Accurate Estimation of Tracheal Tube Depth in Pediatric Patients using Machine Learning Models

Data to Paper

February 20, 2024

Abstract

Determining the optimal tracheal tube depth (OTTD) is crucial in pediatric patients undergoing mechanical ventilation. Existing methods for estimating OTTD have limitations, necessitating the exploration of improved models. This study conducts a comparative analysis of Random Forest (RF) and Elastic Net (EN) models for OTTD estimation in a dataset of 969 pediatric patients who underwent postoperative mechanical ventilation. The RF and EN models leverage patient features including age, height, weight, and sex to estimate OTTD. Both models demonstrate accurate estimations with mean squared residuals of 1.48 cm and 1.35 cm, respectively. The comparative analysis using a paired t-test reveals no significant difference in performance between the RF and EN models. Accurate estimation of OTTD can reduce complications and improve patient outcomes. Further validation and enhancements of the models are needed, advancing the use of machine learning in clinical decision-making for pediatric patients requiring mechanical ventilation.

Results

To evaluate the performance of the models for estimating tracheal tube depth in pediatric patients, we conducted a comparative analysis using the Random Forest (RF) and Elastic Net (EN) models. The goal was to determine their accuracy in estimating the optimal tracheal tube depth (OTTD) based on patient features such as sex, age, height, and weight.

First, to understand the performance of the RF model, we constructed and tuned the model using a grid search approach. The RF model was trained and evaluated using a 5-fold cross-validation method. The optimal

parameters for the RF model were found to be a maximum depth of 5, minimum samples split of 10, and 200 estimators. The mean squared residual for the RF model was 1.48 cm, indicating that, on average, the RF model's predictions deviated from the true OTTD by 1.48 cm (Table 1).

Table 1: Performance of the Random Forest Model

	Model	Mean Squared Residual
RF	Random Forest	1.48

RF: Random Forest model

Next, we examined the performance of the EN model. Similar to the RF model, the EN model was trained and evaluated using a 5-fold cross-validation method. The best parameters for the EN model were found to be an alpha value of 0.1 and an L1 ratio of 0.2. The mean squared residual for the EN model was 1.35 cm, indicating that, on average, the EN model's predictions deviated from the true OTTD by 1.35 cm (Table 2).

Table 2: Performance of the Elastic Net Model

	Model	Mean Squared Residual
EN	Elastic Net	1.35

EN: Elastic Net model

Finally, we compared the performance of the RF and EN models using a paired t-test. The paired t-test was conducted using the mean squared residuals of both models as the dependent variable. The t-statistic was 1.32, with a p-value of 0.188 (Table 3). These results indicate that there is no significant difference in performance between the RF and EN models.

Table 3: Comparative Performance of the Random Forest and Elastic Net Models

	Compared Models	t-statistic	p-value
Comparison	RF vs EN	1.32	0.188

RF denotes Random Forest model, EN denotes Elastic Net model.
Compared Models: Pair of models compared
t-statistic: Statistical measure for the Hypothesis Test
p-value: Significance measure for the Hypothesis Test

In this study, we analyzed a dataset of 969 pediatric patients who un-

derwent post-operative mechanical ventilation. The RF and EN models demonstrated similar accuracy in estimating the tracheal tube depth, with mean squared residuals of 1.48 cm and 1.35 cm, respectively. These values represent the average difference between the predicted and true OTTD in centimeters.

In summary, our comparative analysis of the RF and EN models, using a dataset of 969 pediatric patients, showed that both models accurately estimated the tracheal tube depth. The RF and EN models achieved mean squared residuals of 1.48 cm and 1.35 cm, respectively. Furthermore, the paired t-test indicated no significant difference in performance between the two models. These results highlight the potential of both RF and EN models for improving clinical decision-making in determining the tracheal tube depth in pediatric patients requiring mechanical ventilation.

