HAP823 Spring 2024 Final Project Source Code Part 2 - Amitriptyline

April 21, 2024

1 Data Checkpoint

1.1 Restore needed files from the data checkpoint

[1]:
```
# Import libraries with short names
import pandas as pd
import numpy as np
import os
```

[2]:
```
# Open each file and store in a dictionary of dataframes
ad_list = ['Amitriptyline',
           'Bupropion',
           'Citalopram',
           'Desvenlafaxine',
           'Doxepin',
           'Duloxetine',
           'Escitalopram',
           'Fluoxetine',
           'Mirtazapine',
           'Nortriptyline',
           'Paroxetine',
           'Sertraline',
           'Trazodone',
           'Venlafaxine',
           'Other']

analysis_df_dict = dict()
for ad in ad_list:
    filename = f'./checkpoint/hap823_analysis_subset_{ad}.csv'
    analysis_df_dict[ad] = pd.read_csv(filename, low_memory = False)
```

[3]:
```
# Check the info and head of each file
# for ad in ad_list:
#    print(f"\n\nAntidepressant: {ad}")
#    print(analysis_df_dict[ad].info())
#    print(analysis_df_dict[ad].head())
```

2 Include Data Required by AI Model

2.1 All of Us Data Import

[4]:
```
import pandas
import os

# This query represents dataset "Amitriptyline_dxi" for domain "condition" and␣
 ↪was generated for All of Us Registered Tier Dataset v7
dataset_98674129_condition_sql = """
    SELECT
        c_occurrence.person_id,
        c_occurrence.condition_start_datetime,
        visit.concept_name as visit_occurrence_concept_name,
        c_source_concept.concept_code as source_concept_code,
        c_source_concept.vocabulary_id as source_vocabulary
    FROM
        ( SELECT
            *
        FROM
            `""" + os.environ["WORKSPACE_CDR"] + """.condition_occurrence`␣
 ↪c_occurrence
        WHERE
            (
                condition_source_concept_id IN (
                    SELECT
                        DISTINCT c.concept_id
                    FROM
                        `""" + os.environ["WORKSPACE_CDR"] + """.cb_criteria` c
                    JOIN
                        (
                            SELECT
                                CAST(cr.id as string) AS id
                            FROM
                                `""" + os.environ["WORKSPACE_CDR"] + """.
 ↪cb_criteria` cr
                            WHERE
                                concept_id IN (
                                    44826642
                                )
                                AND full_text LIKE '%_rank1]%'
                        ) a
                            ON (
                                c.path LIKE CONCAT('%.',
                            a.id,
                            '.%')
                            OR c.path LIKE CONCAT('%.',
```

```
                            a.id)
                            OR c.path LIKE CONCAT(a.id,
                            '.%')
                            OR c.path = a.id)
                        WHERE
                            is_standard = 0
                            AND is_selectable = 1
                        )
                )
                AND (
                    c_occurrence.PERSON_ID IN (
                        SELECT
                            distinct person_id
                        FROM
                            `""" + os.environ["WORKSPACE_CDR"] + """.
↪cb_search_person` cb_search_person
                        WHERE
                            cb_search_person.person_id IN (
                                SELECT
                                    person_id
                                FROM
                                    `""" + os.environ["WORKSPACE_CDR"] + """.
↪person` p
                                WHERE
                                    race_concept_id IN (8516)
                            )
                            AND cb_search_person.person_id IN (
                                SELECT
                                    criteria.person_id
                                FROM
                                    (SELECT
                                        DISTINCT person_id,
                                        entry_date,
                                        concept_id
                                    FROM
                                        `""" + os.environ["WORKSPACE_CDR"] +␣
↪"""".cb_search_all_events`
                                    WHERE
                                        (
                                            concept_id IN(
                                                SELECT
                                                    DISTINCT c.concept_id
                                                FROM
                                                    `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` c
                                                JOIN
                                                    (
```

```
                                                    SELECT
                                                        CAST(cr.id as␣
↪string) AS id
                                                    FROM
                                                        `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` cr
                                                    WHERE
                                                        concept_id IN␣
↪(4152280)
                                                        AND full_text LIKE␣
↪'%_rank1]%'
                                                ) a
                                                    ON (
                                                        c.path LIKE␣
↪CONCAT('%.',
                                                    a.id,
                                                    '.%')
                                                    OR c.path LIKE␣
↪CONCAT('%.',
                                                    a.id)
                                                    OR c.path LIKE CONCAT(a.
↪id,
                                                    '.%')
                                                    OR c.path = a.id)
                                                WHERE
                                                    is_standard = 1
                                                    AND is_selectable = 1
                                                )
                                                AND is_standard = 1
                                            )
                                        ) criteria
                                    )
                                    AND cb_search_person.person_id NOT IN (
                                        SELECT
                                            criteria.person_id
                                        FROM
                                            (SELECT
                                                DISTINCT person_id,
                                                entry_date,
                                                concept_id
                                            FROM
                                                `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_search_all_events`
                                            WHERE
                                                (
                                                    concept_id IN(
```

```
                                                    SELECT
                                                        DISTINCT c.
↪concept_id
                                                    FROM
                                                        `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` c
                                                    JOIN
                                                        (
                                                            SELECT
                                                                CAST(cr.id␣
↪as string) AS id
                                                            FROM
                                                                `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` cr
                                                            WHERE
                                                                concept_id␣
↪IN (436665)
                                                                AND␣
↪full_text LIKE '%_rank1]%'
                                                        ) a
                                                            ON (
                                                                c.path LIKE␣
↪CONCAT('%.',
                                                                a.id,
                                                                '.%')
                                                                OR c.path LIKE␣
↪CONCAT('%.',
                                                                a.id)
                                                                OR c.path LIKE␣
↪CONCAT(a.id,
                                                                '.%')
                                                                OR c.path = a.
↪id)
                                                    WHERE
                                                            is_standard = 1
                                                            AND␣
↪is_selectable = 1
                                                    )
                                                    AND is_standard = 1
                                                )
                                            ) criteria
                                        ) ))
                                ) c_occurrence
                            LEFT JOIN
                                `""" + os.environ["WORKSPACE_CDR"] + """.
↪visit_occurrence` v
```

```
                                    ON c_occurrence.visit_occurrence_id = v.
 ↪visit_occurrence_id
                                LEFT JOIN
                                    `""" + os.environ["WORKSPACE_CDR"] + """.
 ↪concept` visit
                                    ON v.visit_concept_id = visit.concept_id
                                LEFT JOIN
                                    `""" + os.environ["WORKSPACE_CDR"] + """.
 ↪concept` c_source_concept
                                    ON c_occurrence.condition_source_concept_id␣
 ↪= c_source_concept.concept_id"""

dataset_98674129_condition_df = pandas.read_gbq(
    dataset_98674129_condition_sql,
    dialect="standard",
    use_bqstorage_api=("BIGQUERY_STORAGE_API_ENABLED" in os.environ),
    progress_bar_type="tqdm_notebook")

dataset_98674129_condition_df.head(5)
```

```
Downloading:   0%|          | 0/9099 [00:00<?, ?rows/s]
```

[4]:

	person_id	condition_start_datetime	visit_occurrence_concept_name	source_concept_code	source_vocabulary
0	2487423	2013-10-08 00:00:00+00:00	None	428.0	ICD9CM
1	1122724	2009-12-17 04:00:00+00:00	None	428.0	ICD9CM
2	2036147	2012-08-15 15:31:00+00:00	None	428.0	ICD9CM
3	6971152	2008-10-22 21:58:00+00:00	None	428.0	ICD9CM
4	1936151	2015-08-02 03:03:52+00:00	None	428.0	ICD9CM

[5]:
```
import pandas
import os

# This query represents dataset "Amitriptyline_dxo" for domain "condition" and␣
 ↪was generated for All of Us Registered Tier Dataset v7
dataset_04948107_condition_sql = """
    SELECT
        c_occurrence.person_id,
        c_occurrence.condition_start_datetime,
        visit.concept_name as visit_occurrence_concept_name,
        c_source_concept.concept_code as source_concept_code,
```

```
        c_source_concept.vocabulary_id as source_vocabulary
    FROM
        ( SELECT
            *
        FROM
            `""" + os.environ["WORKSPACE_CDR"] + """.condition_occurrence`␣
↪c_occurrence
        WHERE
            (
                condition_source_concept_id IN (
                    SELECT
                        DISTINCT c.concept_id
                    FROM
                        `""" + os.environ["WORKSPACE_CDR"] + """.cb_criteria` c
                    JOIN
                        (
                            SELECT
                                CAST(cr.id as string) AS id
                            FROM
                                `""" + os.environ["WORKSPACE_CDR"] + """.
↪cb_criteria` cr
                            WHERE
                                concept_id IN (
                                    44820052, 44820682, 44822975, 44825349,␣
↪44825716, 44829926, 44831089, 44833421, 44835785, 44836929
                                )
                                AND full_text LIKE '%_rank1]%'
                        ) a
                            ON (
                                c.path LIKE CONCAT('%.',
                            a.id,
                            '.%')
                            OR c.path LIKE CONCAT('%.',
                            a.id)
                            OR c.path LIKE CONCAT(a.id,
                            '.%')
                            OR c.path = a.id)
                        WHERE
                            is_standard = 0
                            AND is_selectable = 1
                        )
                )
                AND (
                    c_occurrence.PERSON_ID IN (
                        SELECT
                            distinct person_id
                        FROM
```

```
                            `""" + os.environ["WORKSPACE_CDR"] + """.
↪cb_search_person` cb_search_person
                        WHERE
                            cb_search_person.person_id IN (
                                SELECT
                                    person_id
                                FROM
                                    `""" + os.environ["WORKSPACE_CDR"] + """.
↪person` p
                                WHERE
                                    race_concept_id IN (8516)
                            )
                            AND cb_search_person.person_id IN (
                                SELECT
                                    criteria.person_id
                                FROM
                                    (SELECT
                                        DISTINCT person_id,
                                        entry_date,
                                        concept_id
                                    FROM
                                        `""" + os.environ["WORKSPACE_CDR"] +␣
↪""".cb_search_all_events`
                                    WHERE
                                        (
                                            concept_id IN(
                                                SELECT
                                                    DISTINCT c.concept_id
                                                FROM
                                                    `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` c
                                                JOIN
                                                    (
                                                        SELECT
                                                            CAST(cr.id as␣
↪string) AS id
                                                        FROM
                                                            `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` cr
                                                        WHERE
                                                            concept_id IN␣
↪(4152280)
                                                            AND full_text LIKE␣
↪'%_rank1]%'
                                                    ) a
                                                        ON (
```

```
                                                    c.path LIKE␣
↪CONCAT('%.',
                                                a.id,
                                                '.%')
                                                OR c.path LIKE␣
↪CONCAT('%.',
                                                a.id)
                                                OR c.path LIKE CONCAT(a.
↪id,
                                                '.%')
                                                OR c.path = a.id)
                                            WHERE
                                                is_standard = 1
                                                AND is_selectable = 1
                                            )
                                            AND is_standard = 1
                                    )
                                ) criteria
                            )
                            AND cb_search_person.person_id NOT IN (
                                SELECT
                                    criteria.person_id
                                FROM
                                    (SELECT
                                        DISTINCT person_id,
                                        entry_date,
                                        concept_id
                                    FROM
                                        `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_search_all_events`
                                    WHERE
                                        (
                                            concept_id IN(
                                                SELECT
                                                    DISTINCT c.
↪concept_id
                                                FROM
                                                    `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` c
                                                JOIN
                                                    (
                                                        SELECT
                                                            CAST(cr.id␣
↪as string) AS id
                                                        FROM
```

```
                                                    `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` cr
                                                WHERE
                                                    concept_id␣
↪IN (436665)
                                                    AND␣
↪full_text LIKE '%_rank1]%'
                                            ) a
                                                ON (
                                                    c.path LIKE␣
↪CONCAT('%.',
                                                    a.id,
                                                    '.%')
                                                    OR c.path LIKE␣
↪CONCAT('%.',
                                                    a.id)
                                                    OR c.path LIKE␣
↪CONCAT(a.id,
                                                    '.%')
                                                    OR c.path = a.
↪id)
                                            WHERE
                                                    is_standard = 1
                                                    AND␣
↪is_selectable = 1
                                            )
                                            AND is_standard = 1
                                        )
                                    ) criteria
                                ) ))
                        ) c_occurrence
                    LEFT JOIN
                        `""" + os.environ["WORKSPACE_CDR"] + """.
↪visit_occurrence` v
                            ON c_occurrence.visit_occurrence_id = v.
↪visit_occurrence_id
                    LEFT JOIN
                        `""" + os.environ["WORKSPACE_CDR"] + """.
↪concept` visit
                            ON v.visit_concept_id = visit.concept_id
                    LEFT JOIN
                        `""" + os.environ["WORKSPACE_CDR"] + """.
↪concept` c_source_concept
                            ON c_occurrence.condition_source_concept_id␣
↪= c_source_concept.concept_id"""
```

```
dataset_04948107_condition_df = pandas.read_gbq(
    dataset_04948107_condition_sql,
    dialect="standard",
    use_bqstorage_api=("BIGQUERY_STORAGE_API_ENABLED" in os.environ),
    progress_bar_type="tqdm_notebook")

dataset_04948107_condition_df.head(5)
```

```
Downloading:   0%|          | 0/63907 [00:00<?, ?rows/s]
```

[5]:

	person_id	condition_start_datetime	visit_occurrence_concept_name	source_concept_code	source_vocabulary
0	1459235	2019-12-12 00:00:00+00:00	None	300.02	ICD9CM
1	5306013	2013-05-26 08:25:08+00:00	None	300.02	ICD9CM
2	3789920	2021-01-09 10:00:00+00:00	None	300.02	ICD9CM
3	2857783	2022-03-24 00:00:00+00:00	None	300.02	ICD9CM
4	5306013	2013-09-08 08:16:00+00:00	None	300.02	ICD9CM

