Capstone Project Session 1

Derrick Mills

Student

Michigan Engineering Professional Education AIML – Cohort 3

Aura: Intelligent Prediction and Recommendation Engine for Digital Health Analytics

Introduction

The development of Aura, an intelligent prediction and recommendation engine for healthcare-focused digital marketing, requires a robust and reliable data pipeline capable of handling complex, heterogeneous datasets. The NSMES1988 dataset serves as the initial input for establishing the fundamental data ingestion, preprocessing, and optimization procedures that will support Aura's analytical, visualization, and machine-learning components.

The present report details the activities completed during Session 1, including data importation, structural evaluation, preliminary preprocessing, type refinement, memory footprint analysis, and exportation of optimized data artifacts. These steps establish a clean, validated, and computationally efficient foundation for subsequent modeling and system integration.

Methods:

Data Importation Environment

All tasks were conducted within Google Colab, using Python's scientific computing stack (pandas, NumPy). The dataset was imported from the file path /NSMES1988.csv. Colab provides a cloud-hosted execution environment that supports reproducible, scalable computational workflows suitable for healthcare analytics.

Initial Structural Inspection

Upon loading, the dataset was examined to characterize its dimensionality, variable composition, data types, and overall integrity. The inspection process involved:

- Evaluating dataset shape
- Reviewing variable data types
- Displaying summary statistics for numerical variables
- Assessing missingness patterns
- Identifying anomalous or non-standard values
- Characterizing categorical vs. continuous variables

Data Cleanliness Assessment

A complete analysis was performed using df.isna().sum(). Logical consistency checks were conducted to identify possible encoding abnormalities or out-of-range values. Special focus was given to socioeconomic variables such as income and clinical utilization variables that influence predictive modeling.

Type Optimization and Memory Reduction

Given Aura's requirement to scale to larger datasets in future phases, memory efficiency was prioritized. The following transformations were applied:

1. **Removal of redundant variables**
   - Unnamed: 0, an auto-generated index column, was removed.
2. **Categorical encoding**
   Variables representing discrete states (e.g., health, adl, region, gender, married, employment, insurance, medicaid) were cast to pandas category types.
3. **Numeric downcasting**
   - Integer columns were downcast from int64 to smaller integer types (int16 or int8), where appropriate.
   - Floating-point columns (age, income) were downcast from float64 to float32.

Memory usage was measured before and after optimization using df.info(memory_usage="deep").

Data Exportation

To support downstream processing pipelines, two export formats were generated:

- The original dataset was exported as JSON (NSMES1988.json), enabling API-friendly ingestion.
- The cleaned, optimized dataset was exported as CSV (NSMES1988new.csv) for analytical and modeling tasks.

Results:

Dataset Structure and Composition

The dataset contains 4,406 observations and 19 variables, encompassing:

- **Healthcare utilization** - visits, non-physician visits, physician visits, emergency services, hospital admissions
- **Health status indicators** - chronic conditions, functional limitations (ADL), self-reported health

- **Demographic factors** - age (expressed in decades), gender, marital status
- **Socioeconomic indicators** - education, income (in units of $10,000), employment status
- **Insurance variables** - private insurance, Medicaid coverage
- **Geographic region** - urban, rural, or regional classifications

The dataset exhibits no missing values, indicating complete records across all variables.

Observed Anomalies and Considerations

Two key observations emerged from the structural assessment:

1. **Age Encoding**
   The age variable ranges from approximately 6.6 to 10.9, indicating that age is measured in decades rather than years. This transformation will require reconversion or careful interpretative handling in later modeling stages.
2. **Negative Income Values**
   Income ranges from 1.0 to 54.8 (i.e., –$10,000 to $548,000). Negative values may reflect special coding, reporting errors, or debt and should be evaluated through business rules in subsequent preprocessing sessions.

