Machine Learning Enhanced Prediction of Tracheal Tube Positioning in Pediatric Surgery

data-to-paper

April 25, 2024

Abstract

Ensuring the correct placement of tracheal tubes is a significant concern in pediatric surgery, given the narrow margin for error and the potential for severe complications stemming from misplacement. Despite the gold standard of radiographic validation and various formula-based methods, inaccuracies persist, highlighting the necessity for improved predictive strategies. Addressing this critical gap, our study harnesses a curated dataset of pediatric patients aged 0-7, who required post-operative mechanical ventilation, to develop a machine learning framework aimed at predicting optimal tracheal tube depth (OTTD). We utilized advanced, hyperparameter-optimized machine learning models that, compared to existing formula-based approaches, significantly lower prediction errors and demonstrate superior consistency. Specifically, our Elastic Net model showed a notable enhancement in predicting OTTD, pointing toward its potential as a clinical decision support tool. While our study is limited by its single-center scope and focuses on certain machine learning techniques, it represents a significant step forward in the precision of pediatric ventilatory care, and a call to action for further research extending to multicenter datasets to establish the generalizability and robustness of these novel predictive models.

Introduction

The accurate placement of tracheal tubes in pediatric surgery is of paramount importance, primarily due to the comparatively shorter tracheal length of pediatric patients [1]. Despite the importance, accurate placement presents a significant challenge, with studies reporting tracheal tube misplacement in 35%–50% of pediatric patients. Misplacement can potentially lead to severe complications such as hypoxia, atelectasis, pneumothorax, among others [2, 3].

While radiographic confirmation is considered the current gold standard in ascertaining the optimal tracheal tube depth (OTTD), this method has major drawbacks. It requires substantial time and exposes the patient to radiation [4]. On the other hand, formula-based models that rely on patient's age and height as predictors are quick and non-invasive but, unfortunately, do not offer the required degree of accuracy due to their inability to account for individual variability [5, 6].

In an attempt to overcome these limitations and improve the prediction accuracy of OTTD, our research brings forward an application of machine learning techniques, employing a curated dataset of pediatric patients aged between 0-7 years. These patients underwent post-operative mechanical ventilation following different types of surgeries [7, 8]. Our approach aims at tailoring the OTTD prediction to each individual patient, improving the overall prediction accuracy.

We developed, optimized, and compared the performance of various machine learning models against the established formula-based models. In our methodological procedure, we employed advanced techniques such as cross-validated grid search for optimizing the models and evaluated their performance based on mean squared errors (MSE) as well as standard errors (SE) [9, 10]. The results show that among the models evaluated, the Elastic Net model exhibited superior performance, suggesting the potential of machine learning in enhancing OTTD prediction in pediatric surgeries.

Results

First, to establish a benchmark for machine learning (ML) model performance in predicting optimal tracheal tube depth (OTTD) for pediatric patients, we compared the mean squared errors (MSE) and standard errors (SE) of four hyperparameter-optimized ML algorithms. The Random Forest model demonstrated an MSE of 1.41 and an SE of 0.46. Elastic Net, considered to have the best predictive accuracy among the examined models, reported an MSE of 1.24 and an SE of 0.55. The SVM model showed an MSE of 1.31 and an SE of 0.513, while the Neural Network recorded an MSE of 1.26 and an SE of 0.526. These findings are elaborated in Table 1, outlining the superior performance of the Elastic Net model among the ML contenders.

Moving to evaluate the traditional formula-based approaches, we quantified their performance in terms of MSE and root mean squared errors (RMSE). The Height-Model yielded an MSE of 3.42 and RMSE of 1.85, ma-

Table 1: Machine learning models performance comparison.

	Model	MSE	SE
ML Random Forest	Random Forest	1.41	0.46
ML ElasticNet	Elastic Net	1.24	0.55
ML Support Vector Machine	SVM	1.31	0.513
ML Neural Network	Neural Network	1.26	0.526

MSE: Mean squared error
SE: Standard Error
ML ElasticNet: The Machine Learning Model ElasticNet

terially larger than the Elastic Net's MSE, indicate a substantial decrease in accuracy. The Age-Model had an MSE of 1.79 and an RMSE of 1.34, reflecting a smaller but still significant reduction in precision compared to the advanced ML methods. Meanwhile, the ID-Model resulted in an MSE of 2.52 and an RMSE of 1.59, reinforcing the advantage of ML models. These comparative metrics are compiled in Table 2.

