DATA ANALYTICS REPORT

Rancel Hernandez

D214 - DATA ANALYTICS GRADUATE CAPSTONE

March 20, 2025

Introduction

Healthcare systems are constantly under pressure from high congestion and could benefit from optimized resource allocation while maintaining quality patient care. Hospital admission duration, defined as the time between a patient's admission and discharge, is an influential factor that impacts resource allocation, operational efficiency, and patient outcomes. Accurately predicting admission duration can improve bed management, staff scheduling, and discharge planning, leading to better patient care.

This report presents an analysis of hospital admission duration prediction using random forest regression modeling on a large critical care database. By identifying patterns between patient characteristics, clinical measurements, and admission information, the goal is to develop a precise model to support healthcare resource management. In addition to improving operational efficiency, better estimates of expected hospital stays could allow earlier room preparation for incoming patients and provide insights to help lower healthcare costs related to prolonged stays. The following sections cover the research question, data collection methodology, preparation process, analysis techniques, and implications of the findings.

Research Question

The research question for the analysis is: “Can a random forest regression model accurately predict patient admission duration based on demographic data, clinical measurements, and admission information from a publicly available critical care database?” This question benefits from a data analysis approach because accurately predicting patient admission duration

can help alleviate hospital strain and improve resource management. Some hospitals receive hundreds of patients requiring care in the emergency room within a 24-hour period, and this strain can intensify during holidays when accidents are more likely to occur [1]. This surge in patient flow can overwhelm doctors and nurses, making the need for a model that reliably estimates a patient's stay urgent in order to reduce their workload.

Rooms can be prepared earlier for incoming patients, staff can be assigned more efficiently, and patients can receive a clearer estimate of their stay. Additionally, identifying the factors that influence admission duration could offer insights into how to allocate resources more effectively and reduce the length of hospital stays, easing strain on the healthcare system and improving patient outcomes.

This question will be addressed using a public critical care dataset containing de-identified patient information. The analysis will focus specifically on admissions from the emergency department because these cases often arise abruptly through medical emergencies and contribute significantly to healthcare strain. However, predicting the duration of a patient's stay is a complex task influenced by many factors, including demographics, vital signs, current patient flow, and other contextual variables. While many of these features are available in the dataset, the analysis will present challenges due to the multifaceted nature of hospital admissions.

The null hypothesis for the analysis is that the random forest model cannot predict patient admission duration within a root mean squared error (RMSE) of 5 hours, while the alternative hypothesis is that the model can achieve this level of accuracy. This threshold was selected before the data preparation and was based on limited clinical context rather than statistical

considerations. A narrow 5-hour prediction window would allow hospitals to efficiently schedule staff shifts, prepare rooms, and manage patient expectations.

While this target was ambitious, a high standard for predictive accuracy was prioritized to maximize potential benefits for healthcare resource management and patient satisfaction. Data exploration later revealed that admission duration had a standard deviation of approximately 100 hours, indicating high variability and suggesting that this goal would be challenging to achieve [2]. However, the threshold was maintained to evaluate how well a random forest model could overcome the inherent variability in hospital admission duration modeling using the available critical care data.

Data Collection

The data that will be used in the analysis comes from a public critical care dataset. It contains de-identified patient information from the intensive care unit and emergency department for a hospital that will remain anonymous. The data includes diagnostics, vital signs, procedures, medication information, and admission details relevant for training the model. One advantage of this data-gathering methodology is the use of a large, pre-existing dataset that would be difficult to collect independently. This approach ensures the model has sufficient training data to capture as much of the underlying structure as possible for this complex task. However, a disadvantage of using this pre-existing dataset is the lack of control over which variables were collected, restricting predictive power to the available data, which may include missing or inconsistent entries.

A significant challenge in using this dataset was gaining access because it contains sensitive protected health information and requires a credentialing process despite being publicly available. This process included completing an online training course on research ethics, privacy regulations, and confidentiality standards for human subjects research. Additionally, a data use agreement was signed that permitted the exploration of patient admission duration and readmission patterns, including the focus of this analysis. The thorough credentialing process ensured ethical compliance and responsible handling of the clinical data, reflecting the importance of maintaining high standards when working with sensitive information.

Data Extraction and Preparation

The data extraction and preparation process follows a ten-step plan involving data acquisition, exploration, transformation, and preparation for analysis. This analysis utilized various techniques and tools, with the primary ones being pgAdmin, PostgreSQL, Python, Pandas, Z-score outlier detection, and category aggregation. These tools and techniques were essential for the analysis but had their own limitations.

The first phase of the extraction and preparation process involves acquiring the data. For this analysis, the data is a public critical care dataset that can be downloaded from the publisher's website after completing the credentialing process. After logging in with a verified account, the dataset can be found by typing its name into the search bar. The research question explores patient admission duration for visits to the emergency department, and this information is contained in the 'hosp' module, as shown in *Figure 1*. Within this module, only the corresponding CSV files were downloaded into an organized folder:

admissions: Each entry represents an admission tied to a patient and includes additional admission information. This file serves as the core of the training data because the target variable is admission duration, meaning the data is at the admission level.

drgcodes: Contains diagnosis-related group codes used to classify hospital services for insurance purposes. Admissions sharing similar ICD codes may have similar durations, making this data potentially useful for admission prediction.

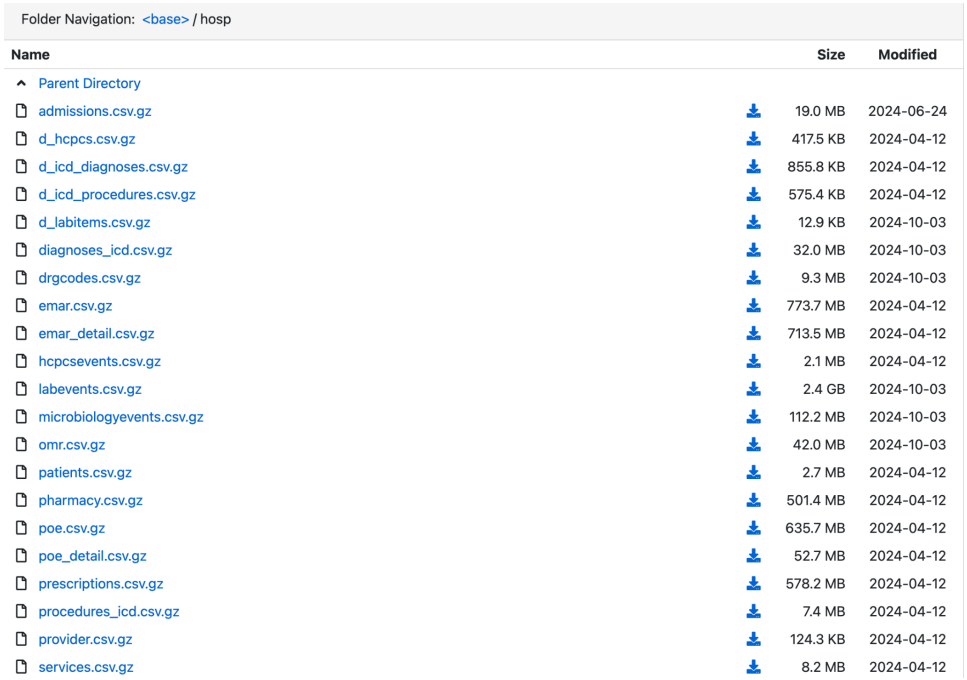
emar: Records the start, scheduled, and end times of medications administered to a patient. These features are used to calculate medication delays and durations, which can reflect a patient's diagnosis complexity or hospital congestion, both potentially related to admission duration.

omr: Includes vital signs, such as height, weight, blood pressure, and body mass index (BMI). These vital signs are commonly available during a patient's admission and may correlate with the duration of their stay.

patients: Contains demographic information, such as age and gender.

procedures_icd: Identifies the primary procedure for the admission and provides an international classification of diseases (ICD) code. These codes will be aggregated into procedure categories for analysis.

services: Captures the sequence of services provided to a patient during their stay.



Folder Navigation: <base> / hosp

Name	Size	Modified
Parent Directory		
admissions.csv.gz	19.0 MB	2024-06-24
d_hcpcs.csv.gz	417.5 KB	2024-04-12
d_icd_diagnoses.csv.gz	855.8 KB	2024-04-12
d_icd_procedures.csv.gz	575.4 KB	2024-04-12
d_labitems.csv.gz	12.9 KB	2024-10-03
diagnoses_icd.csv.gz	32.0 MB	2024-10-03
drgcodes.csv.gz	9.3 MB	2024-10-03
emar.csv.gz	773.7 MB	2024-04-12
emar_detail.csv.gz	713.5 MB	2024-04-12
hcpcsevents.csv.gz	2.1 MB	2024-04-12
labevents.csv.gz	2.4 GB	2024-10-03
microbiologyevents.csv.gz	112.2 MB	2024-10-03
omr.csv.gz	42.0 MB	2024-10-03
patients.csv.gz	2.7 MB	2024-04-12
pharmacy.csv.gz	501.4 MB	2024-04-12
poe.csv.gz	635.7 MB	2024-04-12
poe_detail.csv.gz	52.7 MB	2024-04-12
prescriptions.csv.gz	578.2 MB	2024-04-12
procedures_icd.csv.gz	7.4 MB	2024-04-12
provider.csv.gz	124.3 KB	2024-04-12
services.csv.gz	8.2 MB	2024-04-12

Figure 1: The hosp module containing the relevant CSV files for the analysis.

After decompressing the files, they were loaded into a PostgreSQL database management system for exploration and preparation. Using pgAdmin, a table was created for each file, and appropriate column names and data types were assigned, as shown in *Figure 2*. The import wizard was used to populate the tables and simplify the process, as shown in *Figure 3*. The initial set of features was identified through an exploratory analysis, with a focus on those relevant to the research question. Some features were extracted directly from the dataset, while others were engineered using the available information. Features with significant missing values were excluded from consideration because the missing values were not imputed to avoid reducing the overall data quality. The selected features, which reflect a focus on demographics, vital signs, and admission information, include:

patient_id: A unique 8-digit identifier for each patient, used to link patient information to their admissions.

admission_id: A unique 8-digit identifier for each admission.

admission_time: The start of a patient's stay, indicating the admission date and time.

discharge_time: The date and time when the patient was discharged.

admission_type: Describes the reason for the admission, such as a planned observation or emergency visit.

admission_location: Indicates how the patient was admitted, such as a walk-in or through a hospital transfer.

insurance: The patient's type of insurance, such as private or Medicaid.

primary_language: The patient's primary language.

marital_status: The patient's marital status.

race: The patient's race.

age: The patient's age.

gender: The patient's gender.

drg_severity: The severity level of a diagnosis, measured on a scale from one to four, with higher values indicating more serious conditions.

drg_mortality: The mortality risk associated with a diagnosis, measured on a scale from one to four, where higher values represent a greater risk of death.

procedure_count (Engineered): The number of procedures recorded for a patient during their admission. This feature was created by grouping entries in the procedures_icd table by admission ID and counting the total number of procedure records per admission.

diagnoses_count (Engineered): the number of diagnoses assigned to the patient during their admission. This feature was created by grouping entries in the diagnoses_icd table by admission ID and counting the total number of diagnosis records per admission.

first_service (Engineered): The first type of service the patient received, such as surgical or psychological. This feature was created using the services table by selecting the earliest service entry for each admission, identified by rows where the prev_service field was null.

medications_ordered (Engineered): The total number of medications ordered during the admission. This feature was created using the emar table by grouping entries by admission ID and counting all medication orders.

medications_given (Engineered): Not all ordered medications are actually administered, so medications_given was included to represent the total number of medications that were actually administered. This feature was created using the emar table by grouping entries by admission ID and counting all administered medications.

readmission_status (Engineered): Identifies whether the current admission is a readmission, with readmissions being defined as an admission within 30 days of the previous admission. This feature was created by ordering admissions by patient ID and admission time, then using a lag function to calculate the interval between the current admission time and the previous discharge time. If the interval was less than 30 days, the admission was marked as a readmission.

duration_hours (Engineered): The total duration of admission in hours. This feature was created by subtracting the admission time from the discharge time and converting the resulting time interval into a floating-point number representing hours.

medication_delays (Engineered): The difference between the scheduled and actual start times for administered medications.

medication_duration (Engineered): The duration of medication administration, calculated as the difference between the actual start and end times of administered medications.

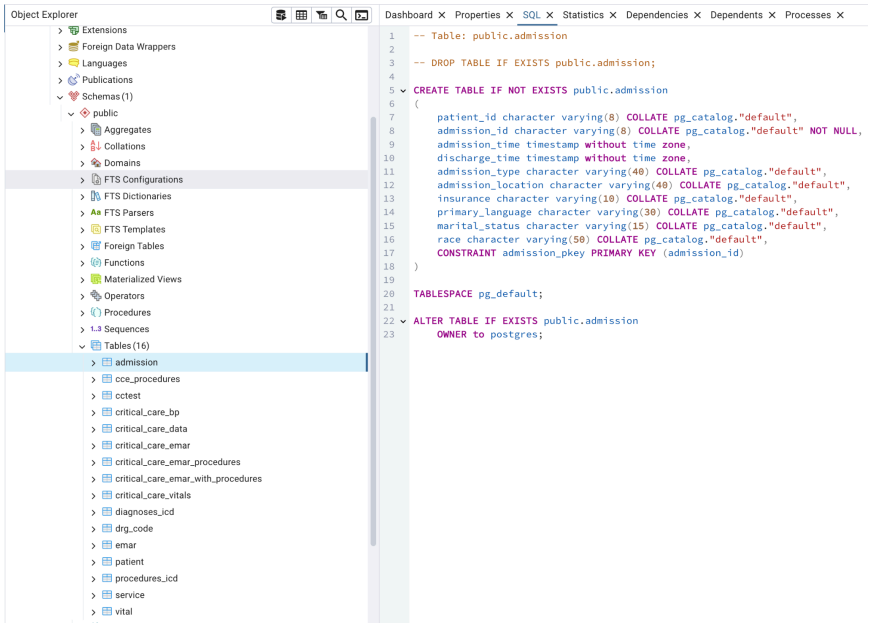


Figure 2: Example of a table structure in pgAdmin, showing column names and data types.

