Enhancing Tracheal Tube Placement Accuracy in Pediatric Patients Using Data-Driven Models

Data to Paper

January 7, 2024

Abstract

Accurate determination of the optimal tracheal tube depth (OTTD) is crucial for pediatric patients undergoing mechanical ventilation. However, existing formula-based methods have limited success in predicting the OTTD. In this study, we propose a data-driven approach to improve tracheal tube placement accuracy in pediatric patients. We analyze a dataset of 969 patients who received post-operative mechanical ventilation, employing machine learning models and formula-based predictions based on patient features to estimate the OTTD. Our results demonstrate the superiority of machine learning models over formula-based models in accurately predicting the OTTD. These findings underscore the potential of data-driven models to enhance the accuracy of tracheal tube placement in pediatric patients. While this study has limitations, such as the restricted age range and specific population considered, the results provide valuable insights for clinical practice and call for further validation and exploration of data-driven approaches in pediatric tracheal tube placement.

Results

In this section, we report the results of our analyses to establish an effective method for the determination of the optimal tracheal tube depth (OTTD) in pediatric patients. We specifically discuss the predictive accuracy of machine learning models compared to formula-based models, the mean squared error (MSE) for each model, and the R-squared values for machine learning models.

To evaluate the performance of different models, we conducted an analysis comparing machine learning models and formula-based models. As depicted in Table 1, among the machine learning models that encompass the

Random Forest Regressor, Elastic Net Regressor, Support Vector Machine, and Neural Network, the Elastic Net Regressor outperformed the rest with the lowest MSE value of 1.25. The other models attained MSE values of 1.54, 1.28, and 1.41 respectively. This suggests the superior accuracy of the Elastic Net Regressor in predicting the OTTD.

Table 1: Mean Squared Error for each Machine Learning Model

	MSE
Random Forest Regressor	1.54
Elastic Net Regressor	1.25
Support Vector Machine	1.28
Neural Network	1.41

Machine Learning Models include: Random Forest, Elastic Net, Support Vector Machine, Neural Network
MSE: Mean Squared Error, value closer to zero is better

Subsequently, we determined the efficiency of the established formula-based models to ascertain the OTTD. Table 2 indicates the MSE for each formula-based model, which includes patient height, patient age, and tube internal diameter (ID). The patient height formula showed the highest MSE (3.48), followed by tube ID model (2.34), and the patient age formula (1.89). Inferentially, it indicates a limited success of formula-based models in accurately predicting OTTD when compared to machine learning models.

Table 2: Mean Squared Error for each Formula-Based Model

	MSE
Patient Height	3.48
Patient Age	1.89
Tube ID	2.34

Formula-Based Models include: ID (Tube ID Model), Height (Height Formula Model), Age (Age Formula Model)
MSE: Mean Squared Error, value closer to zero is better
Tube ID: internal diameter of the tracheal tube in millimeter
Patient Height: Height of the patient in cm
Patient Age: Age of the patient in years

In addition, we performed a paired T-test comparison to establish the difference in prediction accuracy between the machine learning and formula-based model predictions. As outlined in Table 3, all the machine learning

models demonstrated p-values consistently below 10^{-6} when compared to the formula-based models. This implies a statistically significant difference in the prediction accuracy between the machine learning models and the formula-based models.

Table 3: Paired T-Test Comparison between Machine Learning and Formula-Based Model Predictions

	p-values
Random Forest vs Height	$<10^{-6}$
Random Forest vs Age	$7.14\ 10^{-6}$
Random Forest vs ID	$<10^{-6}$
Elastic Net vs Height	$<10^{-6}$
Elastic Net vs Age	$<10^{-6}$
Elastic Net vs ID	$<10^{-6}$
SVM vs Height	$<10^{-6}$
SVM vs Age	$<10^{-6}$
SVM vs ID	$<10^{-6}$
Neural Network vs Height	$<10^{-6}$
Neural Network vs Age	0.178
Neural Network vs ID	$<10^{-6}$

P-values formatted to be less than 10^{-6} if it's less than 10^{-6}

Furthermore, we evaluated the R-squared values for each machine learning model. The Elastic Net Regressor exhibited the highest R-squared value of 0.6061, followed by the Support Vector Machine (0.5988), Neural Network (0.5579), and Random Forest Regressor (0.5163). These values illustrate the proportion of the variance in the OTTD that can be attributed to the respective machine learning models, with higher values suggesting a more enhanced model performance.

