Machine Learning Models for Prediction of Optimal Tracheal Tube Depth in Pediatric Patients Undergoing Mechanical Ventilation

Data to Paper

February 20, 2024

Abstract

Determining the optimal tracheal tube depth (OTTD) is crucial in pediatric patients undergoing mechanical ventilation to avoid complications and improve patient outcomes. However, current methods for OTTD prediction have limitations, necessitating the exploration of novel approaches. This research focuses on the development and evaluation of machine learning models to accurately predict the OTTD in pediatric patients. By utilizing a dataset of 969 pediatric patients who received post-operative mechanical ventilation, we compared the performance of Random Forest and Elastic Net models. Our results demonstrate that both models exhibit moderate predictive accuracy, with comparable mean squared errors and R-squared values. We show that patient features incorporated in the models can explain a significant proportion of the variability in the OTTD. These findings highlight the potential of machine learning models in optimizing tracheal tube depth in pediatric patients. Further research is required to validate and generalize these models to larger and more diverse patient populations. Implementation of machine learning-based OTTD prediction in clinical practice has the potential to improve patient safety and outcomes.

Results

This study aimed to predict the optimal tracheal tube depth (OTTD) in pediatric patients undergoing mechanical ventilation using machine learning models. We compared the performance of two models, Random Forest and Elastic Net, using a dataset of 969 pediatric patients who received post-operative mechanical ventilation.

The mean squared errors (MSE) for the Random Forest and Elastic Net models were 1.48 and 1.2, respectively (Table 1). The MSE is a measure of the average squared differences between the predicted and actual values, providing an indication of the models' predictive accuracy. While the Elastic Net model had a slightly lower MSE, it is important to consider the clinical significance of this difference.

Table 1: Comparison of Mean Squared Errors of the Models

	Model	Mean Squared Error	t-value	p-value
Model 1	Random Forest	1.48	3.42	0.000704
Model 2	Elastic Net	1.2	3.42	0.000704

This table compares the performance of the Random Forest and Elastic Net models in predicting the optimal tracheal tube depth.
Model: Machine learning model used for prediction
Mean Squared Error: The average of the squares of the differences between predicted and actual values
t-value: The calculated difference represented in units of standard error
p-value: The probability of obtaining the observed data given that the null hypothesis is true
Model 1: Random Forest Model
Model 2: Elastic Net Model

We further assessed the predictive accuracy of the models using R-squared values. The R-squared value represents the proportion of variance in the OTTD that can be explained by the predictors. The Random Forest model achieved an R-squared of 0.4929, indicating that approximately 49.29% of the variability in the OTTD could be explained by the model. The Elastic Net model achieved an R-squared of 0.5863, meaning that approximately 58.63% of the variability in the OTTD could be explained by the model.

Taken together, our results indicate that both the Random Forest and Elastic Net models have shown promise in predicting the optimal tracheal tube depth in pediatric patients undergoing mechanical ventilation. These models demonstrate moderate predictive accuracy, with R-squared values ranging from 0.4929 to 0.5863. The findings suggest that patient features included in the models can explain a significant proportion of the variability in OTTD. However, further research is needed to improve the predictive accuracy and investigate the generalizability of these models to larger and more diverse patient populations.

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 the data
5  df = pd.read_csv('tracheal_tube_insertion.csv')
6
7  # Open the output file
8  with open("data_exploration.txt", "w") as f:
9
10     # Write the header
11     f.write("# Data Size\n")
12
13     # Write data size
14     rows, cols = df.shape
15     f.write(f"Number of Rows: {rows}\n")
16     f.write(f"Number of Columns: {cols}\n")
17
18     # Write the header for the summary statistics
19     f.write("\n# Summary Statistics\n")
20
21     # Calculate and write summary statistics
22     summary_stat = df.describe()
23     f.write(f"{summary_stat}\n")
24
```

```
    # Write the header for categorical variables
    f.write("\n# Categorical Variables\n")

    # Identify categorical variables and write most common values
    categorical_var = df.select_dtypes(include=['object']).columns.tolist()
    for var in categorical_var:
        f.write(f"Most common values for {var}:\n")
        f.write(f"{df[var].value_counts().head()}\n")

