Enhancing Tracheal Tube Placement in Pediatric Patients: A Machine Learning Approach

Data to Paper

January 4, 2024

Abstract

Accurate positioning of tracheal tubes is critical for pediatric patients undergoing mechanical ventilation. However, determining the optimal tracheal tube depth (OTTD) in this population remains challenging. Existing methods rely on chest X-rays or formula-based models with limited success. To address this gap, we present a machine learning approach to predict OTTD in pediatric patients. Utilizing a dataset of 969 patients who underwent post-operative mechanical ventilation, our random forest regression model outperforms conventional formula-based models in predicting OTTD. Our analysis reveals strong correlations between patient features and OTTD, highlighting the importance of a personalized approach. Importantly, the machine learning model demonstrates greater precision with significantly smaller residuals compared to conventional models. Although chest X-rays were used and older patients were excluded in this study, our findings have substantial implications for improving the safety and efficacy of tracheal tube placement in pediatric patients. This work represents a significant step towards enhancing clinical outcomes in this population.

Results

To evaluate the distribution of relevant variables in our dataset, we stratified the patient features and the Optimal Tracheal Tube Depth (OTTD) by sex. Our dataset composed of approximately an equal number of male and female patients. Table 1 displays the mean and standard deviation of tube size, age, height, weight, and OTTD for both genders. The distribution exhibits no significant differences in the key features and OTTD between the male and female populations.

Building on this, we created a machine-learning model for predicting the OTTD in pediatric patients. We trained a Random Forest regression

Table 1: Mean and standard deviation of Optimal Tube Depth based on Sex

sex	female	male
tube mean	3.68	3.7
tube std	0.552	0.582
Mean Age	0.732	0.781
Age SD	1.4	1.47
ht mean	65.4	66.5
ht std	18.7	19.4
wt mean	6.84	7.37
wt std	4.57	4.94
Tube Mean Depth	10.1	10.3
Tube Depth SD	1.65	1.86

Tube Mean Depth: cm via X-ray
Tube Depth SD: cm via X-ray
Mean Age: Years, rounded
Age SD: Years, rounded

model with 4 predictors (patient sex, age, height, weight) utilizing a dataset comprising 969 observations. Importantly, the model considers tube placement as a function of patient's characteristics and excludes tube diameter, a feature typically included in traditional formula-based models.

To assess the predictive accuracy of our model, we compared the residuals of our machine learning model to those of the conventional formula-based model. The Machine Learning model recorded a mean residual of 0.03 with a standard deviation of 1.19, indicating its greater precision. Conversely, the formula-based model exhibited a larger mean residual and higher standard deviation, as shown in Table 2. The observed P-value, less than 10^{-6}, affirms the statistical significance of the difference between these residuals.

Table 2: Comparison of residuals between the Machine Learning model and Formula-based model

	mean	std	P-value
Random Forest Model	0.0328	1.19	$<10^{-6}$
Height Formula Model	-1.36	1.26	$<10^{-6}$

The residuals from the ML model and the formula-based model are compared using t-test. The displayed P-value is for the t-test comparison.

In summary, these results demonstrate that the machine learning model offers enhanced prediction accuracy for OTTD in pediatric patients. The superior performance of our model is demonstrated by the significantly smaller residuals in comparison to the traditional approach. Specifically, the model's predictive capability is not biased towards any gender group, reinforcing its utility in diverse patient demographics.

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:

```

import pandas as pd

# Load the data
dataset = pd.read_csv('tracheal_tube_insertion.csv')

# Open the output file
with open("data_exploration.txt", "w") as file:

    # Data size
    file.write("# Data Size\n")
    file.write(f"Number of rows: {dataset.shape[0]}\n")
    file.write(f"Number of columns: {dataset.shape[1]}\n\n")

    # Summary Statistics
    file.write("# Summary Statistics\n")
    stats = dataset.describe(include='all')
    file.write(str(stats) + '\n\n')

    # Categorical Variables
    file.write("# Categorical Variables\n")
    categorical_cols =
    ↪  dataset.select_dtypes(['object','bool']).columns
    for col in categorical_cols:
        file.write(f"{col} most common values:\n")
        file.write(str(dataset[col].value_counts().head()) +
         ↪  "\n\n")

    # Missing values
    file.write("# Missing Values\n")
    missing_vals = dataset.isnull().sum()
    file.write(str(missing_vals) + "\n")

    # use replace to count special numeric values that stand
     ↪  for unknown/undefined if any
    # For the current dataset, it does not seem to apply.

```

B.2 Code Description

The provided code performs data exploration on the "tracheal_tube_insertion.csv" dataset. It retrieves information about the dataset, computes summary statistics, examines categorical variables, and identifies missing values.

First, the code loads the dataset into a pandas DataFrame using the 'read_csv' function.

Next, it opens the output file "data_exploration.txt" in write mode to store the results of the data exploration.

