Machine Learning Predicts Optimal Tracheal Tube Depth in Pediatric Patients

Data to Paper

January 3, 2024

Abstract

Accurate determination of the Optimal Tracheal Tube Depth (OTTD) is crucial for pediatric patients undergoing post-operative mechanical ventilation to prevent complications. Formula-based models and chest X-rays have limited success in precisely estimating OTTD, indicating the need for an improved approach. To address this, we developed machine learning models, Random Forest (RF) and Elastic Net (EN), to predict OTTD using electronic health records of 969 pediatric patients aged 0-7 years. The RF and EN models achieved accurate predictions of OTTD, as demonstrated by low mean squared errors. Furthermore, a paired t-test revealed comparable predictive power between the RF and EN models. These results highlight the potential of machine learning in improving tracheal tube placement for enhanced post-operative mechanical ventilation outcomes. However, validation in larger cohorts and real-time clinical assessment are needed to further evaluate the generalizability and performance of these models.

Results

The objective of this study was to develop machine learning models, specifically Random Forest (RF) and Elastic Net (EN), for predicting optimal tracheal tube depth (OTTD) in pediatric patients undergoing post-operative mechanical ventilation, with data from 969 patients included in the analysis. Both RF and EN models were trained using a training set and their performance was subsequently evaluated on a test set.

The prediction errors of the trained models were quantified using Mean Squared Error (MSE), a measure where lower values indicate better performance due to smaller discrepancies between the model's predictions and the observed OTTD values. The RF model achieved an MSE of 1.51 and

the EN model demonstrated slightly lower error with an MSE of 1.14, as depicted in Table 1. This comparison of the prediction errors indicates that both models can provide accurate predictions of OTTD, although the EN model shows marginally enhanced performance.

Table 1: Mean Squared Error Comparisons for Machine Learning Models

	Random Forest MSE	Elastic Net MSE
Mean Squared Residuals	1.51	1.14

Random Forest MSE: Mean Squared Error for Random Forest Model Predictions
Elastic Net MSE: Mean Squared Error for Elastic Net Model Predictions

Further comparison of the RF and EN models was accomplished through a paired t-test, with the test's results presented in Table 2. This test assessed whether there was a statistically significant difference in the predictive performance of the two models. A t-statistic value of -1.1 was obtained along with a consequent p-value of 0.273. The p-value, being greater than the commonly accepted significance level of 0.05, indicates no statistically significant difference in the predictive powers of the two models.

Table 2: Statistical Test Comparisons for Model Predictions

	T-statistic	P-value
Paired t-test	-1.1	0.273

T-statistic: T-statistic for paired t-test comparison of model predictions
P-value: Significance value for paired t-test comparison of model predictions

In addition to the accuracy assessment of the models, an exploration of the hyperparameters providing the optimal model performance was conducted. The best RF model comprised of 100 estimators and used 'sqrt' as the maximum number of features, whereas the optimal EN model was achieved with an alpha value of 0.1 and an L1 ratio of 0.1, as noted in the "Additional Results".

In summary, these results demonstrate the effective application of machine learning, particularly using the RF and EN models, in accurately predicting OTTD for pediatric patients who require post-operative mechanical ventilation. Despite minor differences, both models deliver comparable predictive reliability, offering the potential to improve the precision of tracheal tube placement and thus enhancing the outcomes of post-operative mechanical ventilation in the pediatric population.

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 into a pandas DataFrame
6  df = pd.read_csv("tracheal_tube_insertion.csv")
7
8  # Open an output file to write results
9  with open("data_exploration.txt", "w") as output_file:
10
11     # Data size
12     output_file.write("# Data Size\n")
13     output_file.write(f"Number of rows: {df.shape[0]}\n")
14     output_file.write(f"Number of columns: {df.shape[1]}\n\n")
15
16     # Summary Statistics
17     output_file.write("# Summary Statistics\n")
18     summary_stats = df.describe(include="all") # summary stats
       ↪  for all variables
19     output_file.write(f"{summary_stats}\n\n")
20
21     # Categorical Variables
22     output_file.write("# Categorical Variables\n")
```

```
    categorical_vars = df.select_dtypes(include=["object", "bool"]).columns # Get categorical variables
    for var in categorical_vars:
        output_file.write(f"Variable '{var}' most common values:\n")

        output_file.write(f"{df[var].value_counts().head()}\n")
    if len(categorical_vars) == 0:
        output_file.write("There are no categorical variables in the dataset.\n")
    output_file.write("\n")

    # Missing Values
    output_file.write("# Missing Values\n")
    missing_values = df.isnull().sum() # Get missing values
    output_file.write(f"{missing_values}\n")
    if missing_values.sum() == 0:
        output_file.write("There are no missing values in the dataset.\n")
    output_file.write("\n")

