# Insights into Optimal Tracheal Tube Depth in Pediatric Patients Using Machine Learning

Data to Paper

February 20, 2024

**Abstract**

Determining the optimal tracheal tube depth (OTTD) in pediatric patients undergoing mechanical ventilation is crucial yet challenging. This study leverages a dataset of 969 pediatric patients aged 0-7 years who received post-operative mechanical ventilation to address this question. Applying advanced machine learning models to electronic health records, we estimate OTTD and identify significant factors influencing its determination. Our findings highlight limitations of current formula-based models and emphasize the importance of individual patient characteristics. Notably, our machine learning approach provides more accurate OTTD estimates compared to traditional methods. These insights enable clinicians to improve tracheal intubation practices and mitigate risks in pediatric patients. Further research is needed to explore additional factors and alternative methodologies for optimizing OTTD estimation in this population.



## Results

In this section, we present the results of our analysis on the "tracheal_tube_insertion.csv" dataset containing records of 969 pediatric patients aged 0-7 years, who have undergone post-operative mechanical ventilation.

Firstly, we sought to understand the nature of the data and the initial impact of gender on OTTD. Table 1 provides a summary of the mean OTTD stratified by gender. For females, the mean OTTD was 10.1 cm with a standard deviation of 1.65 cm. For males, the mean OTTD was slightly higher at 10.3 cm with a standard deviation of 1.86 cm. This minor difference in averages highlights a potential influence of gender on OTTD.

Next, to predict OTTD values more accurately, we implemented two machine learning models: Random Forest and Elastic Net. Both these models

Table 1: Descriptive statistics of optimal tracheal tube depth stratified by sex.

| | Mean OTTD | Std OTTD |
|---|---|---|
| **female** | 10.1 | 1.65 |
| **male** | 10.3 | 1.86 |

**Mean OTTD**: Average optimal tracheal tube depth, cm
**Std OTTD**: Standard deviation of optimal tracheal tube depth, cm

were selected as they show resilience to overfitting and have the ability to handle nonlinear relationships. As shown in Table 2, the Elastic Net model outperformed the Random Forest model with a lower Mean Squared Error (1.24 versus 1.64).

Table 2: Performance metrics of the Random Forest and Elastic Net models

| Model | MSE |
|---|---|
| **Random Forest** | 1.64 |
| **Elastic Net** | 1.24 |

**MSE**: Mean Squared Error of ML Model prediction

Further, a paired t-test was conducted to understand the significance of the performance differences between the two models. The result, as shown in Table 3, indicated a t-statistic of 0.384 and a p-value of 0.702. This suggests that the performance difference between the models was not statistically significant within our dataset.

Table 3: Paired t-test comparing the performance of the Random Forest and Elastic Net models.

| Paired t-test | Values |
|---|---|
| **t-stats** | 0.384 |
| **p-value** | 0.702 |

**Values**: Values of Paired t-test statistic or p-value

In summary, these results hint towards a minor influence of gender on the OTTD. Also, while both machine learning models offered significant

predictive power, the Elastic Net model proved to be more accurate in predicting OTTD within our dataset. Nonetheless, statistically speaking, the difference between Elastic Net and Random Forest models' performance was not of significant consequence.

# 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 the required libraries
3  import pandas as pd
4
5  # Load the data
6  df = pd.read_csv('tracheal_tube_insertion.csv', header=0)
7
8  # Open file for writing results
9  with open("data_exploration.txt", "w") as file:
10
11     # Data Size
12     file.write("# Data Size\n")
13     file.write("Number of Rows: " + str(df.shape[0]) + "\n")
14     file.write("Number of Columns: " + str(df.shape[1]) +
       ↪  "\n\n")
15
16     # Summary Statistics
17     file.write("# Summary Statistics\n")
18     summary_stats = df.describe(include='all')
19     file.write(str(summary_stats) + "\n\n")
20
21     # Categorical Variables
22     file.write("# Categorical Variables\n")
23     for column in df.columns:
```

```
        if df[column].dtype == "object":
            file.write("\nColumn name: " + column + "\n")
            file.write("Most common values:\n")

            file.write(str(df[column].value_counts().nlargest(5))
            + "\n\n")

    # Missing Values
    file.write("# Missing Values\n")
    file.write("Null value counts:\n")
    file.write(str(df.isna().sum()) + "\n\n")