The code then proceeds with the data exploration process:

1. Data Size: It obtains the number of rows and columns in the dataset using the 'shape' attribute of the DataFrame and writes this information to the output file.

2. Summary Statistics: The code calculates the summary statistics of the dataset using the 'describe' method of the DataFrame with the 'include' parameter set to 'all'. This includes statistics such as count, mean, standard deviation, minimum, quartiles, and maximum for both numeric and categorical columns. The resulting summary statistics are written to the output file.

3. Categorical Variables: The code identifies the categorical variables in the dataset by selecting columns with data types 'object' and 'bool'. For each categorical variable, it determines the most common values by using the 'value_counts' method and selects the top values using the 'head' function. The resulting information about the most common values for each categorical variable is written to the output file.

4. Missing Values: The code calculates the number of missing values

in each column of the dataset using the 'isnull' method of the DataFrame, which returns a boolean DataFrame indicating whether each value is missing or not. The 'sum' function is then used to count the number of missing values in each column. The resulting counts of missing values per column are written to the output file.

Finally, if applicable, the code could include a step to replace special numeric values that stand for unknown/undefined values and count these values. However, for the current dataset, this step is not necessary.

The code writes all the obtained information, such as data size, summary statistics, most common values for categorical variables, and the counts of missing values, into the "data_exploration.txt" file, providing a comprehensive exploration of the dataset.

This data exploration process helps to gain a better understanding of the dataset, identify any potential data quality issues, and inform further analysis or preprocessing steps in the research study.

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
```

C Data Analysis

C.1 Code

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

```python

# IMPORT
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from scipy.stats import ttest_rel
import pickle

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

# DATASET PREPARATIONS
## No dataset preparations are needed.

# DESCRIPTIVE STATISTICS
## Table 0: "Mean and standard deviation of tube, age, height,
↪  weight, and tube_depth_G stratified by sex"
df0 = df.groupby('sex').aggregate(['mean',
↪  'std']).rename(index={0: 'female', 1: 'male'})
df0.columns = [' '.join(col).strip() for col in
↪  df0.columns.values]
df0.index.name='sex'
df0.to_pickle('table_0.pkl')

# PREPROCESSING
## Creating dummy variables for sex
df = pd.get_dummies(df, columns=['sex'], drop_first=True)
```

```

# ANALYSIS
## Create a new column for the formula-based model
df['ht_form'] = df['ht'] / 10 + 5

## Split data into train and test sets
X_train, X_test, y_train, y_test =
↪  train_test_split(df.drop(['tube_depth_G', 'ht_form'],
↪  axis=1), df['tube_depth_G'], test_size=0.2,
↪  random_state=42)

## Initialize the model
rf_model = RandomForestRegressor(n_estimators=200,
↪  max_depth=5, random_state=42)

## Train the model
rf_model.fit(X_train, y_train)

## Making predictions
rf_predictions = rf_model.predict(X_test)
ht_predictions = X_test.ht / 10 + 5

## Comparing ML model residuals and formula-based model
↪  residuals
rf_res = y_test - rf_predictions
ht_res = y_test - ht_predictions

## Table 1: "Comparison of residuals between the ML model and
↪  formula-based model"
df_res = pd.DataFrame( {'RF_model' : rf_res , 'Height_Formula'
↪  : ht_res})
df1 = df_res.aggregate(['mean', 'std']).T
df1['p_val'] = [ttest_rel(rf_res, ht_res).pvalue]*len(df1)
df1.to_pickle('table_1.pkl')

# SAVE ADDITIONAL RESULTS
additional_results = {
 'Total number of observations': df.shape[0],
 'Test sample size': X_test.shape[0],
}
```

```

with open('additional_results.pkl', 'wb') as f:
 pickle.dump(additional_results, f)

```

C.2 Code Description

The provided code performs an analysis on a dataset of pediatric patients who received post-operative mechanical ventilation after undergoing surgery. The goal of the analysis is to determine the optimal tracheal tube depth (OTTD) using a machine learning model and a formula-based model.

First, the code loads the dataset and prepares it for analysis. It calculates descriptive statistics by stratifying the data by sex and saving the results in a pickled file ("table_0.pkl").

Next, the code preprocesses the dataset by creating dummy variables for the sex feature.

The analysis begins by creating a new column, "ht_form", which represents the height feature in the formula-based model. The dataset is then split into training and testing sets.

A Random Forest Regression model is initialized and trained using the training set. The model is set to use 200 estimators and a maximum depth of 5, and the random state is fixed for reproducibility.

Predictions are made using the trained model on the testing set, and a formula-based prediction is generated for comparison. The residuals from both the machine learning model and the formula-based model are then compared, and the results are stored in a pickled file ("table_1.pkl"). The comparison includes the mean and standard deviation of the residuals and a p-value from a paired sample t-test.

Lastly, additional results are saved in a pickled file ("additional_results.pkl"). These results include the total number of observations in the dataset and the size of the testing set.

Overall, the code conducts exploratory data analysis, prepares the dataset for analysis, trains a machine learning model, compares it to a formula-based model, and saves the results for further analysis or reporting.

C.3 Code Output

table_0.pkl