    # Check if the numeric columns contain any special values that mean unknown/undefined
    output_file.write("# Special numeric values (unknown/undefined)\n")
    numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist() # list of all numeric columns
    check_cols = ['tube', 'age_c', 'ht', 'wt', 'tube_depth_G'] # list of numeric columns to check for special values
    special_values = [col for col in check_cols if col in numeric_cols]
    if len(special_values) == 0:
        output_file.write("There are no special numeric values that mean unknown or undefined in the dataset.\n")
    else:
        for col in special_values:
            special_val_counts = df.loc[df[col] < 0, col].count() # assuming that negative values are special values
```

```
                output_file.write(f"{col}:
                ↪  {special_val_counts}\n")
```

B.2 Code Description

The provided code performs data exploration on the "tracheal_tube_insertion.csv" dataset. It aims to gain an understanding of the dataset's characteristics by performing several analysis steps.

First, the code loads the dataset into a pandas DataFrame. It then proceeds with the following analysis steps:

1. Data Size: The code calculates the number of rows and columns in the dataset using the shape attribute of the DataFrame. These values are written to the "data_exploration.txt" file.

2. Summary Statistics: The code computes summary statistics for all variables in the dataset using the describe() method. These statistics include count, mean, standard deviation, minimum, 25th percentile, median, 75th percentile, and maximum. The summary statistics are written to the "data_exploration.txt" file.

3. Categorical Variables: The code identifies the categorical variables in the dataset by selecting columns with data types "object" or "bool" using the select_dtypes() method. For each categorical variable, the code determines the most common values using the value_counts() method and writes this information to the "data_exploration.txt" file.

4. Missing Values: The code checks for missing values in the dataset by using the isnull().sum() method. It counts the number of missing values for each variable and writes these counts to the "data_exploration.txt" file.

5. Special Numeric Values: The code identifies special numeric values that may indicate unknown or undefined values. It selects the numeric columns in the dataset using the select_dtypes() method. Then, it checks for special values in specific columns ('tube', 'age_c', 'ht', 'wt', 'tube_depth_G') by filtering values less than 0 and counting them. If any special values are found, the code writes the counts to the "data_exploration.txt" file.

Overall, the code provides valuable insights into the dataset, including its size, summary statistics, information about categorical variables, missing values, and potential special numeric values. These insights are written to the "data_exploration.txt" file, allowing researchers to have a comprehensive understanding of the dataset's characteristics and potential data quality issues.

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
There are no categorical variables in the dataset.

# Missing Values
tube            0
sex             0
age_c           0
ht              0
wt              0
tube_depth_G    0
dtype: int64
There are no missing values in the dataset.

# Special numeric values (unknown/undefined)
tube: 0
age_c: 0
ht: 0
wt: 0
tube_depth_G: 0
```

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
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import ElasticNet
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import mean_squared_error
from scipy import stats
import pickle
import warnings

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

# DATASET PREPARATIONS
# No dataset preparations are needed.

# DESCRIPTIVE STATISTICS
# No descriptive statistics table is needed.

# PREPROCESSING

# Dummifying categorical variable 'sex' for modeling
df_original = pd.get_dummies(df_original, columns=['sex'], drop_first=True)

# ANALYSIS

# ========= RF Model =========
# Preparing target and feature sets for training
X = df_original[['sex_1', 'age_c', 'ht', 'wt']]
y = df_original['tube_depth_G']
```

```
34
35  # Splitting the data into training and testing sets
36  X_train, X_test, y_train, y_test = train_test_split(X, y,
    ↪  test_size=0.33, random_state=42)
37
38  # Instantiate the RF model
39  rf = RandomForestRegressor(random_state = 42)
40
41  # Set up parameter grid for tuning
42  param_grid_rf = {
43      "n_estimators" : [10, 50, 100],
44      "max_features" : ['sqrt', 'log2']
45      }
46
47  # Build GridSearch
48  CV_rf = GridSearchCV(estimator=rf, param_grid=param_grid_rf,
    ↪  cv= 5)
49  CV_rf.fit(X_train, y_train)
50
51  # Predict OTTD with the tuned RF model
52  rf_pred = CV_rf.predict(X_test)
53  rf_mse = mean_squared_error(y_test,rf_pred)
54
55  # ========= EN Model =========
56  # Instantiate the EN model
57  en = ElasticNet(random_state=42)
58
59  # Set up hyperparameter grid for tuning
60  param_grid_en = {
61      "alpha": [0.1, 0.5, 1],
62      "l1_ratio": [0.1, 0.3, 0.5, 0.7, 0.9]
63      }
64
65  # Build GridSearch
66  CV_en = GridSearchCV(estimator=en, param_grid=param_grid_en,
    ↪  cv= 5)
67  CV_en.fit(X_train, y_train)
68
69  # Predict OTTD with the tuned EN model
70  en_pred = CV_en.predict(X_test)
```

```python
en_mse = mean_squared_error(y_test,en_pred)

# Build dataframes for the scientific tables

# Table 1: Mean squared residuals of both models
df1 = pd.DataFrame(
    {"RF_model": [rf_mse],
     "EN_model": [en_mse]},
     index=["Mean_squared_residuals"])

df1.to_pickle('table_1.pkl')

