Accurate Prediction of Optimal Tracheal Tube Depth in Pediatric Patients during Mechanical Ventilation

Data to Paper

February 20, 2024

Abstract

Determining the optimal tracheal tube depth (OTTD) is crucial for pediatric patients undergoing mechanical ventilation. Existing methods, such as chest X-ray and formula-based models, have limitations in accuracy and feasibility. In this study, we compared the performance of a Random Forest Model (RFM) with a Height Formula-Based Model (HFMB) in predicting OTTD in a dataset of pediatric patients aged 0-7 years who underwent post-operative mechanical ventilation. The RFM outperformed the HFMB, providing a more accurate estimation of OTTD. We identified patient weight as the most influential factor in determining OTTD, with height, age, and sex also contributing to a lesser extent. The RFM model demonstrated good performance and generalization to unseen data. Our findings have implications for enhancing patient safety and optimizing mechanical ventilation in pediatric intensive care settings.

Results

Accurately determining the optimal tracheal tube depth (OTTD) is crucial for pediatric patients undergoing mechanical ventilation. In this study, we compared the performance of a Random Forest Model (RFM) with a Height Formula-Based Model (HFMB) in predicting OTTD in pediatric patients.

We evaluated the performance of the RFM and HFMB models using mean squared residuals, t-statistic, and p-value (Table 1). The RFM outperformed the HFMB, with lower mean squared residuals (1.35 versus 3.42) and a statistically significant t-statistic of -6.31 (p-value $< 10^{-6}$). These results indicate that the RFM provides a more accurate estimation of OTTD compared to the HFMB.

Table 1: Performance comparison of Random Forest Model (RFM) and Height Formula-Based Model (HFMB)

	Mean Squared Residuals	t-statistic	p-value
Model			
Random Forest Model	1.35	-6.31	$<10^{-6}$
Height Formula-Based Model	3.42	-6.31	$<10^{-6}$

Mean Squared Residuals: Squared residuals (prediction - target)**2
t-statistic: Statistical t value
p-value: Two-tailed p value for hypothesis test

We further investigated the feature importance of the RFM model (Table 2). Weight was identified as the most influential factor in determining OTTD, contributing an importance value of 0.884. This suggests that patient weight plays a critical role in determining the tracheal tube depth in pediatric patients. Height, age, and sex also showed some contribution, although to a lesser extent.

Table 2: Feature Importance of Random Forest Model (RFM)

	Importance
Feature	
Sex	0.000261
Age	0.00343
Height	0.112
Weight	0.884

Importance: Feature importance as computed by the Random Forest model
Sex: Patient sex (0=female, 1=male)
Age: Patient's age in years
Height: Patient's height in cm
Weight: Patient's weight in kg

Additionally, the RFM model demonstrated good performance as indicated by a training score of 0.5916 and a test score of 0.6068. These scores suggest that the RFM model can generalize well to unseen data and reliably predict OTTD in pediatric patients.

In summary, our findings suggest that the Random Forest Model outperforms the Height Formula-Based Model in accurately estimating the optimal tracheal tube depth in pediatric patients. Patient weight emerges as the most important factor in determining OTTD, followed by height, age, and sex. These results have important implications for enhancing patient safety and optimizing mechanical ventilation in pediatric intensive care settings.

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

# Loading the data
df = pd.read_csv("tracheal_tube_insertion.csv")

# Creating a summary text file
with open("data_exploration.txt", "w") as f:
    # Data size
    f.write("# Data Size\n")
    f.write(f"Number of rows: {len(df)}\n")
    f.write(f"Number of columns: {len(df.columns)}\n\n")

    # Summary statistics
    f.write("# Summary Statistics\n")
    f.write(df.describe(include='all').to_string())
    f.write("\n\n")

    # Categorical variable(s)
    f.write("# Categorical Variables\n")
    f.write(df['sex'].value_counts().to_string())
    f.write("\n\n")

    # Missing values
    f.write("# Missing Values\n")
    f.write(df.isnull().sum().to_string())
    f.write("\n\n")

    # Checking for special numeric values that stand for unknown/undefined
    f.write("# Special Numeric Values\n")
    special_vals = df.applymap(np.isreal).sum() != len(df)
    f.write(special_vals[special_vals == True].to_string())
    f.write("\n")

```

B.2 Code Description

The provided code performs data exploration on the given dataset. The main steps of the analysis are as follows:

1. Loading the data: The code reads the data from the "tracheal_tube_insertion.csv" file into a pandas DataFrame.

2. Data Size: The code calculates the number of rows and columns in the DataFrame and writes this information to the "data_exploration.txt" file.

3. Summary Statistics: The code generates summary statistics for all the columns in the DataFrame using the ‘describe()‘ function. These statistics include count, mean, standard deviation, minimum, 25th percentile, median, 75th percentile, and maximum. The code writes these summary statistics to the "data_exploration.txt" file.

4. Categorical Variables: The code calculates the number of occurrences of each unique value in the 'sex' column, which represents the patient's sex. The code writes this information to the "data_exploration.txt" file.

5. Missing Values: The code determines the number of missing values in each column of the DataFrame using the ‘isnull().sum()‘ function. The code writes this information to the "data_exploration.txt" file.

6. Special Numeric Values: The code identifies any special numeric values in the DataFrame that might represent unknown or undefined values. It does this by checking if each element in the DataFrame is a real number using the ‘np.isreal()‘ function and counting the number of elements that are not real numbers. The code writes this information to the "data_exploration.txt" file.

The "data_exploration.txt" file contains the following information: -

Data size: Number of rows and columns in the dataset. - Summary statistics: Descriptive statistics for each column in the dataset. - Categorical variables: Number of occurrences of each unique value in the 'sex' column. - Missing values: Number of missing values in each column. - Special Numeric Values: Any special numeric values that represent unknown or undefined values.

This data exploration is an essential step in understanding the characteristics of the dataset and identifying any potential issues, such as missing values or special numeric values. The generated summary statistics provide a summary of the distribution of the data, while the information on missing and special values helps in determining the data quality and potential data preprocessing steps that may be required.

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
sex
1    522
0    447

# Missing Values
tube            0
sex             0
age_c           0
```

```
ht              0
wt              0
tube_depth_G    0

# Special Numeric Values
Series([], )
```

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 sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from scipy import stats
import pickle

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

# DATASET PREPARATIONS
# No dataset preparations are needed.

# DESCRIPTIVE STATISTICS
# No descriptive statistics table is needed.

# PREPROCESSING
# No preprocessing is needed because all features are numerical

# ANALYSIS
# Table 1: "Performance comparison of Random Forest Model (RFM) and Height Formula-Based Model (HFMB)"
features = df.drop(columns=['tube_depth_G', 'tube'])
target = df['tube_depth_G']

# Split into train and test set
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size = 0.2, random_state = 42)

# Random Forest Model with GridSearchCV for hyperparameter tuning
parameters = {'n_estimators':[50,100,200], 'max_depth':[3,5,7, None]}
rf = RandomForestRegressor(random_state = 42)
clf = GridSearchCV(rf, parameters, cv=5)
clf.fit(X_train, y_train)
rf_predictions = clf.predict(X_test)

# Height Formula-Based Model
heights = X_test['ht']
hfmb_predictions = heights/10 + 5

# Compute Residuals square
rf_residuals_sq = (rf_predictions - y_test)**2
hfmb_residuals_sq = (hfmb_predictions - y_test)**2

# Perform paired t-test
ttest_results = stats.ttest_rel(rf_residuals_sq, hfmb_residuals_sq)

df1 = pd.DataFrame({
    'Model': ['RFM', 'HFMB'],
    'Mean_Squared_Residuals': [rf_residuals_sq.mean(), hfmb_residuals_sq.mean()],
    't_statistic': [ttest_results.statistic, ttest_results.statistic],
    'p_value': [ttest_results.pvalue, ttest_results.pvalue]
})

df1.set_index('Model', inplace=True)
```

```python
df1.to_pickle('table_1.pkl')

# Table 2: "Feature Importance of Random Forest Model"
importances = pd.DataFrame({
    'Feature': X_train.columns,
    'Importance': clf.best_estimator_.feature_importances_
})
importances.set_index('Feature', inplace=True)
importances.to_pickle('table_2.pkl')

# SAVE ADDITIONAL RESULTS
additional_results = {
    'Total number of observations': len(df),
    'RFM_best_params': clf.best_params_,
    'RFM_training_score': clf.score(X_train, y_train),
    'RFM_test_score': clf.score(X_test, y_test)
}
with open('additional_results.pkl', 'wb') as f:
    pickle.dump(additional_results, f)


```

C.2 Code Description

The provided code performs the analysis of the pediatric patient dataset to determine the optimal tracheal tube depth (OTTD) using a Random Forest Model (RFM) and compares it with a Height Formula-Based Model (HFMB). The code follows the following steps:

1. Load the dataset: The code loads the dataset containing patient information and OTTD determined by chest X-ray.

2. Prepare the dataset: No dataset preparations are needed as all the features are already in numerical format.

3. Analysis: a. Performance comparison of RFM and HFMB (Table 1): i. Split the dataset into training and testing sets. ii. Apply the RFM using the Random Forest Regressor with hyperparameter tuning using GridSearchCV. iii. Make predictions with the RFM on the testing set. iv. Calculate the squared residuals between the RFM predictions and the actual OTTD values for the testing set. v. Calculate the squared residuals between the HFMB predictions (computed using patient heights) and the actual OTTD values for the testing set. vi. Perform a paired t-test on the

squared residuals to compare the performance of RFM and HFMB. vii. Create Table 1 containing the mean squared residuals, t-statistic, and p-value for RFM and HFMB.

b. Feature Importance of RFM (Table 2): i. Obtain the feature importances from the best estimator of the RFM after hyperparameter tuning. ii. Create Table 2 containing the feature names and their corresponding importances.

4. Save Additional Results: - Save the total number of observations in the dataset. - Save the best parameters of the RFM. - Save the training and testing scores of the RFM. - Write these additional results into the 'additional_results.pkl' file.

The code allows for the comparison of the RFM and HFMB models in terms of their performance in determining OTTD for pediatric patients. The RFM is trained on the dataset to predict OTTD based on patient features, while the HFMB computes the predictions using a height-based formula. The code outputs tables summarizing the comparison results as well as the feature importances of the RFM. Additionally, it saves the total number of observations and RFM-related results in the 'additional_results.pkl' file.

C.3 Code Output

table_1.pkl

```
       Mean_Squared_Residuals  t_statistic    p_value
Model
RFM                  1.354749    -6.307568  1.893e-09
HFMB                 3.418890    -6.307568  1.893e-09
```

table_2.pkl

```
         Importance
Feature
sex        0.000261
age_c      0.003430
ht         0.111983
wt         0.884326
```

additional_results.pkl

```
{
    'Total number of observations': 969,
    'RFM_best_params': {'max_depth': 3, 'n_estimators': 50},
    'RFM_training_score': 0.5916              ,
    'RFM_test_score': 0.6068            ,
}
```

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 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
shared_mapping: AbbrToNameDef = {
}

# 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 |= {
 'Mean_Squared_Residuals': ('Mean Squared Residuals', 'Squared residuals (prediction - target)**2'),
 't_statistic': ('t-statistic', 'Statistical t value'),
 'p_value': ('p-value', 'Two-tailed p value for hypothesis test'),
 'RFM': ('Random Forest Model', None),
 'HFMB': ('Height Formula-Based Model', None)
}
```

```python

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="Performance comparison of Random Forest Model (RFM) and Height Formula-Based Model (HFMB)",
 label='table:rf_hfmb_comparison',
 note=None,
 legend=legend)

# TABLE 2
df = pd.read_pickle('table_2.pkl')

# RENAME ROWS AND COLUMNS
mapping = {k: v for k, v in shared_mapping.items() if is_str_in_df(df, k)}
mapping |= {
 'Importance': ('Importance', 'Feature importance as computed by the Random Forest model'),
 'sex': ('Sex', 'Patient sex (0=female, 1=male)'),
 'age_c': ('Age', "Patient's age in years"),
 'ht': ('Height', "Patient's height in cm"),
 'wt': ('Weight', "Patient's weight in kg")
}

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_2.tex',
 caption="Feature Importance of Random Forest Model (RFM)",
 label='table:rfm_feature_importance',
 note=None,
 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_1.tex

```
\begin{table}[h]
\caption{Performance comparison of Random Forest Model (RFM) and Height Formula-
    Based Model (HFMB)}
\label{table:rf_hfmb_comparison}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{lrrl}
\toprule
 & Mean Squared Residuals & t-statistic & p-value \\
Model &  &  &  \\
\midrule
\textbf{Random Forest Model} & 1.35 & -6.31 & $<$1e-06 \\
\textbf{Height Formula-Based Model} & 3.42 & -6.31 & $<$1e-06 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item \textbf{Mean Squared Residuals}: Squared residuals (prediction -
    target)**2
\item \textbf{t-statistic}: Statistical t value
\item \textbf{p-value}: Two-tailed p value for hypothesis test
\end{tablenotes}
\end{threeparttable}
\end{table}
```

table_2.tex

```
\begin{table}[h]
\caption{Feature Importance of Random Forest Model (RFM)}
\label{table:rfm_feature_importance}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
```

```
\makebox[\linewidth]{%
\begin{tabular}{lr}
\toprule
 & Importance \\
Feature &  \\
\midrule
\textbf{Sex} & 0.000261 \\
\textbf{Age} & 0.00343 \\
\textbf{Height} & 0.112 \\
\textbf{Weight} & 0.884 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item \textbf{Importance}: Feature importance as computed by the Random Forest
    model
\item \textbf{Sex}: Patient sex (0=female, 1=male)
\item \textbf{Age}: Patient's age in years
\item \textbf{Height}: Patient's height in cm
\item \textbf{Weight}: Patient's weight in kg
\end{tablenotes}
\end{threeparttable}
\end{table}
```