```
        tube mean  tube std  age_c mean  age_c std    ht mean     ht std   wt
    mean    wt std  tube_depth_G mean  tube_depth_G std
```

```
sex
female   3.681208  0.551846    0.731544   1.402500  65.400447  18.701462
    6.841902  4.568146          10.062416           1.645478
male     3.704598  0.582023    0.780651   1.472808  66.514368  19.403722
    7.370556  4.935102          10.298276           1.857778
```

table_1.pkl

```
                    mean       std      p_val
RF_model        0.032771  1.192936  8.269e-64
Height_Formula -1.355155  1.261208  8.269e-64
```

additional_results.pkl

```
{
    'Total number of observations': 969,
    'Test sample size': 194,
}
```

D LaTeX Table Design

D.1 Code

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

```python

# IMPORT
import pandas as pd
from my_utils import to_latex_with_note, format_p_value,
↪  is_str_in_df, split_mapping, AbbrToNameDef

# PREPARATION FOR ALL TABLES
shared_mapping: AbbrToNameDef = {
    'sex': ('Sex', '0: Female, 1: Male'),
    'age_c': ('Age', 'Years, rounded'),
    'ht': ('Height', 'cm'),
    'wt': ('Weight', 'kg'),
    'tube_depth_G mean': ('Tube Mean Depth', 'cm via X-ray'),
    'tube_depth_G std': ('Tube Depth SD', 'cm via X-ray'),
    'age_c mean': ('Mean Age', 'Years, rounded'),
    'age_c std': ('Age SD', 'Years, rounded')}
```

```

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

# FORMATTING VALUES (none required)

# 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 = df.rename(columns=abbrs_to_names, index=abbrs_to_names)

# Transposition of dataset
df = df.T

# SAVE AS LATEX
to_latex_with_note(
 df,
 'table_0.tex',
 caption='Mean and standard deviation of Optimal Tube Depth
  ↪  based on Sex',
 label='table:table0',
 legend=legend)

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

# FORMATTING VALUES
df['p_val'] = df['p_val'].apply(format_p_value)

# RENAME ROWS and COLUMNS
table_1_mapping = {
    'Height_Formula': ('Height Formula Model', None),
    'RF_model': ('Random Forest Model', None),
    'p_val': ('P-value', None)}

mapping = {**shared_mapping, **table_1_mapping}
mapping = {k: v for k, v in mapping.items() if
 ↪  is_str_in_df(df, k)}

```

```
abbrs_to_names, legend = split_mapping(mapping)
df = df.rename(columns=abbrs_to_names, index=abbrs_to_names)

# SAVE AS LATEX
to_latex_with_note(
 df,
 'table_1.tex',
 caption='Comparison of residuals between the Machine Learning
↪  model and Formula-based model',
 label='table:table1',
 note='The residuals from the ML model and the formula-based
↪  model are compared using t-test. The displayed P-value is
↪  for the t-test comparison.',
 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{Mean and standard deviation of Optimal Tube Depth based on Sex}
\label{table:table0}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{lrr}
\toprule
sex & female & male \\
\midrule
\textbf{tube mean} & 3.68 & 3.7 \\
\textbf{tube std} & 0.552 & 0.582 \\
\textbf{Mean Age} & 0.732 & 0.781 \\
\textbf{Age SD} & 1.4 & 1.47 \\
\textbf{ht mean} & 65.4 & 66.5 \\
\textbf{ht std} & 18.7 & 19.4 \\
\textbf{wt mean} & 6.84 & 7.37 \\
\textbf{wt std} & 4.57 & 4.94 \\
\textbf{Tube Mean Depth} & 10.1 & 10.3 \\
\textbf{Tube Depth SD} & 1.65 & 1.86 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item \textbf{Tube Mean Depth}: cm via X-ray
\item \textbf{Tube Depth SD}: cm via X-ray
\item \textbf{Mean Age}: Years, rounded
\item \textbf{Age SD}: Years, rounded
\end{tablenotes}
\end{threeparttable}
\end{table}
```

table_1.tex

```
\begin{table}[h]
\caption{Comparison of residuals between the Machine Learning model and Formula-
    based model}
\label{table:table1}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{lrrl}
\toprule
 & mean & std & P-value \\
\midrule
\textbf{Random Forest Model} & 0.0328 & 1.19 & $<$1e-06 \\
\textbf{Height Formula Model} & -1.36 & 1.26 & $<$1e-06 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item The residuals from the ML model and the formula-based model are compared
    using t-test. The displayed P-value is for the t-test comparison.
\end{tablenotes}
\end{threeparttable}
```

```
\end{table}
```