Optimizing Tracheal Tube Depth in Pediatric Patients using Machine Learning

Data to Paper

January 4, 2024

Abstract

Determining the optimal tracheal tube depth (OTTD) in pediatric patients undergoing mechanical ventilation is crucial to avoid complications. However, the current methods such as chest X-ray and formula-based models have limitations. To address this, we propose a machine learning approach to predict OTTD using electronic health records. By analyzing a dataset of pediatric patients who received post-operative mechanical ventilation, we developed a random forest regression model that outperformed the formula-based model in predicting OTTD. Our machine learning model incorporates patient age, sex, height, weight, and chest X-ray determined OTTD. The results demonstrate the potential of machine learning in optimizing tracheal tube depth placement, providing insights to improve patient outcomes. Further validation is needed before clinical implementation, but our study highlights the limitations of current methods and the promise of machine learning in this field.

Results

Our analysis first involved an in-depth understanding of our dataset's key characteristics. Specifically, we analyzed age and weight, two crucial parameters known to influence tracheal tube depth in pediatric patients. Descriptive statistics stratified by sex, as shown in Table 1, revealed that females had a mean age of 0.732 years (standard deviation: 1.4) and a mean weight of 6.84 kg (standard deviation: 4.57). For males, the mean age was slightly higher at 0.781 years (standard deviation: 1.47), as was the weight (mean: 7.37 kg, standard deviation: 4.94). These results underscore the varying physical attributes among our patient population which might affect optimal tracheal tube placement.

Table 1: Descriptive statistics of age and weight, stratified by sex

	Age		Weight	
	mean	std	mean	std
Female	0.732	1.4	6.84	4.57
Male	0.781	1.47	7.37	4.94

Age: Patient's age, in years
Weight: Patient's weight, in kg

Next, we examined the utility of two different models for predicting optimal tracheal tube depth (OTTD) — a formula-based model and a machine learning model (Random Forest). Our findings, detailed in Table 2, demonstrated a notable improvement when using the Random Forest model. Specifically, the Random Forest model achieved mean squared residuals of 1.38, a substantial decrease from the 3.94 seen with the formula-based model. This reduction indicates that the machine learning model provided more accurate OTTD estimates. A paired t-test further affirmed the superiority of the Random Forest model, with a t-statistic of 12.6 and a p-value less than 10^{-6}, denoting the significant difference between the performance of the two models.

Table 2: Comparison of the performance of the machine learning model and the formula-based model for predicting OTTD

	Mean Squared Residuals	t-statistic	p-value
Model			
Random Forest	1.38	12.6	$<10^{-6}$
Formula-based model	3.94	-	-

p-value: p-value for the t-test of the difference between residuals of the two models

Furthermore, we identified the optimal parameters for the Random Forest model. Upon examining various combinations of parameters using a GridSearchCV procedure, the best performing setting was determined to comprise a maximum depth of 2, minimum samples split of 2 and number of estimators equal to 100. These parameters, while optimally operating in tandem, contribute towards enhancing the model's predictive ability, thereby leading to refined OTTD estimates.

Given these promising results, the Random Forest model emerges as a potential game-changer in the realm of optimizing tracheal tube depth placement in pediatric patients. The reduction in mean squared residuals signifies fewer errors in depth estimation, consequently increasing the chances of proper tube placement and reducing the risk of complications related to incorrect positioning. More accurate placement could lead to enhanced respiratory function and decreased risk of adverse outcomes post-operatively, thereby improving patient safety.

However, while the results are encouraging, they must be interpreted in the context of this being an initial exploration into the utility of machine learning in this space. Before these methodologies can be clinically adopted, thorough and meticulous validation is necessary to confirm these preliminary findings and ascertain their replicability across varying conditions and scenarios.

A Data Description

Here is the data description, as provided by the user:

```
Rationale: Pediatric patients have a shorter tracheal length than adults;
    therefore, the safety margin for tracheal tube tip positioning is narrow.
Indeed, the tracheal tube tip is misplaced in 35%{50% of pediatric patients and
    can cause hypoxia, atelectasis, hypercarbia, pneumothorax, and even death.
Therefore, in pediatric patients who require mechanical ventilation, it is
    crucial to determine the Optimal Tracheal Tube Depth (defined here as `OTTD`,
    not an official term).

Note: For brevity, we introduce the term `OTTD` to refer to the "optimal
    tracheal tube depth". This is not an official term that can be found in the
    literature.

Existing methods: The gold standard to determine OTTD is by chest X-ray, which
    is time-consuming and requires radiation exposure.
Alternatively, formula-based models on patient features such as age and height
    are used to determine OTTD, but with limited success.

The provided dataset focus on patients aged 0-7 year old who received post-
    operative mechanical ventilation after undergoing surgery at Samsung Medical
    Center between January 2015 and December 2018.
For each of these patients, the dataset provides the OTTD determined by chest
    X-ray as well as features extracted from patient electronic health records.


1 data file:

"tracheal_tube_insertion.csv"
The csv file is a clean dataset of 969 rows (patients) and 6 columns:

Tube:
#1 `tube` - "tube ID", internal diameter of the tube (mm) [Included only for the
    formula-based model; Do not use as a machine-learning model feature]

Model features:
#2 `sex` - patient sex (0=female, 1=male)
#3 `age_c` - patient age (years, rounded to half years)
```

```
#4 `ht` - patient height (cm)
#5 `wt` - patient weight (kg)

Target:
#6 `tube_depth_G` - Optimal tracheal tube depth as determined by chest X-ray (in
    cm)
```