Memory Utilization Before and After Optimization

- Original dataset memory footprint **-** 2.4 MB
- Optimized dataset memory footprint **-** 123 KB

This represents a 95% reduction in memory usage, achieved through categorical encoding and numeric downcasting. Although this dataset is relatively small, the optimization approach establishes a scalable pattern suitable for large-volume healthcare datasets expected in later stages of Aura's development.

Exported Artifacts

Two export files were successfully generated:

- NSMES1988.json - original dataset
- NSMES1988new.csv - cleaned and optimized dataset

These files are now available within the Colab working directory for downstream analysis.

Discussion

The preprocessing performed in Session 1 establishes a rigorous and efficient foundation for integrating the NSMES1988 dataset into Aura's analytical framework. The absence of missing data simplifies early modeling tasks; the presence of encoded or transformed values (e.g., age in decades, negative income values) warrants domain-specific treatment in later sessions.

The dramatic memory footprint reduction demonstrates the importance of dtype optimization in scalable analytics architectures. As Aura evolves to accommodate larger, multi-source, and potentially real-time healthcare datasets, such optimizations will play a pivotal role in supporting low-latency processing, cost-efficient cloud workloads, and smooth visualization rendering.

Furthermore, the structured export of both original and cleaned datasets ensures compatibility with diverse components of the Aura pipeline, including RESTful ingestion endpoints, visualization engines, and machine-learning model trainers.

Conclusion

Session 1 accomplished the foundational tasks of data importation, structural assessment, data type optimization, memory analysis, and exportation of both original and cleaned datasets. The dataset is now prepared for more advanced phases of processing, including exploratory data analysis (Session 2), feature engineering, and predictive model development within the Aura engine.

The dataset's completeness, coupled with the systematic preprocessing conducted, provides a strong basis for subsequent analytical and modeling activities central to Aura's mission of delivering intelligent, data-driven insights for healthcare marketers.

Recommendations for Future Work

To prepare for predictive modeling and visualization in subsequent sessions, the following steps are recommended:

1. **Address anomalous income values** through imputation, capping, or transformation based on domain policy.
2. **Normalize key continuous variables**, particularly income and age (if converted back to years).
3. **Create a correlation matrix and univariate distributions** to identify predictive signal strength.
4. **Explore visit utilization clusters** (e.g., high-utilizers vs. low-utilizers) as potential segmentation insights.
5. **Establish metadata documentation** for variable semantics and business rules to support auditability and reproducibility.

aura_session1.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

[2]

```python
import numpy as np
import pandas as pd

# === CONFIG ===
CSV_PATH = "/NSMES1988.csv"  # update if your file is elsewhere, e.g. "NSMES1988.csv"


def load_data(csv_path: str) -> pd.DataFrame:
    """Load the NSMES1988 dataset from CSV."""
    df = pd.read_csv(csv_path)
    return df


def basic_inspection(df: pd.DataFrame) -> None:
    """Print basic info and cleanliness checks."""
    print("\n=== BASIC INSPECTION ===")
    print(f"Shape (rows, columns): {df.shape}\n")

    print("Column dtypes:")
    print(df.dtypes, "\n")

    print("First 5 rows:")
    print(df.head(), "\n")

    print("Summary statistics (numeric):")
    print(df.describe(), "\n")

    print("Missing values per column:")
    print(df.isna().sum(), "\n")

    if "age" in df.columns:
        print("Age stats:")
        print(df["age"].describe(), "\n")

    if "income" in df.columns:
        print("Income stats:")
        print(df["income"].describe(), "\n")


def memory_report(df: pd.DataFrame, title: str) -> None:
    """Print DataFrame.info with deep memory usage."""
    print(f"\n=== MEMORY REPORT: {title} ===")
    df.info(memory_usage="deep")


def optimize_dtypes(df: pd.DataFrame) -> pd.DataFrame:
    """
    Optimize dtypes:
      - drop artificial index column if present
      - convert object columns to category
      - downcast integer and float columns
    """
    df_opt = df.copy()

    # Drop the 'Unnamed: 0' index column if present
    if "Unnamed: 0" in df_opt.columns:
```

How can I install Python libraries? | Load data from Google Drive | Show an example of training

What can I help you build?

