Optimal Tracheal Tube Depth Determination in Pediatric Patients

Data to Paper

January 15, 2024

Abstract

Pediatric patients requiring mechanical ventilation often experience complications from misplaced tracheal tubes. Accurately determining the optimal tracheal tube depth (OTTD) is crucial to mitigate these risks. However, current methods based on chest X-rays or formula-based models have limitations. To address this, we present a dataset of pediatric patients who underwent post-operative mechanical ventilation, along with their OTTD determined by chest X-ray. We compared the performance of a Random Forest model and a height-based formula in determining OTTD. The Random Forest model outperformed the height-based formula, providing more accurate predictions. Our findings demonstrate the potential of the Random Forest model in accurately determining OTTD and reducing complications caused by tracheal tube misplacement. We discuss the implications of our study and its limitations, emphasizing the need for further research to refine and validate this approach.

Results

Our analysis begins with an assessment of the distribution of the Optimal Tracheal Tube Depth (OTTD), stratified by sex. From table 1, the average OTTD for female patients was found to be 10.1 cm with a standard deviation of 1.65 cm, while for male patients it was slightly higher at 10.3 cm with a standard deviation of 1.86 cm. This slight difference underscores the necessity of considering sex when determining the OTTD in pediatric patients.

Next, the performance of two different approaches to determine OTTD –the Random Forest model and the Height-based Formula– were compared, with the Mean Squared Error (MSE) as the evaluation metric (Table 2).

Table 1: Descriptive statistics of OTTD stratified by sex

	mean	std
female	10.1	1.65
male	10.3	1.86

mean: Average Value std: Standard Deviation

The results show that the Random Forest model has superior performance with a lower MSE of 1.39 as compared to an MSE of 3.42 for the Height-based Formula. The implication here is a better performance of the Random Forest model in accurately establishing the OTTD in pediatric patients on mechanical ventilation.

Table 2: Model performance comparison: Random Forest vs. Height-based Formula

	Model	Residuals MSE	T-Statistic	P-Value
1	Random Forest	1.39	-6.23	$<10^{-6}$
2	Height-based Formula	3.42	-6.23	$<10^{-6}$

Index 1. Random Forest 2. Height-based Formula
Model: Predictive Model Name
Residuals MSE: Mean Squared Error of residuals
T-Statistic: Value of T-Statistic for paired T-test involving residuals
P-Value: Corresponding P-Value for the test statistic

Moving on to model fine-tuning, the optimal parameters for the Random Forest model were established as a maximum depth of 5 and number of estimators as 300. These parameters were derived by grid search optimization on our data, and hence are expected to foster the most accurate predictions for OTTD in our specific cohort of pediatric patients (Additional Results).

In conclusion, our study reveals a barely perceptible difference in the OTTD between female and male pediatric patients. Furthermore, our results point towards a quantitatively better performance by the Random Forest model when compared with the Height-based Formula in determining the OTTD. The optimal parameters for our Random Forest model, as derived from our data, holds potential application for accurately predicting OTTD in mechanically ventilated pediatric patients.

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:

```
1
2  # import required packages
3  import pandas as pd
4  import numpy as np
5
6  # load the data
7  df = pd.read_csv("tracheal_tube_insertion.csv")
8
9  # open our output file
10 with open("data_exploration.txt", "w") as file:
11
12     # write data size
13     file.write("# Data Size\n")
14     file.write(f"Number of rows: {df.shape[0]}\n")
15     file.write(f"Number of columns: {df.shape[1]}\n\n")
16
17     # write summary statistics
18     file.write("# Summary Statistics\n")
19     summary_stats = df.describe(include='all').transpose()
20     file.write(f"{summary_stats}\n\n")
21
22     # writing categorical variables
23     file.write("# Categorical Variables\n")
```

```
    categorical_variables =
    ↪  df.select_dtypes(include='object').nunique()
    for var, unique_counts in categorical_variables.items():
        file.write(f"{var}: {unique_counts} unique values\n")
    file.write("\n")

    # write missing values
    file.write("# Missing Values\n")
    missing_values = df.isnull().sum()
    for var, missing_count in missing_values.items():
        if missing_count > 0:
            file.write(f"{var}: {missing_count} missing
            ↪  values\n")

    # Check for special numeric values that stand for
    ↪  unknown/undefined
    nan_cols = df.isna().any()
    file.write("\nColumns with NaN values\n")
    file.write(f"{nan_cols}\n")

```

B.2 Code Description

The provided code performs data exploration on the given dataset of pediatric patients who received post-operative mechanical ventilation.