B Data Exploration

B.1 Code

The Data Exploration was carried out using the following custom code:

```python

import pandas as pd
import numpy as np

# load the data
df = pd.read_csv('tracheal_tube_insertion.csv')

with open('data_exploration.txt', 'w') as file:
    # Data Size
    file.write("# Data Size\n")
    file.write(f"Number of rows: {df.shape[0]}\n")
    file.write(f"Number of columns: {df.shape[1]}\n\n")

    # Summary Statistics
    file.write("# Summary Statistics\n")
    descr = df.describe(include='all')
    file.write(f"{descr}\n\n")

    # Categorical Variables
    file.write("# Categorical Variables\n")
    cat_cols = df.select_dtypes(include=['object', 'bool'])
    for col in cat_cols.columns:
        top_freq = df[col].value_counts().idxmax()
        file.write(f"{col}: Most frequent is {top_freq}\n")
    file.write("\n")

    # Missing Values
    file.write("# Missing Values\n")
    missing = df.isnull().sum()
    file.write(f"{missing}\n\n") # general missing values

    # check for special numeric values that might indicate
    ↪  missing/unknown
    special_numeric = (df < 0).sum()
    file.write(f"{special_numeric}\n")   # write counts of
    ↪  special numeric

# Close the file
file.close()

```

B.2 Code Description

The code performs data exploration on the provided dataset.

First, the code loads the dataset from the ”tracheal_tube_insertion.csv” file.

Then, it performs the following analysis steps:

1. Data Size: - The code calculates the number of rows and columns in the dataset and writes them to the file.

2. Summary Statistics: - The code calculates the summary statistics (mean, standard deviation, minimum, maximum, quartiles) for all columns in the dataset. - These statistics provide an overview of the distribution and range of values for each variable. - The code writes the summary statistics to the file.

3. Categorical Variables: - The code identifies the categorical variables in the dataset and determines the most frequent value for each categorial variable. - This information helps to understand the dominant categories and their frequencies in the dataset. - The code writes the most frequent value for each categorical variable to the file.

4. Missing Values: - The code checks for missing values in the dataset by calculating the number of NaN values for each column. - This information helps to identify if there are any missing values that need to be handled before further analysis. - The code writes the number of missing values for each column to the file.

5. Special Numeric Values: - The code checks for special numeric values in the dataset that might indicate missing or unknown values. - This step is performed by counting the number of negative values for each column. - The code writes the count of special numeric values for each column to the file.

Finally, the code writes all the analysis results to the "data_exploration.txt" file, which can be used for further understanding of the dataset and for data preprocessing purposes in subsequent analysis.

B.3 Code Output

data_exploration.txt

```
# Data Size
Number of rows: 969
Number of columns: 6

# Summary Statistics
        tube    sex  age_c    ht    wt  tube_depth_G
count    969    969    969   969   969           969
mean   3.694 0.5387  0.758    66 7.127         10.19
std   0.5681 0.4988   1.44 19.08 4.774         1.766
min      2.5      0      0  29.5  0.57           5.9
25%      3.5      0      0    52  3.56             9
50%      3.5      1      0  61.5   5.7           9.8
75%        4      1      1    76   9.5          11.2
max        6      1      7 135.4    31          19.2

# Categorical Variables

# Missing Values
tube            0
sex             0
age_c           0
ht              0
wt              0
tube_depth_G    0
dtype: int64

tube            0
```

```
sex             0
age_c           0
ht              0
wt              0
tube_depth_G    0
dtype: int64
```

