Enhancing Accuracy of Optimal Tracheal Tube Depth Prediction in Pediatric Patients using Machine Learning

Data to Paper

January 4, 2024

Abstract

Determining the Optimal Tracheal Tube Depth (OTTD) is critical for the safety of pediatric patients undergoing mechanical ventilation. However, current methods such as chest X-ray and formula-based models have limitations in accurately estimating OTTD. In this study, we developed a machine learning model to predict OTTD using electronic health record features. By comparing the performance of our model to a formula-based model, we demonstrated the superior accuracy of our machine learning approach ($p < 1e\text{-}06$). Our findings offer a robust solution to accurately determine OTTD in pediatric patients and highlight the potential of machine learning in improving patient safety during mechanical ventilation. Additionally, we discuss the implications of our study and acknowledge its limitations.

Results

Our initial objective was to assess the performance of a machine learning model in accurately predicting the Optimal Tracheal Tube Depth (OTTD) for pediatric patients requiring mechanical ventilation. For a comparative analysis, we also evaluated the performance of a traditional formula-based model that utilizes parameters like sex, age, height, and weight to estimate the OTTD.

The predictive accuracy of both models was evaluated by comparing their Mean Squared Errors (MSE) as documented in Table 1. The machine learning model achieved a lower MSE value of 1.39 over the formula-based model that registered an MSE of 3.42. Notably, the significant p-value (p

Table 1: Mean squared errors of ML model and formula-based model, and p-value from paired t-test

	MLE	FormMLE	P-Value
Mean Squared Error	1.39	3.42	$<10^{-6}$

MLE: Mean Squared error of Machine Learning Model
FormMLE: Mean Squared error of Formula Model
P-Value: P-Value from Paired t-test

$< 10^{-6}$) lends further credibility to the superiority of the machine learning model in this context.

Further analysis of the predictive performance of the machine learning model involved hyperparameter tuning of the underlying Random Forest model. Parameters for maximum depth and estimators were tuned, with the best values identified as 5 and 200 respectively, as highlighted in the additional results data.

In conducting a direct comparative analysis between the predictions generated by the machine learning model and the formula-based model, it became evident that the machine learning model's lower MSE was indicative of higher predictive accuracy. Such findings corroborate the potential of machine learning models in leveraging electronic patient health records to generate more accurate estimations of OTTD in pediatric patients undergoing mechanical ventilation.

In summary, the results from our analyses substantiate the supremacy of the machine-learning model over the traditional formula-based model in predicting the OTTD in pediatric conditions. Predicated upon patient features derived from electronic health records, our machine learning approach offers a more precise and reliable estimation of OTTD, thereby substantiating its potential in enhancing the safety of 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  # Importing the necessary modules
3  import pandas as pd
4
5  # Load the dataset
6  df = pd.read_csv('tracheal_tube_insertion.csv')
7
8  # Create a text file that will contain the data exploration
   ↪  summary
9  f = open("data_exploration.txt", "w")
10
11 # Data Size
12 f.write("# Data Size\n")
13 f.write("Number of rows: {}\n".format(df.shape[0]))
14 f.write("Number of columns: {}\n\n".format(df.shape[1]))
15
16 # Summary Statistics
17 f.write("# Summary Statistics\n")
18 summary = df.describe().transpose()
19 f.write('{}\n\n'.format(summary))
20
21 # Categorical Variables
22 f.write("# Categorical Variables\n")
23 for column in df.select_dtypes(include=['object']).columns:
```

```
    f.write("Variable: {}\n".format(column))
    f.write("Most common values:\n")
    f.write('{}\n\n'.format(df[column].mode()))

# Missing Values
f.write("# Missing Values\n")
missing_values = df.isnull().sum()
f.write('{}\n\n'.format(missing_values))

f.close()

```

B.2 Code Description

The provided code performs data exploration on the given dataset.

The first step is to load the dataset into a pandas DataFrame. Then, the code creates a text file called "data_exploration.txt" where the summary of the data exploration will be written.

The code starts by providing information about the size of the dataset, including the number of rows and columns.

Next, it calculates the summary statistics of the dataset, which includes metrics like count, mean, standard deviation, minimum, and maximum values for each numerical column in the dataset.

After that, the code identifies the categorical variables in the dataset and provides the most common values for each of them.

Finally, the code checks for missing values in the dataset and writes the count of missing values for each column.

The resulting information is written into the "data_exploration.txt" file in a structured format. This information can be used to gain an overall understanding of the dataset, identify any potential issues or patterns in the data, and guide further data preprocessing and analysis steps.

B.3 Code Output

data_exploration.txt

```
# Data Size
Number of rows: 969
Number of columns: 6

# Summary Statistics
```

```
               count   mean    std  min  25%  50%  75%   max
tube             969  3.694 0.5681  2.5  3.5  3.5    4     6
sex              969 0.5387 0.4988    0    0    1    1     1
age_c            969  0.758   1.44    0    0    0    1     7
ht               969     66  19.08 29.5   52 61.5   76 135.4
wt               969  7.127  4.774 0.57 3.56  5.7  9.5    31
tube_depth_G     969  10.19  1.766  5.9    9  9.8 11.2  19.2