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
4  # Load dataset
5  df = pd.read_csv('tracheal_tube_insertion.csv')
6
7  # Open a new text file
8  with open('data_exploration.txt', 'w') as f:
9
10     # Data size
11     f.write('# Data Size\n')
12     f.write('Number of rows: ' + str(df.shape[0])+'\n')
13     f.write('Number of columns: ' + str(df.shape[1])+'\n\n')
14
15     # Summary statistics
16     f.write('# Summary Statistics\n')
17     f.write(df.describe().to_string()+'\n\n')
18
19     # Categorical variables
20     f.write('# Categorical Variables\n')
21     df_categorical = df.select_dtypes(include=['object'])
22     for col in df_categorical.columns:
23         f.write(str(col) + ' most common values: \n' +
```

```
                f.write(df_categorical[col].value_counts().head(5).to_string()
                + '\n\n')

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

    # Check for special numeric values that stand for unknown/undefined
    f.write('# Special Numeric Values that Stand for Unknown/Undefined\n')
    special_values = ['?', '-', 'N/A', 'NA', 'na', 'nan', 'NaN', 'Nan', '--', 'unknown',
                      'Unknown', 'undefined', 'Undefined', 'null', 'None', 'none', 'Null']
    for special in special_values:
        if df.isin([special]).sum().sum() > 0:
            f.write(df.isin([special]).sum().to_string()+'\n\n')

f.close()
```

B.2 Code Description

The purpose of the provided code is to perform data exploration on a given dataset and write the findings into a text file named "data_exploration.txt".

The code begins by loading the dataset from a CSV file into a pandas DataFrame. It then proceeds to perform several analysis steps:

1. Data Size: The code calculates the number of rows and columns in the dataset using the 'shape' attribute of the DataFrame and writes these values into the text file.

2. Summary Statistics: The code generates summary statistics for the dataset using the 'describe()' function of the DataFrame. This includes count, mean, standard deviation, minimum, quartiles, and maximum values for each numerical column in the dataset. The summary statistics are then written into the text file.

3. Categorical Variables: The code identifies the most common value of the 'sex' column in the dataset using the 'value_counts()' function and

writes this information, along with the number of occurrences, into the text file.

4. Missing Values: The code checks for missing values in the dataset by using the 'isnull()' function to generate a Boolean mask indicating missing values and then sums up the total number of missing values using the 'sum()' function. If there are missing values, the code writes the number of missing values into the text file; otherwise, it writes that there are no missing values.

The code writes the findings of each analysis step into a text file named "data_exploration.txt". This file serves as a record of the data exploration process and provides a summary of important information about the dataset, such as its size, summary statistics, the most common categorical variable value, and the presence of missing values. This information can later be referenced and used for further analysis or reporting purposes 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

# Special Numeric Values that Stand for Unknown/Undefined
```

C Data Analysis

C.1 Code

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

```

# IMPORT
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import ElasticNet
from sklearn.model_selection import GridSearchCV
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
# No descriptive statistics table is needed.

# PREPROCESSING
# No preprocessing is needed.

# ANALYSIS
# The analysis will comprise the construction of Random Forest and Elastic Net models,
# hyperparameter tuning, calculation of the mean squared residuals of the models, and
```

```
# comparative testing of the models using a paired t-test.

# Prepare data for model training
features = df[['sex', 'age_c', 'ht', 'wt']]
target = df['tube_depth_G']
X_train, X_test, y_train, y_test = train_test_split(features,
    ↪  target, test_size=0.2, random_state=0)

## Table 1: "Performance of the Random Forest Model"
# Construct and tune the Random Forest model
rf = RandomForestRegressor()
rf_params = {'n_estimators': [50, 100, 200], 'max_depth':
    ↪  [None, 5, 10], 'min_samples_split': [2, 5, 10]}
rf_grid = GridSearchCV(rf, rf_params, cv=5)
rf_grid.fit(X_train, y_train)

# Predict and calculate the mean squared residuals
rf_preds = rf_grid.predict(X_test)
rf_errors = (rf_preds - y_test) ** 2
mean_rf_error = rf_errors.mean()
df1 = pd.DataFrame({'Model': ['Random Forest'], 'Mean Squared
    ↪  Residual': [mean_rf_error]}, index=['RF'])
df1.to_pickle('table_1.pkl')

## Table 2: "Performance of the Elastic Net Model"
# Construct and tune the Elastic Net model
en = ElasticNet()
en_params = {'alpha': [0.1, 0.5, 1.0, 2.0], 'l1_ratio': [0.2,
    ↪  0.5, 0.8]}
en_grid = GridSearchCV(en, en_params, cv=5)
en_grid.fit(X_train, y_train)

# Predict and calculate the mean squared residuals
en_preds = en_grid.predict(X_test)
en_errors = (en_preds - y_test) ** 2
mean_en_error = en_errors.mean()
df2 = pd.DataFrame({'Model': ['Elastic Net'], 'Mean Squared
    ↪  Residual': [mean_en_error]}, index=['EN'])
df2.to_pickle('table_2.pkl')