To summarize, our results underline the superior efficiency of the Elastic Net Regressor over other machine learning and formula-based models in accurately determining the OTTD in pediatric patients. The MSE values provide further substantiation towards this, affirming that the Elastic Net Regressor displays the highest predictive accuracy. On performing a paired T-test, we find the p-values further validate the significantly better performance of the machine learning models. The R-squared values, concurrently, indicate a substantial proportion of the variance in OTTD is explained by the Elastic Net Regressor. Collectively, these results emphasize the potential of data-driven models, such as the Elastic Net Regressor, in enhancing

the accuracy of tracheal tube placement in 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 pandas as pd
3  import numpy as np
4
5  # Load the data
6  df = pd.read_csv("tracheal_tube_insertion.csv")
7
8  # Writing all the summaries to a text file
9  with open('data_exploration.txt', 'w') as f:
10
11     # Data Size
12     f.write("# Data Size\n")
13     f.write("Number of rows: " + str(df.shape[0]) + "\n")
14     f.write("Number of columns: " + str(df.shape[1]) + "\n\n")
15
16     # Summary Statistics
17     f.write("# Summary Statistics\n")
18     summary_stats = df.describe(include=[np.number])
19     f.write(str(summary_stats) + "\n\n")
20
21     # Categorical Variables
22     f.write("# Categorical Variables\n")
23     categorical_vars = df.select_dtypes(include=['object'])
24     for column in categorical_vars:
```

```
25          f.write(str(df[column].value_counts().head()) +
            ↪  "\n\n")
26
27      # Missing Values
28      f.write("# Missing Values\n")
29      missing_values = df.isnull().sum()
30      f.write(str(missing_values[missing_values > 0]) + "\n\n")
31
32      # Check if there is any special numeric value that might
        ↪  stand for unknown/undefined
33      f.write("# Investigation for Special Numeric Values\n")
34      special_values = df[df < 0].count()
35      f.write(str(special_values[special_values > 0]) + "\n") #
        ↪  Assuming negative values are not expected
36
```

B.2 Code Description

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

Firstly, the code loads the dataset and then proceeds to perform various analysis steps.

The code generates a summary of the data size by reporting the number of rows and columns in the dataset.

Next, the code computes summary statistics for the numeric variables in the dataset using the ‘describe‘ function. These statistics include measures such as count, mean, standard deviation, minimum, 25th percentile, median, 75th percentile, and maximum. The summary statistics provide an overview of the distribution and range of values for each numeric variable.

The code also examines the categorical variables in the dataset. For each categorical variable, it counts the occurrences of each unique value and reports the top few values. This provides insights into the frequency and distribution of categorical variables.

Furthermore, the code identifies any missing values in the dataset by using the ‘isnull‘ function to generate a Boolean mask. It then sums up the missing values for each variable and reports the variables with non-zero missing values. This step helps to identify the completeness of the dataset and potential challenges related to missing data.

Additionally, the code investigates if there are any special numeric values that might represent unknown or undefined data. It specifically checks if

there are any negative values in the dataset, assuming that negative values are not expected in certain variables. This step helps to ensure data integrity and identify potential issues with data entry or coding.

Finally, the code writes the generated analysis summaries into a text file named "data_exploration.txt". This file contains the data size, summary statistics, information about categorical variables, missing values, and investigation results for special numeric values. The file serves as a comprehensive record of the initial data exploration process.

Overall, the code provides valuable insights into the structure, distribution, and quality of the dataset, helping to inform subsequent data analysis and modeling processes.

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
Series([], dtype: int64)

# Investigation for Special Numeric Values
Series([], dtype: int64)
```

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
import warnings
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import ElasticNet
from sklearn.svm import SVR
from sklearn.exceptions import ConvergenceWarning
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error
from scipy.stats import ttest_rel

# To ignore ConvergenceWarning during model training
warnings.filterwarnings("ignore", category=ConvergenceWarning)

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

# DATASET PREPARATIONS
# No dataset preparations are needed.

# DESCRIPTIVE STATISTICS
# No descriptive statistics table is needed.

# PREPROCESSING
data['gender'] = data['sex'].map({0: 'female', 1:'male'})  # create a new series
data = pd.get_dummies(data)  # create dummy variables

# ANALYSIS
X = data[['gender_male', 'age_c', 'ht', 'wt']]
y = data['tube_depth_G']
```

```
tube = data['tube']