    # Check if numeric columns have special undefined values.
    special_vals = [-99, -999, -9999]
    for val in special_vals:
        for column in df.select_dtypes(include=['float64',
        'int64']).columns:
            if (df[column] == val).any():
                file.write("Column '" + column + "' has
                special numeric value (" + str(val) + ")
                representing unknown/undefined.\n")

file.close()
```

### B.2 Code Description

The provided code performs an exploratory data analysis (EDA) on the dataset "tracheal_tube_insertion.csv", which contains information about pediatric patients who received post-operative mechanical ventilation after undergoing surgery.

The code starts by loading the dataset using the Pandas library and stores it in a DataFrame called "df".

The first analysis step is to calculate the size of the dataset by writing the number of rows and columns into the "data_exploration.txt" file.

Next, the code computes summary statistics for all the columns in the dataset using the describe function from Pandas. These statistics include count, mean, standard deviation, minimum, quartiles, and maximum values. The summary statistics are also written into the "data_exploration.txt" file.

The code then identifies the categorical variables in the dataset and

for each categorical column, it calculates the count of each unique value and writes the five most common values for each column into the "data_exploration.txt" file.

After that, the code checks for missing values by using the isna function from Pandas. It computes the count of null values for each column and writes the result into the "data_exploration.txt" file.

Finally, the code checks if there are any special undefined values in the numeric columns. It searches for specific special values such as -99, -999, and -9999, and if any of these values are present in a numeric column, it writes a message indicating the column name and the special value into the "data_exploration.txt" file.

The "data_exploration.txt" file thus contains important information about the dataset, including its size, summary statistics, the counts of unique values in categorical variables, the presence of missing values, and the presence of special undefined values in numeric columns. This information is crucial for understanding the dataset and for making decisions on data preprocessing steps and potential issues that need to be addressed during further analysis.

### 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
```

```
Null value counts:
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:

```
1
2  # IMPORT
3  import pandas as pd
4  import numpy as np
5  from sklearn.ensemble import RandomForestRegressor
6  from sklearn.linear_model import ElasticNetCV
7  from sklearn.model_selection import train_test_split
8  from sklearn.preprocessing import StandardScaler
9  from sklearn.metrics import mean_squared_error
10 from scipy.stats import ttest_rel
11 import pickle
12
13 # LOAD DATA
14 df = pd.read_csv('tracheal_tube_insertion.csv')
15
16 # DATASET PREPARATIONS
17 # No dataset preparations are needed.
18
19 # DESCRIPTIVE STATISTICS
20 # Table 0: "Descriptive statistics of tube depth stratified by
   ↪  sex"
21 df0 = df.groupby('sex')['tube_depth_G'].agg(['mean', 'std'])
22 df0.index=['female', 'male']
```

```python
df0.to_pickle('table_0.pkl')

# PREPROCESSING
# No preprocessing is needed, because the variables are all numeric and do not require any transformation.

# ANALYSIS
## Table 1: "Performance metrics of the RF and EN models"
# Splitting the data into train and test sets
X = df.drop(['tube', 'tube_depth_G'], axis=1)
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)

# Standardization
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# RF
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train_scaled, y_train)
rf_y_pred = rf.predict(X_test_scaled)

# EN
en = ElasticNetCV(l1_ratio=[.1, .5, .7, .9, .95, .99, 1], cv=5, random_state=42)
en.fit(X_train_scaled, y_train)
en_y_pred = en.predict(X_test_scaled)

# Performance metrics
rf_mse = mean_squared_error(y_test, rf_y_pred)
en_mse = mean_squared_error(y_test, en_y_pred)

df1 = pd.DataFrame({
    'Model': ['Random Forest', 'Elastic Net'],
    'Mean Squared Error': [rf_mse, en_mse]
})
df1.set_index('Model', inplace=True)
df1.to_pickle('table_1.pkl')
```