Variables Terminal 11:46 AM Python 3

aura_session1.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

[2]

```python
    """
    df_opt = df.copy()

    # Drop the 'Unnamed: 0' index column if present
    if "Unnamed: 0" in df_opt.columns:
        df_opt = df_opt.drop(columns=["Unnamed: 0"])

    # Categorical columns from the data dictionary
    cat_cols = [
        "health",
        "adl",
        "region",
        "gender",
        "married",
        "employed",
        "insurance",
        "medicaid",
    ]

    for col in cat_cols:
        if col in df_opt.columns:
            df_opt[col] = df_opt[col].astype("category")

    # Downcast integer columns
    int_cols = df_opt.select_dtypes(include=["int64"]).columns
    df_opt[int_cols] = df_opt[int_cols].apply(
        pd.to_numeric, downcast="integer"
    )

    # Downcast float columns
    float_cols = df_opt.select_dtypes(include=["float64"]).columns
    df_opt[float_cols] = df_opt[float_cols].apply(
        pd.to_numeric, downcast="float"
    )

    return df_opt


def export_files(df_original: pd.DataFrame, df_clean: pd.DataFrame) -> None:
    """
    Export:
      - original data to JSON (NSMES1988.json)
      - cleaned / optimized data to CSV (NSMES1988new.csv)
    """
    print("\n=== EXPORTING FILES ===")

    # JSON export of original dataframe
    json_path = "NSMES1988.json"
    df_original.to_json(json_path, orient="records")
    print(f"Exported original DataFrame to JSON: {json_path}")

    # CSV export of cleaned / optimized dataframe
    csv_new_path = "NSMES1988new.csv"
    df_clean.to_csv(csv_new_path, index=False)
    print(f"Exported cleaned DataFrame to CSV: {csv_new_path}")


# === MAIN EXECUTION IN COLAB ===
```

How can I install Python libraries? | Load data from Google Drive | Show an example of training

What can I help you build?

Variables Terminal 11:46 AM Python 3

aura_session1.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```python
            df_opt[col] = df_opt[col].astype("category")

    # Downcast integer columns
    int_cols = df_opt.select_dtypes(include=["int64"]).columns
    df_opt[int_cols] = df_opt[int_cols].apply(
        pd.to_numeric, downcast="integer"
    )

    # Downcast float columns
    float_cols = df_opt.select_dtypes(include=["float64"]).columns
    df_opt[float_cols] = df_opt[float_cols].apply(
        pd.to_numeric, downcast="float"
    )

    return df_opt


def export_files(df_original: pd.DataFrame, df_clean: pd.DataFrame) -> None:
    """
    Export:
      - original data to JSON (NSMES1988.json)
      - cleaned / optimized data to CSV (NSMES1988new.csv)
    """
    print("\n=== EXPORTING FILES ===")

    # JSON export of original dataframe
    json_path = "NSMES1988.json"
    df_original.to_json(json_path, orient="records")
    print(f"Exported original DataFrame to JSON: {json_path}")

    # CSV export of cleaned / optimized dataframe
    csv_new_path = "NSMES1988new.csv"
    df_clean.to_csv(csv_new_path, index=False)
    print(f"Exported cleaned DataFrame to CSV: {csv_new_path}")


# === MAIN EXECUTION IN COLAB ===
df = load_data(CSV_PATH)

# 1. Basic inspection & cleanliness check
basic_inspection(df)

# 2. Memory usage before optimization
memory_report(df, title="Original DataFrame")

# 3. Optimize dtypes and clean structure
df_opt = optimize_dtypes(df)

# 4. Memory usage after optimization
memory_report(df_opt, title="Optimized DataFrame")

# 5. Export JSON and new CSV back to Colab workspace
export_files(df, df_opt)

print("\nDone. Files now in the current working directory:")
!ls -lh
```

How can I install Python libraries? | Load data from Google Drive | Show an example of training

What can I help you build?

Variables Terminal

11:46 AM Python 3

aura_session1.ipynb

File Edit View Insert Runtime Tools Help

mmands + Code + Text Run all