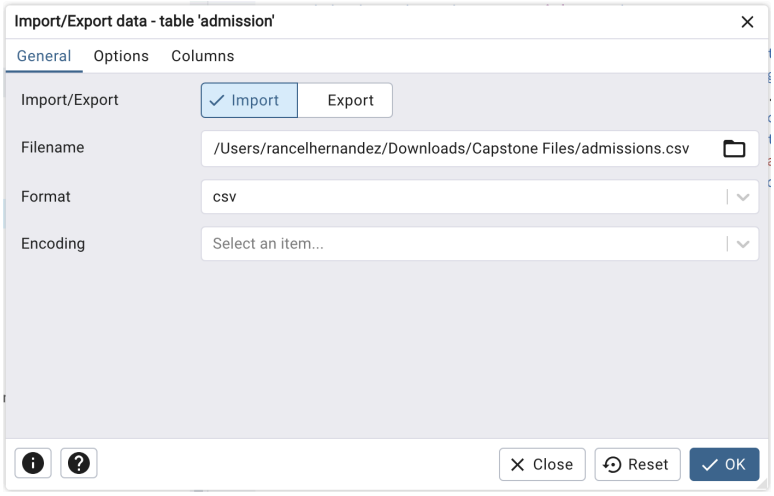


Figure 3: The import wizard used to populate the tables.

All the features listed above are included in every training set. However, multiple training sets were created because certain features were not available for all the admissions. Rather than excluding these features entirely, separate training sets were produced to make use of them. This approach allowed the analysis to incorporate more granular clinical data while still preserving a larger dataset for training on the more consistently available features. It should be noted that height and weight are not typically classified as vital signs, but they were included with the vital signs data to remain consistent with the original data source, which grouped them with other clinical measurements. The following features were included only in their respective training sets due to their limited availability:

height (vitals): The patient's height in inches.

weight (vitals): The patient's weight in pounds.

max_bmi (vitals) (Engineered): The patient's maximum BMI. This feature was created by using the omr table to group entries by admission ID, filter for BMI measurements, and select the highest value.

min_bmi (vitals) (Engineered): The patient's minimum BMI. This feature was created by using the omr table to group entries by admission ID, filter for BMI measurements, and select the lowest value.

procedure_category (procedures): The procedure category based on the admission's procedure ICD code. This feature was created by using the procedures_icd table to group procedure codes into their corresponding ICD-9-CM procedure categories. The mappings can be seen in Table 1.

sbp (Engineered): The patient's systolic blood pressure (SBP). This feature was created by extracting the numerator from blood pressure measurements in the omr table.

dbp (Engineered): The patient's diastolic blood pressure (DBP). This feature was created by extracting the denominator from blood pressure measurements in the omr table.

map (Engineered): The patient's mean arterial pressure (MAP), a calculated measure of blood flow pressure using the formula (SBP + 2 × DBP) / 3 [3].

ICD-9-CM Codes	Section
00-00	Procedures and interventions, not elsewhere classified
01–05	Operations on the nervous system
06–07	Operations on the endocrine system
08–16	Operations on the eye
17–17	Other miscellaneous diagnostic and therapeutic procedures
18–20	Operations on the ear
21–29	Operations on the nose, mouth, and pharynx
30–34	Operations on the respiratory system
35–39	Operations on the cardiovascular system
40–41	Operations on the hemic and lymphatic system
42–54	Operations on the digestive system
55–59	Operations on the urinary system
60–64	Operations on the male genital organs
65–71	Operations on the female genital organs
72–75	Obstetrical procedures
76–84	Operations on the musculoskeletal system
85–86	Operations on the integumentary system
87–99	Miscellaneous diagnostic and therapeutic procedures

Table 1 [4]

Some of the features contained data entry errors, which were addressed by removing the affected records. For example, there were admissions with negative duration intervals where the discharge time was recorded before the admission time. A similar issue occurred in the medication duration feature, where some medications were logged as being administered after they were already given. Additionally, the service data contained inconsistencies because some entries listed a previous service even though no prior service was recorded.

The remaining features showed no further data quality issues, so the dataset was ready to be split into separate tables. The main table, which included all the base features, was named critical_care_emar, as shown in *Figures 4 - 5*. This table served as the foundation for creating additional tables because all the admissions in the subsets were present in the base table. This is because the base table had fewer restrictions by excluding the additional features with limited availability.

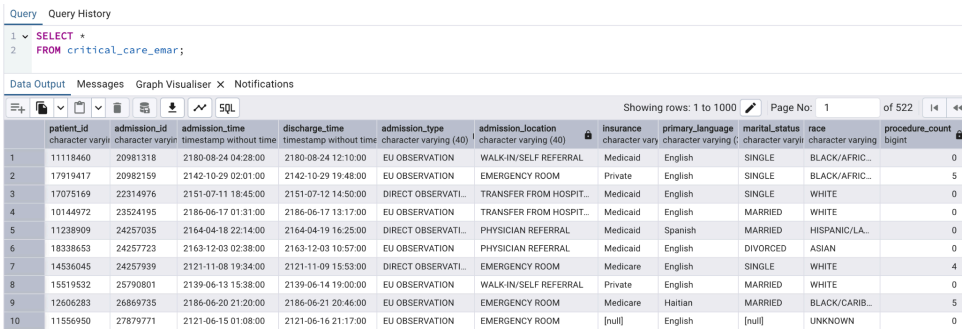
```
Query  Query History
28  -- add the extra medication cols
29  CREATE TABLE critical_care_emar AS
30  WITH add_emar_cols AS (
31      SELECT
32          CCD.admission_id,
33          -- add the medication delays/duration but use a 0 intervals for nulls
34          -- as a substitute b/c entries with no medications given have 0 delay/duration
35          COALESCE(SUM(E.chart_time - E.scheduletime), '0'::INTERVAL) AS medication_delays,
36          -- exclude medication entries where the start time is after the application time
37          -- those are data entry errors
38          COALESCE(SUM(CASE WHEN E.storetime > E.chart_time
39                            THEN E.storetime - E.chart_time
40                            ELSE NULL END), '0'::INTERVAL) AS medication_duration
41      FROM critical_care_data CCD
42      LEFT JOIN emar E
43          USING(admission_id)
44      GROUP BY CCD.admission_id
45  )
46
47  SELECT
48      CCD.*,
49      AEC.medication_delays,
50      AEC.medication_duration
51  FROM critical_care_data CCD
52  INNER JOIN add_emar_cols AEC
53      USING(admission_id);
54
55

Data Output  Messages  Graph Visualiser ×  Notifications

SELECT 544190

Query returned successfully in 33 secs 392 msec.
```

Figure 4: A query creating the critical_care_emar table by joining medication delay and duration features to the admissions data. Null values were replaced with zero intervals, and entries with invalid medication times were excluded.



```
Query  Query History
1  SELECT *
2  FROM critical_care_emar;

Data Output  Messages  Graph Visualiser ×  Notifications
Showing rows: 1 to 1000  Page No: 1  of 522
```

	patient_id	admission_id	admission_time	discharge_time	admission_type	admission_location	insurance	primary_language	marital_status	race	procedure_count
	character varyi	character varyin	timestamp without time	timestamp without time	character varying (40)	character varying (40)	character vary	character varying (	character varyi	character varying	bigint
1	11118460	20981318	2180-08-24 04:28:00	2180-08-24 12:10:00	EU OBSERVATION	WALK-IN/SELF REFERRAL	Medicaid	English	SINGLE	BLACK/AFRIC...	0
2	17919417	20982159	2142-10-29 02:01:00	2142-10-29 19:48:00	EU OBSERVATION	EMERGENCY ROOM	Private	English	SINGLE	BLACK/AFRIC...	5
3	17075169	22314976	2151-07-11 18:45:00	2151-07-12 14:50:00	DIRECT OBSERVATI...	TRANSFER FROM HOSPIT...	Medicaid	English	SINGLE	WHITE	0
4	10144972	23524195	2186-06-17 01:31:00	2186-06-17 13:17:00	EU OBSERVATION	TRANSFER FROM HOSPIT...	Medicaid	English	MARRIED	WHITE	0
5	11238909	24257035	2164-04-18 22:14:00	2164-04-19 16:25:00	DIRECT OBSERVATI...	PHYSICIAN REFERRAL	Medicaid	Spanish	MARRIED	HISPANIC/LA...	0
6	18338653	24257723	2163-12-03 02:38:00	2163-12-03 10:57:00	EU OBSERVATION	PHYSICIAN REFERRAL	Medicaid	English	DIVORCED	ASIAN	0
7	14536045	24257939	2121-11-08 19:34:00	2121-11-09 15:53:00	DIRECT OBSERVATI...	EMERGENCY ROOM	Medicare	English	SINGLE	WHITE	4
8	15519532	25790801	2139-06-13 15:38:00	2139-06-14 19:00:00	EU OBSERVATION	WALK-IN/SELF REFERRAL	Private	English	MARRIED	WHITE	0
9	12606283	26869735	2186-06-20 21:20:00	2186-06-21 20:46:00	EU OBSERVATION	EMERGENCY ROOM	Medicare	Haitian	MARRIED	BLACK/CARIB...	5
10	11556950	27879771	2121-06-15 01:08:00	2121-06-16 21:17:00	EU OBSERVATION	EMERGENCY ROOM	[null]	English	[null]	UNKNOWN	0

Figure 5: A SELECT query on the critical_care_emar table showing a subset of the base features.

Three subsets were created from critical_care_emar: critical_care_vitals, critical_care_bp, and critical_care_emar_procedures. The base table contained 544,190 entries. The critical_care_vitals table included vital sign features such as height, weight, min_bmi, and max_bmi, and contained 89,669 entries. The critical_care_bp table included blood pressure-related features such as SBP, DBP, and MAP, and contained 48,928 entries. The critical_care_emar_procedures table included the procedure_category feature and contained 162,992 entries.

The vital signs and blood pressure data were sourced from the omr table. However, they were not directly linked to admissions because each measurement included only a patient ID and chart date. To assign these measurements to the appropriate admissions, the entries from the omr table were linked by patient ID and filtered based on whether the measurement timestamp fell within the admission time frame. A 48-hour window prior to admission was also included because measurements within that period were considered relevant to the upcoming admission.

Once the data preparation in pgAdmin was complete, the tables were exported as CSV files and loaded into Pandas DataFrames, as shown in *Figures 6 - 7*. Each DataFrame was checked for duplicate entries and missing values. Duplicate entries can introduce bias in the training process because they are more likely to be included in the bootstrap samples for individual trees, overrepresenting certain observations and potentially skewing predictions. These entries were identified using Pandas' .duplicated() method, which returns duplicate rows. The process and results are shown in *Figure 8*. Since there were relatively few duplicate entries, they were removed.

[2]:
```
file_path = '/Users/rancelhernandez/Downloads/Capstone Files/critical_care_base.csv'

# this is the base dataFrame that achieved the lowest RMSE ~57 hours
critical_care_df = pd.read_csv(file_path)

file_path = '/Users/rancelhernandez/Downloads/Capstone Files/critical_care_vitals.csv'

# same columns as critical_care_df,
# but includes ['max_bmi', 'min_bmi', 'weight', 'height']
critical_care_vitals = pd.read_csv(file_path)

file_path = '/Users/rancelhernandez/Downloads/Capstone Files/critical_care_bp.csv'

# includes blood pressure as ['sbp', 'dbp', 'map']
critical_care_bp = pd.read_csv(file_path)

file_path = '/Users/rancelhernandez/Downloads/Capstone Files/critical_care_procedures.csv'

# includes ['procedure_category'] for surgeries
critical_care_pro = pd.read_csv(file_path)
```

[3]:
```
critical_care_df.head()
```

[3]:

ender	duration_hours	medication_delays	medication_duration	drg_severity	drg_mortality	medication_delays_hours	medication_duration_hours
F	49.3833	00:00:00	00:00:00	0	0	0.000000	0.0
M	15.7333	00:00:00	00:00:00	0	0	0.000000	0.0
M	9.5333	00:32:00	00:06:00	0	0	0.533333	0.1
F	4.5333	00:00:00	00:00:00	0	0	0.000000	0.0
M	29.0833	00:00:00	00:00:00	0	0	0.000000	0.0

Figure 6: Loading the derived datasets into their corresponding DataFrames, along with a preview of the critical_care_df DataFrame.

In [4]:
```
print("Base:", critical_care_df.shape)
print("Vitals:", critical_care_vitals.shape)
print("Blood Pressure:", critical_care_bp.shape)
print("Procedures:", critical_care_pro.shape)
```

```
Base: (544190, 26)
Vitals: (89669, 30)
Blood Pressure: (48928, 29)
Procedures: (162992, 27)
```

Figure 7: The shape of each loaded dataset.

```
In [5]: # all entries should have a unique admission id
if sum(critical_care_df['admission_id'].duplicated()) == 0:
    print('There are no duplicated values in the Base DataFrame.')
else:
    print('Duplicated entries detected in Base DataFrame.')
    print('Count:', critical_care_df.shape[0] - len(critical_care_df['admission_id'].unique()))

# do the same for the other DataFrames
if sum(critical_care_vitals['admission_id'].duplicated()) == 0:
    print('There are no duplicated values in Vitals DataFrame.')
else:
    print('Duplicated entries detected in Vitals DataFrame.')
    print('Count:', critical_care_vitals.shape[0] - len(critical_care_vitals['admission_id'].unique()))

if sum(critical_care_bp['admission_id'].duplicated()) == 0:
    print('There are no duplicated values in Blood Pressure DataFrame.')
else:
    print('Duplicated entries detected in Blood Pressure DataFrame.')
    print('Count:', critical_care_bp.shape[0] - len(critical_care_bp['admission_id'].unique()))

if sum(critical_care_pro['admission_id'].duplicated()) == 0:
    print('There are no duplicated values in Procedures DataFrame.')
else:
    print('Duplicated entries detected in Procedures DataFrame.')
    print('Count:', critical_care_pro.shape[0] - len(critical_care_pro['admission_id'].unique()))
```

```
Duplicated entries detected in Base DataFrame.
Count: 10
Duplicated entries detected in Vitals DataFrame.
Count: 1868
Duplicated entries detected in Blood Pressure DataFrame.
Count: 3730
Duplicated entries detected in Procedures DataFrame.
Count: 1
```

Figure 8: The process used to identify duplicate values. The vital sign–related datasets contained the most duplicates, likely due to multiple measurements recorded throughout a single admission.