```

```
## Table 3: "Comparative Performance of the Random Forest and Elastic Net Models"
# Compare the performance of the two models using a paired t-test
ttest_results = ttest_rel(rf_errors, en_errors)
df3 = pd.DataFrame({'Model Pair': ['RF vs EN'], 't-statistic': [ttest_results.statistic], 'p-value': [ttest_results.pvalue]}, index=['Comparison'])
df3.to_pickle('table_3.pkl')

# SAVE ADDITIONAL RESULTS
additional_results = {
 'Number of Observations': len(df),
 'Best Parameters for RF': rf_grid.best_params_,
 'Best Parameters for EN': en_grid.best_params_,
}
with open('additional_results.pkl', 'wb') as f:
    pickle.dump(additional_results, f)

```

C.2 Code Description

The code performs an analysis to determine the optimal tracheal tube depth (OTTD) for pediatric patients who require mechanical ventilation. The analysis involves constructing and evaluating two different models, namely Random Forest and Elastic Net, using patient features such as sex, age, height, and weight.

First, the dataset is loaded into a Pandas dataframe. Then, the data is split into training and testing sets using the train_test_split function from the sklearn library. The features and target variables are extracted from the dataframe to be used in the model training and evaluation.

The analysis includes three main steps: 1. Performance of the Random Forest Model: - The Random Forest model is constructed using the RandomForestRegressor class from the sklearn library. - Hyperparameter tuning is performed using GridSearchCV to find the optimal combination of parameters (n_estimators, max_depth, and min_samples_split). - The model is evaluated by predicting the target variable on the test set and calculating the mean squared residuals. - The mean squared residuals are saved in a dataframe, along with the name of the model (Random Forest), and stored in a pickle file (table_1.pkl).

2. Performance of the Elastic Net Model: - The Elastic Net model is constructed using the ElasticNet class from the sklearn library. - Hyperparameter tuning is performed using GridSearchCV to find the optimal combination of parameters (alpha and l1_ratio). - The model is evaluated by predicting the target variable on the test set and calculating the mean squared residuals. - The mean squared residuals are saved in a dataframe, along with the name of the model (Elastic Net), and stored in a pickle file (table_2.pkl).

3. Comparative Performance of the Random Forest and Elastic Net Models: - The performance of the Random Forest and Elastic Net models is compared using a paired t-test. - The mean squared residuals of both models are used as the paired samples. - The t-test is performed using the ttest_rel function from the scipy library. - The t-statistic and p-value are saved in a dataframe and stored in a pickle file (table_3.pkl).

Additionally, the code saves additional results in the "additional_results.pkl" file. This includes the number of observations in the dataset and the best parameters found for the Random Forest and Elastic Net models during hyperparameter tuning.

The analysis provides insights into the performance of different models in determining the optimal tracheal tube depth for pediatric patients. These results can contribute to improving patient safety and clinical decision-making in pediatric mechanical ventilation.

C.3 Code Output

table_1.pkl

	Model	Mean Squared Residual
RF	Random Forest	1.476233

table_2.pkl

	Model	Mean Squared Residual
EN	Elastic Net	1.349365

table_3.pkl

	Model Pair	t-statistic	p-value
Comparison	RF vs EN	1.322153	0.1877

additional_results.pkl

```
{
    'Number of Observations': 969,
    'Best Parameters for RF': {'max_depth': 5, 'min_samples_split': 10,
    'n_estimators': 200},
    'Best Parameters for EN': {'alpha': 0.1, 'l1_ratio': 0.2},
}
```

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, Tuple, Optional
from my_utils import to_latex_with_note, format_p_value,
↪  is_str_in_df, split_mapping, AbbrToNameDef