[6]:
```
import pandas
import os

# This query represents dataset "Amitriptyline_dxo" for domain "measurement"␣
 ↪and was generated for All of Us Registered Tier Dataset v7
dataset_04948107_measurement_sql = """
    SELECT
        measurement.person_id,
        measurement.measurement_datetime,
        m_visit.concept_name as visit_occurrence_concept_name,
        m_source_concept.concept_code as source_concept_code,
        m_source_concept.vocabulary_id as source_vocabulary
    FROM
        ( SELECT
            *
        FROM
            `""" + os.environ["WORKSPACE_CDR"] + """.measurement` measurement
        WHERE
            (
                measurement_source_concept_id IN (
                    SELECT
                        DISTINCT c.concept_id
                    FROM
                        `""" + os.environ["WORKSPACE_CDR"] + """.cb_criteria` c
```

```
                    JOIN
                        (
                            SELECT
                                CAST(cr.id as string) AS id
                            FROM
                                `""" + os.environ["WORKSPACE_CDR"] + """.
↪cb_criteria` cr
                            WHERE
                                concept_id IN (
                                    44836711
                                )
                                AND full_text LIKE '%_rank1]%'
                        ) a
                            ON (
                                c.path LIKE CONCAT('%.',
                            a.id,
                            '.%')
                            OR c.path LIKE CONCAT('%.',
                            a.id)
                            OR c.path LIKE CONCAT(a.id,
                            '.%')
                            OR c.path = a.id)
                        WHERE
                            is_standard = 0
                            AND is_selectable = 1
                        )
                )
                AND (
                    measurement.PERSON_ID IN (
                        SELECT
                            distinct person_id
                        FROM
                            `""" + os.environ["WORKSPACE_CDR"] + """.
↪cb_search_person` cb_search_person
                        WHERE
                            cb_search_person.person_id IN (
                                SELECT
                                    person_id
                                FROM
                                    `""" + os.environ["WORKSPACE_CDR"] + """.
↪person` p
                                WHERE
                                    race_concept_id IN (8516)
                            )
                            AND cb_search_person.person_id IN (
                                SELECT
                                    criteria.person_id
```

```
                                FROM
                                    (SELECT
                                        DISTINCT person_id,
                                        entry_date,
                                        concept_id
                                    FROM
                                        `""" + os.environ["WORKSPACE_CDR"] +␣
↪""".cb_search_all_events`
                                    WHERE
                                        (
                                            concept_id IN(
                                                SELECT
                                                    DISTINCT c.concept_id
                                                FROM
                                                    `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` c
                                                JOIN
                                                    (
                                                        SELECT
                                                            CAST(cr.id as␣
↪string) AS id
                                                        FROM
                                                            `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` cr
                                                        WHERE
                                                            concept_id IN␣
↪(4152280)
                                                            AND full_text LIKE␣
↪'%_rank1]%'
                                                    ) a
                                                        ON (
                                                            c.path LIKE␣
↪CONCAT('%.',
                                                        a.id,
                                                        '.%')
                                                        OR c.path LIKE␣
↪CONCAT('%.',
                                                        a.id)
                                                        OR c.path LIKE CONCAT(a.
↪id,
                                                        '.%')
                                                        OR c.path = a.id)
                                                    WHERE
                                                        is_standard = 1
                                                        AND is_selectable = 1
                                                    )
```

```
                                                AND is_standard = 1
                                        )
                                    ) criteria
                                )
                                AND cb_search_person.person_id NOT IN (
                                    SELECT
                                        criteria.person_id
                                    FROM
                                        (SELECT
                                            DISTINCT person_id,
                                            entry_date,
                                            concept_id
                                        FROM
                                            `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_search_all_events`
                                        WHERE
                                            (
                                                concept_id IN(
                                                    SELECT
                                                        DISTINCT c.
↪concept_id
                                                    FROM
                                                        `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` c
                                                    JOIN
                                                        (
                                                            SELECT
                                                                CAST(cr.id␣
↪as string) AS id
                                                            FROM
                                                                `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` cr
                                                            WHERE
                                                                concept_id␣
↪IN (436665)
                                                                AND␣
↪full_text LIKE '%_rank1]%'
                                                        ) a
                                                            ON (
                                                                c.path LIKE␣
↪CONCAT('%.',
                                                                a.id,
                                                                '.%')
                                                                OR c.path LIKE␣
↪CONCAT('%.',
                                                                a.id)
```

```
                                                                    OR c.path LIKE␣
↪CONCAT(a.id,
                                                                    '.%')
                                                                    OR c.path = a.
↪id)
                                                                WHERE
                                                                    is_standard = 1
                                                                    AND␣
↪is_selectable = 1
                                                                )
                                                                AND is_standard = 1
                                                            )
                                                        ) criteria
                                                    ) ))
                                    ) measurement
                                LEFT JOIn
                                    `""" + os.environ["WORKSPACE_CDR"] + """.
↪visit_occurrence` v
                                        ON measurement.visit_occurrence_id = v.
↪visit_occurrence_id
                                LEFT JOIN
                                    `""" + os.environ["WORKSPACE_CDR"] + """.
↪concept` m_visit
                                        ON v.visit_concept_id = m_visit.concept_id
                                LEFT JOIN
                                    `""" + os.environ["WORKSPACE_CDR"] + """.
↪concept` m_source_concept
                                        ON measurement.
↪measurement_source_concept_id = m_source_concept.concept_id"""

dataset_04948107_measurement_df = pandas.read_gbq(
    dataset_04948107_measurement_sql,
    dialect="standard",
    use_bqstorage_api=("BIGQUERY_STORAGE_API_ENABLED" in os.environ),
    progress_bar_type="tqdm_notebook")

dataset_04948107_measurement_df.head(5)
```

```
Downloading:   0%|          | 0/94 [00:00<?, ?rows/s]
```

[6]:

	person_id	measurement_datetime	visit_occurrence_concept_name	\
0	1676132	2012-03-28 00:00:00+00:00		None
1	5870979	2011-10-18 00:00:00+00:00		None
2	1676132	2007-11-11 00:00:00+00:00		None
3	5049841	2009-04-12 00:00:00+00:00		None
4	5049841	2011-09-10 00:00:00+00:00		None

```
  source_concept_code source_vocabulary
0               V76.12            ICD9CM
1               V76.12            ICD9CM
2               V76.12            ICD9CM
3               V76.12            ICD9CM
4               V76.12            ICD9CM
```

[7]:
```
import pandas
import os

# This query represents dataset "Amitriptyline_dxo" for domain "procedure" and␣
 ↪was generated for All of Us Registered Tier Dataset v7
dataset_04948107_procedure_sql = """
    SELECT
        procedure.person_id,
        procedure.procedure_datetime,
        p_visit.concept_name as visit_occurrence_concept_name,
        p_source_concept.concept_code as source_concept_code,
        p_source_concept.vocabulary_id as source_vocabulary
    FROM
        ( SELECT
            *
        FROM
            `""" + os.environ["WORKSPACE_CDR"] + """.procedure_occurrence`␣
 ↪procedure
        WHERE
            (
                procedure_source_concept_id IN (
                    SELECT
                        DISTINCT c.concept_id
                    FROM
                        `""" + os.environ["WORKSPACE_CDR"] + """.cb_criteria` c
                    JOIN
                        (
                            SELECT
                                CAST(cr.id as string) AS id
                            FROM
                                `""" + os.environ["WORKSPACE_CDR"] + """.
 ↪cb_criteria` cr
                            WHERE
                                concept_id IN (
                                    44828600
                                )
                                AND full_text LIKE '%_rank1]%'
                        ) a
                            ON (
                                c.path LIKE CONCAT('%.',
```

```
                            a.id,
                            '.%')
                            OR c.path LIKE CONCAT('%.',
                            a.id)
                            OR c.path LIKE CONCAT(a.id,
                            '.%')
                            OR c.path = a.id)
                        WHERE
                            is_standard = 0
                            AND is_selectable = 1
                        )
                )
                AND (
                    procedure.PERSON_ID IN (
                        SELECT
                            distinct person_id
                        FROM
                            `""" + os.environ["WORKSPACE_CDR"] + """.
↪cb_search_person` cb_search_person
                        WHERE
                            cb_search_person.person_id IN (
                                SELECT
                                    person_id
                                FROM
                                    `""" + os.environ["WORKSPACE_CDR"] + """.
↪person` p
                                WHERE
                                    race_concept_id IN (8516)
                            )
                            AND cb_search_person.person_id IN (
                                SELECT
                                    criteria.person_id
                                FROM
                                    (SELECT
                                        DISTINCT person_id,
                                        entry_date,
                                        concept_id
                                    FROM
                                        `""" + os.environ["WORKSPACE_CDR"] +␣
↪"""".cb_search_all_events`
                                    WHERE
                                        (
                                            concept_id IN(
                                                SELECT
                                                    DISTINCT c.concept_id
                                                FROM
```

```
                                                `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` c
                                            JOIN
                                                (
                                                    SELECT
                                                        CAST(cr.id as␣
↪string) AS id
                                                    FROM
                                                        `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` cr
                                                    WHERE
                                                        concept_id IN␣
↪(4152280)
                                                        AND full_text LIKE␣
↪'%_rank1]%'
                                                ) a
                                                    ON (
                                                        c.path LIKE␣
↪CONCAT('%.',
                                                    a.id,
                                                    '.%')
                                                    OR c.path LIKE␣
↪CONCAT('%.',
                                                    a.id)
                                                    OR c.path LIKE CONCAT(a.
↪id,
                                                    '.%')
                                                    OR c.path = a.id)
                                                WHERE
                                                    is_standard = 1
                                                    AND is_selectable = 1
                                                )
                                                AND is_standard = 1
                                        )
                                    ) criteria
                                )
                                AND cb_search_person.person_id NOT IN (
                                    SELECT
                                        criteria.person_id
                                    FROM
                                        (SELECT
                                            DISTINCT person_id,
                                            entry_date,
                                            concept_id
                                        FROM
```

```
                                                `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_search_all_events`
                                            WHERE
                                                (
                                                    concept_id IN(
                                                        SELECT
                                                            DISTINCT c.
↪concept_id
                                                        FROM
                                                            `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` c
                                                        JOIN
                                                            (
                                                                SELECT
                                                                    CAST(cr.id␣
↪as string) AS id
                                                                FROM
                                                                    `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` cr
                                                                WHERE
                                                                    concept_id␣
↪IN (436665)
                                                                    AND␣
↪full_text LIKE '%_rank1]%'
                                                            ) a
                                                                ON (
                                                                    c.path LIKE␣
↪CONCAT('%.',
                                                                    a.id,
                                                                    '.%')
                                                                    OR c.path LIKE␣
↪CONCAT('%.',
                                                                    a.id)
                                                                    OR c.path LIKE␣
↪CONCAT(a.id,
                                                                    '.%')
                                                                    OR c.path = a.
↪id)
                                                            WHERE
                                                                is_standard = 1
                                                                AND␣
↪is_selectable = 1
                                                            )
                                                            AND is_standard = 1
                                                    )
                                                ) criteria
```

```
                                            ) ))
                                ) procedure
                            LEFT JOIN
                                `""" + os.environ["WORKSPACE_CDR"] + """.
 ↪visit_occurrence` v
                                    ON procedure.visit_occurrence_id = v.
 ↪visit_occurrence_id
                            LEFT JOIN
                                `""" + os.environ["WORKSPACE_CDR"] + """.
 ↪concept` p_visit
                                    ON v.visit_concept_id = p_visit.concept_id
                            LEFT JOIN
                                `""" + os.environ["WORKSPACE_CDR"] + """.
 ↪concept` p_source_concept
                                    ON procedure.procedure_source_concept_id =␣
 ↪p_source_concept.concept_id"""

dataset_04948107_procedure_df = pandas.read_gbq(
    dataset_04948107_procedure_sql,
    dialect="standard",
    use_bqstorage_api=("BIGQUERY_STORAGE_API_ENABLED" in os.environ),
    progress_bar_type="tqdm_notebook")

dataset_04948107_procedure_df.head(5)
```

```
Downloading:   0%|          | 0/3407 [00:00<?, ?rows/s]
```

[7]:

	person_id	procedure_datetime	visit_occurrence_concept_name	source_concept_code	source_vocabulary
0	1466135	2013-02-06 04:00:00+00:00	None	V72.31	ICD9CM
1	1138906	2013-12-12 00:00:00+00:00	None	V72.31	ICD9CM
2	3162368	2005-01-26 17:15:00+00:00	None	V72.31	ICD9CM
3	1919301	2014-04-29 00:00:00+00:00	None	V72.31	ICD9CM
4	3415504	2006-01-18 15:16:00+00:00	None	V72.31	ICD9CM

[8]:
```
import pandas
import os

# This query represents dataset "Amitriptyline_px" for domain "procedure" and␣
 ↪was generated for All of Us Registered Tier Dataset v7
dataset_57232984_procedure_sql = """
```

```
   SELECT
       procedure.person_id,
       procedure.procedure_datetime,
       p_visit.concept_name as visit_occurrence_concept_name,
       p_source_concept.concept_code as source_concept_code,
       p_source_concept.vocabulary_id as source_vocabulary
   FROM
       ( SELECT
           *
       FROM
           `""" + os.environ["WORKSPACE_CDR"] + """.procedure_occurrence`␣
↪procedure
       WHERE
           (
               procedure_source_concept_id IN (
                   SELECT
                       DISTINCT c.concept_id
                   FROM
                       `""" + os.environ["WORKSPACE_CDR"] + """.cb_criteria` c
                   JOIN
                       (
                           SELECT
                               CAST(cr.id as string) AS id
                           FROM
                               `""" + os.environ["WORKSPACE_CDR"] + """.
↪cb_criteria` cr
                           WHERE
                               concept_id IN (
                                   2213521, 2213523, 2213550
                               )
                               AND full_text LIKE '%_rank1]%'
                       ) a
                           ON (
                               c.path LIKE CONCAT('%.',
                           a.id,
                           '.%')
                           OR c.path LIKE CONCAT('%.',
                           a.id)
                           OR c.path LIKE CONCAT(a.id,
                           '.%')
                           OR c.path = a.id)
                       WHERE
                           is_standard = 0
                           AND is_selectable = 1
                       )
               )
               AND (
```

```
                    procedure.PERSON_ID IN (
                        SELECT
                            distinct person_id
                        FROM
                            `""" + os.environ["WORKSPACE_CDR"] + """.
↪cb_search_person` cb_search_person
                        WHERE
                            cb_search_person.person_id IN (
                                SELECT
                                    person_id
                                FROM
                                    `""" + os.environ["WORKSPACE_CDR"] + """.
↪person` p
                                WHERE
                                    race_concept_id IN (8516)
                            )
                            AND cb_search_person.person_id IN (
                                SELECT
                                    criteria.person_id
                                FROM
                                    (SELECT
                                        DISTINCT person_id,
                                        entry_date,
                                        concept_id
                                    FROM
                                        `""" + os.environ["WORKSPACE_CDR"] +␣
↪""".cb_search_all_events`
                                    WHERE
                                        (
                                            concept_id IN(
                                                SELECT
                                                    DISTINCT c.concept_id
                                                FROM
                                                    `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` c
                                                JOIN
                                                    (
                                                        SELECT
                                                            CAST(cr.id as␣
↪string) AS id
                                                        FROM
                                                            `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` cr
                                                        WHERE
                                                            concept_id IN␣
↪(4152280)
```

```
                                                        AND full_text LIKE␣
↪'%_rank1]%'
                                                ) a
                                                    ON (
                                                        c.path LIKE␣
↪CONCAT('%.',
                                                    a.id,
                                                    '.%')
                                                    OR c.path LIKE␣
↪CONCAT('%.',
                                                    a.id)
                                                    OR c.path LIKE CONCAT(a.
↪id,
                                                    '.%')
                                                    OR c.path = a.id)
                                                WHERE
                                                    is_standard = 1
                                                    AND is_selectable = 1
                                                )
                                                AND is_standard = 1
                                        )
                                    ) criteria
                                )
                                AND cb_search_person.person_id NOT IN (
                                    SELECT
                                        criteria.person_id
                                    FROM
                                        (SELECT
                                            DISTINCT person_id,
                                            entry_date,
                                            concept_id
                                        FROM
                                            `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_search_all_events`
                                        WHERE
                                            (
                                                concept_id IN(
                                                SELECT
                                                    DISTINCT c.
↪concept_id
                                                FROM
                                                    `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` c
                                                JOIN
                                                    (
                                                        SELECT
```

```
                                                                    CAST(cr.id␣
↪as string) AS id
                                                                FROM
                                                                    `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` cr
                                                                WHERE
                                                                    concept_id␣
↪IN (436665)
                                                                    AND␣
↪full_text LIKE '%_rank1]%'
                                                            ) a
                                                                ON (
                                                                    c.path LIKE␣
↪CONCAT('%.',
                                                                    a.id,
                                                                    '.%')
                                                                    OR c.path LIKE␣
↪CONCAT('%.',
                                                                    a.id)
                                                                    OR c.path LIKE␣
↪CONCAT(a.id,
                                                                    '.%')
                                                                    OR c.path = a.
↪id)
                                                                WHERE
                                                                    is_standard = 1
                                                                    AND␣
↪is_selectable = 1
                                                                )
                                                                AND is_standard = 1
                                                        )
                                                    ) criteria
                                                ) ))
                                    ) procedure
                                LEFT JOIN
                                    `""" + os.environ["WORKSPACE_CDR"] + """.
↪visit_occurrence` v
                                        ON procedure.visit_occurrence_id = v.
↪visit_occurrence_id
                                LEFT JOIN
                                    `""" + os.environ["WORKSPACE_CDR"] + """.
↪concept` p_visit
                                        ON v.visit_concept_id = p_visit.concept_id
                                LEFT JOIN
                                    `""" + os.environ["WORKSPACE_CDR"] + """.
↪concept` p_source_concept
```

```
                                    ON procedure.procedure_source_concept_id =␣
 ↪p_source_concept.concept_id"""

dataset_57232984_procedure_df = pandas.read_gbq(
    dataset_57232984_procedure_sql,
    dialect="standard",
    use_bqstorage_api=("BIGQUERY_STORAGE_API_ENABLED" in os.environ),
    progress_bar_type="tqdm_notebook")

dataset_57232984_procedure_df.head(5)
```

```
Downloading:   0%|          | 0/6003 [00:00<?, ?rows/s]
```

[8]:

	person_id	procedure_datetime	visit_occurrence_concept_name	source_concept_code	source_vocabulary
0	1860641	2010-11-05 00:00:00+00:00	None	90807	CPT4
1	3169446	2009-06-14 00:00:00+00:00	None	90807	CPT4
2	1860641	2011-09-18 00:00:00+00:00	None	90807	CPT4
3	1860641	2011-10-02 00:00:00+00:00	None	90807	CPT4
4	1986747	2012-06-17 00:00:00+00:00	None	90807	CPT4