Missing values can raise errors in several of the methods used throughout the analysis, such as Z-score calculation and model training. Therefore, entries with missing values were identified and removed. Their removal is justified due to the abundance of available training data. Missing values were identified using Pandas' .isna().sum() method, and part of this process is shown in *Figure 9*. Notably, each DataFrame contained missing values in the same features: insurance, primary_language, and marital_status. Additionally, the base DataFrame had one missing value in the admission_location feature. The number of missing values appeared to be proportional to the size of the DataFrame, with larger DataFrames containing more missing values. However, none of the DataFrames had a concerning proportion of missing values, so their removal was appropriate.

```
In [10]: # print the missing values in each DataFrame
         print("Base:\n", critical_care_df.isna().sum())

         print("\nVitals:\n", critical_care_vitals.isna().sum())

         print("\nBlood Pressure:\n", critical_care_bp.isna().sum())

         print("\nProcedures:\n", critical_care_pro.isna().sum())
```

```
Base:
 patient_id                      0
admission_id                     0
admission_time                   0
discharge_time                   0
admission_type                   0
admission_location               1
insurance                     9328
primary_language               767
marital_status               13566
race                             0
procedure_count                  0
diagnoses_count                  0
first_service                    0
medications_ordered              0
medications_given                0
readmission_status               0
duration                         0
age                              0
gender                           0
duration_hours                   0
medication_delays                0
medication_duration              0
drg_severity                     0
drg_mortality                    0
medication_delays_hours          0
medication_duration_hours        0
dtype: int64
```

Figure 9: The method for identifying missing values in the DataFrames, with the output shown for the base DataFrame.

The next step in the data preparation process was to aggregate categories in categorical features that contained an excessive number of unique values. Some features had over 20 distinct categories, as shown in the pre-aggregation overview in *Figure 10*, which can be problematic during the encoding process due to a significant increase in dimensionality. Specifically, One-Hot Encoding was utilized, which encodes the categorical variables by creating a binary feature for each category. As a result, a categorical feature with 20 categories would generate 20 additional binary features, potentially introducing irrelevant features and adding noise to the

training process. The category mappings are shown in *Figure 11*, the results of the aggregation are summarized in Table 2, and the post-aggregation output is shown in *Figure 12*.

```
In [12]: # categorical features are type string(object)
string_columns = []

# excluded features
excluded_features = ['admission_time', 'discharge_time', 'duration', 'medication_delays', 'medication_duration']

# at each iteration, check if the current col is type object or an exluded feature
# the print the unique categories and save the current col to string_columns
for col in critical_care_df.columns:

    if critical_care_df[col].dtype == 'object' and col not in excluded_features:
        print(f'{col}: {list(critical_care_df[col].unique())}\n')
        string_columns.append(col)

# this cetegorical variable has integer based groups
# where for a range of ICD procedure codes
# mappings found in the report
print(f'procedure_group: {list(critical_care_pro["procedure_category"].unique())}\n')
```

```
admission_type: ['EU OBSERVATION', 'AMBULATORY OBSERVATION', 'DIRECT OBSERVATION', 'URGENT', 'EW EMER.', 'OBSERVATION ADMIT', 'SURGICAL SAME DAY ADMISSION', 'DIRECT EME
R.', 'ELECTIVE']

admission_location: ['EMERGENCY ROOM', 'PACU', 'TRANSFER FROM HOSPITAL', 'WALK-IN/SELF REFERRAL', 'PHYSICIAN REFERRAL', 'PROCEDURE SITE', 'CLINIC REFERRAL', 'TRANSFER F
ROM SKILLED NURSING FACILITY', 'INFORMATION NOT AVAILABLE', 'AMBULATORY SURGERY TRANSFER', 'INTERNAL TRANSFER TO OR FROM PSYCH']

insurance: ['Private', 'Medicare', 'Medicaid', 'Other', 'No charge']

primary_language: ['English', 'Other', 'Spanish', 'Kabuverdianu', 'American Sign Language', 'Russian', 'Italian', 'Haitian', 'Portuguese', 'Chinese', 'Korean', 'Vietnam
ese', 'Arabic', 'Modern Greek (1453-)', 'Thai', 'Bengali', 'Somali', 'Polish', 'Persian', 'French', 'Amharic', 'Hindi', 'Armenian', 'Khmer', 'Japanese']

marital_status: ['MARRIED', 'SINGLE', 'WIDOWED', 'DIVORCED']

race: ['WHITE', 'HISPANIC/LATINO - MEXICAN', 'BLACK/AFRICAN AMERICAN', 'OTHER', 'BLACK/CARIBBEAN ISLAND', 'BLACK/AFRICAN', 'HISPANIC/LATINO - PUERTO RICAN', 'WHITE - OT
HER EUROPEAN', 'HISPANIC/LATINO - COLUMBIAN', 'SOUTH AMERICAN', 'BLACK/CAPE VERDEAN', 'PATIENT DECLINED TO ANSWER', 'HISPANIC OR LATINO', 'ASIAN', 'ASIAN - CHINESE', 'W
HITE - EASTERN EUROPEAN', 'MULTIPLE RACE/ETHNICITY', 'HISPANIC/LATINO - DOMINICAN', 'HISPANIC/LATINO - HONDURAN', 'WHITE - RUSSIAN', 'HISPANIC/LATINO - GUATEMALAN', 'UN
KNOWN', 'AMERICAN INDIAN/ALASKA NATIVE', 'PORTUGUESE', 'WHITE - BRAZILIAN', 'ASIAN - ASIAN INDIAN', 'ASIAN - SOUTH EAST ASIAN', 'NATIVE HAWAIIAN OR OTHER PACIFIC ISLAND
ER', 'UNABLE TO OBTAIN', 'ASIAN - KOREAN', 'HISPANIC/LATINO - SALVADORAN', 'HISPANIC/LATINO - CUBAN', 'HISPANIC/LATINO - CENTRAL AMERICAN']

first_service: ['MED', 'VSURG', 'NSURG', 'CMED', 'OBS', 'NMED', 'OMED', 'SURG', 'TRAUM', 'GYN', 'ORTHO', 'GU', 'TSURG', 'PSURG', 'CSURG', 'ENT', 'PSYCH', 'EYE', 'DENT']

gender: ['F', 'M']

procedure_group: [15, 10, 17, 16, 18, 8, 13, 11, 14, 7, 1, 12, 2, 9, 6, 3, 5]
```

Figure 10: An overview of the unique categories in each categorical feature before aggregation.

Feature	Original	Aggregated	Reduction	Justification

	Categories	Categories		
admission_type	9	3	67%	These groups reflect the general purpose and urgency of each admission type.
admission_location	11	5	55%	“Emergency Room”and “Information Not Available” were kept as separate categories due to their distinct clinical significance, and the remaining categories were grouped based on the nature of their transfer.
primary_language	25	6	76%	Grouped based on the most commonly spoken languages in the United States, which is appropriate given that the dataset comes from a hospital in Boston, Massachusetts [5]. Despite this grouping, significant imbalance remains, with English accounting for over 90% of observations. Less common languages, such as Vietnamese, have much smaller representation. These categories were retained because they reflect the typical language distribution seen in U.S. healthcare settings.
race	33	5	85%	These groupings reflect the largest demographic groups in the U.S. population, as defined by the U.S. Census Bureau [6].
first_service	19	5	74%	These groups were formed based on naming conventions and clinical similarity. For example, services containing “MED” were grouped under medical because they represent non-surgical inpatient care.

Table 2

In [15]:
```
# first_service aggregation groups
medical_services = ['MED', 'CMED', 'NMED', 'OMED']
surgical_services = ['SURG', 'NSURG', 'VSURG', 'TSURG', 'CSURG', 'PSURG']
specialized = ['ORTHO', 'ENT', 'GU', 'GYN', 'EYE', 'DENT', 'OBS']

# admission_type aggregation groups
urgent = ['URGENT', 'EW EMER.', 'DIRECT EMER.']
observation = ['EU OBSERVATION', 'OBSERVATION ADMIT', 'DIRECT OBSERVATION', 'AMBULATORY OBSERVATION']
planned = ['ELECTIVE', 'SURGICAL SAME DAY ADMISSION']

# admission_location aggregation groups
referrals = ['PHYSICIAN REFERRAL', 'CLINIC REFERRAL', 'WALK-IN/SELF REFERRAL']
external_transfers = ['TRANSFER FROM HOSPITAL', 'TRANSFER FROM SKILLED NURSING FACILITY']
internal_procedural_transfers = ['PACU', 'PROCEDURE SITE', 'INTERNAL TRANSFER TO OR FROM PSYCH', 'AMBULATORY SURGERY TRANSFER']
```

In [16]:
```
# use the aggregation groups to create mappings for the categories
race_mappings = {
    **{race: 'Asian' for race in asian},
    **{race: 'Hispanic/Latino' for race in hispanic_latino},
    **{race: 'White' for race in white},
    **{race: 'Black' for race in black},
    **{race: 'Other' for race in other_races}
}

language_mappings = {
    **{language: 'Other' for language in other_languages}
}

first_service_mappings = {
    **{code: 'Medical' for code in medical_services},
    **{code: 'Surgical' for code in surgical_services},
    **{code: 'Specialized' for code in specialized},
}

admission_type_mappings = {
    **{admission_type: 'Urgent' for admission_type in urgent},
    **{admission_type: 'Observation' for admission_type in observation},
    **{admission_type: 'Planned' for admission_type in planned}
}

admission_location_mappings = {
    **{admission_location: 'Referrals' for admission_location in referrals},
    **{admission_location: 'External Transfers' for admission_location in external_transfers},
    **{admission_location: 'Internal Procedural Transfers' for admission_location in internal_procedural_transfers}
}
```

Figure 11: The categorical feature aggregation mappings.

In [18]:
```
# confirm the mapping process
excluded_features = ['admission_time', 'discharge_time', 'duration', 'medication_delays', 'medication_duration']
for col in critical_care_aggregated.columns:
    if critical_care_aggregated[col].dtype == 'object' and col not in excluded_features:
        print(f'{col}: {list(critical_care_aggregated[col].unique())}\n')
```

```
admission_type: ['Observation', 'Urgent', 'Planned']

admission_location: ['EMERGENCY ROOM', 'Internal Procedural Transfers', 'External Transfers', 'Referrals', 'INFORMATION NOT AVAILABLE']

insurance: ['Private', 'Medicare', 'Medicaid', 'Other', 'No charge']

primary_language: ['English', 'Other', 'Spanish', 'Chinese', 'Vietnamese', 'Arabic']

marital_status: ['MARRIED', 'SINGLE', 'WIDOWED', 'DIVORCED']

race: ['White', 'Hispanic/Latino', 'Black', 'Other', 'Asian']

first_service: ['Medical', 'Surgical', 'Specialized', 'TRAUM', 'PSYCH']

gender: ['F', 'M']
```

Figure 12: The results of the aggregation process, showing the reduction in unique categories for each categorical feature.

To include the categorical features in the analysis, One-Hot encoding was utilized because none of the unencoded categorical features contained ranked categories that would

require ordinal encoding. The encoding process, illustrated in *Figure 13*, began by creating a deep copy of the aggregated procedures DataFrame, which contains all the categorical features used across the datasets. For each categorical feature, a OneHotEncoder was initialized with parameters configured to handle unknown categories during transformation and to drop the first category of each feature to prevent multicollinearity.

These encoders were fitted on the deep copy to learn all the possible categories and were stored in a dictionary to ensure consistent encoding across all the datasets. A transformation function was then defined to process all the DataFrames. This function creates a copy of the input DataFrame, transforms each categorical column using its corresponding fitted encoder, generates appropriate feature names for the new binary columns, concatenates the resulting binary features to the DataFrame, and removes the original categorical features. The encoding process was then applied to all DataFrames in the analysis, and the difference in the number of columns before and after transformation is shown in *Figure 14*.

In [19]:
```
# create a copy for the One-Hot encoding
# use the procedure DataFrame since it has all the cetegorical varibles plus procedure_category
critical_care_transformed = critical_care_aggregated_pro.copy(deep=True)

# append procedure_category to string_columns to ensure it gets encoded
string_columns.append('procedure_category')

# create and fit all encoders on the main DataFrame
encoders = {}
for col in string_columns:

    # create an encoder for each column
    encoders[col] = OneHotEncoder(sparse_output=False, handle_unknown='ignore', drop='first')

    # fit on the base DataFrame
    encoders[col].fit(critical_care_transformed[col].values.reshape(-1, 1))

# this function transforms a DataFrame using our fitted encoders
def transform_df(df):

    # create a copy to use for the transformations
    df_copy = df.copy(deep=True)

    for col in string_columns:

        # should avoid errors where procedure_category is called for the other DataFrames
        if col in df_copy.columns:
            # transform using the fitted encoder
            encoded_values = encoders[col].transform(df_copy[col].values.reshape(-1, 1))

            # create a DataFrame with encoded values
            feature_names = encoders[col].get_feature_names_out([col])
            transformed_values = pd.DataFrame(encoded_values, columns=feature_names, index=df_copy.index)

            # add it to the DataFrame and drop original column
            df_copy = pd.concat([df_copy, transformed_values], axis=1)
            df_copy = df_copy.drop(col, axis=1)

    return df_copy

# transform all DataFrames
critical_care_transformed = transform_df(critical_care_aggregated)
critical_care_transformed_vitals = transform_df(critical_care_aggregated_vitals)
critical_care_transformed_bp = transform_df(critical_care_aggregated_bp)
critical_care_transformed_pro = transform_df(critical_care_aggregated_pro)
```

Figure 13: The One-Hot encoding process on all the DataFrames.