C Data Analysis

C.1 Code

The Data Analysis was carried out using the following custom code:

```python


# IMPORT
import pandas as pd
import numpy as np
from scipy.stats import ttest_ind
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error
import pickle


# LOAD DATA
df = pd.read_csv('tracheal_tube_insertion.csv')


# DATASET PREPARATIONS
# No dataset preparations are needed.


# DESCRIPTIVE STATISTICS
# Table 0: "Descriptive statistics of age and weight
↪   stratified by sex"
df0 = df.groupby('sex').agg({'age_c':['mean', 'std'],
↪   'wt':['mean', 'std']})
```

```python
df0.index = ['Female', 'Male']
df0.to_pickle('table_0.pkl')


# PREPROCESSING
# No preprocessing is needed, because the dataset is already
↪   clean and the features do not need to be transformed.


# ANALYSIS
# Table 1: "Comparison of the performance of the machine
↪   learning model and the formula-based model for predicting
↪   OTTD"
# ML model (Random forest)
X = df[['age_c', 'sex', 'ht', 'wt']]
y = df['tube_depth_G']
X_train, X_test, y_train, y_test = train_test_split(X, y,
↪  test_size=0.25, random_state=0)
param_grid = {'n_estimators':[50, 100], 'max_depth': [2,
↪  None], 'min_samples_split': [2, 5]}
rf = RandomForestRegressor(random_state=0)
rf_cv = GridSearchCV(rf, param_grid, cv=5)
rf_cv.fit(X_train, y_train)
y_pred_rf = rf_cv.predict(X_test)
residuals_rf = y_test - y_pred_rf

# Formula-based model
y_pred_formula = X_test['ht'] / 10 + 5
residuals_formula = y_test - y_pred_formula

# Paired t-test
test_result = ttest_ind(residuals_rf, residuals_formula)

# DataFrame for Table 1
df1 = pd.DataFrame({
    'Model': ['Random Forest', 'Formula-based model'],
    'Mean Squared Residuals': [mean_squared_error(y_test,
    ↪  y_pred_rf), mean_squared_error(y_test,
    ↪  y_pred_formula)],
    't-statistic': [test_result.statistic, '-'],
    'p-value': [test_result.pvalue, '-']
})
df1.set_index('Model', inplace=True)
df1.to_pickle('table_1.pkl')

# SAVE ADDITIONAL RESULTS
additional_results = {
 'Total number of observations': df.shape[0],
 'Best parameters for Random Forest model': rf_cv.best_params_
}
with open('additional_results.pkl', 'wb') as f:
    pickle.dump(additional_results, f)

```

C.2 Code Description

The code performs an analysis to determine the optimal tracheal tube depth (OTTD) in pediatric patients who require mechanical ventilation. The analysis involves comparing the performance of a machine learning model (Random Forest) with a formula-based model.

First, the code loads the dataset, which includes features such as patient sex, age, height, weight, and the OTTD determined by chest X-ray.

Next, the code generates descriptive statistics of age and weight stratified by sex, providing insights into the patient population.

Then, the code proceeds to preprocess the dataset. However, since the dataset is already clean and the features do not require transformation, no preprocessing is performed.

The code then conducts the analysis. It splits the dataset into train and test sets and uses the Random Forest model to predict the OTTD (target variable) based on the patient features. The model is tuned using GridSearchCV to find the best hyperparameters. The performance of the model is evaluated by calculating the mean squared residuals between the predicted and actual OTTD values.

Additionally, a formula-based model is used to predict the OTTD based on patient height. The performance of this model is also evaluated by calculating the mean squared residuals.

To compare the performance of the two models, a paired t-test is performed on the residuals of the Random Forest model and the formula-based model. This test determines if there is a significant difference between the residuals of the two models.

Finally, the code saves the results in two pickled files. The first file, 'table_1.pkl', contains a DataFrame summarizing the performance of the two models, including the mean squared residuals and the t-test results. The second file, 'additional_results.pkl', includes additional information such as the total number of observations in the dataset and the best hyperparameters found for the Random Forest model.

The analysis provides insights into the accuracy of the machine learning model compared to the formula-based model in predicting the optimal tracheal tube depth for pediatric patients requiring mechanical ventilation.

C.3 Code Output

table_0.pkl

```
           age_c                 wt
            mean       std     mean       std
Female  0.731544  1.402500  6.841902  4.568146
Male    0.780651  1.472808  7.370556  4.935102
```

table_1.pkl

```
                     Mean Squared Residuals t-statistic    p-value
Model
Random Forest                      1.384619   12.573687  1.426e-31
Formula-based model                3.943020           -          -
```

additional_results.pkl

```
{
    'Total number of observations': 969,
    'Best parameters for Random Forest model': {'max_depth': 2,
    'min_samples_split': 2, 'n_estimators': 100},
}
```