[9]:
```
import pandas
import os

# This query represents dataset "Amitriptyline_rx" for domain "drug" and was␣
 ↪generated for All of Us Registered Tier Dataset v7
dataset_70579010_drug_sql = """
    SELECT
        d_exposure.person_id,
        d_standard_concept.concept_name as standard_concept_name,
        d_standard_concept.concept_code as standard_concept_code,
        d_standard_concept.vocabulary_id as standard_vocabulary,
        d_exposure.drug_exposure_start_datetime
    FROM
        ( SELECT
            *
        FROM
            `""" + os.environ["WORKSPACE_CDR"] + """.drug_exposure` d_exposure
        WHERE
            (
                drug_concept_id IN (
```

```
                    SELECT
                        DISTINCT ca.descendant_id
                    FROM
                        `""" + os.environ["WORKSPACE_CDR"] + """.
↪cb_criteria_ancestor` ca
                    JOIN
                        (
                            SELECT
                                DISTINCT c.concept_id
                            FROM
                                `""" + os.environ["WORKSPACE_CDR"] + """.
↪cb_criteria` c
                            JOIN
                                (
                                    SELECT
                                        CAST(cr.id as string) AS id
                                    FROM
                                        `""" + os.environ["WORKSPACE_CDR"] +␣
↪""".cb_criteria` cr
                                    WHERE
                                        concept_id IN (
                                            1110410, 1124957, 1125315, 1153013,␣
↪1174888, 1332418, 1592085, 703547, 710062, 715233, 738156, 749910, 750982,␣
↪766814, 791967, 798874
                                        )
                                        AND full_text LIKE '%_rank1]%'
                                ) a
                                    ON (
                                        c.path LIKE CONCAT('%.',
                                    a.id,
                                    '.%')
                                    OR c.path LIKE CONCAT('%.',
                                    a.id)
                                    OR c.path LIKE CONCAT(a.id,
                                    '.%')
                                    OR c.path = a.id)
                                WHERE
                                    is_standard = 1
                                    AND is_selectable = 1
                                ) b
                                    ON (
                                        ca.ancestor_id = b.concept_id
                                    )
                            )
                        )
                        AND (
                            d_exposure.PERSON_ID IN (
```

```
                                SELECT
                                    distinct person_id
                                FROM
                                    `""" + os.environ["WORKSPACE_CDR"] + """.
↪cb_search_person` cb_search_person
                                WHERE
                                    cb_search_person.person_id IN (
                                        SELECT
                                            person_id
                                        FROM
                                            `""" + os.environ["WORKSPACE_CDR"] +␣
↪"""".person` p
                                        WHERE
                                            race_concept_id IN (8516)
                                    )
                                    AND cb_search_person.person_id IN (
                                        SELECT
                                            criteria.person_id
                                        FROM
                                            (SELECT
                                                DISTINCT person_id,
                                                entry_date,
                                                concept_id
                                            FROM
                                                `""" + os.environ["WORKSPACE_CDR"]␣
↪+ """.cb_search_all_events`
                                            WHERE
                                                (
                                                    concept_id IN(
                                                        SELECT
                                                            DISTINCT c.concept_id
                                                        FROM
                                                            `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` c
                                                        JOIN
                                                            (
                                                                SELECT
                                                                    CAST(cr.id as␣
↪string) AS id
                                                                FROM
                                                                    `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` cr
                                                                WHERE
                                                                    concept_id IN␣
↪(4152280)
```

```
                                                                    AND full_text␣
↪LIKE '%_rank1]%'
                                                            ) a
                                                                ON (
                                                                    c.path LIKE␣
↪CONCAT('%.',
                                                                    a.id,
                                                                    '.%')
                                                                    OR c.path LIKE␣
↪CONCAT('%.',
                                                                    a.id)
                                                                    OR c.path LIKE␣
↪CONCAT(a.id,
                                                                    '.%')
                                                                    OR c.path = a.id)
                                                            WHERE
                                                                is_standard = 1
                                                                AND is_selectable =␣
↪1
                                                            )
                                                            AND is_standard = 1
                                                    )
                                                ) criteria
                                            )
                                            AND cb_search_person.person_id NOT IN (
                                                SELECT
                                                    criteria.person_id
                                                FROM
                                                    (SELECT
                                                        DISTINCT person_id,
                                                        entry_date,
                                                        concept_id
                                                    FROM
                                                        `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_search_all_events`
                                                    WHERE
                                                        (
                                                            concept_id IN(
                                                                SELECT
                                                                    DISTINCT c.
↪concept_id
                                                                FROM
                                                                    `""" + os.
↪environ["WORKSPACE_CDR"] + """.cb_criteria` c
                                                                JOIN
                                                                    (
```

```
                                                                    SELECT
                                                                        CAST(cr.
 ↪id as string) AS id
                                                                    FROM
                                                                        `""" +␣
 ↪os.environ["WORKSPACE_CDR"] + """.cb_criteria` cr
                                                                    WHERE
                                                                       ␣
 ↪concept_id IN (436665)
                                                                        AND␣
 ↪full_text LIKE '%_rank1]%'
                                                                ) a
                                                                    ON (
                                                                        c.path␣
 ↪LIKE CONCAT('%.',
                                                                    a.id,
                                                                    '.%')
                                                                    OR c.path␣
 ↪LIKE CONCAT('%.',
                                                                    a.id)
                                                                    OR c.path␣
 ↪LIKE CONCAT(a.id,
                                                                    '.%')
                                                                    OR c.path =␣
 ↪a.id)
                                                                WHERE
                                                                    is_standard␣
 ↪= 1
                                                                    AND␣
 ↪is_selectable = 1
                                                                )
                                                                AND is_standard␣
 ↪= 1
                                                        )
                                                    ) criteria
                                                ) ))
                                    ) d_exposure
                                LEFT JOIN
                                    `""" + os.environ["WORKSPACE_CDR"] + """.
 ↪concept` d_standard_concept
                                        ON d_exposure.drug_concept_id =␣
 ↪d_standard_concept.concept_id"""

dataset_70579010_drug_df = pandas.read_gbq(
    dataset_70579010_drug_sql,
    dialect="standard",
```

```
    use_bqstorage_api=("BIGQUERY_STORAGE_API_ENABLED" in os.environ),
    progress_bar_type="tqdm_notebook")

dataset_70579010_drug_df.head(5)
```

```
Downloading:   0%|          | 0/599654 [00:00<?, ?rows/s]
```

[9]:

	person_id	standard_concept_name	standard_concept_code	standard_vocabulary	drug_exposure_start_datetime
0	1425125	30 ML morphine sulfate 1 MG/ML Injection	1728999	RxNorm	2008-09-06 05:00:00+00:00
1	3476915	acetaminophen 500 MG / hydrocodone bitartrate …	857109	RxNorm	2011-02-26 05:00:00+00:00
2	1679094	30 ML morphine sulfate 1 MG/ML Injection	1728999	RxNorm	2014-10-05 16:49:00+00:00
3	1754365	30 ML morphine sulfate 1 MG/ML Injection	1728999	RxNorm	2010-10-10 15:13:46+00:00
4	1808098	30 ML morphine sulfate 1 MG/ML Injection	1728999	RxNorm	2021-08-20 09:07:00+00:00

2.2 Rename imported dataframes

[10]:
```
# Import libraries
import pandas as pd
import numpy as np
```

[11]:
```
# Note that different datasets may exists for different antidepressants
dxi_condition_df = dataset_98674129_condition_df
dxo_condition_df = dataset_04948107_condition_df
dxo_measurement_df = dataset_04948107_measurement_df
dxo_procedure_df = dataset_04948107_procedure_df
px_df = dataset_57232984_procedure_df
rx_df = dataset_70579010_drug_df
```

2.3 Updating the Dataset

2.3.1 Define target antidepressant

[12]:
```
# Change for your own antidepressant
ad = 'Amitriptyline'
```

2.3.2 Antidepressant History

[13]:
```
# Import library
import pandas as pd

# Get reference antidepressant dates data
```

```
ad_dates_df = pd.read_csv('./checkpoint/hap823_ad_dates_df.csv', low_memory =␣
 ↪False)

ad_dates_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28304 entries, 0 to 28303
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   person_id          28304 non-null  int64
 1   ad_grouping        28304 non-null  object
 2   trial_number       28304 non-null  int64
 3   earliest_ad_start  28304 non-null  object
 4   earliest_ad_end    28304 non-null  object
 5   dosage_days        28304 non-null  int64
 6   dose_70p           28304 non-null  int64
 7   other_ad           28304 non-null  int64
 8   Remission          28304 non-null  int64
dtypes: int64(6), object(3)
memory usage: 1.9+ MB
```

[14]:
```
# Change dates data types to make sure they are dates
ad_dates_df['earliest_ad_start'] = pd.
 ↪to_datetime(ad_dates_df['earliest_ad_start'], errors='coerce')
ad_dates_df['earliest_ad_end'] = pd.to_datetime(ad_dates_df['earliest_ad_end'],␣
 ↪errors='coerce')

ad_dates_df.head()
```

[14]:

	person_id	ad_grouping	trial_number	earliest_ad_start	earliest_ad_end	dosage_days	dose_70p	other_ad	Remission
0	3445002	Other	1	2009-12-20 08:48:00+00:00	2009-12-20 08:48:00+00:00	0	0	0	0
1	3445002	Trazodone	1	2011-12-11 22:23:00+00:00	2012-01-17 00:33:00+00:00	6	0	0	0
2	3445002	Trazodone	2	2012-11-16 19:36:00+00:00	2013-01-20 04:18:00+00:00	57	0	0	0
3	3445002	Trazodone	3	2013-03-22 13:37:00+00:00	2013-03-22 17:22:00+00:00	1	0	0	0
4	3445002	Escitalopram	1	2018-01-12 15:00:00+00:00	2018-07-11 14:59:00+00:00	180	1	0	1

We need to find, the following antidepressant histories: - Same antidepressant no remission (SN) - Same antidepressant with remission (SR) - Different antidepressant no remission (DN) - Different antidepressant with remission (DR)

[15]:
```
# Initialize columns
ad_dates_df['SN'] = 0
ad_dates_df['SR'] = 0
ad_dates_df['DN'] = 0
ad_dates_df['DR'] = 0
```

[16]:
```
# Create a function to get all different antidepressant history
## hist_type values are 'SN', 'SR', DN' and 'DR'
def get_ad_hist (row, hist_type):

    # If it's the first trial and SN or SR, return 0 (no history)
    if row['trial_number'] == 1 and (hist_type == 'SN' or hist_type == 'SR'):
        return 0

    # Check if processing for SN or SR is needed
    elif hist_type == 'SN' or hist_type == 'SR':
        # Check previous trials
        previous_trials = ad_dates_df[(ad_dates_df['person_id'] ==␣
 ↪row['person_id']) &
                                      (ad_dates_df['trial_number'] <␣
 ↪row['trial_number']) &
                                      (ad_dates_df['ad_grouping'] ==␣
 ↪row['ad_grouping'])]

        if hist_type == 'SN':
            # Check for any previous trial with no remission
            history = previous_trials[previous_trials['Remission'] == 0]
        else: # 'SR'
            # Check for any previous trial with no remission
            history = previous_trials[previous_trials['Remission'] == 1]

        if history.empty:
            return 0 # no history found
        else:
            return 1 # history found

    # Check if processing for DN or DR is needed
    elif hist_type == 'DN' or hist_type == 'DR':

        # Get the start of the trial
        trial_start = row["earliest_ad_start"]

        # Check other rows for the same person_id for other antidepressants
        other_rows = ad_dates_df[(ad_dates_df['person_id'] == row['person_id'])␣
 ↪&
                                 (ad_dates_df['ad_grouping'] !=␣
 ↪row['ad_grouping'])]
```

```
        # Retain only those with end date before trial start (ended before␣
↪trial began)
        other_rows = other_rows[other_rows['earliest_ad_end'] < trial_start]

        if hist_type == 'DN':
            # Check for any previous other antidepressant with no remission
            history = other_rows[other_rows['Remission'] == 0]
        else: # 'DR'
            # Check for any previous other antidepressant with remission
            history = other_rows[other_rows['Remission'] == 1]

        if history.empty:
            return 0 # no history found
        else:
            return 1 # history found

    return 0 # default return is 0
```

[17]:
```
# Fill values for history of antidepressant use and reaction
for hist_type in ['SN', 'SR', 'DN', 'DR']:
    ad_dates_df[hist_type] = ad_dates_df.apply(get_ad_hist, axis=1,␣
↪args=(hist_type, ))

ad_dates_df.head()
```

[17]:

	person_id	ad_grouping	trial_number	earliest_ad_start	earliest_ad_end	dosage_days	dose_70p	other_ad	Remission	SN	SR	DN	DR
0	3445002	Other	1	2009-12-20 08:48:00+00:00	2009-12-20 08:48:00+00:00	0	0	0	0	0	0	0	0
1	3445002	Trazodone	1	2011-12-11 22:23:00+00:00	2012-01-17 00:33:00+00:00	6	0	0	0	0	0	1	0
2	3445002	Trazodone	2	2012-11-16 19:36:00+00:00	2013-01-20 04:18:00+00:00	57	0	0	0	1	0	1	0
3	3445002	Trazodone	3	2013-03-22 13:37:00+00:00	2013-03-22 17:22:00+00:00	1	0	0	0	1	0	1	0
4	3445002	Escitalopram	1	2018-01-12 15:00:00+00:00	2018-07-11 14:59:00+00:00	180	1	0	1	0	0	1	0

[18]:
```
# Create a function to count episodes per trial
def get_episodes_count (row):

    # get rows that ended before the current trial start for the person to get␣
 ↪all medication history
    previous_episodes = ad_dates_df[(ad_dates_df['person_id'] ==␣
 ↪row['person_id']) &
                                    (ad_dates_df['earliest_ad_end'] <␣
 ↪row['earliest_ad_start'])]

    # if no history then return 0 since no previous episodes exist
    if previous_episodes.empty:
        return 0

    # arrange based on earliest_ad_start
    previous_episodes.sort_values(by=["earliest_ad_start"], inplace=True)

    # initialize variables
    episode_count = 1 # at least 1 episode if there is at least 1 row in␣
 ↪previous_episodes
    current_start_date = None
    current_end_date = None

    # count episodes
    for index, previous_row in previous_episodes.iterrows():
        if current_start_date is None:  # First row
            # store start dates for comparison with succeeding rows
            current_start_date = previous_row['earliest_ad_start']
            current_end_date = previous_row['earliest_ad_end']
        else:
            if previous_row['earliest_ad_start'] <= current_end_date:  # Within␣
 ↪current episode
                current_end_date = max(current_end_date,␣
 ↪previous_row['earliest_ad_end'])
            else:  # New episode
                episode_count += 1
                current_start_date = previous_row['earliest_ad_start']
                current_end_date = previous_row['earliest_ad_end']

    return episode_count
```

[19]:
```
# Store episode history count data
ad_dates_df['num_episodes'] = ad_dates_df.apply(get_episodes_count, axis=1)

ad_dates_df.head(20)
```