In [20]:
```
# the number of columns increased from 20 to 54
print("Base Number of original columns:", critical_care_df.shape[1])
print("Base Number of columns after encoding:", critical_care_transformed.shape[1])

print("Vitals Number of original columns:", critical_care_vitals.shape[1])
print("Vitals Number of columns after encoding:", critical_care_transformed_vitals.shape[1])

print("Blood Pressure Number of original columns:", critical_care_bp.shape[1])
print("Blood Pressure Number of columns after encoding:", critical_care_transformed_bp.shape[1])

print("Procedures Number of original columns:", critical_care_pro.shape[1])
print("Procedures Number of columns after encoding:", critical_care_transformed_pro.shape[1])
```

```
Base Number of original columns: 26
Base Number of columns after encoding: 45
Vitals Number of original columns: 30
Vitals Number of columns after encoding: 49
Blood Pressure Number of original columns: 29
Blood Pressure Number of columns after encoding: 48
Procedures Number of original columns: 27
Procedures Number of columns after encoding: 61
```

Figure 14: The difference in number of columns before and after encoding for each DataFrame.

Random Forest can be robust to outliers because the individual trees are trained on subsets of the data, limiting the influence of outliers to only those trees. However, outliers should still be removed because they can introduce noise into the training process by skewing certain trees. Outliers were detected by calculating the Z-scores of the continuous features, which standardize the data to have a mean of zero and a standard deviation of one. Any data point that is three standard deviations away from the mean is considered an outlier because it significantly deviates from the expected distribution.

Although this method is ideally suited for features that are normally distributed, it was also applied to non-normally distributed features in this analysis because values beyond three standard deviations were rare and clearly deviated from the majority of values clustered near the mean. However, standard statistical approaches are not appropriate for clinical features with physiological limitations. For these features, domain-specific thresholds based on supported research were applied instead of relying on statistical deviations. These features and their outlier ranges are displayed in Table 3.

Feature	Lower Limit	Upper Limit	Source
BMI	10	100	BMI values below 10 and above 100 kg/m² were treated as outliers because values outside this range are implausible and likely due to data entry errors [7 - 8].
Height	36	84	Adult height values below 36 inches or above 84 inches were flagged as outliers because heights outside this range fall within 1% of the adult population [9]. This is further supported by the fact that the youngest patient in the dataset is 18 years old.
Weight	50	800	An upper limit of 800 pounds is statistically justified because this weight falls in the 99th percentile, meaning patients above this weight exceed 99% of the population [10]. Conversely, 50 pounds was set as the lower limit because the youngest patient is 18 years old, and 50 pounds corresponds to the 1st percentile for adult males [11].
SBP	70	200	High systolic blood pressure exceeding 180 mm Hg can lead to a hypertensive crisis and may be fatal [12]. Conversely, low blood pressure can result in hypotension, typically defined as a systolic pressure below 90 mm Hg [13].
DBP	30	130	Diastolic pressure above 110 mm Hg may indicate a hypertensive crisis [12], and below 60 mm Hg is considered hypotension [13].

Table 3

The results of the outlier detection are shown in Table 4. Notably, there were 26,640 entries containing outliers in the base DataFrame, 5,078 entries from the vital signs DataFrame, 2,932 entries in the blood pressure DataFrame, and 8,847 entries in the procedures DataFrame. One of the more extreme outliers came from the admission duration because the highest duration was approximately 515 days and this greatly deviates from the mean of 4 days, as shown in *Figure 15*. Additionally, the highest value for medication delays is equivalent to 950 years, while the mean excluding these outliers is only 18 hours. These values were too extreme, likely due to data entry errors, and would have certainly skewed the predictions.

Feature	Outliers Count	Outliers Min	Outliers Max	Outliers Mean	Outliers Std
Procedure_count	10,928	9	41	12.55	4.55
Diagnoses_count	6,479	35	57	37.54	1.57
Duration_hours	9,159	622.43	12,373.50	1,001.22	523.37
Medication_delays_hours	5,328	329,853.38	8,326,692.95	556,549.31	464,056.96
Medication_duration_hours	6,303	207.40	18,549.32	392.96	367.35
Max_bmi	313	0	126,540.0	2,429.31	9,787.74
Min_bmi	348	0	126,540.0	2,173.27	9,303.21
Height	259	0	73,252.0	517.56	4,605.22
Weight	80	0.22	181,580.69	3,379.10	20,642.11
Sbp	239	40	250	183.45	64.28
Dbp	113	0	199	69.41	70.04
Map	373	13.33	203.67	127.73	42.49
Medications_ordered	7,674	720	27,890	1,320.58	911.25
Medications_given	7,651	501	19,538	914.42	638.69

Table 4

```
Duration_hours
Count: 512600
Min: 0.0
Max: 622.4167
Mean: 97.18
Std: 100.52

Outliers:
Count: 9159
Min: 622.4333
Max: 12373.5
Mean: 1001.22
Std: 523.37
```

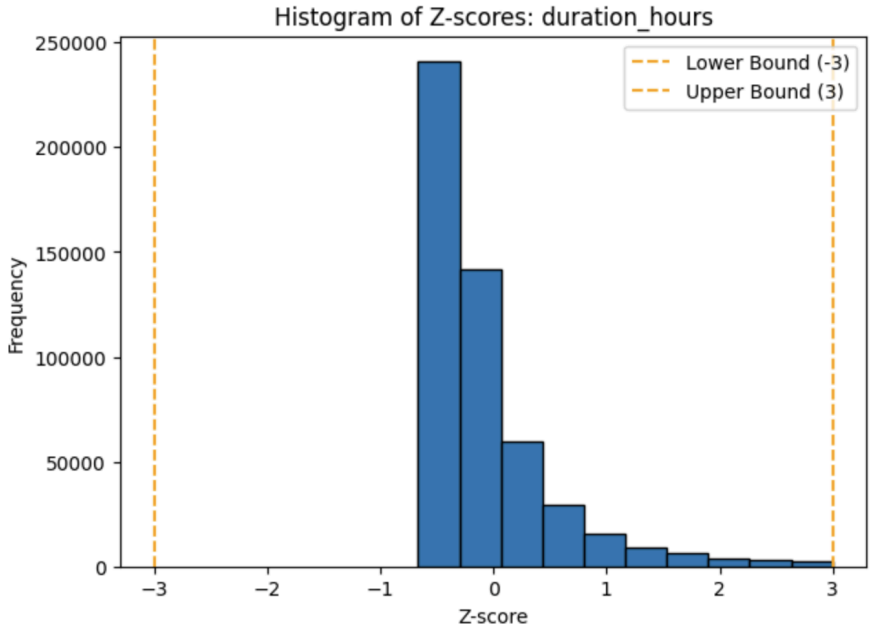


Figure 15: The statistical analysis of admission duration, showing that the maximum outlier value was 12,373 hours, which is approximately 515 days. The histogram excludes outliers to better visualize the statistical distribution. Including these outliers in the plot significantly skewed the scale and made the histogram less interpretable.

The distributions of the categorical features are shown in *Figure 16*. Most of the categories are severely imbalanced, with the exception of gender. Some of these imbalances are expected, such as the predominance of white individuals in the race category, since the data is based in the United States. As a result, only categories with a significantly low number of observations will be removed. This approach is appropriate given the large size of the dataset,

where even less common categories often have thousands of observations. Entries with the category ‘INFORMATION NOT AVAILABLE’ in admission_location will be removed due to having only 288 observations, along with the ‘No charge’ category in insurance, which includes only 420. Despite the imbalance in the procedure_category feature, rare categories within it will be retained for analysis because this feature is specific to the procedures model, and some categories may be dropped later in the reduced feature set if they are not identified as relevant during training.

```
         admission_type
Urgent         241619
Observation    225584
Planned         54556
Name: count, dtype: int64

         admission_location
EMERGENCY ROOM                   234080
Referrals                        211853
External Transfers                56912
Internal Procedural Transfers     18626
INFORMATION NOT AVAILABLE           288
Name: count, dtype: int64

         insurance
Medicare     237678
Private      169052
Medicaid     101211
Other         13398
No charge       420
Name: count, dtype: int64

         primary_language
English       471547
Other          23996
Spanish        17098
Chinese         7378
Vietnamese      1118
Arabic           622
Name: count, dtype: int64

         marital_status
MARRIED     226186
SINGLE      199615
WIDOWED      56206
DIVORCED     39752
Name: count, dtype: int64
```

```
         race
White              351226
Black               86765
Other               33722
Hispanic/Latino     30998
Asian               19048
Name: count, dtype: int64

         first_service
Medical        356500
Surgical        86660
Specialized     62887
PSYCH            8611
TRAUM            7101
Name: count, dtype: int64

         gender
F    273689
M    248070
Name: count, dtype: int64

         procedure_category
18    41772
10    25628
17    23110
8     21259
15    17200
14     6880
1      6739
7      3364
13     3178
11     2957
2      1101
16     1089
12     1007
6       788
9       782
3       186
5        53
Name: count, dtype: int64
```

Figure 16: Distributions of all categorical features included in the analysis for the base dataset. These distributions also apply to the derived datasets because they were subsets of the base dataset.

The final step in the data preparation process was to split the datasets into training and test sets. An additional validation set was not required because Random Forest models provide an out-of-bag (OOB) validation score [14]. This model utilizes bootstrap sampling for each individual tree, meaning it trains on different subsets of the data. As a result, the remaining data not used to train a given tree can serve as an OOB validation set, providing an internal and independent measure of the model's generalization performance.

The training size for each dataset was determined based on the number of observations it contained. The base DataFrame had the largest number of entries, but a smaller training size of 0.3 was used. Several training sizes were tested for this dataset, but increasing the training size did not improve model performance. This was likely due to the limited predictive power of the available features, so the smaller size was chosen to reduce the overall training time. The remaining datasets used a training size of 0.8 because they contained fewer observations and required as much training data as possible. A standard test size of 0.2 was used for all datasets to ensure there was enough data available for evaluating the models' performance. *Figure 17* shows the process for splitting the data into the training and test sets. Only the base DataFrame is shown, but the same process was applied to all the datasets.

Split the Data for the Base Model

In [23]:
```
# extend the excluded features to remove irrelevant features
excluded_features.extend(['patient_id', 'admission_id', 'duration_hours'])

# get the set of intial features by subtracting the excluded set from the columns
initial_features = list(set(critical_care_transformed.columns) - set(excluded_features))

# reduced feature set with feature importance of > 0.01
# this was assessed separately and the results recorded below
# to avoid an additional 4 models
reduced_features = ['diagnoses_count', 'medication_delays_hours', 'age', 'drg_mortality',
                    'medication_duration_hours', 'procedure_count', 'first_service_PSYCH',
                    'drg_severity', 'admission_type_Planned', 'admission_type_Urgent']

# create X using the initial set of features
X = critical_care_transformed[initial_features]

# set y to duration in hours
y = critical_care_transformed["duration_hours"]

# split the data set into a 30/20 train/test sets
# ~156,000 train size
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.3, test_size=0.2, random_state=42)
```

Figure 17: The process for splitting the data into training and test sets for the base model.

Various tools and techniques were essential for the analysis, each with their own advantages and limitations, including:

pgAdmin: A graphical user interface designed specifically for PostgreSQL, the database management system used in this analysis. Its visual interface made it easier to manage and explore the data. However, pgAdmin can become slow when working with very large tables or result sets. This limitation was observed when querying the vital signs table, which contained over seven million entries.

PostgreSQL: A database management system that allows for data manipulation and storage using Structured Query Language. One advantage of PostgreSQL is its simple and efficient syntax, which improves readability when working with long and complex queries, many of which were created during the data preparation process. A limitation of PostgreSQL is that it can select suboptimal join strategies, which may significantly increase query runtime, particularly when working with large datasets, as was the case in this analysis [15].

Python: An object-oriented programming language. One advantage of using Python is its vast selection of libraries for data analysis and machine learning. For example, Scikit-Learn, a Python library, was used to implement the Random Forest Regressor in this analysis. One limitation of Python is that data manipulation can be less efficient compared to using a database management system, which is why PostgreSQL was used to handle large-scale data transformations more effectively.

Pandas: A Python library for data management and analysis. One advantage of using Pandas DataFrames is its compatibility with other Python data analytics libraries. For example, Seaborn, which was used in this analysis to visualize the model's feature

importances, is designed to work with Pandas. One limitation of Pandas is that certain operations can be unintuitive or overly complex. For instance, the syntax for aggregating categorical features was not easily interpretable and required examining the documentation to fully understand, which was time-consuming and distracting from the analysis. The mapping function was used to aggregate categories, but any category not included in the mapping dictionary was converted to a NaN value. To preserve these distinct categories, the .fillna() method was chained to repopulate them with their original values. This syntax is shown in *Figure 18*.

```
critical_care_aggregated['race'] = critical_care_df['race'].map(race_mappings).fillna(critical_care_df['race'])
```

Figure 18: The code used to aggregate the categories for the race feature.

Category Aggregation: A dimensionality reduction technique that reduces the number of categories by grouping them based on similarity. One advantage of using category aggregation is that it significantly reduces the dimensionality of features when applying One-Hot Encoding because each additional category generates a new binary column. However, aggregation can reduce granularity to the point that it obscures important information by combining categories that may have distinct admission durations.