```
--- BASIC INSPECTION ---
Shape (rows, columns): (4406, 19)

Column dtypes:
Unnamed: 0      int64
visits          int64
nvisits         int64
ovisits         int64
novisits        int64
emergency       int64
hospital        int64
health         object
chronic         int64
adl            object
region         object
age           float64
gender         object
married        object
school          int64
income        float64
employed       object
insurance      object
medicaid       object
dtype: object

First 5 rows:
   Unnamed: 0  visits  nvisits  ovisits  novisits  emergency  hospital  \
0           1       5        0        0         0          0         1
1           2       1        0        2         0          2         0
2           3      13        0        0         0          3         3
3           4      16        0        5         0          1         1
4           5       3        0        0         0          0         0

    health  chronic      adl region  age  gender married  school  income  \
0  average        2   normal  other  6.9    male     yes       6  2.8810
1  average        2   normal  other  7.4  female     yes      10  2.7478
2     poor        4  limited  other  6.6  female      no      10  0.6532
3     poor        2  limited  other  7.6    male     yes       3  0.6588
4  average        2  limited  other  7.9  female     yes       6  0.6588

  employed insurance medicaid
0      yes       yes       no
1       no       yes       no
2       no        no      yes
3       no       yes       no
4       no       yes       no

Summary statistics (numeric):
        Unnamed: 0       visits      nvisits      ovisits     novisits  \
count  4406.000000  4406.000000  4406.000000  4406.000000  4406.000000
mean   2203.500000     5.774399     1.618021     0.750794     0.536087
std    1272.046972     6.759225     5.317056     3.652759     3.879506
min       1.000000     0.000000     0.000000     0.000000     0.000000
25%    1102.250000     1.000000     0.000000     0.000000     0.000000
50%    2203.500000     4.000000     0.000000     0.000000     0.000000
75%    3304.750000     8.000000     1.000000     0.000000     0.000000
max    4406.000000    89.000000   104.000000   141.000000   155.000000
```

How can I install Python libraries? | Load data from Google Drive | Show an example of training

aura_session1.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text ▷ Run all

```
          emergency     hospital      chronic          age       school  \
count  4406.000000  4406.000000  4406.000000  4406.000000  4406.000000
mean      0.263504     0.295960     1.541988     7.402406    10.290286
std       0.703659     0.746398     1.349632     0.633405     3.738736
min       0.000000     0.000000     0.000000     6.600000     0.000000
25%       0.000000     0.000000     1.000000     6.900000     8.000000
50%       0.000000     0.000000     1.000000     7.300000    11.000000
75%       0.000000     0.000000     2.000000     7.800000    12.000000
max      12.000000     8.000000     8.000000    10.900000    18.000000

            income
count  4406.000000
mean      2.527132
std       2.924648
min      -1.012500
25%       0.912150
50%       1.698150
75%       3.172850
max      54.835100

Missing values per column:
Unnamed: 0    0
visits        0
nvisits       0
ovisits       0
novisits      0
emergency     0
hospital      0
health        0
chronic       0
adl           0
region        0
age           0
gender        0
married       0
school        0
income        0
employed      0
insurance     0
medicaid      0
dtype: int64

Age stats:
count    4406.000000
mean        7.402406
std         0.633405
min         6.600000
25%         6.900000
50%         7.300000
75%         7.800000
max        10.900000
Name: age, dtype: float64

Income stats:
count    4406.000000
mean        2.527132
std         2.924648
min        -1.012500
```

How can I install Python libraries? | Load data from Google Drive | Show an example of training

aura_session1.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text ▷ Run all

```
Income stats:
count    4406.000000
mean        2.527132
std         2.924648
min        -1.012500
25%         0.912150
50%         1.698150
75%         3.172850
max        54.835100
Name: income, dtype: float64