[19]:

	person_id	ad_grouping	trial_number	earliest_ad_start	earliest_ad_end	dosage_days	dose_70p	other_ad	Remission	SN	SR	DN	DR	num_episodes
0	3445002	Other	1	2009-12-20 08:48:00+00:00	2009-12-20 08:48:00+00:00	0	0	0	0	0	0	0	0	0
1	3445002	Trazodone	1	2011-12-11 22:23:00+00:00	2012-01-17 00:33:00+00:00	6	0	0	0	0				
2	3445002	Trazodone	2	2012-11-16 19:36:00+00:00	2013-01-20 04:18:00+00:00	57	0	0	0	1				
3	3445002	Trazodone	3	2013-03-22 13:37:00+00:00	2013-03-22 17:22:00+00:00	1	0	0	0	1				
4	3445002	Escitalopram	1	2018-01-12 15:00:00+00:00	2018-07-11 14:59:00+00:00	180	1	0	1	0	1	0	0	1
5	3445002	Escitalopram	2	2018-09-23 15:00:00+00:00	2019-03-22 14:59:00+00:00	180	1	0	1	0				
6	8509505	Bupropion	1	2017-07-19 08:43:00+00:00	2019-03-19 00:00:00+00:00	664	1	0	1	0	1	0	0	1
7	8509505	Citalopram	1	2014-04-22 00:00:00+00:00	2015-10-26 23:59:00+00:00	2169	1	0	1	0	1	0	0	1
8	8509505	Citalopram	2	2016-01-09 10:24:00+00:00	2019-01-21 00:00:00+00:00	1107	1	0	1	0				
9	8509505	Fluoxetine	1	2021-03-24 16:47:00+00:00	2021-03-24 16:47:00+00:00	0	0	0	0	0	0	0	0	0
10	8509505	Trazodone	1	2017-07-10 13:57:00+00:00	2019-03-10 00:00:00+00:00	607	1	1	0	0	1	0	0	1
11	8509505	Venlafaxine	1	2019-01-21 17:16:00+00:00	2019-10-20 00:00:00+00:00	324	1	0	1	0	1	0	0	1
12	8509505	Sertraline	1	2012-08-05 00:00:00+00:00	2012-08-28 00:00:00+00:00	3	0	0	0	0	0	0	0	0
13	8509505	Sertraline	2	2012-10-09 00:00:00+00:00	2012-10-14 00:00:00+00:00	2	0	0	0	1				
14	8509505	Sertraline	3	2012-11-18 00:00:00+00:00	2012-12-09 00:00:00+00:00	2	0	0	0	1				
15	1328157	Other	1	2012-07-09 05:00:00+00:00	2013-02-05 11:59:59+00:00	213	1	0	1	0	1	0	0	1
16	1328157	Other	2	2013-04-08 05:00:00+00:00	2016-03-14 11:59:59+00:00	1078	1	0	1	0				
17	1328157	Paroxetine	1	2010-08-21 05:00:00+00:00	2016-12-06 11:59:59+00:00	2510	1	0	1	0	1	0	0	1
18	1328157	Doxepin	1	2003-03-14 05:00:00+00:00	2003-03-14 05:00:00+00:00	0	0	0	0	0	0	0	0	0
19	1328157	Sertraline	1	1999-04-10 05:00:00+00:00	2003-09-22 11:59:59+00:00	1627	1	0	1	0	1	0	0	1

```
2    0   1   0             2
3    0   1   0             3
4    0   1   0             4
5    1   1   0             5
6    0   1   1             4
7    0   1   0             3
8    1   1   0             4
9    0   1   1             5
10   0   1   1             4
11   0   1   1             5
12   0   0   0             0
13   0   0   0             1
14   0   0   0             2
15   0   1   1             1
16   1   1   1             2
17   0   1   1             1
18   0   0   0             0
19   0   0   0             0
```

[20]:
```
# Create a function to count remission history per trial
def get_remission_count (row):

    # get rows that ended before the current trial start for the person to get␣
 ↪all medication history
    previous_episodes = ad_dates_df[(ad_dates_df['person_id'] ==␣
 ↪row['person_id']) &
                                    (ad_dates_df['earliest_ad_end'] <␣
 ↪row['earliest_ad_start'])]

    # if no history then return 0 since no previous episodes exist
    if previous_episodes.empty:
        return 0

    # get the sum of the remission column to get the number of time patient␣
 ↪acheived remission
    remission_count = previous_episodes['Remission'].sum()

    return remission_count
```

[21]:
```
# Store remission history count data
ad_dates_df['num_remissions'] = ad_dates_df.apply(get_remission_count, axis=1)

ad_dates_df.head(20)
```

[21]:
```
    person_id   ad_grouping  trial_number         earliest_ad_start  \
0     3445002         Other             1 2009-12-20 08:48:00+00:00
1     3445002     Trazodone             1 2011-12-11 22:23:00+00:00
```

```
2     3445002     Trazodone             2 2012-11-16 19:36:00+00:00
3     3445002     Trazodone             3 2013-03-22 13:37:00+00:00
4     3445002  Escitalopram             1 2018-01-12 15:00:00+00:00
5     3445002  Escitalopram             2 2018-09-23 15:00:00+00:00
6     8509505     Bupropion             1 2017-07-19 08:43:00+00:00
7     8509505    Citalopram             1 2014-04-22 00:00:00+00:00
8     8509505    Citalopram             2 2016-01-09 10:24:00+00:00
9     8509505    Fluoxetine             1 2021-03-24 16:47:00+00:00
10    8509505     Trazodone             1 2017-07-10 13:57:00+00:00
11    8509505   Venlafaxine             1 2019-01-21 17:16:00+00:00
12    8509505    Sertraline             1 2012-08-05 00:00:00+00:00
13    8509505    Sertraline             2 2012-10-09 00:00:00+00:00
14    8509505    Sertraline             3 2012-11-18 00:00:00+00:00
15    1328157         Other             1 2012-07-09 05:00:00+00:00
16    1328157         Other             2 2013-04-08 05:00:00+00:00
17    1328157    Paroxetine             1 2010-08-21 05:00:00+00:00
18    1328157       Doxepin             1 2003-03-14 05:00:00+00:00
19    1328157    Sertraline             1 1999-04-10 05:00:00+00:00

              earliest_ad_end  dosage_days  dose_70p  other_ad  Remission  SN  \
0  2009-12-20 08:48:00+00:00            0         0         0          0   0
1  2012-01-17 00:33:00+00:00            6         0         0          0   0
2  2013-01-20 04:18:00+00:00           57         0         0          0   1
3  2013-03-22 17:22:00+00:00            1         0         0          0   1
4  2018-07-11 14:59:00+00:00          180         1         0          1   0
5  2019-03-22 14:59:00+00:00          180         1         0          1   0
6  2019-03-19 00:00:00+00:00          664         1         0          1   0
7  2015-10-26 23:59:00+00:00         2169         1         0          1   0
8  2019-01-21 00:00:00+00:00         1107         1         0          1   0
9  2021-03-24 16:47:00+00:00            0         0         0          0   0
10 2019-03-10 00:00:00+00:00          607         1         1          0   0
11 2019-10-20 00:00:00+00:00          324         1         0          1   0
12 2012-08-28 00:00:00+00:00            3         0         0          0   0
13 2012-10-14 00:00:00+00:00            2         0         0          0   1
14 2012-12-09 00:00:00+00:00            2         0         0          0   1
15 2013-02-05 11:59:59+00:00          213         1         0          1   0
16 2016-03-14 11:59:59+00:00         1078         1         0          1   0
17 2016-12-06 11:59:59+00:00         2510         1         0          1   0
18 2003-03-14 05:00:00+00:00            0         0         0          0   0
19 2003-09-22 11:59:59+00:00         1627         1         0          1   0

    SR  DN  DR  num_episodes  num_remissions
0    0   0   0             0               0
1    0   1   0             1               0
2    0   1   0             2               0
3    0   1   0             3               0
4    0   1   0             4               0
```

```
5    1   1   0            5              1
6    0   1   1            4              1
7    0   1   0            3              0
8    1   1   0            4              1
9    0   1   1            5              4
10   0   1   1            4              1
11   0   1   1            5              2
12   0   0   0            0              0
13   0   0   0            1              0
14   0   0   0            2              0
15   0   1   1            1              1
16   1   1   1            2              2
17   0   1   1            1              1
18   0   0   0            0              0
19   0   0   0            0              0
```

[22]:
```
# Create a function to count previous antidepressants taken per trial
def get_ad_count (row):

    # get rows that ended before the current trial start for the person to get␣
 ↪all medication history
    previous_episodes = ad_dates_df[(ad_dates_df['person_id'] ==␣
 ↪row['person_id']) &
                                    (ad_dates_df['earliest_ad_end'] <␣
 ↪row['earliest_ad_start'])]

    # if no history then return 0 since no previous episodes exist
    if previous_episodes.empty:
        return 0

    # get the number of unique antidepressants
    ad_count = previous_episodes['ad_grouping'].nunique()

    return ad_count
```

[23]:
```
# Store antidepressant number in their columns
ad_dates_df['num_antidepressants'] = ad_dates_df.apply(get_ad_count, axis=1)

ad_dates_df.head(20)
```

[23]:
```
    person_id   ad_grouping  trial_number         earliest_ad_start  \
0     3445002         Other             1 2009-12-20 08:48:00+00:00
1     3445002     Trazodone             1 2011-12-11 22:23:00+00:00
2     3445002     Trazodone             2 2012-11-16 19:36:00+00:00
3     3445002     Trazodone             3 2013-03-22 13:37:00+00:00
4     3445002  Escitalopram             1 2018-01-12 15:00:00+00:00
5     3445002  Escitalopram             2 2018-09-23 15:00:00+00:00
```

6	8509505	Bupropion	1	2017-07-19 08:43:00+00:00
7	8509505	Citalopram	1	2014-04-22 00:00:00+00:00
8	8509505	Citalopram	2	2016-01-09 10:24:00+00:00
9	8509505	Fluoxetine	1	2021-03-24 16:47:00+00:00
10	8509505	Trazodone	1	2017-07-10 13:57:00+00:00
11	8509505	Venlafaxine	1	2019-01-21 17:16:00+00:00
12	8509505	Sertraline	1	2012-08-05 00:00:00+00:00
13	8509505	Sertraline	2	2012-10-09 00:00:00+00:00
14	8509505	Sertraline	3	2012-11-18 00:00:00+00:00
15	1328157	Other	1	2012-07-09 05:00:00+00:00
16	1328157	Other	2	2013-04-08 05:00:00+00:00
17	1328157	Paroxetine	1	2010-08-21 05:00:00+00:00
18	1328157	Doxepin	1	2003-03-14 05:00:00+00:00
19	1328157	Sertraline	1	1999-04-10 05:00:00+00:00

	earliest_ad_end	dosage_days	dose_70p	other_ad	Remission	SN	\
0	2009-12-20 08:48:00+00:00	0	0	0	0	0	
1	2012-01-17 00:33:00+00:00	6	0	0	0	0	
2	2013-01-20 04:18:00+00:00	57	0	0	0	1	
3	2013-03-22 17:22:00+00:00	1	0	0	0	1	
4	2018-07-11 14:59:00+00:00	180	1	0	1	0	
5	2019-03-22 14:59:00+00:00	180	1	0	1	0	
6	2019-03-19 00:00:00+00:00	664	1	0	1	0	
7	2015-10-26 23:59:00+00:00	2169	1	0	1	0	
8	2019-01-21 00:00:00+00:00	1107	1	0	1	0	
9	2021-03-24 16:47:00+00:00	0	0	0	0	0	
10	2019-03-10 00:00:00+00:00	607	1	1	0	0	
11	2019-10-20 00:00:00+00:00	324	1	0	1	0	
12	2012-08-28 00:00:00+00:00	3	0	0	0	0	
13	2012-10-14 00:00:00+00:00	2	0	0	0	1	
14	2012-12-09 00:00:00+00:00	2	0	0	0	1	
15	2013-02-05 11:59:59+00:00	213	1	0	1	0	
16	2016-03-14 11:59:59+00:00	1078	1	0	1	0	
17	2016-12-06 11:59:59+00:00	2510	1	0	1	0	
18	2003-03-14 05:00:00+00:00	0	0	0	0	0	
19	2003-09-22 11:59:59+00:00	1627	1	0	1	0	

	SR	DN	DR	num_episodes	num_remissions	num_antidepressants
0	0	0	0	0	0	0
1	0	1	0	1	0	1
2	0	1	0	2	0	2
3	0	1	0	3	0	2
4	0	1	0	4	0	2
5	1	1	0	5	1	3
6	0	1	1	4	1	2
7	0	1	0	3	0	1
8	1	1	0	4	1	2

9	0	1	1	5	4	5
10	0	1	1	4	1	2
11	0	1	1	5	2	2
12	0	0	0	0	0	0
13	0	0	0	1	0	1
14	0	0	0	2	0	1
15	0	1	1	1	1	2
16	1	1	1	2	2	3
17	0	1	1	1	1	2
18	0	0	0	0	0	0
19	0	0	0	0	0	0

[24]:
```python
# Create dummy variables for number of episodes
# Find the minimum number of episodes
min_epi = ad_dates_df['num_episodes'].min()

# Find the maximum number of episodes
max_epi = ad_dates_df['num_episodes'].max()

print(f"Minimum number of episodes: {min_epi}")
print(f"Maximum number of episodes: {max_epi}")
```

```
Minimum number of episodes: 0
Maximum number of episodes: 19
```

[25]:
```python
# Define bins for episode ranges
bins = [0, 1, 19]

# Define labels for episode groups
labels = ['epi_01', 'epi_2p']

# Create a categorical column for episode ranges
ad_dates_df['epi_range'] = pd.cut(ad_dates_df['num_episodes'],
                                  bins=bins,
                                  labels=labels,
                                  right=True,
                                  include_lowest=True)

# Create dummy variables for the episode ranges
epi_dummies = pd.get_dummies(ad_dates_df['epi_range']).astype(int)

# Concatenate the dummy variables with the original DataFrame
ad_dates_df = pd.concat([ad_dates_df, epi_dummies], axis=1)

# Drop unneeded columns
cols_to_drop = ['num_episodes',
                'epi_range',
                'epi_01'] # drop to prevent dummy variable trap
```

```
ad_dates_df = ad_dates_df.drop(columns=cols_to_drop)

ad_dates_df.head()
```

[25]:

	person_id	ad_grouping	trial_number	earliest_ad_start	earliest_ad_end	dosage_days	dose_70p	other_ad	Remission	SN	SR	DN	DR	num_remissions	num_antidepressants	epi_2p
0	3445002	Other	1	2009-12-20 08:48:00+00:00	2009-12-20 08:48:00+00:00	0	0	0	0	0	0	0	0	0	0	0
1	3445002	Trazodone	1	2011-12-11 22:23:00+00:00	2012-01-17 00:33:00+00:00	6	0	0	0	0	0	1	0	0	1	0
2	3445002	Trazodone	2	2012-11-16 19:36:00+00:00	2013-01-20 04:18:00+00:00	57	0	0	0	1	0	1	0	0	2	1
3	3445002	Trazodone	3	2013-03-22 13:37:00+00:00	2013-03-22 17:22:00+00:00	1	0	0	0	1	0	1	0	0	2	1
4	3445002	Escitalopram	1	2018-01-12 15:00:00+00:00	2018-07-11 14:59:00+00:00	180	1	0	1	0	0	1	0	0	2	1

[26]:
```
# Create dummy variables for number of remissions
# Find the minimum number of remissions
min_rem = ad_dates_df['num_remissions'].min()

# Find the maximum number of remissions
max_rem = ad_dates_df['num_remissions'].max()

print(f"Minimum number of remissions: {min_rem}")
print(f"Maximum number of remissions: {max_rem}")
```

```
Minimum number of remissions: 0
Maximum number of remissions: 13
```

[27]:
```
# Define bins for remission ranges
bins = [0, 1, 13]

# Define labels for remission groups
labels = ['rem_01', 'rem_2p']

# Create a categorical column for remission ranges
ad_dates_df['rem_range'] = pd.cut(ad_dates_df['num_remissions'],
                                  bins=bins,
```

```
                                    labels=labels,
                                    right=True,
                                    include_lowest=True)

# Create dummy variables for the remission ranges
rem_dummies = pd.get_dummies(ad_dates_df['rem_range']).astype(int)

# Concatenate the dummy variables with the original DataFrame
ad_dates_df = pd.concat([ad_dates_df, rem_dummies], axis=1)

# Drop unneeded columns
cols_to_drop = ['num_remissions',
                'rem_range',
                'rem_01'] # drop to prevent dummy variable trap

ad_dates_df = ad_dates_df.drop(columns=cols_to_drop)

ad_dates_df.head()
```

[27]:

	person_id	ad_grouping	trial_number	earliest_ad_start	earliest_ad_end	dosage_days	dose_70p	other_ad	Remission	SN	SR	DN	DR	num_antidepressants	epi_2p	rem_2p
0	3445002	Other	1	2009-12-20 08:48:00+00:00	2009-12-20 08:48:00+00:00	0	0	0	0	0	0	0	0	0	0	0
1	3445002	Trazodone	1	2011-12-11 22:23:00+00:00	2012-01-17 00:33:00+00:00	6	0	0	0	0	0	1	0	1	0	0
2	3445002	Trazodone	2	2012-11-16 19:36:00+00:00	2013-01-20 04:18:00+00:00	57	0	0	0	1	0	1	0	2	1	0
3	3445002	Trazodone	3	2013-03-22 13:37:00+00:00	2013-03-22 17:22:00+00:00	1	0	0	0	1	0	1	0	2	1	0
4	3445002	Escitalopram	1	2018-01-12 15:00:00+00:00	2018-07-11 14:59:00+00:00	180	1	0	1	0	0	1	0	2	1	0