The extraction and preparation process followed a ten-step plan to prepare the data for analysis. The critical care dataset was downloaded from the publisher's website and loaded into pgAdmin, using PostgreSQL to efficiently transform the data. A thorough feature selection process resulted in four separate datasets, each with its own set of features: a base dataset containing 544,190 entries, a vital signs subset with 89,669 entries, a blood pressure measurements subset with

48,928 entries, and a procedure categories subset with 162,992 entries. Additional clinically relevant features were engineered using existing information, such as readmission status, medication delays, and procedure counts.

To address data quality issues, missing values, duplicates, outliers, and extremely rare categories were removed using statistical methods and domain-specific thresholds. The dimensionality was reduced through categorical feature aggregation while preserving the distinct meaning of each group. The final datasets were cleaned, encoded, and prepared for model training. By utilizing tools like PostgreSQL for large-scale transformations, Pandas for data manipulation, and appropriate statistical techniques for outlier detection, the resulting datasets were well-suited for the predictive modeling of hospital admission duration.

Analysis

The primary techniques used to analyze the data include a Random Forest Regressor, hyperparameter tuning with Grid Search, feature importance analysis, and RMSE for performance evaluation. A Random Forest Regressor is an ensemble method that is composed of a diverse set of decision trees. Each tree is made up of branches and nodes, where each node splits the data based on the feature and threshold that minimize prediction error [16]. These binary trees grow in layers, with each node having at most two children. The goal at each split is to increase the homogeneity of the resulting subsets, and a node that cannot be split further is called a leaf.

Overfitting is reduced and model diversity is encouraged by using bootstrap sampling, which selects a random subset of training data for each tree. Additionally, each node is limited to

a random subset of features when determining the best split. This forces the model to explore different feature combinations and reduces its reliance on dominant predictors, potentially improving performance by capturing different perspectives of the data. For regression models, each tree outputs a prediction by passing an observation through the tree to a leaf node and returning the average outcome of the training samples in that node. The final prediction from the random forest is the average of the predictions from all individual trees in the ensemble.

The calculations and outputs of the Random Forest Regressor include the model hyperparameters, feature importances, RMSE score, predictions, and R-squared value. This section discusses the predictions and R-squared score, with the remaining topics covered in the following sections. The vitals model achieved the lowest RMSE, so its predictions were plotted against the actual values to visually assess accuracy, as shown in *Figure 19*. The red dashed line represents the equation y = x, indicating the line of perfect prediction where predicted values exactly match actual values.

The scatter plot shows a high degree of variance between predicted and actual values, suggesting the model has a wide margin of error. However, this model still demonstrates predictive power because the predictions are not completely randomly scattered. Instead, the points show an upward trend, with predicted values increasing as the actual durations increase. This trend suggests that the model has learned a general relationship between the input features and the target variable because the predictions relatively align with the actual values.

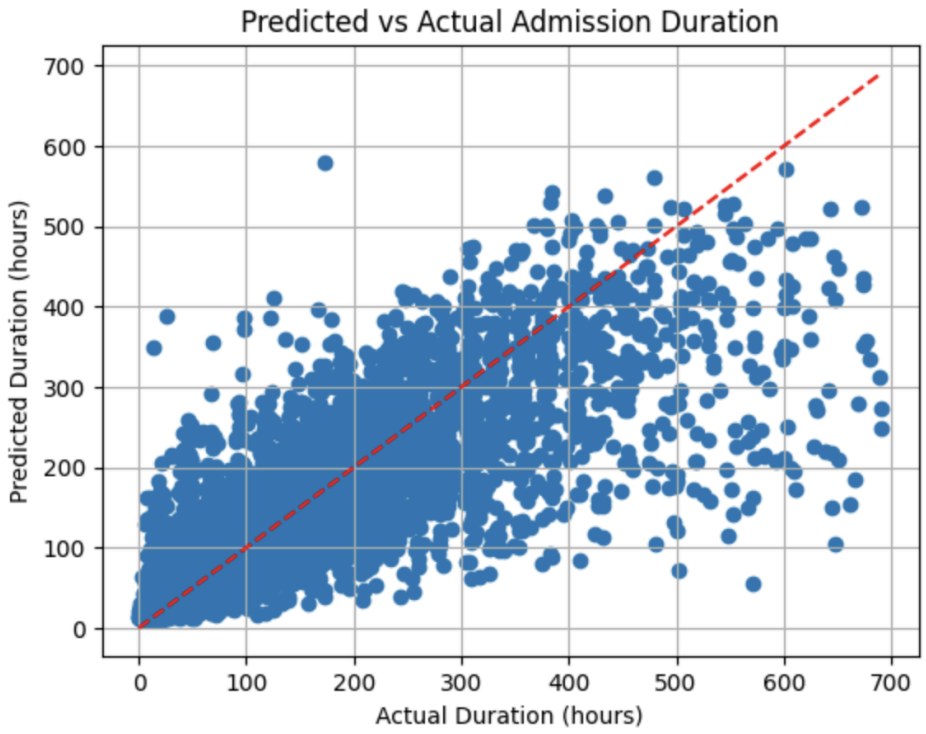


Figure 19: Scatter plot of predicted versus actual admission durations for the vitals model.

The R-squared score quantifies how well the features explain the variance of the target variable. It ranges from 0, meaning the features do not explain any of the variance, to 1, meaning the features perfectly explain it. A higher R-squared value indicates that the model captures more of the factors influencing admission duration, which increases confidence in the model's predictions and the identified important features. The R-squared scores were calculated for the test, train, and OOB sets to evaluate how well the model generalizes to unseen data, with these scores shown in *Figure 20*. If the training score is significantly better than the test score, then the model is potentially overfitting to noise in the training data and is not able to generalize well to unseen data. The OOB score serves as an internal validation set, and consistency between test and OOB scores suggests that the training and test sets were properly randomized and share

similar underlying distributions, supporting the reliability of the R-squared and RMSE performance metrics.

```
Base R^2 scores:
    Train: 0.89
    Test: 0.65
    OOB: 0.654
Vitals R^2 scores:
    Train: 0.95
    Test: 0.72
    OOB: 0.73
Blood Pressure R^2 scores:
    Train: 0.94
    Test: 0.66
    OOB: 0.66
Procedures R^2 scores:
    Train: 0.83
    Test: 0.49
    OOB: 0.48
```

Figure 20: The R-squared scores from the test, train, and OOB sets for each model.

A Random Forest Regressor was chosen to model admission duration because it can capture complex and nonlinear relationships between features and outcomes, which is crucial given the high variability in clinical data. This includes handling interactions between categorical and continuous features. Unlike linear regression models, which assume a linear relationship

between features and the target variable, Random Forest can model the more sophisticated interactions often present in healthcare data, leading to better predictive performance.

One of the advantages of random forests over alternative methods is that they have minimal assumptions about the data, offering greater flexibility. There are three key assumptions when using random forests: limiting irrelevant features, ensuring sufficient data, and understanding the model's extrapolation limitations. Since each tree in the forest is trained on a subset of features, too many irrelevant features can introduce noise and slow training [17]. However, a thorough feature selection process was performed to minimize this risk. Random forests also require a sufficiently large dataset because multiple trees are trained on different subsets of the data [18]. This is not a concern because the base DataFrame includes over 500,000 entries, and even the smallest subset contains approximately 50,000 entries. Lastly, like all tree-based models, random forests are limited to making predictions within the range of the training data and struggle to extrapolate beyond it [19]. This limitation is not problematic in this context because admission durations are expected to fall within the observed range.

A disadvantage of using random forests is that they are computationally intensive, especially with large datasets. The dataset used in this analysis is considered large because it contains over 500,000 entries. To utilize all available data, the training process becomes demanding because each tree is trained on a sizable subset. This can slow down the debugging phase because training can take several minutes per iteration.

GridSearchCV was used to tune the hyperparameters of the random forest. This technique builds a model for every possible combination of the specified parameters and performs cross-validation to identify the set that yields the best performance based on the selected performance metric. The parameters selected for optimization include: [20]

n_estimators: The number of decision trees in the forest. Too few estimators could prevent the model from capturing the complexity of the relationship between features and the target, resulting in underfitting. However, beyond a certain point, adding more trees does not significantly improve performance and only increases computational cost. The values tested were 100 and 300 to evaluate whether a substantial increase in the number of trees leads to a meaningful improvement in performance.

max_depth: The maximum number of layers each decision tree can have. If a tree is too shallow, the model could fail to capture complex patterns and will have limited predictive power. On the other hand, if the depth is too large, the model could overfit by learning noise rather than meaningful patterns, which can reduce performance on the test set [21]. The values tested were None and 20. Setting the max depth to None allows trees to grow without constraint, while a depth of 20 provides a generous limit that still allows the model to learn complex relationships.

min_samples_split: The minimum number of samples required for a node to split. When the sample reaches a node, it will only split if this threshold is met. If the value is too low, the model may split on noise, leading to overfitting. If it's too high, the tree may underfit because it cannot split enough to capture important distinctions, resulting in overly broad leaves and reduced performance. The values tested were 2 and 10. The default value of 2 allows greater flexibility in learning patterns, while 10 is more conservative and reduces the risk of overfitting [22].

min_samples_leaf: The minimum number of samples required in a leaf node after a split. If this threshold is too low, the model may overfit by creating leaves with very few samples that capture noise instead of meaningful patterns. If it's too high, the model may

underfit by forcing diverse samples into the same leaf, reducing purity and missing important distinctions [22]. The thresholds tested were 1 and 4. The default value of 1 allows the model to capture more detailed patterns, while a higher value of 4 provides a more conservative approach to reduce overfitting.

max_features: The exact number of features considered when splitting a node. The default value in newer versions of Scikit-Learn is 1.0, meaning all features will be considered at each node. This contradicts the fundamental nature of random forests because considering all features for splitting each node leads to less diverse trees that likely won't capture different perspectives of the data. Therefore, the maximum features for each model in the grid search was set to the square root of the total number of features to align with the goal of diversifying the ensemble. This threshold is a standard for random forests because it significantly limits the number of features each node considers, ensuring nodes will all have different subsets of features and increasing model diversity [23].

Cross-validation with three folds and negative mean squared error (MSE) scoring was used in the grid search. The training data was split into three subsets, with the model trained on two folds and validated on the third. This process was repeated so that each fold served as the validation set once. The validation scores were then averaged to produce an overall score for each model [24]. Negative MSE was chosen as the scoring method because this analysis prioritized the RMSE, which can be easily calculated by taking the square root of the MSE. Since cross-validation maximizes the scoring metric, and lower error is better, the MSE had to be negated. Using negative MSE ensures that the model with the lowest error receives the highest score during optimization.

The best hyperparameters identified by the GridSearchCV are as follows: max_depth of 20, min_samples_leaf of 1, min_samples_split of 10, and n_estimators of 300. All hyperparameters were updated from their default values, with the comparison shown in *Figure 21*. A max_depth of 20 and min_samples_split of 10 were more optimal than the default values, likely due to reducing overfitting. In contrast, a min_samples_leaf of 1 was a preferred low threshold that increased the granularity of the leaves, allowing the model to capture more detailed patterns. Increasing the number of estimators did decrease the RMSE, but comparing the RMSE of both models can determine how significant this improvement was. In *Figure 22*, the difference between the best and default RMSE was only 1 hour, so further tuning was not required because it was unlikely to lead to a meaningful improvement.

```
Best parameters: {'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 300}
Default parameters: {'max_depth': None, 'min_samples_split': 2, 'min_samples_leaf': 1, 'n_estimators': 100}
```

Figure 21: Comparison of best and default hyperparameters selected during tuning.

```
Best Model RMSE score: 51.88
Default RMSE score: 52.78
```

Figure 22: RMSE values of the best-tuned model versus the default model.

GridSearchCV was an appropriate method for tuning the random forest's hyperparameters because it provides a systematic approach to evaluating multiple parameter combinations. By searching through a predefined parameter space, it removes the need for manual tuning and helps identify the optimal configuration based on the cross-validation

performance. The use of cross-validation also reduces the risk of overfitting by assessing each combination across multiple subsets of the training data. This approach was suitable for the random forest model because it has several hyperparameters whose optimal values are difficult to determine theoretically.

One advantage of using GridSearchCV is that the optimal set of hyperparameters is guaranteed to be identified because each possible combination is tested. This approach ensures that performance will be improved, unless the default parameters already result in peak performance. Compared to randomized tuning methods, GridSearchCV provides a more reliable search of the parameter space.

A disadvantage of using GridSearchCV is that it can be extremely computationally intensive for random forests. Each forest consists of hundreds of trees, and these models will be trained on the number of cross-validation folds defined while training a model for each possible combination of hyperparameters. Moreover, training just one random forest can already be computationally expensive, so training multiple models is exhaustive. In this analysis, the GridSearchCV process took over five minutes to complete, making it an inconvenient limitation.

A feature importance analysis was conducted because it is an essential component to understanding which factors influence admission duration. Excluding this portion from the analysis would result in less informative actionable insights because the key factors would not be identified. Feature importances quantify the percentage of how often a feature was selected to split a node during training [25]. Since features are selected from random subsets at each split, a higher importance score indicates that the feature frequently provided the best split for predicting the target. Conversely, a low importance score suggests the feature was less useful for explaining the outcome.

To improve interpretability, the feature importances were sorted from greatest to least and plotted using horizontal bar graphs, as shown in *Figure 23*. Due to the randomness of random forests, features with similar feature importance scores sometimes swap places in the visualizations between model runs, but overall the models are quite stable. Since each model included over forty features, those with a feature importance below 0.01 were excluded from the visualizations to improve clarity by decluttering. This was an appropriate threshold because any feature importance below 0.01 was utilized less than 1% to optimally split a node during training, making them irrelevant for predicting admission duration.