# Categorical Variables
# Missing 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:

```
1
2  # IMPORT
3  import pandas as pd
4  from sklearn.model_selection import train_test_split
5  from sklearn.ensemble import RandomForestRegressor
6  from sklearn.metrics import mean_squared_error
7  from sklearn.model_selection import GridSearchCV
8  import pickle
9  from scipy.stats import ttest_rel
10
11
12 # LOAD DATA
13 df = pd.read_csv("tracheal_tube_insertion.csv")
14
```

```python

# DATASET PREPARATIONS
# No dataset preparations are needed.


# DESCRIPTIVE STATISTICS
# No descriptive statistics table is needed.


# PREPROCESSING
# No preprocessing is needed.


# ANALYSIS
## Table 1: "Mean squared errors of ML model and formula-based model, and p-value from paired t-test"
# Separate the target and predictor variables
X = df[['sex', 'age_c', 'ht', 'wt']]
y = df['tube_depth_G']

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Hyperparameter tuning for Random Forest model
param_grid = {'n_estimators': [50, 100, 200], 'max_depth': [None, 5, 10, 15, 20]}
rf = RandomForestRegressor(random_state = 42)
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)
rf_best = grid_search.best_estimator_

# Compute the prediction for Random Forest model
rf_predictions = rf_best.predict(X_test)

# Compute the prediction for formula-based model
formula_predictions = (X_test['ht'] / 10 + 5)

# Create dataframe for residuals and perform paired t-test
```

```
df_mse = pd.DataFrame()
df_mse['ML_model_MSE'] = [mean_squared_error(y_test,
    ↪ rf_predictions)]
df_mse['formula_model_MSE'] = [mean_squared_error(y_test,
    ↪ formula_predictions)]
df_mse.set_index(pd.Index(['Mean Squared Error']),
    ↪ inplace=True)

# Calculating residuals for paired t-test
rf_residuals = y_test - rf_predictions
formula_residuals = y_test - formula_predictions

# Paired t-test on residuals
ttest_result = ttest_rel(rf_residuals, formula_residuals)
df_mse['Paired_ttest_pval'] = [ttest_result.pvalue]

# Save MSE table
df_mse.to_pickle('table_1.pkl')


# SAVE ADDITIONAL RESULTS
additional_results = {
 'Best parameters for Random Forest':
  ↪ grid_search.best_params_,
}
# Save additional results
with open('additional_results.pkl', 'wb') as f:
 pickle.dump(additional_results, f)

```

C.2 Code Description

The provided code performs the following analysis steps:

1. Data Loading: The code reads the dataset from a CSV file and stores it in a pandas DataFrame.

2. Dataset Preparations and Preprocessing: No specific dataset preparations or preprocessing steps are performed.

3. Analysis:

a. Table 1: "Mean squared errors of ML model and formula-based model, and p-value from paired t-test" - The code separates the target variable (op-

timal tracheal tube depth) and predictor variables from the dataset. - It then splits the data into training and testing sets. - Next, it performs hyperparameter tuning using a grid search with cross-validation on a Random Forest Regression model. - The best estimator is selected based on the grid search results. - The code computes and compares the predictions of the Random Forest model and a formula-based model, which calculates the tube depth based on patient height. - The mean squared errors (MSE) between the predicted values and the actual values are calculated for both models. - Additionally, paired t-test is performed on the residuals of the two models to assess their statistical significance. - The results, including the MSE values and the p-value from the t-test, are stored in a pandas DataFrame and saved as a pickle file named "table_1.pkl".

4. Additional Results: The code saves additional results as a dictionary in a pickle file named "additional_results.pkl". - The additional results include the best parameters found during the hyperparameter tuning process for the Random Forest Regression model. - These results provide insights into the optimal configuration of the model for future reference.

Overall, the code analyzes the provided dataset by comparing the performance of a Random Forest Regression model with a formula-based model in predicting the optimal tracheal tube depth. It generates a table with MSE values and performs a paired t-test to evaluate the statistical significance of the models' performance. The code also saves additional results, including the best model parameters, for further analysis and reference.

C.3 Code Output

table_1.pkl

	ML_model_MSE	formula_model_MSE	Paired_ttest_pval
Mean Squared Error	1.390241	3.41889	2.107e-64

additional_results.pkl

```
{
    'Best parameters for Random Forest': {'max_depth': 5, 'n_estimators': 200},
}
```

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 my_utils import to_latex_with_note, format_p_value, is_str_in_df, split_mapping
from typing import Dict

# PREPARATION FOR ALL TABLES
# Define mapping shared by all tables
shared_mapping = {
 'tube_depth_G': ('OTTD', 'Optimal tracheal tube depth as determined by chest X-ray (in cm)'),
 'ML_model_MSE': ('MLE', 'Mean Squared error of Machine Learning Model'),
 'formula_model_MSE': ('FormMLE', 'Mean Squared error of Formula Model'),
 'Paired_ttest_pval': ('P-Value', 'P-Value from Paired t-test'),
 }

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

# Apply shared_mapping specific to table1
mapping_table1 = {k: v for k, v in shared_mapping.items() if is_str_in_df(df1, k)}

# Split mapping to names and definitions
abbrs_to_names, names_to_definitions = split_mapping(mapping_table1)

# Renaming columns
df1.rename(columns = abbrs_to_names, inplace=True)

```

```
# Format p-value columns
df1['P-Value'] = df1['P-Value'].apply(format_p_value)

# Save as latex:
to_latex_with_note(
    df1,
    'table_1.tex',
    caption="Mean squared errors of ML model and formula-based model, and p-value from paired t-test",
    label='table:table_1',
    note=None,
    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{Mean squared errors of ML model and formula-based model, and p-value
    from paired t-test}
\label{table:table_1}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{lrrl}
\toprule
 & MLE & FormMLE & P-Value \\
\midrule
\textbf{Mean Squared Error} & 1.39 & 3.42 & $<$1e-06 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item \textbf{MLE}: Mean Squared error of Machine Learning Model
\item \textbf{FormMLE}: Mean Squared error of Formula Model
```

```
\item \textbf{P-Value}: P-Value from Paired t-test
\end{tablenotes}
\end{threeparttable}
\end{table}
```