[28]:
```
ad_dates_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28304 entries, 0 to 28303
Data columns (total 16 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   person_id           28304 non-null  int64
```

```
 1   ad_grouping          28304 non-null  object
 2   trial_number         28304 non-null  int64
 3   earliest_ad_start    27793 non-null  datetime64[ns, UTC]
 4   earliest_ad_end      27833 non-null  datetime64[ns, UTC]
 5   dosage_days          28304 non-null  int64
 6   dose_70p             28304 non-null  int64
 7   other_ad             28304 non-null  int64
 8   Remission            28304 non-null  int64
 9   SN                   28304 non-null  int64
 10  SR                   28304 non-null  int64
 11  DN                   28304 non-null  int64
 12  DR                   28304 non-null  int64
 13  num_antidepressants  28304 non-null  int64
 14  epi_2p               28304 non-null  int64
 15  rem_2p               28304 non-null  int64
dtypes: datetime64[ns, UTC](2), int64(13), object(1)
memory usage: 3.5+ MB
```

[29]:
```python
# Create dummy variables for number of antidepressants
# range of antidepressants
# Find the minimum number of antidepressants
min_adep = ad_dates_df['num_antidepressants'].min()

# Find the maximum number of antidepressants
max_adep = ad_dates_df['num_antidepressants'].max()

print(f"Minimum number of antidepressants: {min_adep}")
print(f"Maximum number of antidepressants: {max_adep}")
```

```
Minimum number of antidepressants: 0
Maximum number of antidepressants: 12
```

[30]:
```python
# Define bins for antidepressant ranges
bins = [0, 1, 3, 12]

# Define labels for antidepressant groups
labels = ['adep_01', 'adep_23', 'adep_4p']

# Create a categorical column for antidepressant ranges
ad_dates_df['adep_range'] = pd.cut(ad_dates_df['num_antidepressants'],
                                   bins=bins,
                                   labels=labels,
                                   right=True,
                                   include_lowest=True)

# Create dummy variables for the antidepressant ranges
adep_dummies = pd.get_dummies(ad_dates_df['adep_range']).astype(int)
```

```
# Concatenate the dummy variables with the original DataFrame
ad_dates_df = pd.concat([ad_dates_df, adep_dummies], axis=1)

# Drop unneeded columns
cols_to_drop = ['num_antidepressants',
                'adep_range',
                'adep_01'] # drop to prevent dummy variable trap

ad_dates_df = ad_dates_df.drop(columns=cols_to_drop)

ad_dates_df.head()
```

[30]:

	person_id	ad_grouping	trial_number	earliest_ad_start	earliest_ad_end	dosage_days	dose_70p	other_ad	Remission	SN	SR	DN	DR	epi_2p	rem_2p	adep_23	adep_4p
0	3445002	Other	1	2009-12-20 08:48:00+00:00	2009-12-20 08:48:00+00:00	0	0	0	0	0	0	0	0	0	0	0	0
1	3445002	Trazodone	1	2011-12-11 22:23:00+00:00	2012-01-17 00:33:00+00:00	6	0	0	0	0	0	1	0	0	0	0	0
2	3445002	Trazodone	2	2012-11-16 19:36:00+00:00	2013-01-20 04:18:00+00:00	57	0	0	0	1	0	1	0	1	0	1	0
3	3445002	Trazodone	3	2013-03-22 13:37:00+00:00	2013-03-22 17:22:00+00:00	1	0	0	0	1	0	1	0	1	0	1	0
4	3445002	Escitalopram	1	2018-01-12 15:00:00+00:00	2018-07-11 14:59:00+00:00	180	1	0	1	0	0	1	0	1	0	1	0

[31]:
```
# Check for columns in ad_dates_df
ad_dates_df.columns
```

[31]:
```
Index(['person_id', 'ad_grouping', 'trial_number', 'earliest_ad_start',
       'earliest_ad_end', 'dosage_days', 'dose_70p', 'other_ad', 'Remission',
       'SN', 'SR', 'DN', 'DR', 'epi_2p', 'rem_2p', 'adep_23', 'adep_4p'],
      dtype='object')
```

[32]:
```
# Retain only primary key and new columns
cols_to_keep = ['person_id',
                'ad_grouping',
                'trial_number',
                'SN',
                'SR',
```

```
                'DN',
                'DR',
                'epi_2p',
                'rem_2p',
                'adep_23',
                'adep_4p']
ad_dates_df = ad_dates_df[cols_to_keep]
ad_dates_df.head()
```

[32]:

	person_id	ad_grouping	trial_number	SN	SR	DN	DR	epi_2p	rem_2p	\
0	3445002	Other	1	0	0	0	0	0	0	
1	3445002	Trazodone	1	0	0	1	0	0	0	
2	3445002	Trazodone	2	1	0	1	0	1	0	
3	3445002	Trazodone	3	1	0	1	0	1	0	
4	3445002	Escitalopram	1	0	0	1	0	1	0	

	adep_23	adep_4p
0	0	0
1	0	0
2	1	0
3	1	0
4	1	0

[33]:
```
# Left join to analysis dataframe for chosen antidepressant
analysis_df = analysis_df_dict[ad].copy()

analysis_df = pd.merge(analysis_df, ad_dates_df, on=['person_id',␣
 ↪'ad_grouping', 'trial_number'],
                       how='left')

analysis_df.head()
```

[33]:

	person_id	ad_grouping	trial_number	earliest_ad_start	\
0	1000039	Amitriptyline	1	2019-12-18 16:13:00+00:00	
1	1000370	Amitriptyline	1	2019-11-01 00:00:00+00:00	
2	1004308	Amitriptyline	1	2007-08-06 14:27:00+00:00	
3	1005542	Amitriptyline	1	2016-11-08 00:00:00+00:00	
4	1010384	Amitriptyline	1	2020-04-06 16:54:00+00:00	

	Remission	age 13-19	age 20-40	age 41-64	age 65-79	age 80-89	…	\
0	0	0	0	1	0	0	…	
1	0	0	0	0	1	0	…	
2	1	0	0	1	0	0	…	
3	0	0	0	1	0	0	…	
4	0	0	0	1	0	0	…	

	disease_12246311000119109	disease_15960141000119102	SN	SR	DN	DR	\

```
0                          0                          0   0   0   1   0
1                          0                          0   0   0   1   0
2                          0                          0   0   0   0   0
3                          0                          0   0   0   1   0
4                          1                          0   0   0   1   0

   epi_2p  rem_2p  adep_23  adep_4p
0       1       0        1        0
1       0       0        0        0
2       0       0        0        0
3       0       0        0        0
4       0       0        0        0

[5 rows x 1617 columns]
```

2.3.3 Diagnoses

[34]:
```
# Define function for processing diagnoses datasets and add to the analysis␣
 ↪dataframe
## analysis_df - the analysis dataframe
## dataset - the All of Us dataset (eg dxo_conditions_df)
## valid_vals - valid values for either inpatient or outpatient
## visit_type_column - column name that should be filtered to only valid values␣
 ↪for inpatient or outpatient
## date_column - column name for the date of the diagnosis
## prefix - prefix for created dummy variable columns
def process_diagnoses(analysis_df, dataset, visit_type_column, valid_vals,␣
 ↪date_column, prefix):

    # Filter the dataset
    dataset = dataset[dataset[visit_type_column].isin(valid_vals)]

    # Retain only needed columns
    cols_to_keep = ['person_id',
                    date_column,
                    'source_concept_code']
    dataset = dataset[cols_to_keep]

    # Left join to a temporary copy of analysis_df_reduced
    analysis_df_temp = pd.merge(analysis_df, dataset, on=['person_id'],␣
 ↪how='left').copy()

    # Convert the date columns to datetime if they are not already
    analysis_df_temp['earliest_ad_start'] = pd.
 ↪to_datetime(analysis_df_temp['earliest_ad_start'], errors='coerce')
    analysis_df_temp[date_column] = pd.
 ↪to_datetime(analysis_df_temp[date_column], errors='coerce')
```

```
  # Filter the DataFrame retain only entries where dxi is empty or happened␣
↪before antidepressant start
  analysis_df_temp = analysis_df_temp[(analysis_df_temp[date_column].
↪isnull()) |
                                      (analysis_df_temp[date_column] <␣
↪analysis_df_temp['earliest_ad_start'])]

  # Fill missing values in the source_concept_code column with 0
  analysis_df_temp['source_concept_code'] =␣
↪analysis_df_temp['source_concept_code'].fillna('0')

  # drop unneeded column
  analysis_df_temp = analysis_df_temp.drop(columns = [date_column])

  # Create a copy containing only primary key columns and dataset␣
↪source_concept_code to create dummy variables
  dataset_per_trial = analysis_df_temp[['person_id', 'ad_grouping',␣
↪'trial_number', 'source_concept_code']].copy()

  # Create dummy variables for the dxi codes
  dataset_dummies = pd.get_dummies(dataset_per_trial['source_concept_code'],␣
↪prefix=prefix).astype(int)

  # Concatenate the dummy variables with the original DataFrame
  dataset_per_trial = pd.concat([dataset_per_trial, dataset_dummies], axis=1)

  # Drop source_concept_code column
  dataset_per_trial = dataset_per_trial.drop(columns=['source_concept_code'])

  # Aggregate dataset_per_trial based on primary keys, use max value for␣
↪aggregation
  aggregation_dict = {col: 'max'
                      for col in dataset_per_trial.columns if col not in␣
↪['person_id', 'ad_grouping', 'trial_number']}

  # Group by the primary key columns and aggregate
  dataset_per_trial = dataset_per_trial.groupby(['person_id', 'ad_grouping',␣
↪'trial_number'],
                                                as_index=False).
↪agg(aggregation_dict)

  # Left join to the main analysis dataframe
  analysis_df= pd.merge(analysis_df, dataset_per_trial, on=['person_id',␣
↪'ad_grouping', 'trial_number'], how='left')
```

```
    # Drop unneeded columns
    analysis_df = analysis_df.drop(columns=[prefix + '_0'])

    # Get a list of column names that start with prefix + '_'
    dataset_columns = [col for col in analysis_df.columns if col.
 ↪startswith(prefix + '_')]

    # Fill missing values in these columns with 0
    analysis_df[dataset_columns] = analysis_df[dataset_columns].fillna(0)


    return analysis_df
```

dxi condition

[35]:
```
# Inspect the dataset to get column names
dataset = dxi_condition_df
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9099 entries, 0 to 9098
Data columns (total 5 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   person_id                      9099 non-null   Int64
 1   condition_start_datetime       9099 non-null   datetime64[ns, UTC]
 2   visit_occurrence_concept_name  8054 non-null   object
 3   source_concept_code            9099 non-null   object
 4   source_vocabulary              9099 non-null   object
dtypes: Int64(1), datetime64[ns, UTC](1), object(3)
memory usage: 364.4+ KB
```

[36]:
```
# Check if everything is ICD9CM only
dataset['source_vocabulary'].unique()

# If not ICD9CM only PANIC! :)
```

[36]:
```
array(['ICD9CM'], dtype=object)
```

[37]:
```
# Get visit type column
visit_type_column = 'visit_occurrence_concept_name'

# Check for unique values of visit type to find values for inpatient visits
dataset[visit_type_column].unique()
```

[37]:
```
array([None, 'Office Visit', 'Inpatient Hospital', 'Emergency Room Visit',
       'Outpatient Visit', 'Non-hospital institution Visit',
       'Ambulatory Radiology Clinic / Center',
       'Emergency Room and Inpatient Visit', 'Nursing Facility',
```

```
       'Observation Room', 'Ambulatory Rehabilitation Visit',
       'Inpatient Visit', 'Unknown Value (but present in data)',
       'Laboratory Visit', 'Outpatient Hospital', 'Telehealth',
       'Ambulatory Surgical Center'], dtype=object)
```

[38]:
```
# Filter based on rows for in-patient diagnoses only
valid_vals = ['Emergency Room and Inpatient Visit',
              'Nursing Facility',
              'Inpatient Hospital',
              'Inpatient Visit']
```

[39]:
```
# Set the date column
date_column = 'condition_start_datetime'

# Set the prefix
prefix = 'dxi'
```

[40]:
```
analysis_df = process_diagnoses(analysis_df, dataset, visit_type_column,␣
 ↪valid_vals, date_column, prefix)
```

[41]:
```
analysis_df.head()
```

[41]:

	person_id	ad_grouping	trial_number	earliest_ad_start	\
0	1000039	Amitriptyline	1	2019-12-18 16:13:00+00:00	
1	1000370	Amitriptyline	1	2019-11-01 00:00:00+00:00	
2	1004308	Amitriptyline	1	2007-08-06 14:27:00+00:00	
3	1005542	Amitriptyline	1	2016-11-08 00:00:00+00:00	
4	1010384	Amitriptyline	1	2020-04-06 16:54:00+00:00	

	Remission	age 13-19	age 20-40	age 41-64	age 65-79	age 80-89	…	\
0	0	0	0	1	0	0	…	
1	0	0	0	0	1	0	…	
2	1	0	0	1	0	0	…	
3	0	0	0	1	0	0	…	
4	0	0	0	1	0	0	…	

	disease_15960141000119102	SN	SR	DN	DR	epi_2p	rem_2p	adep_23	\
0	0	0	0	1	0	1	0	1	
1	0	0	0	1	0	0	0	0	
2	0	0	0	0	0	0	0	0	
3	0	0	0	1	0	0	0	0	
4	0	0	0	1	0	0	0	0	

	adep_4p	dxi_428.0
0	0	0.0
1	0	0.0
2	0	0.0

```
3        0        0.0
4        0        0.0

[5 rows x 1618 columns]
```

[42]:
```
analysis_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1984 entries, 0 to 1983
Columns: 1618 entries, person_id to dxi_428.0
dtypes: float64(1), int64(1615), object(2)
memory usage: 24.5+ MB
```

dxo condition

[43]:
```
# Inspect the dataset to get column names
dataset = dxo_condition_df
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63907 entries, 0 to 63906
Data columns (total 5 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   person_id                      63907 non-null  Int64
 1   condition_start_datetime       63907 non-null  datetime64[ns, UTC]
 2   visit_occurrence_concept_name  52885 non-null  object
 3   source_concept_code            63907 non-null  object
 4   source_vocabulary              63907 non-null  object
dtypes: Int64(1), datetime64[ns, UTC](1), object(3)
memory usage: 2.5+ MB
```

[44]:
```
# Check if everything is ICD9CM only
dataset['source_vocabulary'].unique()

# If not ICD9CM only PANIC! :)
```

[44]:
```
array(['ICD9CM'], dtype=object)
```

[45]:
```
# Get visit type column
visit_type_column = 'visit_occurrence_concept_name'

# Check for unique values of visit type to find values for outpatient visits
dataset[visit_type_column].unique()
```

[45]:
```
array([None, 'Non-hospital institution Visit', 'Inpatient Hospital',
       'Pharmacy visit', 'Ambulatory Rehabilitation Visit',
       'Outpatient Visit', 'Emergency Room Visit',
       'Emergency Room and Inpatient Visit', 'Office Visit', 'Telehealth',
```

```
       'Laboratory Visit', 'Inpatient Visit', 'Outpatient Hospital',
       'Unknown Value (but present in data)', 'Nursing Facility',
       'Ambulatory Radiology Clinic / Center', 'Home Visit',
       'Ambulatory Surgical Center', 'Observation Room'], dtype=object)
```

[46]:
```
# Filter based on rows for outpatient diagnoses only
valid_vals = ['Non-hospital institution Visit',
              'Pharmacy visit',
              'Ambulatory Rehabilitation Visit',
              'Outpatient Visit',
              'Emergency Room Visit',
              'Office Visit',
              'Telehealth',
              'Laboratory Visit',
              'Outpatient Hospital',
              'Ambulatory Radiology Clinic / Center',
              'Home Visit',
              'Ambulatory Surgical Center',
              'Observation Room']
```