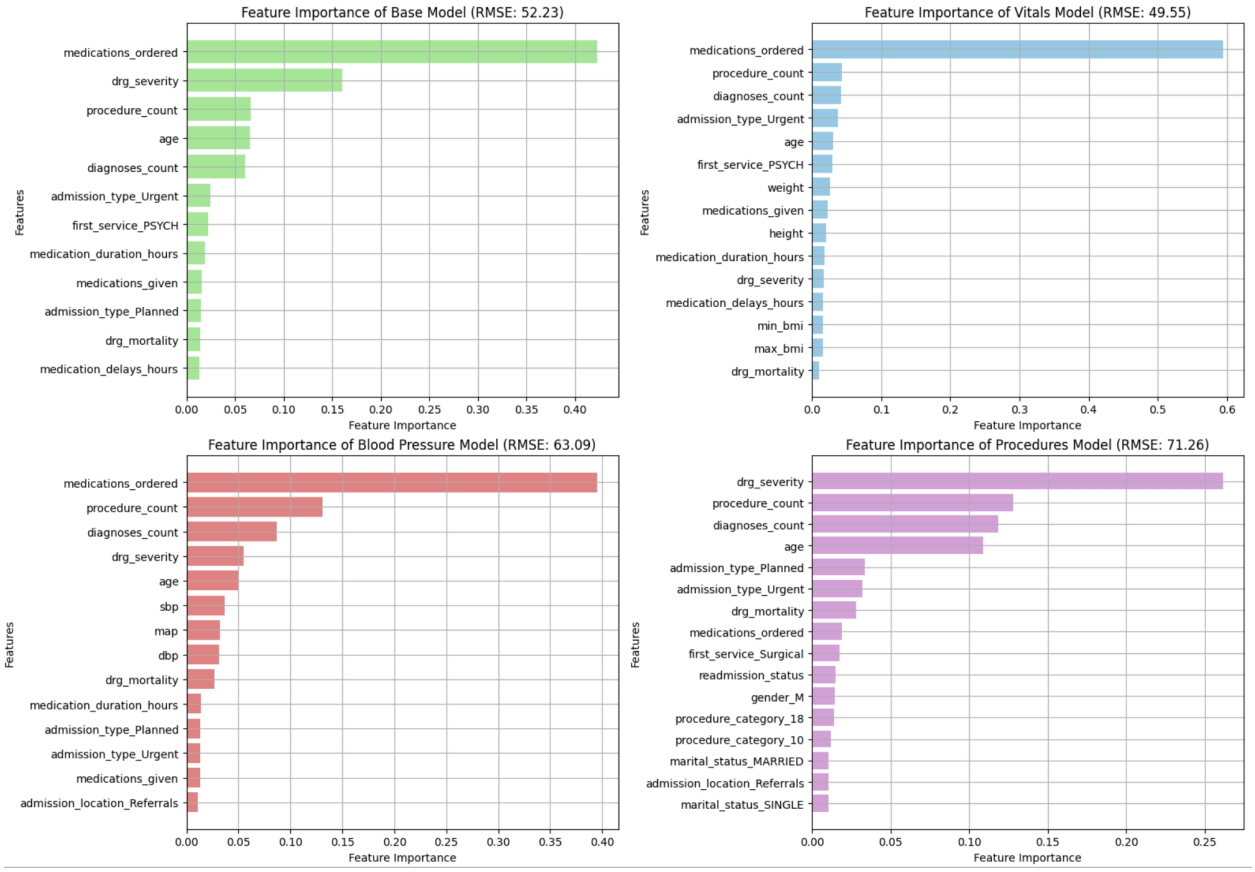


Figure 23: Comparison of feature importances across all evaluated random forest models.

The feature importance analysis was appropriate because it helps identify the key factors influencing admission duration, making the insights more actionable for potentially reducing hospital stays. One advantage of this analysis is that feature importances can be clearly visualized using horizontal bar graphs, making them easy to interpret even for non-technical audiences. This is valuable in healthcare settings because cross-functional teams can include clinical and administrative staff who need to collaborate but could struggle to interpret raw values.

A disadvantage of analyzing feature importances is that they do not explain how the features relate to admission duration, only that they have an impact. This is a limitation because the magnitude and direction of their relationship to the target are unknown. For example, knowing that 'patient age' is important does not reveal whether older patients tend to have longer or shorter stays. Therefore, additional analysis would be needed to explore these relationships further, which can require more time and resources.

The RMSE is an evaluation metric that quantifies the average error of a model. It is calculated by subtracting the actual values from the predictions to get the margin of error, squaring these residuals to penalize larger deviations and eliminate the effect of negative errors, and then taking the square root to bring the measurement back to the original unit [26]. RMSE scores were calculated for the train, test, and OOB sets of each model: base, vital signs, blood pressure, and procedures, with all scores shown in *Figure 24*.

```
Base RMSE scores:
    Train: 30.06
    Test: 52.24
    OOB: 52.39
Vitals RMSE scores:
    Train: 21.60
    Test: 49.55
    OOB: 49.69
Blood Pressure RMSE scores:
    Train: 27.65
    Test: 63.10
    OOB: 65.13
Procedures RMSE scores:
    Train: 39.62
    Test: 71.25
    OOB: 69.75
```

Figure 24: The RMSE scores for all models, including base, vital signs, blood pressure, and procedures.

The RMSE was appropriate as an evaluation metric because it serves as a standard margin of error for the models, penalizing larger units and encouraging more precise predictions. An advantage of using RMSE over other error metrics such as MSE is that it is more interpretable because the square root is taken to scale it back to the original units. Therefore, it has all the benefits of MSE while being more intuitive, which is important when explaining results to a non-technical audience, especially in a healthcare setting.

One disadvantage of using RMSE compared to some other error metrics is that it provides no information about the direction of the errors, such as whether the model tends to overpredict or underpredict. In a healthcare setting, this directional information can be important because overpredicting a patient's length of stay could lead to unnecessary room reservations and reduced hospital efficiency. In contrast, underpredicting could cause staff to begin discharge planning too early, leading to emergency department overcrowding, rushed care transitions, or incomplete treatment [27].

This limitation means that even with a low RMSE, the model could still be biased in one direction, and that bias would not be apparent from the RMSE value alone. Additional analysis would be required to understand the direction of the error, such as creating a residual plot of actual values against residuals. This would help identify whether the model tends to overpredict, which produces negative residuals, or underpredict, which produces positive residuals, across the range of the target variable.

This analysis utilized a Random Forest Regressor to predict hospital admission durations, GridSearchCV for hyperparameter tuning, feature importance analysis to identify key factors, and RMSE for performance evaluation. The Random Forest model was selected because it can capture non-linear relationships in complex healthcare data while requiring minimal assumptions about the data. Although computationally intensive, this approach identified several influential factors, such as medication delays and diagnosis severity.

The R-squared scores were calculated for the test, train, and OOB sets to evaluate how well the features explain the variance of the target on unseen data. The RMSE values offer an intuitive margin of error for hospital resource planning, though the analysis could be improved by exploring the direction of prediction errors to determine whether the models tend to

overpredict or underpredict the lengths of stay. Overall, these techniques were appropriate and essential to the analysis, leading to meaningful insights that can inform recommendations and contribute to improved patient outcomes.

Data Summary and Implications

The results of the data analysis, in relation to the research question, will cover the RMSE values for the test, train, and OOB sets across all models, the hypothesis, influential features, R-squared scores, and a comparison between the initial and reduced feature sets. In addition to summarizing the results, this section will also discuss the limitations of the analysis and provide recommendations for future work. The research question of this analysis is: "Can a random forest regression model accurately predict patient admission duration based on demographic data, clinical measurements, and admission information from a publicly available critical care database?" As a baseline level of performance, a simple model that predicts the average admission duration was used because it provides a rough but realistic estimate. The average admission duration in the base model's test set is approximately 90 hours, and the RMSE for this model is around 89 hours. This indicates a large margin of error of approximately 3.8 days.

The analysis showed that the vitals model achieved the lowest test RMSE among all the evaluated models, with a value of approximately 50 hours, as shown in *Figure 22*. This represents a 44% improvement over the baseline model, calculated as the difference between the two RMSE scores divided by the baseline RMSE. This reduction in error suggests that the model identified features influencing admission duration and learned underlying patterns in the data, significantly lowering the margin of error for predicting hospital admission duration.

The remaining models also outperformed the average-predicting model, as shown in Table 5. The base model achieved a test RMSE of approximately 52 hours, which is higher than the best model despite having the most available training samples compared to all the other models. It is possible that the additional features in the vital signs model helped to slightly improve the predictive power and outperform the base model. However, the base model did perform relative, being behind by only about 3 hours.

The blood pressure model achieved a test RMSE of approximately 63 hours, while the procedures model reached around 71 hours. Both performed better than the baseline model but substantially worse than the top two models, with RMSE values more than 10 hours higher. The additional features in these models also had low feature importance scores, with none exceeding the 5% threshold, and most procedure categories not even surpassing 1%. Therefore, these additional factors are likely not relevant for predicting admission duration.

Model	RMSE (hrs)	Improvement Over Baseline (91.80 hrs)
Base	52.23	42%
Vitals	49.55	44%
Blood Pressure	63.09	30%
Procedures	71.26	21%

Table 5

The train RMSE scores were all significantly lower for each model, with a gap of over 20 hours compared to the test set, with the vitals model achieving the lowest RMSE on the train set. The blood pressure model had the largest gap between train and test scores, likely because of its

higher overall RMSE. The procedures model had the lowest performance on both the train and test sets compared to the other models. Overall, the train RMSE scores showed significant deviation from the test set RMSE scores. This suggests that all the models were overfitting to noise in the training set, likely due to the large number of irrelevant features present in the initial set of predictors.

The OOB scores showed no significant deviation from the test set RMSE scores, with no difference exceeding 2 hours. This supports the validity of the test set results because it suggests that the training and test sets share similar underlying distributions. Since the OOB samples are subsets of the training data not used to train each tree, similar performance on the OOB and test sets indicates good generalization. The comparison of these scores is shown in Table 6.

Model	Train Set RMSE	Test Set RMSE	Difference Train vs. Test	OOB Set RMSE	Difference Test vs. OOB
Base	30.06	52.23	22.17	52.38	0.18
Vitals	21.60	49.55	27.95	49.69	0.13
Blood Pressure	27.65	63.09	35.44	65.13	1.95
Procedures	39.62	71.26	31.64	69.77	1.55

Table 6

The vitals model achieved the lowest test RMSE at approximately 50 hours, which significantly exceeds the 5-hour threshold set in the hypothesis. Therefore, we reject the alternative hypothesis that the random forest model can predict patient admission duration within

an RMSE of 5 hours and fail to reject the null hypothesis that it cannot. The original threshold was excessively ambitious, given that the standard deviation for admission duration is approximately 100 hours. This high variability suggests that achieving such a low margin of error would be difficult, regardless of model complexity or optimization [2].

Although the vitals model did not meet the ambitious 5-hour RMSE threshold, it achieved a reduction in error to approximately 50 hours, representing a 44% improvement compared to the baseline. This improvement has practical implications for patient care communication because clinicians could use the predictions to provide more realistic discharge timeline estimates when discussing care plans with patients and families, potentially improving satisfaction. In addition, the strong correlation between medication orders, diagnosis severity, and length of stay likely aligns with clinical intuition, but quantifying these relationships provides a stronger foundation for resource planning.

While the model's performance is not precise enough for hour-by-hour scheduling, it shows that machine learning approaches can extract meaningful patterns from existing clinical data to support actionable insights. Simply using the average admission duration, which had an RMSE of approximately 90 hours or about 3.8 days, would leave hospitals with greater uncertainty. In contrast, the vitals model reduces this uncertainty by nearly two days, helping hospitals better anticipate patient flow even with the current prediction limitations. Even though the original accuracy goal was not met, all models still substantially outperformed the baseline and significantly improved admission duration prediction.

The feature importance analysis identified the most influential features for predicting admission duration, with the top five features for each model shown in Table 7. Notably, each model relied on many of the same features, though their relative importance varied. In the base,

vitals, and blood pressure models, medications_ordered was the most influential feature, accounting for between 42% and 59% of total feature importance. This suggests that a patient's medication regimen is a strong indicator of their length of stay, likely because medication orders are correlated with the complexity of a patient's diagnosis.

The second most influential feature across models was drg_severity, which rates the severity of a diagnosis. It was the top feature in the procedures model, where it replaced medications_ordered and accounted for approximately 26% of the feature importance. Diagnoses_count, age, and procedure_count also showed consistent relevance, appearing in the top five features for all models. However, their importance was typically less than half of the top feature, indicating noticeably lower predictive power. The remaining features contributed significantly less, with importance values around 5% or lower, suggesting they had minimal individual influence on predicting admission duration.

Medications_ordered was not as influential in the procedures model compared to the other models. Additionally, the procedure categories were not helpful for predicting admission duration because each had a feature importance of approximately 2% or lower. One plausible explanation for why this model relied on different features is that it had the highest total number of features. The procedures model included 51 features, while the other models had only around 40. This increase in total features may have introduced additional noise, which could have reduced the number of times key features were selected during training to split a node.

Many of the procedure categories also had fewer than a thousand observations, which likely limited their predictive value and further diluted the impact of more informative features. However, they were still included because they are unique to the procedures model and were expected to be removed in the reduced feature set if not considered important. As a result, the

model may have relied on alternative features because the presence of weaker ones could have reduced the visibility of more meaningful features. This hypothesis is supported by the procedures model achieving the highest RMSE of 71 hours, which is over 20 hours higher than the best-performing model.

Models	Features	Feature Importances
Base	medications_ordered drg_severity procedure_count age diagnoses_count	0.42 0.16 0.066 0.065 0.061
Vitals	medications_ordered procedure_count diagnoses_count admission_type_Urgent age	0.59 0.043 0.042 0.038 0.031
Blood Pressure	medications_ordered procedure_count diagnoses_count drg_severity age	0.40 0.13 0.09 0.055 0.050
Procedures	drg_severity procedure_count diagnoses_count age admission_type_Planned	0.26 0.13 0.12 0.11 0.03

Table 7

The R-squared values for the test, train, and OOB sets across all the models are shown in *Figure 20*. The vitals model, which produced the lowest RMSE, also achieved the highest R-squared score at approximately 0.72. This means the features explained 72% of the variance in admission duration. Predicting admission duration is inherently challenging due to the influence of external factors not captured in clinical data, such as hospital congestion, varying patient recovery times, and unexpected complications during admission [28]. Given this complexity and high variability, a score of 0.72 is notable and indicates a strong relationship between the input features and the target variable.