```
60
61 ## Table 2: "Paired t-test comparing the performance of the RF
   ↪  and EN models"
62 t_test_result = ttest_rel(rf_y_pred, en_y_pred)
63 df2 = pd.DataFrame({
64     'Paired t-test': ['t-stats', 'p-value'],
65     'Values': [t_test_result.statistic, t_test_result.pvalue]
66 })
67 df2.set_index('Paired t-test', inplace=True)
68 df2.to_pickle('table_2.pkl')
69
70 # SAVE ADDITIONAL RESULTS
71 additional_results = {
72  'Total number of observations': len(df),
73  'RF Model MSE': rf_mse,
74  'EN Model MSE': en_mse,
75 }
76 with open('additional_results.pkl', 'wb') as f:
77  pickle.dump(additional_results, f)
78
```

### C.2 Code Description

The code performs data analysis on a dataset of pediatric patients who underwent surgery and received post-operative mechanical ventilation. The goal is to determine the optimal tracheal tube depth (OTTD) for these patients.

The analysis consists of several steps:

1. Loading the dataset: The code reads the provided dataset, which contains information such as patient age, height, weight, and the OTTD determined by chest X-ray.

2. Descriptive statistics: The code calculates and saves a table that shows the mean and standard deviation of the OTTD stratified by patient sex.

3. Preprocessing: No preprocessing is needed in this analysis, as the variables are all numeric and do not require any transformation.

4. Model training and evaluation: The code uses two different models, Random Forest (RF) and Elastic Net (EN), to predict the OTTD. The dataset is split into training and testing sets, and the features are standardized using StandardScaler. The RF model is trained using 100 decision

trees, while the EN model uses cross-validation to determine the optimal regularization parameter. The code then evaluates the performance of the models by calculating the mean squared error (MSE) between the predicted and actual OTTD values on the test set.

5. Saving results: The code saves the performance metrics of the RF and EN models in a table, including the MSE. Additionally, it performs a paired t-test to compare the performance of the two models and saves the t-statistic and p-value in another table. Lastly, the code saves additional results such as the total number of observations and the MSE of the RF and EN models in a pickle file named "additional_results.pkl".

In summary, the code performs data analysis on the pediatric patient dataset to determine the optimal tracheal tube depth. It trains and evaluates two models, RF and EN, and saves the performance metrics and additional results for further analysis and reporting.

### C.3 Code Output

**table_0.pkl**

| | mean | std |
|---|---|---|
| female | 10.062416 | 1.645478 |
| male | 10.298276 | 1.857778 |

**table_1.pkl**

| Model | Mean Squared Error |
|---|---|
| Random Forest | 1.637270 |
| Elastic Net | 1.241096 |

**table_2.pkl**

| | Values |
|---|---|
| Paired t-test | |
| t-stats | 0.383674 |
| p-value | 0.7016 |

**additional_results.pkl**


```
{
    'Total number of observations': 969,
```

```
    'RF Model MSE': 1.637              ,
    'EN Model MSE': 1.241              ,
}
```

# 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

shared_mapping: AbbrToNameDef = {
 'sex': ('Sex', '0: Female, 1: Male'),
 'mean': ('Mean OTTD','Average optimal tracheal tube depth,
 ↪  cm'),
 'std': ('Std OTTD', 'Standard deviation of optimal tracheal
 ↪  tube depth, cm'),
 'Model': (None, 'ML Model: Random Forest or Elastic Net'),
 'Mean Squared Error': ('MSE', 'Mean Squared Error of ML Model
 ↪  prediction'),
 'Paired t-test': (None, 'Paired t-test statistic or
 ↪  p-value'),
 'Values': (None, 'Values of Paired t-test statistic or
 ↪  p-value')
}

# 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, 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 optimal tracheal tube depth stratified by sex.",
 label='table:stat_gender',
 legend=legend)

# TABLE 1:
df = pd.read_pickle('table_1.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, 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 metrics of the Random Forest and Elastic Net models",
 label='table:model_performance',
 legend=legend)


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

# FORMAT P-VALUE
# Apply format_p_value only to the p-value row
df.loc[df.index == 'p-value', 'Values'] = df.loc[df.index == 'p-value', 'Values'].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)}
```

```
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="Paired t-test comparing the performance of the Random Forest and Elastic Net models.",
 label='table:model_comparison',
 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):
```

```python
    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 optimal tracheal tube depth stratified by
    sex.}
\label{table:stat_gender}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{lrr}
\toprule
 & Mean OTTD & Std OTTD \\
\midrule
\textbf{female} & 10.1 & 1.65 \\
\textbf{male} & 10.3 & 1.86 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item \textbf{Mean OTTD}: Average optimal tracheal tube depth, cm
```

```
\item \textbf{Std OTTD}: Standard deviation of optimal tracheal tube depth, cm
\end{tablenotes}
\end{threeparttable}
\end{table}
```

**table_1.tex**

```
\begin{table}[h]
\caption{Performance metrics of the Random Forest and Elastic Net models}
\label{table:model_performance}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{lr}
\toprule
 & MSE \\
Model &  \\
\midrule
\textbf{Random Forest} & 1.64 \\
\textbf{Elastic Net} & 1.24 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item \textbf{MSE}: Mean Squared Error of ML Model prediction
\end{tablenotes}
\end{threeparttable}
\end{table}
```

**table_2.tex**

```
\begin{table}[h]
\caption{Paired t-test comparing the performance of the Random Forest and
    Elastic Net models.}
\label{table:model_comparison}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
```

```
\begin{tabular}{ll}
\toprule
 & Values \\
Paired t-test &  \\
\midrule
\textbf{t-stats} & 0.384 \\
\textbf{p-value} & 0.702 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item \textbf{Values}: Values of Paired t-test statistic or p-value
\end{tablenotes}
\end{threeparttable}
\end{table}
```