=== MEMORY REPORT: Original DataFrame ===
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4406 entries, 0 to 4405
Data columns (total 19 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  4406 non-null   int64
 1   visits      4406 non-null   int64
 2   nvisits     4406 non-null   int64
 3   ovisits     4406 non-null   int64
 4   novisits    4406 non-null   int64
 5   emergency   4406 non-null   int64
 6   hospital    4406 non-null   int64
 7   health      4406 non-null   object
 8   chronic     4406 non-null   int64
 9   adl         4406 non-null   object
 10  region      4406 non-null   object
 11  age         4406 non-null   float64
 12  gender      4406 non-null   object
 13  married     4406 non-null   object
 14  school      4406 non-null   int64
 15  income      4406 non-null   float64
 16  employed    4406 non-null   object
 17  insurance   4406 non-null   object
 18  medicaid    4406 non-null   object
dtypes: float64(2), int64(9), object(8)
memory usage: 2.2 MB

=== MEMORY REPORT: Optimized DataFrame ===
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4406 entries, 0 to 4405
Data columns (total 18 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   visits     4406 non-null   int8
 1   nvisits    4406 non-null   int8
 2   ovisits    4406 non-null   int16
 3   novisits   4406 non-null   int16
 4   emergency  4406 non-null   int8
 5   hospital   4406 non-null   int8
 6   health     4406 non-null   category
 7   chronic    4406 non-null   int8
 8   adl        4406 non-null   category
 9   region     4406 non-null   category
 10  age        4406 non-null   float32
 11  gender     4406 non-null   category
```

How can I install Python libraries? | Load data from Google Drive | Show an example of training

aura_session1.ipynb

File Edit View Insert Runtime Tools Help

mmands + Code + Text Run all

```
 1   visits      4406 non-null   int64
 2   nvisits     4406 non-null   int64
 3   ovisits     4406 non-null   int64
 4   novisits    4406 non-null   int64
 5   emergency   4406 non-null   int64
 6   hospital    4406 non-null   int64
 7   health      4406 non-null   object
 8   chronic     4406 non-null   int64
 9   adl         4406 non-null   object
 10  region      4406 non-null   object
 11  age         4406 non-null   float64
 12  gender      4406 non-null   object
 13  married     4406 non-null   object
 14  school      4406 non-null   int64
 15  income      4406 non-null   float64
 16  employed    4406 non-null   object
 17  insurance   4406 non-null   object
 18  medicaid    4406 non-null   object
dtypes: float64(2), int64(9), object(8)
memory usage: 2.2 MB

=== MEMORY REPORT: Optimized DataFrame ===
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4406 entries, 0 to 4405
Data columns (total 18 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   visits     4406 non-null   int8
 1   nvisits    4406 non-null   int8
 2   ovisits    4406 non-null   int16
 3   novisits   4406 non-null   int16
 4   emergency  4406 non-null   int8
 5   hospital   4406 non-null   int8
 6   health     4406 non-null   category
 7   chronic    4406 non-null   int8
 8   adl        4406 non-null   category
 9   region     4406 non-null   category
 10  age        4406 non-null   float32
 11  gender     4406 non-null   category
 12  married    4406 non-null   category
 13  school     4406 non-null   int8
 14  income     4406 non-null   float32
 15  employed   4406 non-null   category
 16  insurance  4406 non-null   category
 17  medicaid   4406 non-null   category
dtypes: category(8), float32(2), int16(2), int8(6)
memory usage: 113.9 KB

=== EXPORTING FILES ===
Exported original DataFrame to JSON: NSMES1988.json
Exported cleaned DataFrame to CSV: NSMES1988new.csv

Done. Files now in the current working directory:
total 1.5M
drwx------ 5 root root 4.0K Nov 29 16:41 drive
-rw-r--r-- 1 root root 1.2M Nov 29 16:46 NSMES1988.json
-rw-r--r-- 1 root root 299K Nov 29 16:46 NSMES1988new.csv
drwxr-xr-x 1 root root 4.0K Nov 20 14:30 sample_data
```

How can I install Python libraries? Load data from Google Drive Show an example of training