In a similar study published in BMC Health Services Research, researchers achieved an R-squared score of 0.82 for predicting length of stay in newborns using linear regression [29]. In comparison, this analysis achieved a score of 0.72 for adult patients, demonstrating strong predictive performance, especially given the greater complexity and variability associated with adult admissions. The remaining R-squared scores for the other models ranged between 0.48 and 0.7, reflecting a moderate but still meaningful relationship between the features and admission duration.

The train set R-squared scores were substantially higher than the test and OOB set scores, with most closer to 0.9. The vitals model, which performed the best, also had the highest train R-squared score of approximately 0.95, while the procedures model, which performed the worst, had the lowest train R-squared score around 0.83. The significant difference in scores between the train and test sets further supports that all the models were overfitting to noise in the training set, with the lowest gap still moderately high at around 20%. Additionally, the OOB R-squared scores were nearly identical to their corresponding test scores, with differences under 1% for

each model. This consistency further supports that the training and test sets were properly randomized and share a similar distribution.

Each of the models showed signs of overfitting to noise in the training data when comparing the RMSE and R-squared scores between the train and test sets. Moreover, many features were considered unlikely to contribute meaningfully to predicting admission duration because they were rarely selected as optimal splits during training, indicated by their low feature importance scores. To reduce the initial feature set and minimize potential noise from weak predictors cluttering the feature space, a 1% feature importance threshold was applied, which removed approximately 75% of features and reduced each model's input to around 10 to 15 variables.

The comparison between the initial and reduced RMSE scores is shown in Table 8. In the initial feature set, there were clear signs of significant overfitting for each model, with the smallest gap between training and test sets exceeding 20 hours. However, in the reduced feature set, none of the gaps exceed 20 hours. Moreover, the increase in training RMSE did not worsen performance on the test sets, with no difference greater than 2 hours between the initial and reduced test sets for any model. As a result, it is likely that the irrelevant features were introducing significant noise, leading to the large gaps between training and testing performance in the initial set. While there is still a noticeable gap between the training and test sets in the reduced feature set, the feature reduction relied on an arbitrary but strong initial threshold, suggesting that this gap could likely be reduced further with a more targeted feature selection without hurting performance.

Model RMSE Scores	Initial Train Set	Initial Test Set	Absolute Difference	Reduced Train Set	Reduced Test Set	Absolute Difference
Base	30.06	52.23	22.17	41.72	53.1	11.38
Vitals	21.60	49.55	27.95	33.64	50.9	17.26
Blood Pressure	27.65	63.09	35.44	44.88	63.69	18.81
Procedures	39.62	71.26	31.64	53.58	72.75	19.17

Table 8

The comparison between the initial and reduced R-squared scores is shown in Table 9. The R-squared scores followed the same trend as the RMSE scores, with the difference between the training and test sets decreasing after the feature reduction. In the initial set, the vitals model achieved the highest R-squared score of 0.72 and had the lowest gap of 0.23, yet in the reduced set, even the worst-performing model achieved that same gap of 0.23. Overall, the gaps for each model decreased significantly without a drop in test R-squared scores, with the largest change between the initial and reduced test sets being less than 2%. Therefore, while the significant gaps in the initial feature set were clear signs of overfitting to the training data, the reduced feature set significantly decreased these gaps, supporting that the irrelevant features in the initial set were introducing noise.

Model R-squared Scores	Initial Train Set	Initial Test Set	Absolute Difference	Reduced Train Set	Reduced Test Set	Absolute Difference
Base	0.89	0.65	0.24	0.78	0.64	0.14
Vitals	0.95	0.72	0.23	0.87	0.71	0.16
Blood Pressure	0.94	0.66	0.28	0.84	0.66	0.18
Procedures	0.83	0.49	0.34	0.70	0.47	0.23

Table 9

The OOB RMSE and R-squared scores are shown in Table 10. They were excluded from the earlier tables to reduce clutter. The OOB scores remained consistent between the initial and reduced feature sets, with RMSE values within 2 hours and R-squared scores within 2%. Additionally, the OOB scores were similar to the corresponding test set scores, supporting that the training and test sets were properly randomized and share similar underlying distributions. Overall, the consistency of the OOB scores across both feature sets further reinforces the validity of the test set results.

Model	Initial RMSE OOB Set	Reduced RMSE OOB Set	Initial R-squared OOB Set	Reduced R-squared OOB Set
Base	52.39	53.14	0.654	0.64
Vitals	49.69	50.83	0.73	0.71
Blood Pressure	65.13	65.94	0.66	0.65
Procedures	69.75	70.84	0.48	0.47

Table 10

The updated feature importances are visualized in *Figure 25*. This continues the analysis of the reduced feature set by examining how the feature importances compare to the original models. Compared to the initial feature importances shown in *Figure 23*, there do not appear to be any major differences. All of the top features remained consistent across models, with only a slight reordering in the vitals model. These minor changes likely reflect random variation during training rather than the impact of removing features, especially since the rearranged features had very similar importance values. Reducing the feature set increased the relative importance of the more dominant features because removing the least influential ones allowed the model to select the remaining strong features more frequently during training. The most influential features remained consistent, mainly including medications_ordered, drg_severity, diagnoses_count, age, and procedure_count.

There also appears to be a pattern between model performance and the level of reliance on medications_ordered. The vitals model, which achieved the lowest RMSE of approximately 50 hours, assigned the highest importance to medications_ordered at 66%, while the remaining features were all around 5% or lower. The base model, which was the second-best performing, had an RMSE of around 53 hours and also had medications_ordered as the top feature, though with a lower importance of 48%. In this model, drg_severity followed at 18%, with the remaining features near or below 5%. The blood pressure model, which was the third-best performing with an RMSE of approximately 64 hours, showed a similar structure, with medications_ordered contributing 45% and the remaining features gradually decreasing in importance.

Finally, the procedures model, which had the highest RMSE at approximately 73 hours, showed the weakest reliance on medications_ordered. In this model, drg_severity was the most

important feature, while medications_ordered ranked seventh with an importance below 5%. This pattern suggests that models placing greater emphasis on medications_ordered tend to perform better. The consistency of this relationship across the initial and reduced feature sets highlights the strong predictive value of this feature. Therefore, future modeling approaches could benefit from a deeper focus on medication data because it appears to be a key factor influencing admission duration.

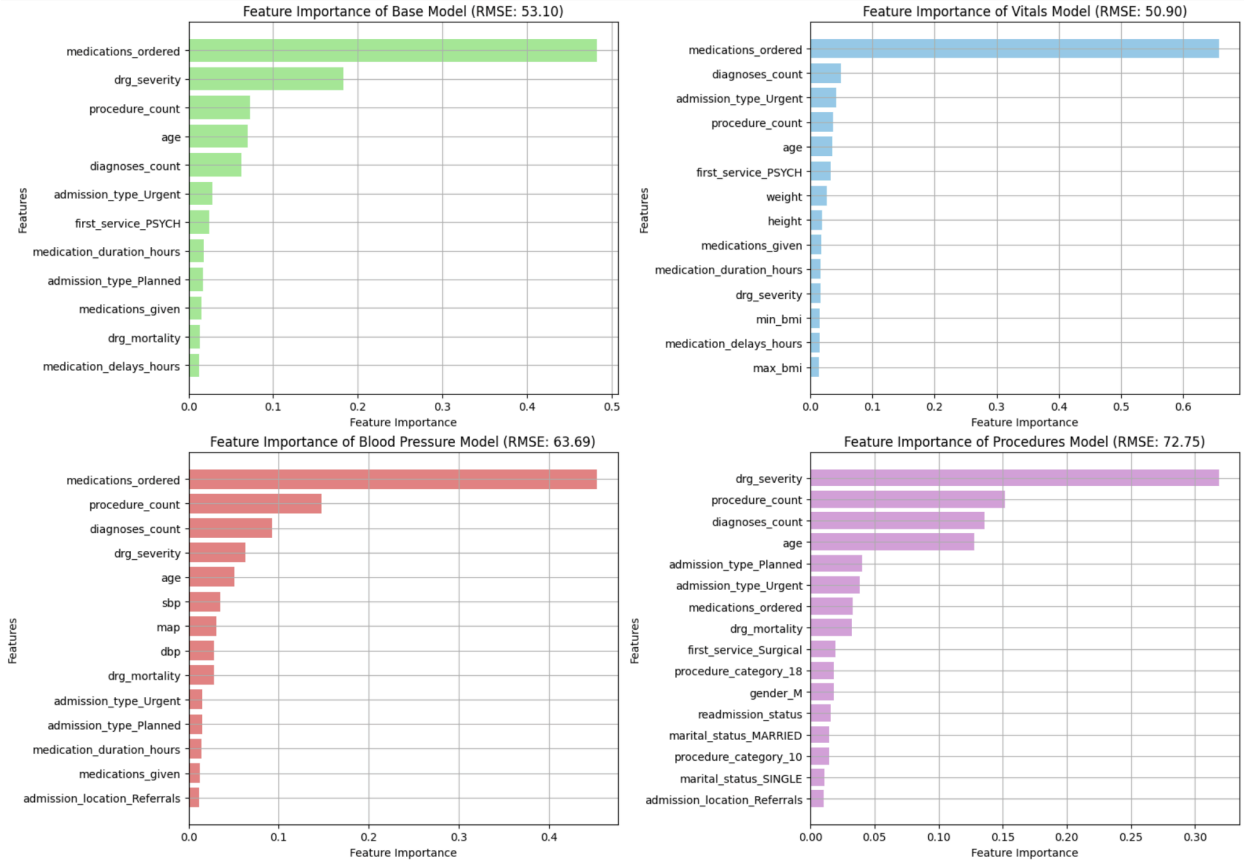


Figure 25: Updated comparison of feature importances across all random forest models using the reduced feature set.

The data analysis concluded with all models surpassing the baseline performance of a model that simply predicts the average admission duration. Their performance was measured using RMSE and R-squared metrics for the test, train, and OOB sets. The vitals model achieved the best performance with a 44% improvement and a test RMSE of approximately 50 hours. Despite this significant improvement, the analysis failed to reject the null hypothesis that a random forest model cannot predict admission duration within an RMSE of 5 hours.

Influential features were identified across all models, including medications_ordered, drg_severity, age, diagnoses_count, and procedure_count. These features were frequently ranked among the top five in each model. The vitals model also achieved the highest test R-squared score at around 0.72, meaning the features explained 72% of the variance in admission duration, which is notable given the high variability of the target. The remaining models achieved more moderate test R-squared scores between 0.48 and 0.66, still supporting the overall good performance and the relevance of the identified features.

Comparing the RMSE and R-squared scores between the train and test sets suggested that the models were overfitting to the training data, with large gaps in performance for each model. To reduce potential noise from irrelevant features cluttering the feature space, an arbitrary threshold of 1% was applied to feature importances, reducing the total number of features by around 75% for each model. After removing the rarely selected features that were likely not contributing meaningfully to predicting admission duration, the gap in performance between the train and test sets decreased significantly. While some overfitting remained, these gaps could likely be reduced further through more targeted feature selection to eliminate noise from the training data or by utilizing regularization techniques that penalize overly complex trees by limiting their growth through maximum depth and stricter node split conditions. The consistency

between the test and OOB scores supported that the training and test sets were properly randomized and shared similar distributions, validating the test set results.

There was also a consistent trend where models that prioritized medications_ordered more heavily tended to perform better. This was most evident in the procedures model, where medications_ordered ranked seventh with an importance below 5%, and the model had the worst performance with an RMSE of approximately 73 hours. Although the models performed well overall, it is important to acknowledge the limitations of this analysis and consider future directions based on these findings.

One limitation of this analysis is that the model was trained on data from a single hospital, which limits how well the findings can be generalized to other healthcare settings. Different hospitals have different patient populations, care protocols, resource availability, and operational factors that were not captured in this dataset. Because the data was mostly limited to demographic information, clinical measurements, and admission details, other variables like staffing, discharge planning, or hospital congestion were not available, even though they could have a real impact on admission duration.

This limitation is reflected in the best model's R-squared score of 0.72, meaning 28% of the variance is still unexplained. The target variable also had a large standard deviation of around 100 hours, which highlights how difficult this task is to predict accurately. While an RMSE of 50 hours was a strong improvement from the baseline, it still represents about 2.1 days, which is already a large margin of error. Therefore, predictions would likely get worse when applying the model to patients outside of this hospital. So even though the model performed well given the challenges, its usefulness in other settings is limited unless validated with data from multiple institutions.

Based on the results of the analysis, it is recommended that the hospital invest resources in improving data gathering and optimizing its database to better structure and organize clinical data. While the dataset was impressive in scope, with various tables and thousands of observations, it was not optimized for admission prediction. Over 60% of the analysis time was spent on data extraction and preparation. Many of the features had to be engineered, tested, and cleaned, and even the target variable itself did not exist in the original dataset.

For example, vital sign measurements were not tied to a specific admission and had to be meticulously connected using patient IDs and date timestamps, despite many of the values being recorded throughout a patient's hospital stay. These entries were also mixed with outpatient records, adding more complexity to the feature engineering. While these features were ultimately included, optimizing their structure could offer more precise insights, potentially capturing a greater portion of the unexplained variance and complexity associated with admission duration. Additionally, many features had to be excluded due to large amounts of missing or messy data, so improving data gathering practices would lead to a broader set of usable features.

This recommendation is realistic because the necessary data is technically available and simply requires the right investments to make it usable. This process would likely involve establishing better integration between systems such as vitals monitors and lab platforms, hiring database engineers to manage the architecture, and adopting standardized formats and terminologies to ensure high data quality and consistency. These improvements would not only benefit admission duration modeling but would also support future analyses that rely on clinical data, reducing long-term analytical costs while increasing insights.