X_train, X_test, y_train, y_test, tube_train, tube_test =
↪  train_test_split(X, y, tube, test_size=0.3,
↪  random_state=42)

ml_models = {'Random Forest': RandomForestRegressor(),
↪  'Elastic Net': ElasticNet(), 'SVM': SVR(), 'Neural
↪  Network': MLPRegressor(max_iter=1000)}
formula_predictions = [X_test['ht'] / 10 + 5,
↪  9+np.clip(X_test['age_c'], 0, 4) + 0.5 *
↪  np.clip(X_test['age_c'] - 2, 0, np.inf), 3*tube_test]
formula_names = ['Height', 'Age', 'ID']

table1 = {}
for model in ml_models:
    regressor = ml_models[model]
    regressor.fit(X_train, y_train)
    y_pred = regressor.predict(X_test)
    table1[model] = mean_squared_error(y_test, y_pred)

df1 = pd.DataFrame.from_dict(table1, orient='index',
↪  columns=['Mean Squared Error'])
df1.to_pickle('table_1.pkl')

table2 = {}
for i in range(len(formula_predictions)):
    table2[formula_names[i]] = mean_squared_error(y_test,
    ↪  formula_predictions[i])

df2 = pd.DataFrame.from_dict(table2, orient='index',
↪  columns=['Mean Squared Error'])
df2.to_pickle('table_2.pkl')

p_values = {}
for model1 in table1:
    for i in range(len(formula_predictions)):
        ttest_result = ttest_rel(y_test -
        ↪  ml_models[model1].predict(X_test), y_test -
        ↪  formula_predictions[i])
```

```
        p_values[model1 + ' vs ' + formula_names[i]] =
        ↪ ttest_result.pvalue

df3 = pd.DataFrame.from_dict(p_values, orient='index',
↪ columns=['p-values'])
df3.to_pickle('table_3.pkl')

# SAVE ADDITIONAL RESULTS
additional_results = {
 'Total number of observations': len(data)
}
for model in ml_models:
    additional_results[model + ' R-squared'] =
    ↪ ml_models[model].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 data analysis on a dataset of pediatric patients who received post-operative mechanical ventilation. The aim of the analysis is to determine the optimal tracheal tube depth (OTTD) for these patients.

The code begins by importing the necessary libraries and ignoring convergence warnings during model training. It then loads the dataset from a CSV file.

Next, the dataset is prepared for analysis. Gender is converted into a categorical variable by creating a new series, and dummy variables are created for categorical features. This ensures that all variables are in a suitable format for model training.

The analysis is performed using machine learning models and formula-based predictions. The dataset is split into train and test sets, and the independent variables (features) and the target variable (tube_depth_G) are identified.

Four machine learning models - Random Forest, Elastic Net, Support Vector Machine (SVM), and Neural Network - are used to train regression models on the training data. The models are then used to predict the tube depth on the test data, and the mean squared error (MSE) between the predicted and actual tube depths is calculated for each model.

Formula-based predictions are also calculated using three different formulas based on height, age, and tube ID. The MSE between the formula-based predictions and the actual tube depths is calculated.

The MSE results for both the machine learning models and the formula-based predictions are saved in separate dataframes, which are then serialized and saved as pickle files ('table_1.pkl' and 'table_2.pkl').

Additionally, t-tests are performed to compare the performance of each machine learning model against each formula-based prediction. The p-values resulting from the t-tests are stored in a dataframe and serialized as a pickle file ('table_3.pkl').

Finally, additional results including the total number of observations and the R-squared values for each machine learning model on the test data are stored in a dictionary and serialized as a pickle file ('additional_results.pkl').

The 'additional_results.pkl' file contains information about the performance of the machine learning models and provides additional context to the analysis.

Overall, the code implements machine learning models and formula-based predictions to determine the optimal tracheal tube depth for pediatric patients using a dataset of patient features and actual tube depths as determined by chest X-ray.

C.3 Code Output

table_1.pkl

	Mean Squared Error
Random Forest	1.537771
Elastic Net	1.252317
SVM	1.275379
Neural Network	1.405439

table_2.pkl

	Mean Squared Error
Height	3.483762
Age	1.888213
ID	2.344570

table_3.pkl

	p-values
Random Forest vs Height	3.911e-81
Random Forest vs Age	7.14e-06
Random Forest vs ID	4.446e-43
Elastic Net vs Height	1.169e-109
Elastic Net vs Age	8.037e-13
Elastic Net vs ID	1.902e-54
SVM vs Height	6.378e-115
SVM vs Age	4.165e-12
SVM vs ID	3.849e-55
Neural Network vs Height	2.873e-128
Neural Network vs Age	0.1783
Neural Network vs ID	2.22e-65

additional_results.pkl

```
{
    'Total number of observations': 969,
    'Random Forest R-squared': 0.5163           ,
    'Elastic Net R-squared': 0.6061           ,
    'SVM R-squared': 0.5988           ,
    'Neural Network R-squared': 0.5579           ,
}
```

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 typing import Dict, Any, Optional, Tuple
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 = {
```

```
    'Mean Squared Error': ('MSE', 'Mean Squared Error, value closer to zero is better'),
    'SVM': ('Support Vector Machine', None),
    'Random Forest': ('Random Forest Regressor', None),
    'Elastic Net': ('Elastic Net Regressor', None),
    'ID': ('Tube ID', 'internal diameter of the tracheal tube in millimeter'),
    'Height': ('Patient Height', 'Height of the patient in cm'),
    'Age': ('Patient Age', 'Age of the patient in years')
}