[47]:
```
# Set the date column
date_column = 'condition_start_datetime'

# Set the prefix
prefix = 'dx'
```

[48]:
```
analysis_df = process_diagnoses(analysis_df, dataset, visit_type_column,␣
 ↪valid_vals, date_column, prefix)
```

[49]:
```
analysis_df.head()
```

[49]:

	person_id	ad_grouping	trial_number	earliest_ad_start	\
0	1000039	Amitriptyline	1	2019-12-18 16:13:00+00:00	
1	1000370	Amitriptyline	1	2019-11-01 00:00:00+00:00	
2	1004308	Amitriptyline	1	2007-08-06 14:27:00+00:00	
3	1005542	Amitriptyline	1	2016-11-08 00:00:00+00:00	
4	1010384	Amitriptyline	1	2020-04-06 16:54:00+00:00	

	Remission	age 13-19	age 20-40	age 41-64	age 65-79	age 80-89	…	\
0	0	0	0	1	0	0	…	
1	0	0	0	0	1	0	…	
2	1	0	0	1	0	0	…	
3	0	0	0	1	0	0	…	
4	0	0	0	1	0	0	…	

	dx_250.01	dx_272.2	dx_296.20	dx_296.30	dx_296.33	dx_300.02	dx_311	\
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

```
1        0.0       0.0        0.0        0.0        0.0        0.0      1.0
2        0.0       0.0        0.0        0.0        0.0        0.0      0.0
3        0.0       0.0        1.0        0.0        1.0        0.0      0.0
4        0.0       0.0        0.0        0.0        0.0        0.0      0.0

   dx_357.2  dx_728.87  dx_780.09
0       0.0        0.0        0.0
1       0.0        0.0        0.0
2       0.0        0.0        0.0
3       0.0        0.0        0.0
4       0.0        0.0        0.0

[5 rows x 1628 columns]
```

[50]:
```
analysis_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1984 entries, 0 to 1983
Columns: 1628 entries, person_id to dx_780.09
dtypes: float64(11), int64(1615), object(2)
memory usage: 24.6+ MB
```

dxo measurement

[51]:
```
# Inspect the dataset to get column names
dataset = dxo_measurement_df
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 94 entries, 0 to 93
Data columns (total 5 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   person_id                      94 non-null     Int64
 1   measurement_datetime           94 non-null     datetime64[ns, UTC]
 2   visit_occurrence_concept_name  20 non-null     object
 3   source_concept_code            94 non-null     object
 4   source_vocabulary              94 non-null     object
dtypes: Int64(1), datetime64[ns, UTC](1), object(3)
memory usage: 3.9+ KB
```

[52]:
```
# Check if everything is ICD9CM only
dataset['source_vocabulary'].unique()

# If not ICD9CM only PANIC! :)
```

[52]:
```
array(['ICD9CM'], dtype=object)
```

[53]:
```
# Get visit type column
visit_type_column = 'visit_occurrence_concept_name'

# Check for unique values of visit type to find values for outpatient visits
dataset[visit_type_column].unique()
```

[53]:
```
array([None, 'Outpatient Visit'], dtype=object)
```

[54]:
```
# Filter based on rows for outpatient diagnoses only
valid_vals = ['Outpatient Visit']
```

[55]:
```
# Set the date column
date_column = 'measurement_datetime'

# Set the prefix
prefix = 'dx'
```

[56]:
```
analysis_df = process_diagnoses(analysis_df, dataset, visit_type_column,␣
 ↪valid_vals, date_column, prefix)
```

[57]:
```
analysis_df.head()
```

[57]:

	person_id	ad_grouping	trial_number	earliest_ad_start	Remission	age 13-19	age 20-40	age 41-64	age 65-79	age 80-89	…	dx_272.2	dx_296.20	dx_296.30	dx_296.33	dx_300.02	dx_311	dx_357.2	dx_728.87	dx_780.09	dx_V76.12
0	1000039	Amitriptyline	1	2019-12-18 16:13:00+00:00	0	0	0	1	0	0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	1000370	Amitriptyline	1	2019-11-01 00:00:00+00:00	0	0	0	0	1	0	…	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
2	1004308	Amitriptyline	1	2007-08-06 14:27:00+00:00	1	0	0	1	0	0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1005542	Amitriptyline	1	2016-11-08 00:00:00+00:00	0	0	0	1	0	0	…	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
4	1010384	Amitriptyline	1	2020-04-06 16:54:00+00:00	0	0	0	1	0	0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
3        0.0        0.0        0.0
4        0.0        0.0        0.0

[5 rows x 1629 columns]
```

[58]:
```
analysis_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1984 entries, 0 to 1983
Columns: 1629 entries, person_id to dx_V76.12
dtypes: float64(12), int64(1615), object(2)
memory usage: 24.7+ MB
```

dxo procedure

[59]:
```
# Inspect the dataset to get column names
dataset = dxo_procedure_df
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3407 entries, 0 to 3406
Data columns (total 5 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   person_id                       3407 non-null   Int64
 1   procedure_datetime              3407 non-null   datetime64[ns, UTC]
 2   visit_occurrence_concept_name   2914 non-null   object
 3   source_concept_code             3407 non-null   object
 4   source_vocabulary               3407 non-null   object
dtypes: Int64(1), datetime64[ns, UTC](1), object(3)
memory usage: 136.5+ KB
```

[60]:
```
# Check if everything is ICD9CM only
dataset['source_vocabulary'].unique()

# If not ICD9CM only PANIC! :)
```

[60]:
```
array(['ICD9CM'], dtype=object)
```

[61]:
```
# Get visit type column
visit_type_column = 'visit_occurrence_concept_name'

# Check for unique values of visit type to find values for outpatient visits
dataset[visit_type_column].unique()
```

[61]:
```
array([None, 'Office Visit', 'Outpatient Visit', 'Emergency Room Visit',
       'Inpatient Hospital', 'Non-hospital institution Visit',
       'Home Visit', 'Inpatient Visit'], dtype=object)
```

[62]:
```
# Filter based on rows for outpatient diagnoses only
valid_vals = ['Office Visit',
              'Outpatient Visit',
              'Emergency Room Visit',
              'Non-hospital institution Visit',
              'Home Visit']
```

[63]:
```
# Set the date column
date_column = 'procedure_datetime'

# Set the prefix
prefix = 'dx'
```

[64]:
```
analysis_df = process_diagnoses(analysis_df, dataset, visit_type_column,␣
 ↪valid_vals, date_column, prefix)
```

[65]:
```
analysis_df.head()
```

[65]:

	person_id	ad_grouping	trial_number	earliest_ad_start	\
0	1000039	Amitriptyline	1	2019-12-18 16:13:00+00:00	
1	1000370	Amitriptyline	1	2019-11-01 00:00:00+00:00	
2	1004308	Amitriptyline	1	2007-08-06 14:27:00+00:00	
3	1005542	Amitriptyline	1	2016-11-08 00:00:00+00:00	
4	1010384	Amitriptyline	1	2020-04-06 16:54:00+00:00	

	Remission	age 13-19	age 20-40	age 41-64	age 65-79	age 80-89	…	\
0	0	0	0	1	0	0	…	
1	0	0	0	0	1	0	…	
2	1	0	0	1	0	0	…	
3	0	0	0	1	0	0	…	
4	0	0	0	1	0	0	…	

	dx_296.20	dx_296.30	dx_296.33	dx_300.02	dx_311	dx_357.2	dx_728.87	\
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	1.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	1.0	0.0	1.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

	dx_780.09	dx_V76.12	dx_V72.31
0	0.0	0.0	0.0
1	0.0	0.0	0.0
2	0.0	0.0	0.0
3	0.0	0.0	0.0
4	0.0	0.0	0.0

[5 rows x 1630 columns]

[66]:
```
analysis_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1984 entries, 0 to 1983
Columns: 1630 entries, person_id to dx_V72.31
dtypes: float64(13), int64(1615), object(2)
memory usage: 24.7+ MB
```

2.3.4 Procedures

[67]:
```
# Define function for processing procedure datasets and add to the analysis␣
 ↪dataframe
## analysis_df - the analysis dataframe
## dataset - the All of Us dataset (eg dxo_conditions_df)
## date_column - column name for the date of the diagnosis
## prefix - prefix for created dummy variable columns
def process_procedure(analysis_df, dataset, date_column, prefix):

    # Retain only needed columns
    cols_to_keep = ['person_id',
                    date_column,
                    'source_concept_code']
    dataset = dataset[cols_to_keep]

    # Left join to a temporary copy of analysis_df_reduced
    analysis_df_temp = pd.merge(analysis_df, dataset, on=['person_id'],␣
 ↪how='left').copy()

    # Convert the date columns to datetime if they are not already
    analysis_df_temp['earliest_ad_start'] = pd.
 ↪to_datetime(analysis_df_temp['earliest_ad_start'], errors='coerce')
    analysis_df_temp[date_column] = pd.
 ↪to_datetime(analysis_df_temp[date_column], errors='coerce')

    # Filter the DataFrame retain only entries where dxi is empty or happened␣
 ↪before antidepressant start
    analysis_df_temp = analysis_df_temp[(analysis_df_temp[date_column].
 ↪isnull()) |
                                        (analysis_df_temp[date_column] <␣
 ↪analysis_df_temp['earliest_ad_start'])]

    # Fill missing values in the source_concept_code column with 0
    analysis_df_temp['source_concept_code'] =␣
 ↪analysis_df_temp['source_concept_code'].fillna('0')

    # drop unneeded column
    analysis_df_temp = analysis_df_temp.drop(columns = [date_column])
```

```
    # Create a copy containing only primary key columns and dataset␣
↪source_concept_code to create dummy variables
    dataset_per_trial = analysis_df_temp[['person_id', 'ad_grouping',␣
↪'trial_number', 'source_concept_code']].copy()

    # Create dummy variables for the dxi codes
    dataset_dummies = pd.get_dummies(dataset_per_trial['source_concept_code'],␣
↪prefix=prefix).astype(int)

    # Concatenate the dummy variables with the original DataFrame
    dataset_per_trial = pd.concat([dataset_per_trial, dataset_dummies], axis=1)

    # Drop source_concept_code column
    dataset_per_trial = dataset_per_trial.drop(columns=['source_concept_code'])

    # Aggregate dataset_per_trial based on primary keys, use max value for␣
↪aggregation
    aggregation_dict = {col: 'max'
                        for col in dataset_per_trial.columns if col not in␣
↪['person_id', 'ad_grouping', 'trial_number']}

    # Group by the primary key columns and aggregate
    dataset_per_trial = dataset_per_trial.groupby(['person_id', 'ad_grouping',␣
↪'trial_number'],
                                                  as_index=False).
↪agg(aggregation_dict)

    # Left join to the main analysis dataframe
    analysis_df= pd.merge(analysis_df, dataset_per_trial, on=['person_id',␣
↪'ad_grouping', 'trial_number'], how='left')

    # Drop unneeded columns
    analysis_df = analysis_df.drop(columns=[prefix + '_0'])

    # Get a list of column names that start with prefix + '_'
    dataset_columns = [col for col in analysis_df.columns if col.
↪startswith(prefix + '_')]

    # Fill missing values in these columns with 0
    analysis_df[dataset_columns] = analysis_df[dataset_columns].fillna(0)


    return analysis_df
```

[68]:
```
# Inspect the dataset to get column names
dataset = px_df
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6003 entries, 0 to 6002
Data columns (total 5 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   person_id                       6003 non-null   Int64
 1   procedure_datetime              6003 non-null   datetime64[ns, UTC]
 2   visit_occurrence_concept_name   4941 non-null   object
 3   source_concept_code             6003 non-null   object
 4   source_vocabulary               6003 non-null   object
dtypes: Int64(1), datetime64[ns, UTC](1), object(3)
memory usage: 240.5+ KB
```

[69]:
```
# Check if everything is CPT4 only
dataset['source_vocabulary'].unique()

# If not CPT4 only PANIC! :)
```

[69]:
```
array(['CPT4'], dtype=object)
```

[70]:
```
# Set the date column
date_column = 'procedure_datetime'

# Set the prefix
prefix = 'px'
```

[71]:
```
analysis_df = process_procedure(analysis_df, dataset, date_column, prefix)
```

[72]:
```
analysis_df.head()
```

[72]:

	person_id	ad_grouping	trial_number	earliest_ad_start	Remission	age 13-19	age 20-40	age 41-64	age 65-79	age 80-89	…
0	1000039	Amitriptyline	1	2019-12-18 16:13:00+00:00	0	0	0	1	0	0	…
1	1000370	Amitriptyline	1	2019-11-01 00:00:00+00:00	0	0	0	0	1	0	…
2	1004308	Amitriptyline	1	2007-08-06 14:27:00+00:00	1	0	0	1	0	0	…
3	1005542	Amitriptyline	1	2016-11-08 00:00:00+00:00	0	0	0	1	0	0	…
4	1010384	Amitriptyline	1	2020-04-06 16:54:00+00:00	0	0	0	1	0	0	…

```
   dx_300.02  dx_311  dx_357.2  dx_728.87  dx_780.09  dx_V76.12  dx_V72.31  \
0        0.0     0.0       0.0        0.0        0.0        0.0        0.0
1        0.0     1.0       0.0        0.0        0.0        0.0        0.0
2        0.0     0.0       0.0        0.0        0.0        0.0        0.0
3        0.0     0.0       0.0        0.0        0.0        0.0        0.0
4        0.0     0.0       0.0        0.0        0.0        0.0        0.0

   px_90805  px_90807  px_90862
0       0.0       0.0       0.0
1       1.0       0.0       1.0
2       0.0       0.0       0.0
3       0.0       0.0       0.0
4       0.0       0.0       0.0

[5 rows x 1633 columns]
```

[73]:
```
analysis_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1984 entries, 0 to 1983
Columns: 1633 entries, person_id to px_90862
dtypes: float64(16), int64(1615), object(2)
memory usage: 24.7+ MB
```

2.3.5 Medication

[74]:
```
# Define function for processing procedure datasets and add to the analysis␣
 ↪dataframe
## analysis_df - the analysis dataframe
## dataset - the All of Us dataset (eg dxo_conditions_df)
## date_column - column name for the date of the diagnosis
## prefix - prefix for created dummy variable columns
def process_medication(analysis_df, dataset, date_column, prefix):

    # Retain only needed columns
    cols_to_keep = ['person_id',
                    date_column,
                    'rx_code']
    dataset = dataset[cols_to_keep]

    # drop duplicates
    dataset = dataset.drop_duplicates()

    # Left join to a temporary copy of analysis_df_reduced
    analysis_df_temp = pd.merge(analysis_df, dataset, on=['person_id'],␣
 ↪how='left').copy()

    # Convert the date columns to datetime if they are not already
```

```
    analysis_df_temp['earliest_ad_start'] = pd.
↪to_datetime(analysis_df_temp['earliest_ad_start'], errors='coerce')
    analysis_df_temp[date_column] = pd.
↪to_datetime(analysis_df_temp[date_column], errors='coerce')

    # Filter the DataFrame retain only entries where dxi is empty or happened␣
↪before antidepressant start
    analysis_df_temp = analysis_df_temp[(analysis_df_temp[date_column].
↪isnull()) |
                                        (analysis_df_temp[date_column] <␣
↪analysis_df_temp['earliest_ad_start'])]

    # Fill missing values in the source_concept_code column with 0
    analysis_df_temp['rx_code'] = analysis_df_temp['rx_code'].fillna('0')

    # drop unneeded column
    analysis_df_temp = analysis_df_temp.drop(columns = [date_column])

    # Create a copy containing only primary key columns and dataset␣
↪source_concept_code to create dummy variables
    dataset_per_trial = analysis_df_temp[['person_id', 'ad_grouping',␣
↪'trial_number', 'rx_code']].copy()

    # Create dummy variables for the dxi codes
    dataset_dummies = pd.get_dummies(dataset_per_trial['rx_code'],␣
↪prefix=prefix).astype(int)

    # Concatenate the dummy variables with the original DataFrame
    dataset_per_trial = pd.concat([dataset_per_trial, dataset_dummies], axis=1)

    # Drop source_concept_code column
    dataset_per_trial = dataset_per_trial.drop(columns=['rx_code'])

    # Aggregate dataset_per_trial based on primary keys, use max value for␣
↪aggregation
    aggregation_dict = {col: 'max'
                        for col in dataset_per_trial.columns if col not in␣
↪['person_id', 'ad_grouping', 'trial_number']}