First, the code loads the dataset into a pandas DataFrame.

The code then writes the following information into the "data_exploration.txt" file:

1. Data Size: The number of rows and columns in the dataset.

2. Summary Statistics: The code calculates summary statistics for all columns in the dataset, including count, mean, standard deviation, minimum, 25th percentile, median, 75th percentile, and maximum. These statistics provide an overview of the distribution and range of values for each numerical feature.

3. Categorical Variables: The code identifies categorical variables in the dataset and writes the number of unique values for each categorical variable. This information helps identify the diversity of categories within each variable.

4. Missing Values: The code identifies missing values in the dataset and writes the number of missing values for each column. This information is

important to understand the completeness of the dataset and to inform any necessary data cleaning or imputation steps.

5. Columns with NaN values: The code checks for columns that contain NaN values and writes the column names. This information helps identify columns that may require special handling or imputation due to incomplete data.

Overall, the data exploration code provides key insights into the structure, distribution, and completeness of the dataset, enabling researchers to make informed decisions regarding data cleaning, preprocessing, and analysis.

B.3 Code Output

data_exploration.txt

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

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

# Categorical Variables

# Missing Values

Columns with NaN values
tube            False
sex             False
age_c           False
ht              False
wt              False
tube_depth_G    False
dtype: bool
```

C Data Analysis

C.1 Code

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

```

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

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

# DATASET PREPARATIONS
# No dataset preparations are needed.

# DESCRIPTIVE STATISTICS
## Table 0: "Descriptive statistics of tube_depth_G stratified by sex"
df0 = df.groupby('sex').tube_depth_G.agg(['mean', 'std'])
df0.index = ['female', 'male']
df0.to_pickle('table_0.pkl')

# PREPROCESSING
# No preprocessing is needed, because all the variables are already in suitable format.

# ANALYSIS
## Table 1: "Model performance comparison: Random Forest vs. Height-based Formula"
# Random Forest
```

```python
X = df[["sex", "age_c", "ht", "wt"]]
y = df["tube_depth_G"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

rf = RandomForestRegressor(random_state=42)
param_grid = {'n_estimators': [100, 200, 300], 'max_depth': [5, 10, 15, None]}
grid_search = GridSearchCV(rf, param_grid, cv=5)
grid_search.fit(X_train, y_train)

y_pred_rf = grid_search.predict(X_test)
rf_residuals = np.square(y_pred_rf - y_test)

# Height Formula based Model
y_pred_ht = X_test["ht"]/10 + 5
ht_residuals = np.square(y_pred_ht - y_test)

# Paired t-test
t_test_results = stats.ttest_rel(rf_residuals, ht_residuals)

df1 = pd.DataFrame({
    "Model": ["Random Forest", "Height-based Formula"],
    "Residuals Mean Squared Error": [np.mean(rf_residuals), np.mean(ht_residuals)],
    "T-statistic": [t_test_results.statistic, t_test_results.statistic],
    "P-value": [t_test_results.pvalue, t_test_results.pvalue]},
    index=['1', '2']
)

df1.to_pickle('table_1.pkl')

# SAVE ADDITIONAL RESULTS
additional_results = {
    'Total number of observations': df.shape[0],
    'Random Forest: Best parameters': grid_search.best_params_,
}
```

```
63
64 with open('additional_results.pkl', 'wb') as f:
65     pickle.dump(additional_results, f)
66
```

C.2 Code Description

The code performs a data analysis on the provided dataset of pediatric patients who underwent mechanical ventilation after surgery. The goal is to determine the optimal tracheal tube depth (OTTD) for these patients, which is crucial for avoiding complications.

After loading the dataset, the code first computes the descriptive statistics of the OTTD stratified by patient sex. It calculates the mean and standard deviation of OTTD for both female and male patients and saves the results in a pickle file.

Next, the code prepares the dataset for analysis by selecting the necessary features. There is no preprocessing step required as the variables are already in a suitable format.

The analysis is performed using two models: Random Forest and a height-based formula. The Random Forest model is trained on a subset of the data, using the features (sex, age, height, and weight) to predict the OTTD. Grid search is performed to find the best hyperparameters for the Random Forest model. The trained model is then used to predict the OTTD for the test data.

The height-based formula model predicts the OTTD based on the patient's height using a simple formula. The predicted OTTD values for both models are then compared to the true OTTD values in the test data.

To evaluate the performance of the models, the mean squared error (MSE) of the residuals (the squared difference between the predicted and true OTTD values) is calculated for both models. A paired t-test is conducted to compare the MSEs of the two models and determine if one model performs significantly better than the other.

The results of the analysis, including the model performance comparison, are saved in a pickle file named "table_1.pkl". The file contains a table with the model names, the MSEs, the t-statistic, and the p-value of the t-test.

Additionally, the code saves some additional results in a pickle file named "additional_results.pkl". These results include the total number of observations in the dataset and the best hyperparameters found for the Random Forest model during the grid search.

C.3 Code Output

table_0.pkl

