON)
 - Mendelian (NORP)
 - two (CARDINAL)
 - Mendelian (NORP)
 - ALB (ORG)
 - cystatin C (PERSON)
 - CYS (ORG)
 - BUN (ORG)
 - ALB (ORG)
 - NAP (ORG)
 - MR (GPE)
```

[ ]:
```
from sklearn.model_selection import train_test_split

# Read preprocessed dataset (with symptom labels)
df = pd.read_csv("/content/alzheimers_abstracts__symptoms.csv")

# Split the data into training set and temporary set (dev + test)
train_df, temp_df = train_test_split(df, test_size=0.3, random_state=42,␣
 ↪stratify=df["MentionsSymptom"])
# 30% of data goes to dev + test, 70% to train

# Further split the temporary set into dev (validation) and test sets (50% each)
dev_df, test_df = train_test_split(temp_df, test_size=0.5, random_state=42,␣
 ↪stratify=temp_df["MentionsSymptom"])

# Save the splits into separate CSV files
train_df.to_csv("symptom_train.csv", index=False)
dev_df.to_csv("symptom_dev.csv", index=False)
test_df.to_csv("symptom_test.csv", index=False)

print("Symptom recognition train/validation/test sets have been saved.")
```

```
Symptom recognition train/validation/test sets have been saved.
```

[ ]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
import pandas as pd

# Read the previously saved training and testing datasets
train_df = pd.read_csv("/content/symptom_train.csv")
test_df = pd.read_csv("/content/symptom_test.csv")
```

```
# Split features and labels
X_train = train_df["Abstract"] # Extract the abstract text for training
y_train = train_df["MentionsSymptom"] # Extract the label (whether symptom is␣
 ↪mentioned: 0 or 1) for training
X_test = test_df["Abstract"] # Extract the abstract text for testing
y_test = test_df["MentionsSymptom"] # Extract the label for testing

# Apply TF-IDF vectorization to the text
vectorizer = TfidfVectorizer(max_features=5000) # Keep only the top 5000 most␣
 ↪frequent words
X_train_vec = vectorizer.fit_transform(X_train) # Fit on training data and␣
 ↪transform it
X_test_vec = vectorizer.transform(X_test)

# Train a Logistic Regression model
clf = LogisticRegression(max_iter=200) # Initialize the model (allow up to 200␣
 ↪iterations to converge)
clf.fit(X_train_vec, y_train) # Train the model on the vectorized training data

# Predict and output results
y_pred = clf.predict(X_test_vec) # Predict the labels for test data
print(classification_report(y_test, y_pred)) # Print precision, recall,␣
 ↪f1-score, and support for each class
```

```
              precision    recall  f1-score   support

           0       0.88      0.97      0.93      1127
           1       0.85      0.54      0.66       315

    accuracy                           0.88      1442
   macro avg       0.87      0.76      0.80      1442
weighted avg       0.88      0.88      0.87      1442
```

[ ]:
```
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪roc_auc_score

# Predict the labels for the test set
y_pred = clf.predict(X_test_vec) # Model's predicted class (0 or 1) for each␣
 ↪test s
y_prob = clf.predict_proba(X_test_vec)[:, 1] # Model's predicted probability of␣
 ↪class 1 for each sample

# Print the classification report (precision, recall, F1-score, support)
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
# Calculate and print the ROC-AUC score
auc = roc_auc_score(y_test, y_prob) # Area under the Receiver Operating␣
 ↪Characteristic curve
print(f"ROC-AUC: {auc:.4f}")

from sklearn.metrics import confusion_matrix

# Compute the confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.97      0.93      1127
           1       0.85      0.54      0.66       315

    accuracy                           0.88      1442
   macro avg       0.87      0.76      0.80      1442
weighted avg       0.88      0.88      0.87      1442

ROC-AUC: 0.9313
Confusion Matrix:
[[1098   29]
 [ 144  171]]
```

[ ]:
```
! pip install datasets
```

```
Collecting datasets
  Downloading datasets-3.5.0-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-
packages (from datasets) (3.18.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-
packages (from datasets) (2.0.2)
Requirement already satisfied: pyarrow>=15.0.0 in
/usr/local/lib/python3.11/dist-packages (from datasets) (18.1.0)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages
(from datasets) (2.2.2)
Requirement already satisfied: requests>=2.32.2 in
/usr/local/lib/python3.11/dist-packages (from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.11/dist-
packages (from datasets) (4.67.1)
Collecting xxhash (from datasets)
  Downloading
```

```
xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(12 kB)
Collecting multiprocess<0.70.17 (from datasets)
  Downloading multiprocess-0.70.16-py311-none-any.whl.metadata (7.2 kB)
Collecting fsspec<=2024.12.0,>=2023.1.0 (from
fsspec[http]<=2024.12.0,>=2023.1.0->datasets)
  Downloading fsspec-2024.12.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-
packages (from datasets) (3.11.15)
Requirement already satisfied: huggingface-hub>=0.24.0 in
/usr/local/lib/python3.11/dist-packages (from datasets) (0.30.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-
packages (from datasets) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-
packages (from datasets) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (2.6.1)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-
packages (from aiohttp->datasets) (25.3.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.6.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (6.4.3)
Requirement already satisfied: propcache>=0.2.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (0.3.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.20.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.24.0->datasets)
(4.13.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets)
(3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests>=2.32.2->datasets) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets)
(2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets)
(2025.1.31)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-
packages (from pandas->datasets) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-
```

```
packages (from pandas->datasets) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-
packages (from python-dateutil>=2.8.2->pandas->datasets) (1.17.0)
Downloading datasets-3.5.0-py3-none-any.whl (491 kB)
                         491.2/491.2 kB
12.5 MB/s eta 0:00:00
Downloading dill-0.3.8-py3-none-any.whl (116 kB)
                         116.3/116.3 kB
13.1 MB/s eta 0:00:00
Downloading fsspec-2024.12.0-py3-none-any.whl (183 kB)
                         183.9/183.9 kB
19.9 MB/s eta 0:00:00
Downloading multiprocess-0.70.16-py311-none-any.whl (143 kB)
                         143.5/143.5 kB
14.8 MB/s eta 0:00:00
Downloading
xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
                         194.8/194.8 kB
18.3 MB/s eta 0:00:00
Installing collected packages: xxhash, fsspec, dill, multiprocess,
datasets
  Attempting uninstall: fsspec
    Found existing installation: fsspec 2025.3.2
    Uninstalling fsspec-2025.3.2:
      Successfully uninstalled fsspec-2025.3.2
```

```
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.
gcsfs 2025.3.2 requires fsspec==2025.3.2, but you have fsspec 2024.12.0 which is
incompatible.
torch 2.6.0+cu124 requires nvidia-cublas-cu12==12.4.5.8; platform_system ==
"Linux" and platform_machine == "x86_64", but you have nvidia-cublas-cu12
12.5.3.2 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cuda-cupti-cu12==12.4.127; platform_system ==
"Linux" and platform_machine == "x86_64", but you have nvidia-cuda-cupti-cu12
12.5.82 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cuda-nvrtc-cu12==12.4.127; platform_system ==
"Linux" and platform_machine == "x86_64", but you have nvidia-cuda-nvrtc-cu12
12.5.82 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cuda-runtime-cu12==12.4.127; platform_system
== "Linux" and platform_machine == "x86_64", but you have nvidia-cuda-runtime-
cu12 12.5.82 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cudnn-cu12==9.1.0.70; platform_system ==
"Linux" and platform_machine == "x86_64", but you have nvidia-cudnn-cu12
9.3.0.75 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cufft-cu12==11.2.1.3; platform_system ==
"Linux" and platform_machine == "x86_64", but you have nvidia-cufft-cu12
11.2.3.61 which is incompatible.
torch 2.6.0+cu124 requires nvidia-curand-cu12==10.3.5.147; platform_system ==
"Linux" and platform_machine == "x86_64", but you have nvidia-curand-cu12
10.3.6.82 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cusolver-cu12==11.6.1.9; platform_system ==
"Linux" and platform_machine == "x86_64", but you have nvidia-cusolver-cu12
11.6.3.83 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cusparse-cu12==12.3.1.170; platform_system ==
"Linux" and platform_machine == "x86_64", but you have nvidia-cusparse-cu12
12.5.1.3 which is incompatible.
torch 2.6.0+cu124 requires nvidia-nvjitlink-cu12==12.4.127; platform_system ==
"Linux" and platform_machine == "x86_64", but you have nvidia-nvjitlink-cu12
12.5.82 which is incompatible.
Successfully installed datasets-3.5.0 dill-0.3.8 fsspec-2024.12.0
```

```
multiprocess-0.70.16 xxhash-3.5.0
```

[ ]:
```python
from transformers import DistilBertTokenizerFast,␣
 ↪DistilBertForSequenceClassification, Trainer, TrainingArguments
from datasets import Dataset
import pandas as pd
import numpy as np
from sklearn.metrics import classification_report, roc_auc_score

# Load training and testing datasets
df = pd.read_csv("/content/symptom_train.csv")
df_test = pd.read_csv("/content/symptom_test.csv")

# Load the DistilBERT tokenizer
tokenizer = DistilBertTokenizerFast.from_pretrained("distilbert-base-uncased")

# Data Preprocessing: prepare Huggingface Dataset objects
# Rename columns to 'text' and 'label' for Huggingface compatibility
train_dataset = Dataset.from_pandas(df[["Abstract", "MentionsSymptom"]].
 ↪rename(columns={"Abstract": "text", "MentionsSymptom": "label"}))
test_dataset = Dataset.from_pandas(df_test[["Abstract", "MentionsSymptom"]].
 ↪rename(columns={"Abstract": "text", "MentionsSymptom": "label"}))

# Tokenize the datasets
train_dataset = train_dataset.map(lambda x: tokenizer(x["text"],␣
 ↪truncation=True, padding="max_length"), batched=True)
test_dataset = test_dataset.map(lambda x: tokenizer(x["text"], truncation=True,␣
 ↪padding="max_length"), batched=True)

# Define label map and load model
id2label = {0: "No Symptom", 1: "Mentions Symptom"}
label2id = {"No Symptom": 0, "Mentions Symptom": 1}

# Load DistilBERT model for sequence classification
from transformers import AutoModelForSequenceClassification

model = AutoModelForSequenceClassification.from_pretrained(
    "distilbert-base-uncased",
    num_labels=2, # Binary classification (0 or 1)
    id2label=id2label,
    label2id=label2id
)


# Set training arguments
from transformers import TrainingArguments
```

```
from transformers import (AutoTokenizer, DataCollatorWithPadding,␣
 ↪AutoModelForSequenceClassification,
    TrainingArguments, Trainer)

training_args = TrainingArguments(
    output_dir="text_classification_model",
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=4,
    weight_decay=0.01,
    eval_strategy="epoch",                        # run eval at the end of each␣
 ↪epoch
    save_strategy="epoch",
    load_best_model_at_end=True,
    push_to_hub=False,
    report_to="none",
    fp16=True  # # Use FP16 (faster on GPUs) with Colab
)


# Define custom evaluation metrics
def compute_metrics(pred):
    labels = pred.label_ids
    preds = np.argmax(pred.predictions, axis=1)
    auc = roc_auc_score(labels, pred.predictions[:, 1])
    report = classification_report(labels, preds, output_dict=True)
    return {
        "accuracy": report["accuracy"],         # Overall accuracy
        "precision": report["1"]["precision"], # Precision for class '1'␣
 ↪(Mentions Symptom)
        "recall": report["1"]["recall"],        # Recall for class '1'
        "f1": report["1"]["f1-score"],          # F1-score for class '1'
        "roc_auc": auc                          # ROC-AUC score
    }


# Initialize Trainer object
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    compute_metrics=compute_metrics
)
```

```
trainer.train() # Start training
trainer.evaluate() # Final evaluation on the test set
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94:
UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
(https://huggingface.co/settings/tokens), set it as secret in your Google Colab
and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access
public models or datasets.
  warnings.warn(
```

```
tokenizer_config.json:   0%|          | 0.00/48.0 [00:00<?, ?B/s]
```

```
vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]
```

```
tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]
```

```
config.json:   0%|          | 0.00/483 [00:00<?, ?B/s]
```

```
Map:   0%|          | 0/6727 [00:00<?, ? examples/s]
```

```
Map:   0%|          | 0/1442 [00:00<?, ? examples/s]
```

```
Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed.
Falling back to regular HTTP download. For better performance, install the
package with: `pip install huggingface_hub[hf_xet]` or `pip install hf_xet`
WARNING:huggingface_hub.file_download:Xet Storage is enabled for this repo, but
the 'hf_xet' package is not installed. Falling back to regular HTTP download.
For better performance, install the package with: `pip install
huggingface_hub[hf_xet]` or `pip install hf_xet`
```

```
model.safetensors:   0%|          | 0.00/268M [00:00<?, ?B/s]
```

```
Some weights of DistilBertForSequenceClassification were not initialized from
the model checkpoint at distilbert-base-uncased and are newly initialized:
['classifier.bias', 'classifier.weight', 'pre_classifier.bias',
'pre_classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
[ ]: {'eval_loss': 0.06558195501565933,
  'eval_accuracy': 0.9882108183079057,
  'eval_precision': 0.9901315789473685,
  'eval_recall': 0.9555555555555556,
  'eval_f1': 0.9725363489499192,
  'eval_roc_auc': 0.9877776369346912,
```

```
 'eval_runtime': 5.5567,
 'eval_samples_per_second': 259.505,
 'eval_steps_per_second': 16.377,
 'epoch': 4.0}
```

[ ]:
```
import transformers
print(transformers.__version__)
```

```
4.51.3
```

[ ]:
```
! pip install evaluate
```

```
Collecting evaluate
  Downloading evaluate-0.4.3-py3-none-any.whl.metadata (9.2 kB)
Requirement already satisfied: datasets>=2.0.0 in
/usr/local/lib/python3.11/dist-packages (from evaluate) (3.5.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-
packages (from evaluate) (2.0.2)
Requirement already satisfied: dill in /usr/local/lib/python3.11/dist-packages
(from evaluate) (0.3.8)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages
(from evaluate) (2.2.2)
Requirement already satisfied: requests>=2.19.0 in
/usr/local/lib/python3.11/dist-packages (from evaluate) (2.32.3)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.11/dist-
packages (from evaluate) (4.67.1)
Requirement already satisfied: xxhash in /usr/local/lib/python3.11/dist-packages
(from evaluate) (3.5.0)
Requirement already satisfied: multiprocess in /usr/local/lib/python3.11/dist-
packages (from evaluate) (0.70.16)
Requirement already satisfied: fsspec>=2021.05.0 in
/usr/local/lib/python3.11/dist-packages (from fsspec[http]>=2021.05.0->evaluate)
(2024.12.0)
Requirement already satisfied: huggingface-hub>=0.7.0 in
/usr/local/lib/python3.11/dist-packages (from evaluate) (0.30.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-
packages (from evaluate) (24.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-
packages (from datasets>=2.0.0->evaluate) (3.18.0)
Requirement already satisfied: pyarrow>=15.0.0 in
/usr/local/lib/python3.11/dist-packages (from datasets>=2.0.0->evaluate)
(18.1.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-
packages (from datasets>=2.0.0->evaluate) (3.11.15)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-
packages (from datasets>=2.0.0->evaluate) (6.0.2)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.7.0->evaluate)
```

```
(4.13.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.19.0->evaluate)
(3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests>=2.19.0->evaluate) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.19.0->evaluate)
(2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.19.0->evaluate)
(2025.1.31)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas->evaluate) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-
packages (from pandas->evaluate) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-
packages (from pandas->evaluate) (2025.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.11/dist-packages (from
aiohttp->datasets>=2.0.0->evaluate) (2.6.1)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.11/dist-packages (from
aiohttp->datasets>=2.0.0->evaluate) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-
packages (from aiohttp->datasets>=2.0.0->evaluate) (25.3.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.11/dist-packages (from
aiohttp->datasets>=2.0.0->evaluate) (1.6.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.11/dist-packages (from
aiohttp->datasets>=2.0.0->evaluate) (6.4.3)
Requirement already satisfied: propcache>=0.2.0 in
/usr/local/lib/python3.11/dist-packages (from
aiohttp->datasets>=2.0.0->evaluate) (0.3.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in
/usr/local/lib/python3.11/dist-packages (from
aiohttp->datasets>=2.0.0->evaluate) (1.20.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-
packages (from python-dateutil>=2.8.2->pandas->evaluate) (1.17.0)
Downloading evaluate-0.4.3-py3-none-any.whl (84 kB)
                        84.0/84.0 kB
3.8 MB/s eta 0:00:00
Installing collected packages: evaluate
Successfully installed evaluate-0.4.3
```