From a legal and compliance standpoint, the Health Insurance Portability and Accountability Act (HIPAA) allows hospitals to utilize patient data without explicit authorization

when used for treatment, payment, or healthcare operations. This includes the data collection and management practices proposed in this recommendation [30]. Therefore, improving database infrastructure would not only support clinical research but also enhance HIPAA compliance and auditing capabilities.

As a result of the analysis, two directions for future study of the dataset could involve further exploring how medication regimens influence admission duration and combining the current dataset with additional institutional data. The feature importance analysis identified medications_ordered as a consistent top feature across the majority of models, and revealed a pattern where models tended to perform better, with lower RMSE and higher R-squared, when they focused on medications_ordered as a primary feature.

The dataset did include the specific medications that were administered to patients and prescribed after discharge. However, these medications were not included in this analysis because there were hundreds of individual medications, likely requiring aggregation to reduce high dimensionality. A future direction could be to explore how specific medications and their combinations correlate with admission duration because the analysis found a strong relationship between medication regimens and length of stay. This could involve conducting a market basket analysis to identify common medication combinations and examining whether certain medications or combinations are associated with shorter recovery times or longer hospital stays, potentially improving prescription practices and patient outcomes.

As previously mentioned, a limitation of the analysis is that the dataset comes exclusively from one hospital. Various factors such as different patient populations, care protocols, resource availability, and regional health trends between hospitals across the United States could lead to significant differences in admission durations. Currently, if another institution were to use this

model, its predictions would be limited, likely serving only as a benchmark for current performance.

Therefore, combining this dataset with data from other institutions could help develop a model that generalizes well across different healthcare settings and potentially identify universal factors that consistently impact admission duration regardless of the institution. Additionally, there should be a column identifying where each entry originates to create distinct validation sets for each institution, ensuring the model performs consistently across diverse healthcare environments. By investing in improved data infrastructure and exploring these research directions, the hospital could improve the accuracy of admission duration predictions, leading to more efficient resource allocation, improved patient care, and the development of insights that could benefit healthcare institutions nationwide.

Third-Party Code References

Pandas Developers, "pandas.DataFrame.duplicated," Pandas Documentation, 2025. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.duplicated.html [Accessed: Mar. 18, 2025].

Pandas Developers, "pandas.DataFrame.drop_duplicates," Pandas Documentation, 2025. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html#pandas.DataFrame.drop_duplicates [Accessed: Mar. 18, 2025].

GeeksforGeeks, "Python | Unpack a Dictionary," GeeksforGeeks, 2025. [Online]. Available: https://www.geeksforgeeks.org/python-unpack-dictionary/ [Accessed: Mar. 18, 2025].

Pandas Developers, "pandas.DataFrame.map," Pandas Documentation, 2025. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.map.html [Accessed: Mar. 18, 2025].

Pandas Developers, "pandas.DataFrame.fillna," Pandas Documentation, 2025. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.fillna.html [Accessed: Mar. 19, 2025].

Scikit-learn Developers, "OneHotEncoder — scikit-learn Documentation," Scikit-learn, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html [Accessed: Mar. 19, 2025].

GeeksforGeeks, "Machine Learning — One Hot Encoding," GeeksforGeeks, 2025. [Online]. Available: https://www.geeksforgeeks.org/ml-one-hot-encoding/ [Accessed: Mar. 20, 2025].

Matplotlib Developers, "matplotlib.pyplot.hist," Matplotlib Documentation, 2025. [Online]. Available: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html [Accessed: Mar. 20, 2025].

Scikit-learn Developers, "train_test_split — scikit-learn Documentation," Scikit-learn, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html [Accessed: Mar. 22, 2025].

Scikit-learn Developers, "Model Evaluation — scikit-learn Documentation," Scikit-learn, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html [Accessed: Mar. 23, 2025].

Matplotlib Developers, "matplotlib.pyplot.subplots," Matplotlib Documentation, 2025. [Online]. Available: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.subplots.html [Accessed: Mar. 23, 2025].

Matplotlib Developers, "matplotlib.pyplot.barh," Matplotlib Documentation, 2025. [Online]. Available: https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.barh.html [Accessed: Mar. 23, 2025].

Dataset Citation:

A. Johnson, L. Bulgarelli, T. Pollard, B. Gow, B. Moody, S. Horng, L. A. Celi, and R. Mark, "MIMIC-IV (version 3.1)," PhysioNet, 2024. [Online]. Available: https://doi.org/10.13026/kpb9-mt58 [Accessed: Mar. 20, 2025].

A. E. W. Johnson, L. Bulgarelli, L. Shen, et al., "MIMIC-IV, a freely accessible electronic health record dataset," Scientific Data, vol. 10, no. 1, 2023. [Online]. Available: https://doi.org/10.1038/s41597-022-01899-x [Accessed: Mar. 20, 2025].

A. Goldberger, L. Amaral, L. Glass, J. Hausdorff, P. C. Ivanov, R. Mark, et al., "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," Circulation, vol. 101, no. 23, pp. e215–e220, 2000. [Online]. Available: https://physionet.org/static/published-projects/pn2000/pn2000-1.0.0.pdf [Accessed: Mar. 20, 2025].

Reference List

[1] S. Dagar, S. Sahin, Y. Yilmaz, and U. Durak, "Emergency Department During Long Public Holidays," Turk J Emerg Med, vol. 14, no. 4, pp. 165–171, Mar. 2016. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC4909941/ [Accessed: Mar. 20, 2025].

[2] M. Rakrak, "Exploring Variability in Data: The Role of Range, Variance, and Standard Deviation," International Journal of Multidisciplinary Research and Analysis, vol. 4, no. 1, pp. 10–15, 2024. [Online]. Available: https://www.researchgate.net/profile/Mustapha-Rakrak/publication/390127595_INTERNATIONAL_JOURNAL_OF_MULTIDISCIPLINARY_RESEARCH_AND_ANALYSIS_Exploring_Variability_in_Data_The_Role_of_Range_Variance_and_Standard_Deviation/links/67e178c4e2c0ea36cd9bf07b/INTERNATIONAL-JOURNAL-OF-MULTIDISCIPLINARY-RESEARCH-AND-ANALYSIS-Exploring-Variability-in-Data-The-Role-of-Range-Variance-and-Standard-Deviation.pdf [Accessed: Mar. 20, 2025].

[3] B. Friedman, "The Measurement of MAP," Clinical View, 2017. [Online]. Available: https://clinicalview.gehealthcare.com/sites/default/files/39_Monitoring%20Solutions%20DINAMAP%20determines%20MAP_white_paper_JB53172XX_Nov17%20%281%29_0.pdf [Accessed: Mar. 21, 2025].

[4] AAPC, "ICD-9-CM Volume 3 Procedure Codes," AAPC, 2025. [Online]. Available: https://www.aapc.com/codes/icd9-codes-vol3-range/?srsltid=AfmBOoqs77ku_CgkfuAk6Zx8OeIHAM2zP2Z9QJiLMol2D_YX0-NbtM5S [Accessed: Mar. 21, 2025].

[5] S. Dietrich and E. Hernandez, "Nearly 68 Million People Spoke a Language Other Than English at Home in 2019," U.S. Census Bureau, Dec. 6, 2022. [Online]. Available:

https://www.census.gov/library/stories/2022/12/languages-we-speak-in-united-states.html [Accessed: Mar. 21, 2025].

[6] U.S. Census Bureau, "QuickFacts: United States," U.S. Census Bureau, 2024. [Online]. Available: https://www.census.gov/quickfacts/fact/table/US/PST045224 [Accessed: Mar. 22, 2025].

[7] T. Yoshizawa, K. Ishikawa, H. Nagasawa, I. Takeuchi, K. Jitsuiki, K. Omori, H. Ohsaka, and Y. Yanagawa, "A Fatal Case of Super-super Obesity (BMI >80) in a Patient with a Necrotic Soft Tissue Infection," Internal Medicine, vol. 57, no. 10, pp. 1479–1481, May 2018. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC5995720/ [Accessed: Mar. 22, 2025].

[8] M. Suszko, J. Sobocki, and C. Imieliński, "Mortality in Extremely Low BMI Anorexia Nervosa Patients – Implications of Gastrointestinal and Endocrine System Dysfunction," Psychiatria Polska, vol. 52, no. 1, pp. 143–157, 2018. [Online]. Available: https://www.psychiatriapolska.pl/pdf-126233-81794?filename=81794.pdf [Accessed: Mar. 22, 2025].

[9] Centers for Disease Control and Prevention (CDC), "CDC Growth Charts: United States," 2000. [Online]. Available: https://www.cdc.gov/growthcharts/cdc-charts.htm [Accessed: Mar. 22, 2025].

[10] P. K., "Weight Percentile Calculator for Men and Women in the United States," DQYDJ, 2025. [Online]. Available: https://dqydj.com/weight-percentile-calculator-men-women/ [Accessed: Mar. 24, 2025].

[11] Medscape, "CDC Weight-for-Age Percentiles for Boys (2-20 Years)," Medscape Reference, 2025. [Online]. Available:

https://reference.medscape.com/calculator/658/cdc-weight-for-age-percentiles-for-boys-2-20-years# [Accessed: Mar. 24, 2025].

[12] National Heart, Lung, and Blood Institute (NHLBI), "High Blood Pressure," National Heart, Lung, and Blood Institute, 2025. [Online]. Available: https://www.nhlbi.nih.gov/health/high-blood-pressure [Accessed: Mar. 24, 2025].

[13] Mayo Clinic Staff, "Low Blood Pressure (Hypotension) – Symptoms and Causes," Mayo Clinic, 2025. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/low-blood-pressure/symptoms-causes/syc-20355465 [Accessed: Mar. 24, 2025].

[14] Scikit-learn Developers, "OOB Errors for Random Forests," Scikit-learn Documentation, 2025. [Online]. Available: https://scikit-learn.org/stable/auto_examples/ensemble/plot_ensemble_oob.html [Accessed: Mar. 24, 2025].

[15] L. Albe, "Join Strategies and Performance in PostgreSQL," Cybertec, 2025. [Online]. Available: https://www.cybertec-postgresql.com/en/join-strategies-and-performance-in-postgresql/ [Accessed: Mar. 25, 2025].

[16] Scikit-learn Developers, "RandomForestRegressor — scikit-learn Documentation," Scikit-learn, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html [Accessed: Mar. 25, 2025].

[17] GeeksforGeeks, "Feature Selection Using Random Forest," GeeksforGeeks, 2023. [Online]. Available: https://www.geeksforgeeks.org/feature-selection-using-random-forest/ [Accessed: Mar. 30, 2025].

[18] S. Sruthi, "Random Forest Algorithm in Machine Learning," Analytics Vidhya, 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/ [Accessed: Mar. 30, 2025].

[19] T. Hengl, "Extrapolation is Tough for Trees: Tree-Based Learners - Combining Learners of Different Types Makes a Difference," Medium, 2021. [Online]. Available: https://medium.com/nerd-for-tech/extrapolation-is-tough-for-trees-tree-based-learners-combining-learners-of-different-type-makes-659187a6f58d#:~:text=So%20in%20summary%20we%20hope,including%20for%20the%20extrapolation%20space [Accessed: Mar. 30, 2025].

[20] Scikit-learn Developers, "GridSearchCV — scikit-learn Documentation," Scikit-learn, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html [Accessed: Mar. 31, 2025].

[21] GeeksforGeeks, "The Effects of the Depth and Number of Trees in a Random Forest," GeeksforGeeks, 2025. [Online]. Available: https://www.geeksforgeeks.org/the-effects-of-the-depth-and-number-of-trees-in-a-random-forest/ [Accessed: Mar. 31, 2025].

[22] Scikit-learn Developers, "Tree Metrics — Decision Trees Documentation," Scikit-learn, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/tree.html#tree-metrics [Accessed: Apr. 20, 2025].

[23] Scikit-learn Developers, "Random Forest Parameters — scikit-learn Documentation," Scikit-learn, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/ensemble.html#random-forest-parameters [Accessed: Apr. 20, 2025].

[24] Scikit-learn Developers, "Cross-Validation — scikit-learn Documentation," Scikit-learn, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation [Accessed: Apr. 20, 2025].

[25] Scikit-learn Developers, "Feature Importance — Random Forests Documentation," Scikit-learn, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/ensemble.html#forest [Accessed: Apr. 21, 2025].

[26] E. Garr, "3 Ways to Calculate the RMSE in Python," Discovery @ Illinois: Statistics with Python, 2025. [Online]. Available: https://discovery.cs.illinois.edu/guides/Statistics-with-Python/rmse/ [Accessed: Apr. 21, 2025].

[27] A. Naemi, T. Schmidt, M. Mansourvar, and et al., "Quantifying the impact of addressing data challenges in prediction of length of stay," BMC Medical Informatics and Decision Making, vol. 21, no. 298, 2021. [Online]. Available: https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-021-01660-1 [Accessed: Apr. 25, 2025].

[28] K. Stone, R. Zwiggelaar, P. Jones, and N. Mac Parthaláin, "A systematic review of the prediction of hospital length of stay: Towards a unified framework," PLOS Digital Health, vol. 1, no. 4, pp. e0000017, Apr. 2022. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC9931263/ [Accessed: Apr. 25, 2025].

[29] R. Jain, M. Singh, A. R. Rao, and et al., "Predicting hospital length of stay using machine learning on a large open health dataset," BMC Health Services Research, vol. 24, no. 860, 2024. [Online]. Available: https://bmchealthservres.biomedcentral.com/articles/10.1186/s12913-024-11238-y [Accessed: Apr. 27, 2025].

[30] U.S. Department of Health & Human Services (HHS), "Uses and Disclosures for Treatment, Payment, and Health Care Operations," HHS.gov, 2025. [Online]. Available: https://www.hhs.gov/hipaa/for-professionals/privacy/guidance/disclosures-treatment-payment-health-care-operations/index.html [Accessed: Apr. 27, 2025].