# PREPARATION FOR ALL TABLES
# Let's keep the shared mapping empty since these labels are
↪  not shared across the tables.
shared_mapping: AbbrToNameDef = {}

# TABLE 1:

# read table 1 data
df = pd.read_pickle('table_1.pkl')

# Prepare specific labels and definitions for table 1
table_1_specific_mapping: AbbrToNameDef = {
    'RF': ('RF', 'Random Forest model'),
}

# Merge shared mapping and specific mapping
mapping = {**shared_mapping, **table_1_specific_mapping}
abbrs_to_names, names_to_definitions = split_mapping(mapping)
```

```
25
26  # Rename columns and save as a LaTeX table
27  df = df.rename(columns=abbrs_to_names)
28  to_latex_with_note(df, 'table_1.tex', caption='Performance of
    ↪  the Random Forest Model',
29                     label='table:RF_Performance', note=None,
                       ↪  legend=names_to_definitions)
30
31
32  # TABLE 2:
33
34  # read table 2 data
35  df = pd.read_pickle('table_2.pkl')
36
37  # Prepare specific labels and definitions for table 2
38  table_2_specific_mapping: AbbrToNameDef = {
39      'EN': ('EN', 'Elastic Net model'),
40  }
41
42  # Merge shared mapping and specific mapping
43  mapping = {**shared_mapping, **table_2_specific_mapping}
44  abbrs_to_names, names_to_definitions = split_mapping(mapping)
45
46  # Rename columns and save as a LaTeX table
47  df = df.rename(columns=abbrs_to_names)
48  to_latex_with_note(df, 'table_2.tex', caption='Performance of
    ↪  the Elastic Net Model',
49                     label='table:EN_Performance', note=None,
                       ↪  legend=names_to_definitions)
50
51
52  # TABLE 3:
53
54  # read table 3 data
55  df = pd.read_pickle('table_3.pkl')
56
57  # Prepare specific labels and definitions for table 3
58  table_3_specific_mapping: AbbrToNameDef = {
59      'Model Pair': ('Compared Models', 'Pair of models
        ↪  compared'),
```

```
    't-statistic': ('t-statistic', 'Statistical measure for the Hypothesis Test'),
    'p-value': ('p-value', 'Significance measure for the Hypothesis Test'),
}

# Merge shared mapping and specific mapping
mapping = {**shared_mapping, **table_3_specific_mapping}
abbrs_to_names, names_to_definitions = split_mapping(mapping)

# Format p-values and rename columns
df['p-value'] = df['p-value'].apply(format_p_value)
df = df.rename(columns=abbrs_to_names)

# Save as a latex table
to_latex_with_note(df, 'table_3.tex', caption='Comparative Performance of the Random Forest and Elastic Net Models',
                   label='table:Model_Comparison', note='RF denotes Random Forest model, EN denotes Elastic Net model.',
                   legend=names_to_definitions)
```

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 of the Random Forest Model}
\label{table:RF_Performance}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{llr}
\toprule
```

```
 & Model & Mean Squared Residual \\
\midrule
\textbf{RF} & Random Forest & 1.48 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item \textbf{RF}: Random Forest model
\end{tablenotes}
\end{threeparttable}
\end{table}
```

table_2.tex

```
\begin{table}[h]
\caption{Performance of the Elastic Net Model}
\label{table:EN_Performance}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{llr}
\toprule
 & Model & Mean Squared Residual \\
\midrule
\textbf{EN} & Elastic Net & 1.35 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item \textbf{EN}: Elastic Net model
\end{tablenotes}
\end{threeparttable}
\end{table}
```

table_3.tex

```
\begin{table}[h]
\caption{Comparative Performance of the Random Forest and Elastic Net Models}
```

```
\label{table:Model_Comparison}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{llrl}
\toprule
 & Compared Models & t-statistic & p-value \\
\midrule
\textbf{Comparison} & RF vs EN & 1.32 & 0.188 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item RF denotes Random Forest model, EN denotes Elastic Net model.
\item \textbf{Compared Models}: Pair of models compared
\item \textbf{t-statistic}: Statistical measure for the Hypothesis Test
\item \textbf{p-value}: Significance measure for the Hypothesis Test
\end{tablenotes}
\end{threeparttable}
\end{table}
```