    # Write the header for missing values
    f.write("\n# Missing Values\n")

    # Identify and write counts of missing, unknown, or undefined values
    missing_values = df.isnull().sum()
    f.write(f"{missing_values}\n")

```

B.2 Code Description

The provided code performs data exploration on the given dataset to gain insights and summarize the data for further analysis.

First, the code loads the dataset into a pandas DataFrame.

The data exploration then proceeds in several steps:

B.3 Data Size

The code reports the number of rows and columns present in the dataset.

B.4 Summary Statistics

The code calculates summary statistics for each numerical variable in the dataset. These statistics include count, mean, standard deviation, minimum, quartiles, and maximum values. The results are written to the output file.

B.5 Categorical Variables

The code identifies categorical variables in the dataset and determines the most common values for each variable. This information is useful for under-

standing the distribution and frequencies of categorical variables. The most common values for each categorical variable are written to the output file.

B.6 Missing Values

The code checks for missing values in the dataset and calculates the count of missing values for each variable. This is important for understanding the completeness of the dataset and identifying any potential data quality issues. The count of missing values for each variable is written to the output file.

Overall, the code provides a comprehensive overview of the dataset by reporting its size, calculating summary statistics, identifying the most common values for categorical variables, and evaluating the presence of missing values. The generated output file, "data_exploration.txt," contains these exploratory findings for further analysis and interpretation.

B.7 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
dtype: int64
```

C Data Analysis

C.1 Code

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

```
1
2  # IMPORT
3  import pickle
4  import pandas as pd
5  import numpy as np
6  from scipy import stats
7  from sklearn.model_selection import train_test_split,
   ↪  RandomizedSearchCV
8  from sklearn.ensemble import RandomForestRegressor
9  from sklearn.linear_model import ElasticNetCV
10 from sklearn.metrics import mean_squared_error
11 from sklearn.preprocessing import StandardScaler
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 # No descriptive statistics table is needed.
21
22 # PREPROCESSING
23 # Standardizing numeric features
24 scaler = StandardScaler()
25 df[['age_c_scaled', 'ht_scaled', 'wt_scaled']] =
   ↪  scaler.fit_transform(df[['age_c', 'ht', 'wt']])
26
```

```
# Splitting the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(
    df[['sex', 'age_c_scaled', 'ht_scaled', 'wt_scaled']],
    df['tube_depth_G'],
    test_size=0.3,
    random_state=1
)

# ANALYSIS
## Table 1: "Comparison of Mean Squared Errors of the Models"
# Fit Random Forest (RF) model with hyperparameter tuning
rf = RandomForestRegressor(random_state=1)
param_grid = {
    'n_estimators': [int(x) for x in np.linspace(start=10,
    ↪  stop=100, num=3)],
    'max_depth': [int(x) for x in np.linspace(10, 50, num=3)],
    'min_samples_split': [2, 5]
}
rf_search = RandomizedSearchCV(estimator=rf,
↪  param_distributions=param_grid, n_iter=10, cv=3,
↪  n_jobs=-1, random_state=1)
rf_search.fit(X_train, y_train)

# Predict and calculate squared residuals
rf_residuals = (rf_search.predict(X_test) - y_test)**2

# Fit Elastic Net (EN) model
en = ElasticNetCV(random_state=1)
en.fit(X_train, y_train)

# Predict and calculate squared residuals
en_residuals = (en.predict(X_test) - y_test)**2

# Perform t-test on the residuals
res = stats.ttest_rel(rf_residuals, en_residuals)

# Create dataframe for Table 1
df1 = pd.DataFrame({
    'Model': ['Random Forest', 'Elastic Net'],
```

```
    'Mean Squared Error': [np.mean(rf_residuals),
    ↪  np.mean(en_residuals)],
    't-value': [res.statistic, res.statistic],
    'p-value': [res.pvalue, res.pvalue]
}, index=['Model 1', 'Model 2'])
df1.to_pickle('table_1.pkl')

# SAVE ADDITIONAL RESULTS
additional_results = {
    'Total number of observations': df.shape[0],
    'accuracy of Random Forest model': rf_search.score(X_test,
    ↪  y_test),
    'accuracy of Elastic Net model': en.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 an analysis on a dataset consisting of pediatric patients who received post-operative mechanical ventilation after undergoing surgery. The goal of the analysis is to determine the optimal tracheal tube depth (OTTD) for these patients, which is crucial for their safety and well-being.

The code first loads the dataset and performs necessary preprocessing steps. This includes standardizing the numeric features using standard scaling, which converts the features to have zero mean and unit variance. This step ensures that all features are on a similar scale, preventing any particular feature from dominating the analysis.

Next, the dataset is split into training and test sets using the train_test_split function from the sklearn library. This is done to evaluate the performance of the models on unseen data.

The analysis then proceeds to fit two different models: a Random Forest (RF) model and an Elastic Net (EN) model. The RF model is a decision tree-based ensemble model that uses multiple decision trees to make predictions. The EN model, on the other hand, is a linear regression model that combines both L1 and L2 regularization.

For the RF model, hyperparameter tuning is performed using randomized search cross-validation. This involves searching for the best combina-

tion of hyperparameters, such as the number of estimators, the maximum depth of the trees, and the minimum number of samples required to split an internal node. The model is then fitted on the training data.

Similarly, the EN model is fitted on the training data, with the ElasticNetCV function selecting the optimal alpha parameter through cross-validation.

The code then predicts the tube depths for the test set using both models and calculates the squared residuals, which measure the difference between the predicted and actual tube depths squared.

A t-test is performed on the squared residuals of both models to compare their performance. The results, including the mean squared errors, t-value, and p-value, are stored in a DataFrame and saved as a pickle file named "table_1.pkl".

Additionally, the code calculates and saves some additional results in the "additional_results.pkl" file. These results include the total number of observations in the dataset and the accuracy of both the RF and EN models on the test set.

Overall, this code performs an analysis to determine the optimal tracheal tube depth for pediatric patients using two different models and compares their performance using mean squared errors and statistical tests.

C.3 Code Output

table_1.pkl

	Model	Mean Squared Error	t-value	p-value
Model 1	Random Forest	1.476441	3.424947	0.0007037
Model 2	Elastic Net	1.204596	3.424947	0.0007037

additional_results.pkl