# Use paired t-test to compare the mean squared residuals of RF and EN models
ttest_results = stats.ttest_rel(rf_pred, en_pred)

# Table 2: Test results of comparing the predictive power of RF and EN models
df2 = pd.DataFrame({
    "t_statistic": [ttest_results.statistic],
    "p_value": [ttest_results.pvalue]},
    index=["Paired_t_test"])

df2.to_pickle('table_2.pkl')

# SAVE ADDITIONAL RESULTS
additional_results = {
 'Total number of observations': df_original.shape[0],
 'RF_model_tuned_parameters': CV_rf.best_params_,
 'EN_model_tuned_parameters': CV_en.best_params_
}

# Save additional results
with open('additional_results.pkl', 'wb') as f:
 pickle.dump(additional_results, f)

```

C.2 Code Description

The code performs data analysis to determine the optimal tracheal tube depth (OTTD) for pediatric patients who require mechanical ventilation after surgery. The analysis is based on a dataset containing information about patient demographics, such as age, sex, height, weight, as well as the OTTD determined by chest X-ray.

The code starts by loading the dataset and prepares it for analysis by converting the categorical variable 'sex' into dummy variables.

The analysis is then performed using two different models: Random Forest (RF) and Elastic Net (EN).

For the RF model, the dataset is split into training and testing sets, and a grid search is conducted to find the optimal values for the number of estimators and maximum features. The model is tuned using the training set and evaluated using the testing set. The mean squared error (MSE) is calculated as a measure of predictive accuracy.

Similarly, for the EN model, a grid search is conducted to find the optimal values for the alpha and l1_ratio hyperparameters. The model is tuned and evaluated using the same approach as the RF model.

The code then creates two pandas dataframes for the scientific tables: Table 1 shows the mean squared residuals for both models, and Table 2 shows the results of a paired t-test comparing the predictive power of the RF and EN models.

These dataframes are saved in the form of pickled files for further analysis or reporting purposes.

Additionally, the code saves some additional results in a pickled file named 'additional_results.pkl'. These results include the total number of observations in the dataset, as well as the tuned parameters for both the RF and EN models.

In summary, the code analyzes the dataset to determine the OTTD for pediatric patients using RF and EN models. It evaluates the predictive accuracy of these models and saves the results in the form of dataframes and pickled files for further analysis and reporting.

C.3 Code Output

table_1.pkl

	RF_model	EN_model
Mean_squared_residuals	1.505649	1.137364

table_2.pkl

```
               t_statistic p_value
Paired_t_test    -1.098762  0.2727
```

additional_results.pkl

```
{
    'Total number of observations': 969,
    'RF_model_tuned_parameters': {'max_features': 'sqrt', 'n_estimators': 100},
    'EN_model_tuned_parameters': {'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:

```

# 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 = {
    'RF_model': ('Random Forest MSE', 'Mean Squared Error for
    ↪  Random Forest Model Predictions'),
    'EN_model': ('Elastic Net MSE', 'Mean Squared Error for
    ↪  Elastic Net Model Predictions'),
    'Mean_squared_residuals': ('Mean Squared Residuals',
    ↪  None),
    'Paired_t_test': ('Paired t-test', None)
}

# 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 Comparisons for Machine Learning Models",
 label='table:model_mse_comparison',
 legend=legend1)

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

# FORMAT VALUES
df2['p_value'] = df2['p_value'].apply(format_p_value)

# RENAME ROWS AND COLUMNS
mapping2 = {
    't_statistic': ('T-statistic', 'T-statistic for paired t-test comparison of model predictions'),
    'p_value': ('P-value', 'Significance value for paired t-test comparison of model predictions'),
}

mapping2 = {k: v for k, v in shared_mapping.items() if is_str_in_df(df2, k)} | mapping2
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="Statistical Test Comparisons for Model Predictions",
 label='table:model_t_test',
 legend=legend2)
```

D.2 Provided Code

The code above is using the following provided functions:

```python
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 Comparisons for Machine Learning Models}
\label{table:model_mse_comparison}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{lrr}
\toprule
 & Random Forest MSE & Elastic Net MSE \\
\midrule
\textbf{Mean Squared Residuals} & 1.51 & 1.14 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item \textbf{Random Forest MSE}: Mean Squared Error for Random Forest Model
    Predictions
\item \textbf{Elastic Net MSE}: Mean Squared Error for Elastic Net Model
    Predictions
\end{tablenotes}
\end{threeparttable}
\end{table}
```

table_2.tex

```
\begin{table}[h]
\caption{Statistical Test Comparisons for Model Predictions}
\label{table:model_t_test}
\begin{threeparttable}
```

```
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{lrl}
\toprule
 & T-statistic & P-value \\
\midrule
\textbf{Paired t-test} & -1.1 & 0.273 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item \textbf{T-statistic}: T-statistic for paired t-test comparison of model
    predictions
\item \textbf{P-value}: Significance value for paired t-test comparison of model
    predictions
\end{tablenotes}
\end{threeparttable}
\end{table}
```