    # Group by the primary key columns and aggregate
    dataset_per_trial = dataset_per_trial.groupby(['person_id', 'ad_grouping',␣
↪'trial_number'],
                                                  as_index=False).
↪agg(aggregation_dict)

    # Left join to the main analysis dataframe
```

```
    analysis_df = pd.merge(analysis_df, dataset_per_trial, on=['person_id',␣
 ↪'ad_grouping', 'trial_number'], how='left')


    # Check if the column rx_0 exists

    if (prefix + '_0') in analysis_df.columns:
        # Drop unneeded columns
        analysis_df = analysis_df.drop(columns=[prefix + '_0'])

    # Get a list of column names that start with prefix + '_'
    dataset_columns = [col for col in analysis_df.columns if col.
 ↪startswith(prefix + '_')]

    # Fill missing values in these columns with 0
    analysis_df[dataset_columns] = analysis_df[dataset_columns].fillna(0)


    return analysis_df
```

[75]:
```
# Inspect the dataset to get column names
dataset = rx_df
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 599654 entries, 0 to 599653
Data columns (total 5 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   person_id                     599654 non-null  Int64
 1   standard_concept_name         599654 non-null  object
 2   standard_concept_code         599654 non-null  object
 3   standard_vocabulary           599654 non-null  object
 4   drug_exposure_start_datetime  599654 non-null  datetime64[ns, UTC]
dtypes: Int64(1), datetime64[ns, UTC](1), object(3)
memory usage: 23.4+ MB
```

[76]:
```
# create a datraframe for grouping containg all unique medication codes
selected_columns = ['standard_concept_name', 'standard_concept_code',␣
 ↪'standard_vocabulary']
selected_rx_df = rx_df[selected_columns].copy()

# Remove duplicates
selected_rx_df = selected_rx_df.drop_duplicates()

selected_rx_df.head()
```

[76]:
```
                                standard_concept_name standard_concept_code  \
0            30 ML morphine sulfate 1 MG/ML Injection                1728999
1   acetaminophen 500 MG / hydrocodone bitartrate …                 857109
13  24 HR bupropion hydrobromide 348 MG Extended R…                 993567
50  acetaminophen 500 MG / caffeine 65 MG Disinteg…                 702615
67  acetaminophen 325 MG / hydrocodone bitartrate …                 857022

   standard_vocabulary
0               RxNorm
1               RxNorm
13              RxNorm
50              RxNorm
67              RxNorm
```

[77]:
```
# Store as csv file for manual grouping
filename = './data/selected_rx_df_' + ad + '.csv'
selected_rx_df.to_csv(filename, index = False)
```

[78]:
```
# Read rx grouping file after manual grouping
filename = './data/selected_rx_df_' + ad + '_grouped.csv'
selected_rx_df_grouped = pd.read_csv(filename)

selected_rx_df_grouped.head()
```

[78]:
```
                              standard_concept_name standard_concept_code  \
0          30 ML morphine sulfate 1 MG/ML Injection                1728999
1  acetaminophen 500 MG / hydrocodone bitartrate …                 857109
2  24 HR bupropion hydrobromide 348 MG Extended R…                 993567
3  acetaminophen 500 MG / caffeine 65 MG Disinteg…                 702615
4  acetaminophen 325 MG / hydrocodone bitartrate …                 857022

  standard_vocabulary   rx_code
0              RxNorm      7052
1              RxNorm  5489_161
2              RxNorm     42347
3              RxNorm       161
4              RxNorm  5489_161
```

[79]:
```
# Left join to group rx codes
dataset = pd.merge(dataset, selected_rx_df_grouped[['standard_concept_code',␣
 ↪'standard_vocabulary', 'rx_code']],
                   on=['standard_concept_code', 'standard_vocabulary'],
                   how='left')

dataset.head()
```

[79]:
```
   person_id                                standard_concept_name  \
0    1425125          30 ML morphine sulfate 1 MG/ML Injection
1    3476915  acetaminophen 500 MG / hydrocodone bitartrate …
2    1679094          30 ML morphine sulfate 1 MG/ML Injection
3    1754365          30 ML morphine sulfate 1 MG/ML Injection
4    1808098          30 ML morphine sulfate 1 MG/ML Injection

  standard_concept_code standard_vocabulary drug_exposure_start_datetime  \
0               1728999              RxNorm    2008-09-06 05:00:00+00:00
1                857109              RxNorm    2011-02-26 05:00:00+00:00
2               1728999              RxNorm    2014-10-05 16:49:00+00:00
3               1728999              RxNorm    2010-10-10 15:13:46+00:00
4               1728999              RxNorm    2021-08-20 09:07:00+00:00

    rx_code
0      7052
1  5489_161
2      7052
3      7052
4      7052
```

[80]:
```
# Set the date column
date_column = 'drug_exposure_start_datetime'

# Set the prefix
prefix = 'rx'
```

[81]:
```
analysis_df = process_medication(analysis_df, dataset, date_column, prefix)
```

[82]:
```
# Modify interaction columns to include products of root terms
# rx_5489_161
# Check if the interaction and root term columns exist
if all(col in analysis_df.columns for col in ['rx_5489_161', 'rx_5489',␣
 ↪'rx_161']):
    # make sure that operation can be done by changing to float
    analysis_df['rx_5489_161'] = analysis_df['rx_5489_161'].astype(float)
    analysis_df['rx_5489'] = analysis_df['rx_5489'].astype(float)
    analysis_df['rx_161'] = analysis_df['rx_161'].astype(float)

    # If interaction term has a value of zero then change to the product of␣
 ↪root terms
    analysis_df['rx_5489_161'] = analysis_df.apply(
        lambda row: row['rx_5489'] * row['rx_161'] if row['rx_5489_161'] == 0␣
 ↪else row['rx_5489_161'], axis=1)

analysis_df.head()
```

[82]:
```
   person_id    ad_grouping  trial_number          earliest_ad_start  \
0    1000039  Amitriptyline             1  2019-12-18 16:13:00+00:00
1    1000370  Amitriptyline             1  2019-11-01 00:00:00+00:00
2    1004308  Amitriptyline             1  2007-08-06 14:27:00+00:00
3    1005542  Amitriptyline             1  2016-11-08 00:00:00+00:00
4    1010384  Amitriptyline             1  2020-04-06 16:54:00+00:00

   Remission  age 13-19  age 20-40  age 41-64  age 65-79  age 80-89  …  \
0          0          0          0          1          0          0  …
1          0          0          0          0          1          0  …
2          1          0          0          1          0          0  …
3          0          0          0          1          0          0  …
4          0          0          0          1          0          0  …

   rx_42347  rx_51272  rx_5489  rx_5489_161  rx_6470  rx_6472  rx_704  \
0       0.0       0.0      0.0          0.0      1.0      0.0     0.0
1       0.0       0.0      0.0          0.0      0.0      0.0     0.0
2       0.0       0.0      0.0          1.0      0.0      0.0     0.0
3       0.0       0.0      0.0          0.0      0.0      0.0     0.0
4       0.0       0.0      0.0          1.0      0.0      0.0     0.0

   rx_7052  rx_7804  rx_8745
0      0.0      0.0      0.0
1      1.0      0.0      1.0
2      0.0      0.0      0.0
3      0.0      0.0      0.0
4      1.0      1.0      0.0

[5 rows x 1651 columns]
```

[83]:
```
analysis_df.head()
```

[83]:
```
   person_id    ad_grouping  trial_number          earliest_ad_start  \
0    1000039  Amitriptyline             1  2019-12-18 16:13:00+00:00
1    1000370  Amitriptyline             1  2019-11-01 00:00:00+00:00
2    1004308  Amitriptyline             1  2007-08-06 14:27:00+00:00
3    1005542  Amitriptyline             1  2016-11-08 00:00:00+00:00
4    1010384  Amitriptyline             1  2020-04-06 16:54:00+00:00

   Remission  age 13-19  age 20-40  age 41-64  age 65-79  age 80-89  …  \
0          0          0          0          1          0          0  …
1          0          0          0          0          1          0  …
2          1          0          0          1          0          0  …
3          0          0          0          1          0          0  …
4          0          0          0          1          0          0  …

   rx_42347  rx_51272  rx_5489  rx_5489_161  rx_6470  rx_6472  rx_704  \
```

```
0       0.0       0.0       0.0           0.0       1.0       0.0       0.0
1       0.0       0.0       0.0           0.0       0.0       0.0       0.0
2       0.0       0.0       0.0           1.0       0.0       0.0       0.0
3       0.0       0.0       0.0           0.0       0.0       0.0       0.0
4       0.0       0.0       0.0           1.0       0.0       0.0       0.0

   rx_7052  rx_7804  rx_8745
0      0.0      0.0      0.0
1      1.0      0.0      1.0
2      0.0      0.0      0.0
3      0.0      0.0      0.0
4      1.0      1.0      0.0

[5 rows x 1651 columns]
```

[84]:
```
analysis_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1984 entries, 0 to 1983
Columns: 1651 entries, person_id to rx_8745
dtypes: float64(34), int64(1615), object(2)
memory usage: 25.0+ MB
```

3 Data Checkpoint

3.1 Store Updated Datasets in a File

[85]:
```
# Store as csv file for manual grouping
filename = './checkpoint/hap823_analysis_subset_' + ad + '_complete.csv'
analysis_df.to_csv(filename, index = False)
```

3.2 Restore needed files from the data checkpoint

[87]:
```
# Import libraries
import pandas as pd
import numpy as np

# Define target antidepressant
# Change for your own antidepressant
ad = 'Amitriptyline'

filename = './checkpoint/hap823_analysis_subset_' + ad + '_complete.csv'

analysis_df = pd.read_csv(filename)

analysis_df.head()
```

[87]:
```
   person_id    ad_grouping  trial_number          earliest_ad_start  \
0    1000039  Amitriptyline             1  2019-12-18 16:13:00+00:00
1    1000370  Amitriptyline             1  2019-11-01 00:00:00+00:00
2    1004308  Amitriptyline             1  2007-08-06 14:27:00+00:00
3    1005542  Amitriptyline             1  2016-11-08 00:00:00+00:00
4    1010384  Amitriptyline             1  2020-04-06 16:54:00+00:00

   Remission  age 13-19  age 20-40  age 41-64  age 65-79  age 80-89  …  \
0          0          0          0          1          0          0  …
1          0          0          0          0          1          0  …
2          1          0          0          1          0          0  …
3          0          0          0          1          0          0  …
4          0          0          0          1          0          0  …

   rx_42347  rx_51272  rx_5489  rx_5489_161  rx_6470  rx_6472  rx_704  \
0       0.0       0.0      0.0          0.0      1.0      0.0     0.0
1       0.0       0.0      0.0          0.0      0.0      0.0     0.0
2       0.0       0.0      0.0          1.0      0.0      0.0     0.0
3       0.0       0.0      0.0          0.0      0.0      0.0     0.0
4       0.0       0.0      0.0          1.0      0.0      0.0     0.0

   rx_7052  rx_7804  rx_8745
0      0.0      0.0      0.0
1      1.0      0.0      1.0
2      0.0      0.0      0.0
3      0.0      0.0      0.0
4      1.0      1.0      0.0

[5 rows x 1651 columns]
```

4 Use the AI Model to Generate Remission Predictions

[88]:
```
coeff_mapping_df = pd.read_csv('./data/Reference_Antidepressant Coef 8 13 23.
↪csv')
coeff_mapping_df.head()
```

[88]:
```
   Feature Present         antidep cgrp ctype code  coef  likelir  \
0              NaN   AMITRIPTYLINE  dxo  ICD9  311  0.15     1.50
1              NaN      CITALOPRAM  dxo  ICD9  311  0.07     1.02
2              NaN      CITALOPRAM  dxi  ICD9  311  0.13     0.93
3              NaN  DESVENLAFAXINE  dxo  ICD9  311  0.09     1.01
4              NaN      DULOXETINE  dxi  ICD9  311  0.05     0.95

        Body System Class of Code  Short Variable Name  C  \
0  mental disorders           311  Depressive Disorder  c
1  mental disorders           311  Depressive Disorder  c
```

```
2  mental disorders           311  Depressive Disorder  c
3  mental disorders           311  Depressive Disorder  c
4  mental disorders           311  Depressive Disorder  c

                                    Calculation Notes  \
0  Set to 1 if any of the major depressions are p…
1  Set to 1 if any of the major depressions are p…
2  Set to 1 if any of the major depressions are p…
3  Set to 1 if any of the major depressions are p…
4  Set to 1 if any of the major depressions are p…

                                      description Data Available (Y/N)  \
0  DEPRESSIVE DISORDER NOT ELSEWHERE CLASSIFIED                     N
1  DEPRESSIVE DISORDER NOT ELSEWHERE CLASSIFIED                     N
2  DEPRESSIVE DISORDER NOT ELSEWHERE CLASSIFIED                     N
3  DEPRESSIVE DISORDER NOT ELSEWHERE CLASSIFIED                     N
4  DEPRESSIVE DISORDER NOT ELSEWHERE CLASSIFIED                     N

  Data Column            Action Needed Code Mapping
0      dx_311  Retrieve diagnosis data          311
1      dx_311  Retrieve diagnosis data          311
2     dxi_311  Retrieve diagnosis data          311
3      dx_311  Retrieve diagnosis data          311
4     dxi_311  Retrieve diagnosis data          311
```

[89]:
```
coeff_mapping_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1573 entries, 0 to 1572
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Feature Present       0 non-null      float64
 1   antidep               1572 non-null   object
 2   cgrp                  1556 non-null   object
 3   ctype                 1556 non-null   object
 4   code                  1556 non-null   object
 5   coef                  1572 non-null   float64
 6   likelir               1556 non-null   float64
 7   Body System           700 non-null    object
 8   Class of Code         1556 non-null   object
 9   Short Variable Name   1572 non-null   object
 10  C                     767 non-null    object
 11  Calculation Notes     278 non-null    object
 12  description           1573 non-null   object
 13  Data Available (Y/N)  1572 non-null   object
 14  Data Column           1572 non-null   object
 15  Action Needed         1468 non-null   object
```

```
 16  Code Mapping          1572 non-null   object
dtypes: float64(3), object(14)
memory usage: 209.0+ KB
```

[90]:
```
# Retain only rows for your antidepressant
selected_ad = 'AMITRIPTYLINE' # remember this is all caps
coeff_mapping_df = coeff_mapping_df[coeff_mapping_df['antidep'] == selected_ad]

coeff_mapping_df.head()
```

[90]:

	Feature Present	antidep	cgrp	ctype	code	coef	likelir	\
0	NaN	AMITRIPTYLINE	dxo	ICD9	311	0.15	1.50	
57	NaN	AMITRIPTYLINE	dxo	ICD9	2722	0.13	1.35	
117	NaN	AMITRIPTYLINE	dxo	ICD9	3572	0.36	1.93	
141	NaN	AMITRIPTYLINE	dxi	ICD9	4280	-0.27	1.19	
268	NaN	AMITRIPTYLINE	dxo	ICD9	25001	0.31	1.76	

	Body System	Class of Code	\
0	mental disorders	311	
57	endocrine nutritional and metabolic diseases a…	272	
117	diseases of the nervous system and sense organs	357	
141	diseases of the circulatory system	428	
268	endocrine nutritional and metabolic diseases a…	250	

	Short Variable Name	C	\
0	Depressive Disorder	c	
57	Mixed Hyperlipidemia	c	
117	Polyneuropathy In Diabetes	c	
141	Congestive Heart Failure	c	
268	Type I Diabetes Mellitus without Complication	c	

	Calculation Notes	\
0	Set to 1 if any of the major depressions are p…	
57	NaN	
117	NaN	
141	NaN	
268	Set to zero if diabetes with complications are…	

	description	Data Available (Y/N)	\
0	DEPRESSIVE DISORDER NOT ELSEWHERE CLASSIFIED	N	
57	MIXED HYPERLIPIDEMIA	N	
117	POLYNEUROPATHY IN DIABETES	N	
141	CONGESTIVE HEART FAILURE UNSPECIFIED	N	
268	DIABETES MELLITUS WITHOUT MENTION OF COMPLICAT…	N	

	Data Column	Action Needed	Code Mapping
0	dx_311	Retrieve diagnosis data	311