[ ]:
```
# Define evaluation metric
import evaluate
accuracy = evaluate.load("accuracy")
```

```
Downloading builder script:   0%|          | 0.00/4.20k [00:00<?, ?B/s]
```

[ ]:
```
# Custom function to compute metrics during evaluation
def compute_metrics(eval_pred):
    predictions, labels = eval_pred # Unpack predictions and true labels
    preds = np.argmax(predictions, axis=1) # Take the class with the highest␣
 ↪probability
    return accuracy.compute(predictions=preds, references=labels) # Return␣
 ↪computed accuracy
```

[ ]:
```
# Use Trainer to make final predictions on the test dataset
predictions = trainer.predict(test_dataset)
```

```
<IPython.core.display.HTML object>
```

[ ]:
```
print(predictions.predictions)  # Model's predicted logits (2D array: samples x␣
 ↪2 classes)
print(predictions.label_ids)    # True labels (1D array)
print(predictions.metrics)      # Evaluation metrics
```

```
[[-3.9160156  3.7617188]
 [ 2.8984375 -3.5273438]
 [ 3.0234375 -3.6972656]
 …
 [ 2.8496094 -3.4941406]
 [ 3.0117188 -3.6972656]
 [ 2.9199219 -3.5527344]]
[1 0 0 … 0 0 0]
{'test_loss': 0.06558195501565933, 'test_accuracy': 0.9882108183079057,
'test_precision': 0.9901315789473685, 'test_recall': 0.9555555555555556,
'test_f1': 0.9725363489499192, 'test_roc_auc': 0.9877776369346912,
'test_runtime': 7.4158, 'test_samples_per_second': 194.449,
'test_steps_per_second': 12.271}
```

[ ]:
```
# Convert logits to final class predictions
y_pred_accuracy = np.argmax(predictions.predictions, axis=1) # Take the index␣
 ↪of the highest logit (class 0 or 1)
y_true_accuracy = predictions.label_ids # True labels extracted from predictions
```

[ ]:
```
# Print a detailed classification report
print("\nClassification Report:")
report = classification_report(
    y_true_accuracy, y_pred_accuracy,
    labels=[0, 1], # Define the label order
```

```
    target_names=["No Symptom", "Mentions Symptom"], # Label names for␣
 ↪readability
    digits=4 # Display results with 4 decimal places
)
print(report)

from sklearn.metrics import confusion_matrix, accuracy_score
# Print confusion matrix and overall accuracy
print("\nConfusion Matrix:")
print(confusion_matrix(y_true_accuracy, y_pred_accuracy, labels=[0, 1]))

print("\nOverall Accuracy:", accuracy_score(y_true_accuracy, y_pred_accuracy))
```

```
Classification Report:
                   precision    recall  f1-score   support

      No Symptom     0.9877    0.9973    0.9925      1127
Mentions Symptom     0.9901    0.9556    0.9725       315

        accuracy                         0.9882      1442
       macro avg     0.9889    0.9764    0.9825      1442
    weighted avg     0.9882    0.9882    0.9881      1442


Confusion Matrix:
[[1124    3]
 [  14  301]]

Overall Accuracy: 0.9882108183079057
```