# TABLE 1:
df1 = pd.read_pickle('table_1.pkl')

# RENAME ROWS AND COLUMNS
mapping1 = {k: v for k, v in shared_mapping.items() if is_str_in_df(df1, k)}
abbrs_to_names1, legend1 = split_mapping(mapping1)
df1 = df1.rename(columns=abbrs_to_names1, index=abbrs_to_names1)

# Save as latex
to_latex_with_note(
    df1, 'table_1.tex',
    caption="Mean Squared Error for each Machine Learning Model",
    label='table:ml_mse',
    note="Machine Learning Models include: Random Forest, Elastic Net, Support Vector Machine, Neural Network",
    legend=legend1)


# TABLE 2:
df2 = pd.read_pickle('table_2.pkl')

# RENAME ROWS AND COLUMNS
mapping2 = {k: v for k, v in shared_mapping.items() if is_str_in_df(df2, k)}
abbrs_to_names2, legend2 = split_mapping(mapping2)
```

```
df2 = df2.rename(columns=abbrs_to_names2, index=abbrs_to_names2)

# Save as latex
to_latex_with_note(
    df2, 'table_2.tex',
    caption="Mean Squared Error for each Formula-Based Model",
    label='table:formula_mse',
    note="Formula-Based Models include: ID (Tube ID Model), Height (Height Formula Model), Age (Age Formula Model)",
    legend=legend2)


# TABLE 3:
df3 = pd.read_pickle('table_3.pkl')

# FORMAT P-VALUES
df3 = df3.applymap(format_p_value)

# SAVE AS LATEX
to_latex_with_note(
    df3, 'table_3.tex',
    caption="Paired T-Test Comparison between Machine Learning and Formula-Based Model Predictions",
    label='table:p_values',
    note="P-values formatted to be less than 1e-06 if it's less than 1e-06")


```

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{Mean Squared Error for each Machine Learning Model}
```

```
\label{table:ml_mse}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{lr}
\toprule
 & MSE \\
\midrule
\textbf{Random Forest Regressor} & 1.54 \\
\textbf{Elastic Net Regressor} & 1.25 \\
\textbf{Support Vector Machine} & 1.28 \\
\textbf{Neural Network} & 1.41 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item Machine Learning Models include: Random Forest, Elastic Net, Support
    Vector Machine, Neural Network
\item \textbf{MSE}: Mean Squared Error, value closer to zero is better
\end{tablenotes}
\end{threeparttable}
\end{table}
```

table_2.tex

```
\begin{table}[h]
\caption{Mean Squared Error for each Formula-Based Model}
\label{table:formula_mse}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{lr}
\toprule
 & MSE \\
\midrule
\textbf{Patient Height} & 3.48 \\
\textbf{Patient Age} & 1.89 \\
\textbf{Tube ID} & 2.34 \\
\bottomrule
```

```
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item Formula-Based Models include: ID (Tube ID Model), Height (Height Formula
    Model), Age (Age Formula Model)
\item \textbf{MSE}: Mean Squared Error, value closer to zero is better
\item \textbf{Tube ID}: internal diameter of the tracheal tube in millimeter
\item \textbf{Patient Height}: Height of the patient in cm
\item \textbf{Patient Age}: Age of the patient in years
\end{tablenotes}
\end{threeparttable}
\end{table}
```

table_3.tex

```
\begin{table}[h]
\caption{Paired T-Test Comparison between Machine Learning and Formula-Based
    Model Predictions}
\label{table:p_values}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{ll}
\toprule
 & p-values \\
\midrule
\textbf{Random Forest vs Height} & $<$1e-06 \\
\textbf{Random Forest vs Age} & 7.14e-06 \\
\textbf{Random Forest vs ID} & $<$1e-06 \\
\textbf{Elastic Net vs Height} & $<$1e-06 \\
\textbf{Elastic Net vs Age} & $<$1e-06 \\
\textbf{Elastic Net vs ID} & $<$1e-06 \\
\textbf{SVM vs Height} & $<$1e-06 \\
\textbf{SVM vs Age} & $<$1e-06 \\
\textbf{SVM vs ID} & $<$1e-06 \\
\textbf{Neural Network vs Height} & $<$1e-06 \\
\textbf{Neural Network vs Age} & 0.178 \\
\textbf{Neural Network vs ID} & $<$1e-06 \\
\bottomrule
```

```
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item P-values formatted to be less than 1e-06 if it's less than 1e-06
\end{tablenotes}
\end{threeparttable}
\end{table}
```