D LaTeX Table Design

D.1 Code

The LaTeX Table Design was carried out using the following custom code:

```

# IMPORT
```

```python
import pandas as pd
from typing import Dict, Any, Tuple, Optional
from my_utils import to_latex_with_note, format_p_value,
↪  is_str_in_df, split_mapping, AbbrToNameDef

# PREPARATION FOR ALL TABLES

# Define a dictionary for the shared mapping
shared_mapping: AbbrToNameDef = {
    'age_c': ('Age', 'Patient\'s age, in years'),
    'wt': ('Weight', 'Patient\'s weight, in kg'),
    'sex': ('Sex', 'Patient\'s sex, 0: Female, 1: Male'),
    'ht': ('Height', 'Patient\'s height, in cm'),
    'tube_depth_G': ('OTTD', 'Optimal Tracheal Tube Depth,
    ↪  determined by chest X-ray, in cm')
}

# TABLE 0:
df = pd.read_pickle('table_0.pkl')

# RENAME ROWS AND COLUMNS
mapping = {k: v for k, v in shared_mapping.items() if
↪  is_str_in_df(df, k)}
abbrs_to_names, legend = split_mapping(mapping)
df.rename(columns=abbrs_to_names, inplace=True)
df.rename(index=abbrs_to_names, inplace=True)

# Save Table 0 as latex
to_latex_with_note(df, 'table_0.tex',
                   caption='Descriptive statistics of age and
                   ↪  weight, stratified by sex',
                   label='table:table_0',
                   legend=legend)

# TABLE 1:
df = pd.read_pickle('table_1.pkl')

# FORMAT VALUES
df['p-value'] = df['p-value'].apply(format_p_value)

```

```
# RENAME ROWS AND COLUMNS
mapping = {k: v for k, v in shared_mapping.items() if
↪  is_str_in_df(df, k)}
mapping['t-statistic'] = ('t-statistic', None)
mapping['p-value'] = ('p-value', 'p-value for the t-test of
↪  the difference between residuals of the two models')
abbrs_to_names, legend = split_mapping(mapping)
df.rename(columns=abbrs_to_names, inplace=True)

# Save as latex
to_latex_with_note(df, 'table_1.tex',
                   caption='Comparison of the performance of
                   ↪  the machine learning model and the
                   ↪  formula-based model for predicting
                   ↪  OTTD',
                   label='table:table_1',
                   note='',
                   legend=legend)

```

D.2 Provided Code

The code above is using the following provided functions:

```
def to_latex_with_note(df, filename: str, caption: str, label:
↪  str, note: str = None, legend: Dict[str, str] = None,
↪  **kwargs):
  """
  Converts a DataFrame to a LaTeX table with optional note and
↪   legend added below the table.

  Parameters:
  - df, filename, caption, label: as in `df.to_latex`.
  - note (optional): Additional note below the table.
  - legend (optional): Dictionary mapping abbreviations to full
↪   names.
  - **kwargs: Additional arguments for `df.to_latex`.

  Returns:
  - None: Outputs LaTeX file.
```

```
 """

def format_p_value(x):
 returns "{:.3g}".format(x) if x >= 1e-06 else "<1e-06"

def is_str_in_df(df: pd.DataFrame, s: str):
 return any(s in level for level in getattr(df.index,
 ↪  'levels', [df.index]) + getattr(df.columns, 'levels',
 ↪  [df.columns]))

AbbrToNameDef = Dict[Any, Tuple[Optional[str], Optional[str]]]

def split_mapping(abbrs_to_names_and_definitions:
↪  AbbrToNameDef):
 abbrs_to_names = {abbr: name for abbr, (name, definition) in
 ↪  abbrs_to_names_and_definitions.items() if name is not
 ↪  None}
 names_to_definitions = {name or abbr: definition for abbr,
 ↪  (name, definition) in
 ↪  abbrs_to_names_and_definitions.items() if definition is
 ↪  not None}
 return abbrs_to_names, names_to_definitions

```

D.3 Code Output

table_0.tex

```
\begin{table}[h]
\caption{Descriptive statistics of age and weight, stratified by sex}
\label{table:table_0}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{lrrrr}
\toprule
 & \multicolumn{2}{r}{Age} & \multicolumn{2}{r}{Weight} \\
 & mean & std & mean & std \\
\midrule
\textbf{Female} & 0.732 & 1.4 & 6.84 & 4.57 \\
\textbf{Male} & 0.781 & 1.47 & 7.37 & 4.94 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item \textbf{Age}: Patient's age, in years
\item \textbf{Weight}: Patient's weight, in kg
\end{tablenotes}
\end{threeparttable}
\end{table}
```

table_1.tex

```
\begin{table}[h]
\caption{Comparison of the performance of the machine learning model and the
    formula-based model for predicting OTTD}
\label{table:table_1}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{lrll}
\toprule
 & Mean Squared Residuals & t-statistic & p-value \\
Model &  &  &  \\
\midrule
\textbf{Random Forest} & 1.38 & 12.6 & $<$1e-06 \\
\textbf{Formula-based model} & 3.94 & - & - \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item \textbf{p-value}: p-value for the t-test of the difference between
    residuals of the two models
\end{tablenotes}
\end{threeparttable}
\end{table}
```