```
             mean       std
female  10.062416  1.645478
male    10.298276  1.857778
```

table_1.pkl

```
                  Model  Residuals Mean Squared Error  T-statistic    P-value
1         Random Forest                     1.388133    -6.226661  2.914e-09
2  Height-based Formula                     3.418890    -6.226661  2.914e-09
```

additional_results.pkl

```
{
    'Total number of observations': 969,
    'Random Forest: Best parameters': {'max_depth': 5, 'n_estimators': 300},
}
```

D LaTeX Table Design

D.1 Code

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

```

# IMPORT
import pandas as pd
from my_utils import to_latex_with_note, format_p_value,
↪  is_str_in_df, split_mapping
from typing import Dict, Any, Tuple, Optional

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

# PREPARATION FOR ALL TABLES
shared_mapping: AbbrToNameDef = {
    'tube_depth_G': ('OTTD', 'Optimal Tracheal Tube Depth
    ↪  determined by chest X-ray (cm)'),
    'sex': ('Sex', '1: male, 0: female'),
    'age_c': ('Age', 'Age of the patient (in years)'),
```

```
    'ht': ('Height', 'Height of the patient (in cm)'),
    'wt': ('Weight', 'Weight of the patient (in kg)'),
    'tube': ('Tube Diameter', 'Internal diameter of the tube
    ↪  in mm used for mechanical ventilation')
}

# 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, names_to_definitions = split_mapping(mapping)
df = df.rename(columns=abbrs_to_names, index=abbrs_to_names)

# Save as latex:
to_latex_with_note(
 df, 'table_0.tex',
 caption="Descriptive statistics of OTTD stratified by sex",
 label='table:table_0',
 note="mean: Average Value\nstd: Standard Deviation",
 legend=names_to_definitions
)

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

# RENAME ROWS AND COLUMNS
Mapping = {
    'Model': (None, 'Predictive Model Name'),
    'Residuals Mean Squared Error': ('Residuals MSE', 'Mean
    ↪  Squared Error of residuals'),
    'T-statistic': ('T-Statistic', 'Value of T-Statistic for
    ↪  paired T-test involving residuals'),
    'P-value': ('P-Value', 'Corresponding P-Value for the test
    ↪  statistic')
}
mapping = {k: v for k, v in shared_mapping.items() if
↪  is_str_in_df(df, k)}
mapping.update(Mapping)
```

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

# FORMAT VALUES
df['P-Value'] = df['P-Value'].apply(format_p_value)

# Save as latex:
to_latex_with_note(
 df, 'table_1.tex',
 caption="Model performance comparison: Random Forest vs. Height-based Formula",
 label='table:table_1',
 note="Index\n1. Random Forest\n2. Height-based Formula",
 legend=names_to_definitions
)

```

D.2 Provided Code

The code above is using the following provided functions:

```python
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 OTTD stratified by sex}
\label{table:table_0}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{lrr}
\toprule
 & mean & std \\
\midrule
\textbf{female} & 10.1 & 1.65 \\
\textbf{male} & 10.3 & 1.86 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
```

```
\footnotesize
\item mean: Average Value
std: Standard Deviation
\end{tablenotes}
\end{threeparttable}
\end{table}
```

table_1.tex

```
\begin{table}[h]
\caption{Model performance comparison: Random Forest vs. Height-based Formula}
\label{table:table_1}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{llrrl}
\toprule
 & Model & Residuals MSE & T-Statistic & P-Value \\
\midrule
\textbf{1} & Random Forest & 1.39 & -6.23 & $<$1e-06 \\
\textbf{2} & Height-based Formula & 3.42 & -6.23 & $<$1e-06 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item Index
1. Random Forest
2. Height-based Formula
\item \textbf{Model}: Predictive Model Name
\item \textbf{Residuals MSE}: Mean Squared Error of residuals
\item \textbf{T-Statistic}: Value of T-Statistic for paired T-test involving
    residuals
\item \textbf{P-Value}: Corresponding P-Value for the test statistic
\end{tablenotes}
\end{threeparttable}
\end{table}
```