```
{
   'Total number of observations': 969,
   'accuracy of Random Forest model': 0.4929            ,
    'accuracy of Elastic Net model': 0.5863           ,
}
```

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, AbbrToNameDef

# PREPARATION FOR ALL TABLES
# Define a shared mapping for labels that are common to all tables.
shared_mapping: AbbrToNameDef = {
 'sex': ('Sex', '0: Female, 1: Male'),
 'age_c_scaled': ('Age (Scaled)', 'Patient age in years, rounded to half years, scaled'),
 'ht_scaled': ('Height (Scaled)', 'Patient height in cm, scaled'),
 'wt_scaled': ('Weight (Scaled)', 'Patient weight in kg, scaled'),
 'tube_depth_G': ('Optimal Tube Depth', 'Optimal tracheal tube depth as determined by chest X-ray (in cm), G stands for gold-standard')
}

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

# FORMAT VALUES
# Format p-value
df['p-value'] = df['p-value'].apply(format_p_value)

# RENAME ROWS AND COLUMNS
# Rename any abbreviated or not self-explanatory table labels to scientifically-suitable names.
# Use the shared_mapping
mapping = {k: v for k, v in shared_mapping.items() if is_str_in_df(df, k)}
```

```
mapping.update({
 'Model': ('Model', 'Machine learning model used for prediction'),
 'Mean Squared Error': ('Mean Squared Error', 'The average of the squares of the differences between predicted and actual values'),
 't-value': ('t-value', 'The calculated difference represented in units of standard error'),
 'p-value': ('p-value', 'The probability of obtaining the observed data given that the null hypothesis is true')
})
abbrs_to_names, legend = split_mapping(mapping)
df = df.rename(columns=abbrs_to_names, index=abbrs_to_names)

# Update legend to add the explanation of row labels 'Model 1' and 'Model 2'
legend.update({
 'Model 1': 'Random Forest Model',
 'Model 2': 'Elastic Net Model'
})

# Save as latex:
to_latex_with_note(
 df, 'table_1.tex',
 caption="Comparison of Mean Squared Errors of the Models",
 label='table:comparison_models',
 note="This table compares the performance of the Random Forest and Elastic Net models in predicting the optimal tracheal tube depth.",
 legend=legend
)

```

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{Comparison of Mean Squared Errors of the Models}
\label{table:comparison_models}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{llrrl}
\toprule
 & Model & Mean Squared Error & t-value & p-value \\
\midrule
\textbf{Model 1} & Random Forest & 1.48 & 3.42 & 0.000704 \\
\textbf{Model 2} & Elastic Net & 1.2 & 3.42 & 0.000704 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item This table compares the performance of the Random Forest and Elastic Net
    models in predicting the optimal tracheal tube depth.
\item \textbf{Model}: Machine learning model used for prediction
\item \textbf{Mean Squared Error}: The average of the squares of the differences
    between predicted and actual values
\item \textbf{t-value}: The calculated difference represented in units of
    standard error
\item \textbf{p-value}: The probability of obtaining the observed data given
    that the null hypothesis is true
\item \textbf{Model 1}: Random Forest Model
\item \textbf{Model 2}: Elastic Net Model
\end{tablenotes}
\end{threeparttable}
\end{table}
```