```
57     dx_272.2  Retrieve diagnosis data        272.2
117    dx_357.2  Retrieve diagnosis data        357.2
141   dxi_428.0  Retrieve diagnosis data        428.0
268   dx_250.01  Retrieve diagnosis data       250.01
```

[91]:
```
# Create a dictionary with the keys equal to the Data Column column and values␣
 ↪equal to coef column
coeff_dict = dict(zip(coeff_mapping_df['Data Column'], map(float,␣
 ↪coeff_mapping_df['coef'])))
```

[92]:
```
# Tests for dict
# Change to coefficients that are available in your mapping file
print(f"Intercept: {coeff_dict['intercept']}")
print(f"dx_311: {coeff_dict['dx_311']}")
print(f"dxi_428.0: {coeff_dict['dxi_428.0']}")

# check if calculations work (changed to floats)
print(f"sum: {coeff_dict['intercept'] + coeff_dict['dx_311'] +␣
 ↪coeff_dict['dxi_428.0']}")
```

```
Intercept: -4.09
dx_311: 0.15
dxi_428.0: -0.27
sum: -4.21
```

[93]:
```
import math # for the exp function

# Create a function to calculate prediction of remission based on the data and␣
 ↪the mapping file
# Returns 1 if calculated probability is equal to the cutoff or higher,␣
 ↪otherwise 0
def predict_remission(row, coeff_dict, col_names):

    # intialize sum of coefficients
    sumcoeff = 0

    # loop through each key in the coefficient dictionary
    for key in coeff_dict:
        # if the intercept the add value to the sum of coefficents
        if key == 'intercept':
            sumcoeff += coeff_dict[key]
        else: # if variable other than intercept
            # check if column exists
            if key in col_names:
                # add the coefficient if column is present (equal to 1)
                sumcoeff += row[key] * coeff_dict[key]
```

```
    # calculate probability of remission from sum of coefficients
    p_remission = 1 / (1 + math.exp(-sumcoeff))

    return p_remission
```

[95]:
```
# create list of column names
col_names = analysis_df.columns.tolist()

# Create prediction column containing probability of remission
analysis_df['Prediction'] = analysis_df.apply(predict_remission, axis=1,␣
 ↪args=(coeff_dict, col_names,))
```

5 Run Regressions on Remission

5.1 Regress on AI Predictors and Prediction

[96]:
```
# Get list of AI columns
AI_cols = coeff_mapping_df['Data Column'].tolist()

# Get list of available columns in dataset
column_names = analysis_df.columns.tolist()

# Get AI cols that are in the dataset
filtered_AI_cols = [col for col in AI_cols if col in column_names]

# Minimize warning
import warnings
from sklearn.exceptions import ConvergenceWarning

# Suppress ConvergenceWarning
warnings.filterwarnings("ignore", category=ConvergenceWarning)

# Import libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.metrics import log_loss

# Set up variables
X = analysis_df[filtered_AI_cols]
y = analysis_df['Remission']

# Initializing and fitting the logistic regression model
logistic_reg_model = LogisticRegression()
logistic_reg_model.fit(X, y)

# Predicting on the test set
```

```
y_pred = logistic_reg_model.predict(X)

# Evaluating the model
accuracy = accuracy_score(y, y_pred)
print("Accuracy:", accuracy)

# Classification report
print(classification_report(y, y_pred, zero_division=0))

# Calculate log-likelihood of the model
log_likelihood_model = -log_loss(y, y_pred)

# Calculate log-likelihood of the null model
null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

# Calculate McFadden's R-squared
mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print("\n\nMcFadden's R-squared:", mcfadden_r2)

# Displaying regression results
print("Intercept:", logistic_reg_model.intercept_[0])

selected_features = [(feature, coef) for feature, coef in zip(X.columns,␣
 ↪logistic_reg_model.coef_[0]) if coef != 0]

# Print selected feature names and coefficients
print("\nSelected features with coefficients:")
for feature, coef in selected_features:
    print(f"{feature}: {coef}")
```

```
Accuracy: 0.6270161290322581
              precision    recall  f1-score   support

           0       0.64      0.91      0.75      1204
           1       0.58      0.20      0.29       780

    accuracy                           0.63      1984
   macro avg       0.61      0.55      0.52      1984
weighted avg       0.61      0.63      0.57      1984



McFadden's R-squared: 0.0512820512820511
Intercept: -0.5842137688509731

Selected features with coefficients:
```

```
dx_311: -0.12738352338826073
dx_272.2: 0.0891817546004357
dx_357.2: 0.15869538635938613
dxi_428.0: -0.2540711582293475
dx_250.01: -0.07489379812191266
dx_296.30: -0.2595093885543329
dx_296.33: -0.34635765442046706
dx_300.02: 0.024566319547538018
dx_728.87: 0.04803202528397055
dx_780.09: -0.4604369272306751
px_90805: 0.00479724941545526
px_90807: 0.2551692707773834
px_90862: -0.11550412749059748
SN: -0.6108336241899763
DR: -0.05649360224474827
DN: -0.03882767683525345
age 13-19: -0.044121614456370584
age 20-40: -0.320727696094438
age 80-89: -0.3811261492943654
rx_704: 0.4972128847133843
rx_17767: 0.0789157633854937
rx_1292: 0.1266886681996535
rx_42347: -0.19745158658946452
rx_2101: 0.020971282641637497
rx_2598: -0.06595754464060213
rx_3638: 0.19787957279178667
rx_5489_161: 0.16923251193691002
rx_6470: -0.17384360632285772
rx_6472: -0.0143547520211484
rx_7052: 0.005214214895233391
rx_7804: -0.12526860177226717
rx_8745: -0.08388030274838175
rx_51272: 0.013182366455708767
rx_10737: 0.25921151781633767
Female: 0.15110345161710326
epi_2p: -0.09595062363801449
rem_2p: 0.4722962001549113
dx_V72.31: 0.1502633403642134
dx_V76.12: -0.7602037254388627
```

5.2 Regress on AI Predictors, Prediction, and All Other Variables

[98]:
```
# Drop columns that we do not need
cols_to_drop = ['person_id',
                'ad_grouping',
                'trial_number',
                'earliest_ad_start']
```

```
analysis_df = analysis_df.drop(columns=cols_to_drop)
```

[99]:
```
# Calculate the correlation matrix
correlation_matrix = analysis_df.corr()

# Find columns with correlation coefficient of 1 with any other column
multicollinear_columns = set()
for i in range(len(correlation_matrix.columns)):
    for j in range(i+1, len(correlation_matrix.columns)):  # Adjusted range to only process each pair once
        if correlation_matrix.iloc[i, j] == 1.0:
            column_name_i = correlation_matrix.columns[i]
            column_name_j = correlation_matrix.columns[j]

            # Check if either column starts with "disease_" drop them because we want to retain "dx" columns
            if column_name_i.startswith("disease_") and column_name_j.startswith("disease_"):
                multicollinear_columns.add(column_name_i)
            elif column_name_i.startswith("disease_"):
                multicollinear_columns.add(column_name_i)
            elif column_name_j.startswith("disease_"):
                multicollinear_columns.add(column_name_j)
            else:
                multicollinear_columns.add(column_name_i)

# Drop multicollinear columns
regression_df = analysis_df.drop(columns=multicollinear_columns).copy()

regression_df.head()
```

[99]:

	Remission	age 13-19	age 20-40	age 41-64	age 65-79	age 80-89	Female	\
0	0	0	0	1	0	0	1	
1	0	0	0	0	1	0	1	
2	1	0	0	1	0	0	1	
3	0	0	0	1	0	0	0	
4	0	0	0	1	0	0	0	

	disease_450005	disease_563001	disease_928000	…	rx_51272	rx_5489	\
0	0	0	0	…	0.0	0.0	
1	0	0	1	…	0.0	0.0	
2	0	0	0	…	0.0	0.0	
3	0	0	0	…	0.0	0.0	
4	0	0	0	…	0.0	0.0	

	rx_5489_161	rx_6470	rx_6472	rx_704	rx_7052	rx_7804	rx_8745	\
0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	

```
1          0.0      0.0      0.0      0.0      1.0      0.0      1.0
2          1.0      0.0      0.0      0.0      0.0      0.0      0.0
3          0.0      0.0      0.0      0.0      0.0      0.0      0.0
4          1.0      0.0      0.0      0.0      1.0      1.0      0.0

   Prediction
0    0.012370
1    0.044362
2    0.015217
3    0.023431
4    0.024127

[5 rows x 1639 columns]
```

5.2.1 No Cross Validation (10 to 12 Predictors)

[100]:
```
# Minimize warning
import warnings
from sklearn.exceptions import ConvergenceWarning

# Suppress ConvergenceWarning
warnings.filterwarnings("ignore", category=ConvergenceWarning)

# Import libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
from sklearn.metrics import accuracy_score, classification_report
from sklearn.metrics import log_loss

# Set up variables
X = regression_df.drop(columns=['Remission'])
y = regression_df['Remission']

# Create Lasso model
alpha = 0.008 # played with alpha until features are around 10 to 12
lasso = Lasso(alpha = alpha)

# Fit the model
lasso.fit(X, y)

# Get selected features with non zero coefficients
selected_features_names = [feature for feature, coef in zip(X.columns, lasso.
 ↪coef_) if coef != 0]

# Filter X values to just selected features
X = X[selected_features_names]
```

```python
# Fit a logistic regression model
logistic_reg_model = LogisticRegression()
logistic_reg_model.fit(X, y)

# Predict on the test set
y_pred = logistic_reg_model.predict(X)

# Compute classification metrics
accuracy = accuracy_score(y, y_pred)
print("Accuracy:", accuracy)
print(classification_report(y, y_pred, zero_division='warn'))

# Print the selected regularization parameter (alpha)
print("\nSelected regularization parameter (alpha):", alpha)

# Calculate log-likelihood of the model
log_likelihood_model = -log_loss(y, y_pred)

# Calculate log-likelihood of the null model
null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

# Calculate McFadden's R-squared
mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print("\nMcFadden's R-squared:", mcfadden_r2)

# Displaying regression results
print("Intercept:", logistic_reg_model.intercept_[0])

selected_features = [(feature, coef) for feature, coef in zip(X.columns,␣
 ↪logistic_reg_model.coef_[0]) if coef != 0]

# Print selected feature names and coefficients
print("\nSelected features with coefficients:")
for feature, coef in selected_features:
    print(f"{feature}: {coef}")
```

```
Accuracy: 0.6360887096774194
              precision    recall  f1-score   support

           0       0.64      0.90      0.75      1204
           1       0.60      0.23      0.33       780

    accuracy                           0.64      1984
   macro avg       0.62      0.56      0.54      1984
weighted avg       0.62      0.64      0.59      1984
```

```
Selected regularization parameter (alpha): 0.008

McFadden's R-squared: 0.0743589743589742
Intercept: -0.7520860190430002

Selected features with coefficients:
age 41-64: 0.22796958403009232
disease_1201005: 0.22899434280808606
disease_44465007: -0.3926955310970102
disease_70153002: -0.4577379934557015
disease_87522002: 0.29111833324615916
disease_197480006: -0.37949823244825037
disease_373621006: 0.32807812559459376
disease_698247007: -0.31756158732906997
SN: -0.2553174585070953
SR: 0.5338865172848863
rem_2p: 0.3403021035115417
rx_5489_161: 0.20442111430318197
```

Notes on model, intercept and selected features: - McFadden R2: 0.07 - Lasso Alpha (no cross validation): 0.008 - Intercept: -0.75 - Selected features with coefficients: - Age range 41 to 64 - age 41-64: 0.23 - Benign essential hypertension - disease_1201005: 0.23 - Sprain of ankle - disease_44465007: -0.39 - Hemorrhoids - disease_70153002: -0.46 - Iron deficiency anemia - disease_87522002: 0.29 - Anxiety disorder - disease_197480006: -0.38 - Chronic pain syndrome - disease_373621006: 0.33 - Cardiac arrhythmia - disease_698247007: -0.32 - Same Antidepressant No Remission - SN: -0.26 - Same Antidepressant With Remission - SR: 0.54 - Previous Remission 2 or more - rem_2p: 0.34 - Hydrocodone/Acetaminophen - rx_5489_161: 0.20

5.2.2 With Cross Validation

[102]:
```
# Minimize warning
import warnings
from sklearn.exceptions import ConvergenceWarning

# Suppress ConvergenceWarning
warnings.filterwarnings("ignore", category=ConvergenceWarning)

# Import libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LassoCV
from sklearn.metrics import accuracy_score, classification_report
from sklearn.metrics import log_loss

# Set up variables
X = regression_df.drop(columns=['Remission'])
y = regression_df['Remission']
```

```
# Create LassoCV model
lassocv = LassoCV(cv = 5)

# Fit the model
lassocv.fit(X, y)

# Get selected features with non zero coefficients
selected_features_names = [feature for feature, coef in zip(X.columns, lassocv.
 ↪coef_) if coef != 0]

# Filter X values to just selected features
X = X[selected_features_names]

# Fit a logistic regression model
logistic_reg_model = LogisticRegression()
logistic_reg_model.fit(X, y)

# Predict on the test set
y_pred = logistic_reg_model.predict(X)

# Compute classification metrics
accuracy = accuracy_score(y, y_pred)
print("Accuracy:", accuracy)
print(classification_report(y, y_pred, zero_division='warn'))

# Print the selected regularization parameter (alpha)
print("\nSelected regularization parameter (alpha):", lassocv.alpha_)

# Calculate log-likelihood of the model
log_likelihood_model = -log_loss(y, y_pred)

# Calculate log-likelihood of the null model
null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

# Calculate McFadden's R-squared
mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print("\nMcFadden's R-squared:", mcfadden_r2)

# Displaying regression results
print("Intercept:", logistic_reg_model.intercept_[0])

selected_features = [(feature, coef) for feature, coef in zip(X.columns,␣
 ↪logistic_reg_model.coef_[0]) if coef != 0]

# Print selected feature names and coefficients
```

```
print("\nSelected features with coefficients:")
for feature, coef in selected_features:
    print(f"{feature}: {coef}")
```

```
Accuracy: 0.6340725806451613
              precision    recall  f1-score   support

           0       0.65      0.88      0.75      1204
           1       0.58      0.25      0.35       780

    accuracy                           0.63      1984
   macro avg       0.61      0.57      0.55      1984
weighted avg       0.62      0.63      0.59      1984


Selected regularization parameter (alpha): 0.0076971394325567365

McFadden's R-squared: 0.0692307692307691
Intercept: -0.7857225227584292

Selected features with coefficients:
age 41-64: 0.19966577946933675
disease_1201005: 0.23772089138702096
disease_1755008: -0.7010654236304856
disease_44465007: -0.4606937483208233
disease_70153002: -0.48199328547061504
disease_87522002: 0.24296903440170609
disease_95315005: 0.25342627879688184
disease_197480006: -0.37294135727720873
disease_373621006: 0.32389179814595215
disease_399269003: 0.2592360944324592
disease_698247007: -0.3090405415464164
SN: -0.239401831580584
SR: 0.5027093788621657
rem_2p: 0.28454609940708514
rx_10737: 0.12915417343159455
rx_5489_161: 0.20892178128947614
```

Notes on intercept and selected features: - McFadden R2: 0.07 - Intercept: -0.79 - Lasso Alpha (cross validated): 0.0076971394325567365 - Selected features with coefficients: - Age range 41 to 64 - age 41-64: 0.20 - Benign essential hypertension - disease_1201005: 0.24 - Old myocardial infarction - disease_1755008: -0.70 - Sprain of ankle - disease_44465007: -0.46 - Hemorrhoids - disease_70153002: -0.48 - Iron deficiency anemia - disease_87522002: 0.24 - Uterine leiomyoma - disease_95315005: 0.25 - Anxiety disorder - disease_197480006: -0.37 - Chronic pain syndrome - disease_373621006: 0.32 - Arthropathy - disease_399269003: 0.26 - Cardiac arrhythmia - disease_698247007: -0.31 - Same Antidepressant No Remission - SN: -0.24 - Same Antidepressant With Remission - SR: 0.50 - Previous Remission 2 or more - rem_2p: 0.28 - Trazodone - rx_10737: 0.13 - Hydrocodone/Acetaminophen - rx_5489_161: 0.21

6 Population Characteristics

[ ]: