Predictive Modeling of Optimal Tracheal Tube Depth in Pediatric Patients for Enhanced Patient Outcomes

Data to Paper

January 11, 2024

Abstract

Determining the Optimal Tracheal Tube Depth (OTTD) is crucial in pediatric patients requiring mechanical ventilation to mitigate complications and improve patient outcomes. However, existing methods, such as chest X-ray and formula-based models, have limitations in accurately determining OTTD. To address this research gap, we developed predictive models for OTTD using electronic health records of 969 pediatric patients aged 0-7 years who underwent surgery. Our Random Forest and ElasticNet models achieved similar mean squared errors and demonstrated efficacy in predicting OTTD. Descriptive statistics indicated that age and height may not significantly impact optimal tube depth. Our results highlight the potential of machine learning algorithms to improve tracheal tube placement strategies. However, it is important to acknowledge the limitations of these models in accurately determining OTTD in pediatric patients. Further research is needed to enhance prediction methods, ensuring optimal tracheal tube placement and enhancing patient outcomes.

Results

Determining the Optimal Tracheal Tube Depth (OTTD) for pediatric patients undergoing mechanical ventilation is crucial to mitigate complications and improve patient outcomes. To understand the key factors that contribute to OTTD, we first investigated the descriptive statistics of height and age stratified by sex (Table 1). This analysis is important because it provides insight into potential differences in height and age between female and male patients, which could impact the optimal tube depth. We observed that the average age was similar for both females (mean=0.732 years) and males (mean=0.781 years), suggesting that age may not be a differentiating factor in determining OTTD. Additionally, the average height showed no significant difference between sexes, with females having an average height of 65.4 cm and males 66.5 cm. These findings indicate that height may not be a major differentiator in determining the optimal tube depth for pediatric patients.

Table 1: Descriptive statistics of height and age stratified by sex

sex	Female	Male
Average Age (mean)	0.732	0.781
Average Age (std)	1.4	1.47
Average Height (mean)	65.4	66.5
Average Height (std)	18.7	19.4

0: Female, 1: Male
Average Age (mean): Average age in years
Average Age (std): Standard deviation of age
Average Height (mean): Average height in cm
Average Height (std): Standard deviation of height

To evaluate the predictive performance of models for OTTD, we employed Random Forest and ElasticNet models (Table 2). Random Forest is a robust ensemble learning method known for its ability to capture complex relationships in data, while ElasticNet combines L1 and L2 regularization to handle both feature selection and feature correlation. The resulting mean squared errors (MSE) for both models were similar, with the Random Forest achieving an MSE of 1.37 and ElasticNet 1.34. This demonstrates the efficacy of both models in predicting OTTD in pediatric patients.

Table 2: Performance metrics of the Random Forest and ElasticNet models

	MSE
model	
Random Forest	1.37
ElasticNet	1.34

MSE: Mean Squared Error
Random Forest: Random Forest Regression model
ElasticNet: ElasticNet Regression model

The optimal parameters for the Random Forest model were found to be a maximum depth of 3 and 10 estimators, while for ElasticNet, the optimal parameters were an alpha value of 0.1 and an l1 ratio of 0.1. These parameters were determined through a grid search, which seeks to identify the best combination of hyperparameters that minimizes the MSE. In the case of Random Forest, a lower maximum depth and a smaller number of estimators were selected to prevent overfitting and achieve a more generalized model. For ElasticNet, an alpha value of 0.1 balances the trade-off between L1 and L2 regularization and a l1 ratio of 0.1 prioritizes sparsity in the coefficient estimates.

We further compared the residuals of the Random Forest and ElasticNet models (Table 3) to assess any differences in their predictive performance. The paired t-test showed no statistically significant difference between the models, with a t-statistic of 0.466 and a p-value of 0.642. This suggests that both models exhibit similar accuracy in predicting the OTTD residuals.

Table 3: Comparison of residuals of the Random Forest and ElasticNet models

	Statistic	P-value
Comparison	0.466	0.642

Statistic: T-statistic
P-value: P-value from two-tailed paired t-test
Comparison: Comparison of residuals

In summary, our analysis demonstrates that the Random Forest and ElasticNet models are effective in predicting the Optimal Tracheal Tube Depth in pediatric patients undergoing mechanical ventilation. These models achieved similar mean squared errors and exhibited comparable predictive performance. The analysis of descriptive statistics highlighted that age and height may not be substantial factors in determining the optimal tube depth. The results emphasize the potential of machine learning algorithms to improve tracheal tube placement strategies and enhance patient outcomes.

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

# reading the dataset
df = pd.read_csv('tracheal_tube_insertion.csv')

with open('data_exploration.txt', 'w') as file:

    # writing data size
    file.write("# Data Size\n")
    data_size = df.shape
    file.write(f"Number of rows: {data_size[0]}\nNumber of
    ↪  columns: {data_size[1]}\n\n")

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

    # writing categorical variable stats
    file.write("# Categorical Variables\n")
    categorical_variables_freq = df.loc[:, df.dtypes ==
    ↪  object].apply(lambda x: x.value_counts().index[0] if
    ↪  len(x.value_counts())>0 else 'no values present')
    file.write(f"{categorical_variables_freq}\n\n")

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

    # Check for special numeric values which stand for
    ↪  unknown/undefined. Here, we assume that negative
    ↪  values are used to stand for unknown/undefined values.
    file.write(f"# Special Numeric Values\n")
    special_values = df[df<0].count()
    file.write(f"{special_values}\n")

```

B.2 Code Description

The provided code performs data exploration on the "tracheal_tube_insertion.csv" dataset to gain insights into its structure and provide a summary of its contents. The code performs the following analysis steps:

B.3 Data Size

The code first determines the size of the dataset by obtaining the number of rows and columns. This information is then written to the "data_exploration.txt" file.

B.4 Summary Statistics

Next, the code calculates summary statistics for all columns in the dataset using the 'describe' function. These statistics include count, mean, standard deviation, minimum, quartiles, and maximum values. The code then writes the summary statistics to the "data_exploration.txt" file.

B.5 Categorical Variables

The code identifies the categorical variables in the dataset and determines the most frequent value for each variable. This information is useful for understanding the distribution of categorical variables in the dataset. The code writes the categorical variables and their frequencies to the "data_exploration.txt" file.

B.6 Missing Values

The code checks for missing values in the dataset by using the 'isnull' function to identify any null values in each column. The code then counts the number of missing values in each column and writes this information to the "data_exploration.txt" file. This step is important for identifying potential data quality issues and deciding how to handle missing values in subsequent analyses.

B.7 Special Numeric Values

Finally, the code checks for any special numeric values in the dataset that may stand for unknown or undefined values. In this case, the code assumes that negative values are used for this purpose. The code counts the occurrences of negative values in each column and writes this information to the "data_exploration.txt" file.

Overall, the code provides an overview of the dataset's structure, distribution of categorical variables, presence of missing values, and potential use of special numeric values. This information is critical for understanding the dataset before proceeding with further analysis or modeling.

%———————————————————

B.8 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
Series([], dtype: float64)

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

# Special Numeric 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
import numpy as np
import pickle
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import ElasticNet
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import GridSearchCV
from scipy.stats import ttest_rel

```

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

# DATASET PREPARATIONS
# No dataset preparations are needed.

# DESCRIPTIVE STATISTICS
## Table 0: "Descriptive statistics of height and age_c
↪  stratified by sex"
df0 = df.groupby('sex')[['age_c', 'ht']].agg(['mean', 'std'])
df0.columns = [' '.join(col).strip() for col in
↪  df0.columns.values]
df0.index = df0.index.map({0:'Female', 1:'Male'})
df0.to_pickle('table_0.pkl')

# PREPROCESSING
# No preprocessing is needed.

# ANALYSIS
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df[['sex',
↪  'age_c', 'ht', 'wt']], df['tube_depth_G'], test_size=0.2,
↪  random_state=0)

## Table 1: "Performance metrics of the Random Forest and
↪  ElasticNet models"
## Random Forest
rf = RandomForestRegressor(random_state=0)
param_grid_rf = {"n_estimators": np.arange(10, 61, 10),
↪  "max_depth": np.arange(1, 6)}
grid_search_rf = GridSearchCV(rf, param_grid=param_grid_rf,
↪  cv=3, n_jobs=-1)
grid_search_rf.fit(X_train, y_train)
rf_best_model = grid_search_rf.best_estimator_
predictions_rf = rf_best_model.predict(X_test)
mse_rf = mean_squared_error(y_test, predictions_rf)

## ElasticNet
en = ElasticNet(random_state=0)
```

```python
param_grid_en = {'alpha': np.logspace(-4, 1, 6), 'l1_ratio':
 ↪  np.arange(0.1, 0.7, 0.2)}
grid_search_en = GridSearchCV(en, param_grid=param_grid_en,
 ↪  cv=3, n_jobs=-1)
grid_search_en.fit(X_train, y_train)
en_best_model = grid_search_en.best_estimator_
predictions_en = en_best_model.predict(X_test)
mse_en = mean_squared_error(y_test, predictions_en)

df1 = pd.DataFrame({
    "model": ["Random Forest", "ElasticNet"],
    "mse": [mse_rf, mse_en]
})
df1.set_index('model', inplace=True)
df1.to_pickle('table_1.pkl')

## Table 2: "Comparison of residuals of the Random Forest and
 ↪  ElasticNet models"
residuals_rf = (predictions_rf - y_test)**2
residuals_en = (predictions_en - y_test)**2
ttest_res = ttest_rel(residuals_rf, residuals_en)
df2 = pd.DataFrame({
    "statistic": [ttest_res.statistic],
    "pvalue": [ttest_res.pvalue]
}, index=['Comparison'])
df2.to_pickle('table_2.pkl')

# SAVE ADDITIONAL RESULTS
additional_results = {
 'Total number of observations': len(df),
 'Best parameters for Random Forest':
  ↪  grid_search_rf.best_params_,
 'Best parameters for ElasticNet': grid_search_en.best_params_
}
with open('additional_results.pkl', 'wb') as f:
 pickle.dump(additional_results, f)

```

C.2 Code Description

The code performs data analysis on a dataset of pediatric patients who received post-operative mechanical ventilation. The goal is to determine the optimal tracheal tube depth (OTTD) without using chest X-rays.

First, the code loads the dataset and prepares it for analysis. Then, it generates Table 0, which presents the descriptive statistics of patient height and age separated by gender.

Next, the code splits the dataset into training and testing sets for model evaluation. It trains two models, Random Forest and ElasticNet, to predict the OTTD based on patient features such as sex, age, height, and weight. The Random Forest model is optimized by searching through different values of the number of estimators and maximum depth, while the ElasticNet model is optimized for different values of alpha (the regularization parameter) and l1_ratio (the mixing parameter between L1 and L2 penalties).

Table 1 is generated to compare the performance of the two models using mean squared error (MSE) as the evaluation metric. The model with the lower MSE is considered better.

Table 2 compares the residuals of the Random Forest and ElasticNet models. The code calculates the squared difference between the predicted and actual OTTD for each model and performs a t-test to assess if the differences in residuals are statistically significant.

Finally, the code saves additional results in the "additional_results.pkl" file. This includes the total number of observations in the dataset, as well as the best parameters found during the model optimization for Random Forest and ElasticNet.

Overall, the code performs data analysis, model evaluation, and presents the results in the form of tables, providing insights into the performance and comparison of different models in determining the OTTD for pediatric patients undergoing mechanical ventilation.

C.3 Code Output

table_0.pkl

```
        age_c mean  age_c std    ht mean     ht std
sex
Female    0.731544   1.402500  65.400447  18.701462
Male      0.780651   1.472808  66.514368  19.403722
```

table_1.pkl

```
                     mse
model
Random Forest   1.374264
ElasticNet      1.343744
```

table_2.pkl

```
             statistic  pvalue
Comparison    0.465985  0.6418
```

additional_results.pkl

```
{
    'Total number of observations': 969,
    'Best parameters for Random Forest': {'max_depth': 3, 'n_estimators': 10},
    'Best parameters for ElasticNet': {'alpha': 0.1, 'l1_ratio': 0.1},
}
```

D LaTeX Table Design

D.1 Code

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

```

# IMPORTS
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 = {
    'age_c mean': ('Average Age (mean)', 'Average age in
    ↪ years'),
    'age_c std': ('Average Age (std)', 'Standard deviation of
    ↪ age'),
    'ht mean': ('Average Height (mean)', 'Average height in
    ↪ cm'),
    'ht std': ('Average Height (std)', 'Standard deviation of
    ↪  height')
}

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

# TRANPOSE IF REQUIRED
df = df.T

# 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)

# Save as latex:
to_latex_with_note(
 df, 'table_0.tex',
 caption="Descriptive statistics of height and age stratified
  ↪  by sex",
 label='table:age_height_by_sex',
 note="0: Female, 1: Male",
 legend=legend
)

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

# RENAME ROWS AND COLUMNS
mapping = {
    'mse': ('MSE', 'Mean Squared Error'),
    'Random Forest': ('Random Forest', 'Random Forest
    ↪  Regression model'),
    'ElasticNet': ('ElasticNet', 'ElasticNet Regression
    ↪  model')
}

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

```python

# Save as latex:
to_latex_with_note(
 df, 'table_1.tex',
 caption="Performance metrics of the Random Forest and
  ↪  ElasticNet models",
 label='table:model_performance',
 note=None,
 legend=legend
)

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

# FORMAT VALUES
df['pvalue'] = df['pvalue'].apply(format_p_value)

# RENAME ROWS AND COLUMNS
mapping = {
    'statistic': ('Statistic', 'T-statistic'),
    'pvalue': ('P-value', 'P-value from two-tailed paired
     ↪  t-test'),
    'Comparison': ('Comparison', 'Comparison of residuals')
}

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="Comparison of residuals of the Random Forest and
  ↪  ElasticNet models",
 label='table:model_comparison',
 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_0.tex

```
\begin{table}[h]
\caption{Descriptive statistics of height and age stratified by sex}
\label{table:age_height_by_sex}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{lrr}
\toprule
sex & Female & Male \\
\midrule
\textbf{Average Age (mean)} & 0.732 & 0.781 \\
\textbf{Average Age (std)} & 1.4 & 1.47 \\
\textbf{Average Height (mean)} & 65.4 & 66.5 \\
\textbf{Average Height (std)} & 18.7 & 19.4 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item 0: Female, 1: Male
\item \textbf{Average Age (mean)}: Average age in years
\item \textbf{Average Age (std)}: Standard deviation of age
\item \textbf{Average Height (mean)}: Average height in cm
\item \textbf{Average Height (std)}: Standard deviation of height
\end{tablenotes}
\end{threeparttable}
\end{table}
```

table_1.tex

```
\begin{table}[h]
\caption{Performance metrics of the Random Forest and ElasticNet models}
\label{table:model_performance}
\begin{threeparttable}
```

```
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{lr}
\toprule
 & MSE \\
model &  \\
\midrule
\textbf{Random Forest} & 1.37 \\
\textbf{ElasticNet} & 1.34 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item \textbf{MSE}: Mean Squared Error
\item \textbf{Random Forest}: Random Forest Regression model
\item \textbf{ElasticNet}: ElasticNet Regression model
\end{tablenotes}
\end{threeparttable}
\end{table}
```

table_2.tex

```
\begin{table}[h]
\caption{Comparison of residuals of the Random Forest and ElasticNet models}
\label{table:model_comparison}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{lrl}
\toprule
 & Statistic & P-value \\
\midrule
\textbf{Comparison} & 0.466 & 0.642 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item \textbf{Statistic}: T-statistic
\item \textbf{P-value}: P-value from two-tailed paired t-test
```

```
\item \textbf{Comparison}: Comparison of residuals
\end{tablenotes}
\end{threeparttable}
\end{table}
```