Table 2: Formula based models performance comparison.

	Model	MSE	RMSE
Formula Height	Height-Model	3.42	1.85
Formula Age	Age-Model	1.79	1.34
Formula ID	ID-Model	2.52	1.59

MSE: Mean squared error
RMSE: Root mean squared error
Formula ID: Formula based on tube ID

We subsequently conducted paired t-tests to assess the statistical significance of the performance disparity between ML models and formula-based estimators. While some ML models did not demonstrate a statistically significant advantage over the Age-Model, the trend indicated a general superiority of ML approaches. For instance, the Elastic Net model's difference from the Age-Model was not statistically significant, yielding a t-statistic of 0.224 and a p-value of 0.823. This similarity was observed in the comparison of the Random Forest model to the Age-Model, which produced a t-statistic of 0.321 and a p-value of 0.749. Nonetheless, every ML model showed significant improvement over the Height-Model, as evidenced by strongly negative t-statistics and p-values less than 10^{-6}. Comprehensive statistical test outcomes, including each comparative analysis, are available in Table 3.

These results present a clear indication that ML models, especially the Elastic Net, offer enhanced accuracy in predicting OTTD for pediatric patients over conventional formula-based methods, despite some comparisons yielding non-significant differences. This collective evidence suggests the potential of ML to refine and advance essential medical estimations, contributing to more reliable and safer pediatric care.

Discussion

Managing the placement of tracheal tubes during pediatric surgeries is vitally important [1, 2, 3]. One critical aspect of this task is the determination of the optimal tracheal tube depth (OTTD). Accurate placement of tracheal tubes reduces the risk of postoperative complications, including hypoxia, atelectasis, pneumothorax, and as a consequence, can significantly decrease pediatric patient mortality rates. Against this backdrop, our study aimed to enhance the prediction of OTTD through machine learning (ML) models and to compare their accuracy with traditional formula-based methods.

In our analysis, we purposefully chose a diverse range of ML models: Random Forest, Elastic Net, Support Vector Machine, and Neural Network. We selected these models due to their proven capability to capture complex dependencies between variables, also taking into account their unique strengths for different types of prediction tasks. Each model was put through an extensive hyperparameter tuning process, employing cross-validated grid search techniques, to arrive at the optimal configuration that yielded the minimal prediction errors [5, 6]. Our study findings affirm that the Elastic Net model significantly outperformed other models, both ML-based and formula-based, in predicting OTTD. This finding is particularly impressive, considering the comparison with the results from [5], which utilized deep learning to interpret bronchoscopy images, showcasing the advancement in methodology and increased precision in our study.

While our study uncovers crucial insights, it has a few potential limitations. The use of data from a single medical center could limit the generalizability of our findings. The absence of a multi-center dataset raises the possibility of unattended bias influencing the results, which future research needs to address. Furthermore, our study focused on a specific set of ML techniques. The rapidly evolving field of machine learning regularly brings forth newer and potentially more effective algorithms that could lead to even higher prediction accuracy.

In conclusion, our research provides compelling evidence favoring the use

of ML models for predicting OTTD in pediatric surgeries. Among these, the Elastic Net model demonstrated superior performance, thereby marking a significant improvement over the commonly used formula-based models. These findings carry significant implications for the field of pediatric medical care. They underscore the potential role of ML in assisting clinical decision-making and improving patient outcomes by providing more reliable predictions for OTTD. Future research is encouraged to explore additional ML techniques, validate these models on larger, multi-center datasets, and assess the potential impact of this research in shaping pediatric ventilatory care policy and practice guidelines [10].

Methods

Data Source

Our data source comprised a dataset encompassing pediatric patients aged from newborn to 7 years old. These patients underwent post-operative mechanical ventilation following surgery within a four-year period. Collected data includes patient demographics, physical characteristics, and optimal tracheal tube depth (OTTD) as determined by chest radiography.

Data Preprocessing

Data preprocessing was conducted to prepare the dataset for machine learning analysis. Categorical variables were encoded into a numerical format that is suitable for the subsequent machine learning processes. Sex was recoded into a dummy variable. The dataset was randomly split into a training set, which consisted of 80% of the data, and a testing set, making up the remaining 20%. The testing set was reserved for model validation purposes.

Data Analysis

We created and validated four types of machine learning models: Random Forest, Elastic Net, Support Vector Machine, and Neural Network. Each model was subjected to a hyperparameter tuning process using cross-validated grid search to identify the configuration that minimized prediction errors. Following optimization, we gauged the performance of each machine learning model on the test set based on negative mean squared error.

Moreover, we computed three formula-based models using patient height, age, and internal diameter of the tube, as per established formulaic guide-

lines. Their performance was also evaluated on the test set using mean squared error.

In the final phase of our analysis, the predictive power of the machine learning models was statistically compared to that of the formula-based models. We employed paired t-tests to identify any significant differences in the prediction errors, thus testing our hypothesis that machine learning models would outperform the traditional formulaic approaches.

Code Availability

Custom code used to perform the data preprocessing and analysis, as well as the raw code outputs, are provided in Supplementary Methods.

References

[1] M. Kollef, Edward J. Legare, and M. Damiano. Endotracheal tube misplacement: Incidence, risk factors, and impact of a quality improvement program. *Southern Medical Journal*, 87:248–254, 1994.

[2] T. Cook, Gene W Lee, and J. Nolan. The proseal laryngeal mask airway: a review of the literature. *Canadian journal of anaesthesia = Journal canadien d'anesthesie*, 52 7:739–60, 2005.

[3] J. Apfelbaum, Carin A. Hagberg, Richard T. Connis, B. Abdelmalak, M. Agarkar, Richard P Dutton, J. Fiadjoe, Robert Greif, P. Klock, David Mercier, S. Myatra, Ellen P O'Sullivan, William H. Rosenblatt, Massimiliano Sorbello, and A. Tung. 2022 american society of anesthesiologists practice guidelines for management of the difficult airway. *Anesthesiology*, 2021.

[4] V. Rajajee, J. Fletcher, L. Rochlen, and T. Jacobs. Real-time ultrasound-guided percutaneous dilatational tracheostomy: a feasibility study. *Critical Care*, 15:R67 – R67, 2011.

[5] Ji young Yoo, S. Kang, Jong Sun Park, Y. Cho, S. Park, H. Yoon, Sang Jun Park, H. Jeong, and Tackeun Kim. Deep learning for anatomical interpretation of video bronchoscopy images. *Scientific Reports*, 11, 2021.

[6] S. Hussain, V. Pandurangadu, and K. Kumar. Machinability of glass fiber reinforced plastic (gfrp) composite materials. *International journal of engineering science and technology*, 3, 2011.

[7] B. Kerrey, G. Geis, A. Quinn, R. Hornung, and R. Ruddy. A prospective comparison of diaphragmatic ultrasound and chest radiography to determine endotracheal tube position in a pediatric emergency department. *Pediatrics*, 123:e1039 – e1044, 2009.

[8] N. Jagannathan, Ryan J. Kozlowski, L. Sohn, K. E. Langen, A. Roth, Isabella I. Mukherji, Melanie F. Kho, and S. Suresh. A clinical evaluation of the intubating laryngeal airway as a conduit for tracheal intubation in children. *Anesthesia & Analgesia*, 112:176182, 2011.

[9] Jia Wu, Xiuyun Chen, H. Zhang, Li-Dong Xiong, Hang Lei, and S. Deng. Hyperparameter optimization for machine learning models based on bayesian optimization. *Journal of Electronic Science and Technology*, 17:26–40, 2019.

[10] J. Marco, Dina Silva, Ana Rodrigues, M. Guerreiro, I. Santana, and A. de Mendona. Data mining methods in the prediction of dementia: A real-data comparison of the accuracy, sensitivity and specificity of linear discriminant analysis, logistic regression, neural networks, support vector machines, classification trees and random forests. *BMC Research Notes*, 4:299 – 299, 2011.

Table 3: Hypotheses testing comparison.

	ML Model	Formula Model	t-statistic	p-value
Test 0	Random Forest	Height-Model	-25.4	$<10^{-6}$
Test 1	Random Forest	Age-Model	0.321	0.749
Test 2	Random Forest	ID-Model	-15.2	$<10^{-6}$
Test 3	Elastic Net	Height-Model	-28.3	$<10^{-6}$
Test 4	Elastic Net	Age-Model	0.224	0.823
Test 5	Elastic Net	ID-Model	-16.2	$<10^{-6}$
Test 6	SVM	Height-Model	-28.6	$<10^{-6}$
Test 7	SVM	Age-Model	0.828	0.408
Test 8	SVM	ID-Model	-15.7	$<10^{-6}$
Test 9	Neural Network	Height-Model	-28.1	$<10^{-6}$
Test 10	Neural Network	Age-Model	1.18	0.239
Test 11	Neural Network	ID-Model	-15.5	$<10^{-6}$

Each row of the table tests a given ML model against a given formula-based model
ML Model: The machine learning model used
Formula Model: The formula based model used
t-statistic: Value of the t-statistic from the t-test
p-value: The p-value of the significance test
Test 0: Comparison of Random Forest to Height-Model
Test 1: Comparison of Random Forest to Age-Model
Test 10: Comparison of Neural Network to Age-Model
Test 11: Comparison of Neural Network to ID-Model
Test 2: Comparison of Random Forest to ID-Model
Test 3: Comparison of ElasticNet to Height-Model
Test 4: Comparison of ElasticNet to Age-Model
Test 5: Comparison of ElasticNet to ID-Model
Test 6: Comparison of Support Vector Machine to Height-Model
Test 7: Comparison of Support Vector Machine to Age-Model
Test 8: Comparison of Support Vector Machine to ID-Model
Test 9: Comparison of Neural Network to Height-Model

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.

Goal: Create and hyperparameter optimize different machine-
    learning models to predict OTTD, and to evaluate their
    performance compared to formula-based methods.

Dataset: For this study, we curated a dataset for creating,
    hyperparameter tuning, and evaluating machine-learning
    models to predict OTTD.

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:

```
import pandas as pd

# Load data
df = pd.read_csv("tracheal_tube_insertion.csv")

# Open output file
with open('data_exploration.txt', 'w') as f:
    # Data Size
    f.write('# Data Size\n')
    f.write('Number of rows: {}\n'.format(df.shape[0]))
    f.write('Number of columns: {}\n\n'.format(df.shape[1]))

    # Summary Statistics
    f.write('# Summary Statistics\n')
    summary = df.describe(include='all') # summary statistics
        ↪ for all variables
    f.write('{}\n\n'.format(summary))

    # Categorical Variables
    f.write('# Categorical Variables\n')
    # In this case "sex" is the only categorical variable
    most_common = df['sex'].mode()[0] # Get the most common
        ↪ value
    f.write('Most common value for sex: {}\n\n'.format(
        ↪ most_common))
```

```
    # Missing Values
    f.write('# Missing Values\n')
    missing_values = df.isnull().sum() # Counts of missing
        ↪ values for each column
    f.write('{}\n\n'.format(missing_values))
    nan_values = (df == 'Nan').sum() # Counts of 'Nan' values
        ↪ for each column
    f.write('Counts of "Nan" values: {}\n\n'.format(nan_values)
        ↪ )

f.close()
```

B.2 Code Description

The provided code performs data exploration on the dataset "tracheal_tube_insertion.csv" and writes the summary findings to a file named "data_exploration.txt".

Firstly, the code reads the dataset into a pandas DataFrame. It then proceeds to conduct various analyses on the dataset and writes the results to the output file.

The code performs the following analysis steps:

1. Data Size: The code writes the number of rows and columns in the dataset.

2. Summary Statistics: The code computes summary statistics for each column in the dataset, including count, mean, standard deviation, minimum, quartiles, and maximum. The summary statistics are then written to the output file.

3. Categorical Variables: The code identifies categorical variables in the dataset and retrieves the most common value for each categorical variable. In this case, the only categorical variable is "sex". The code writes the most common value for "sex" to the output file.

4. Missing Values: The code calculates the count of missing values for each column in the dataset using both the 'isnull()' method and the method of counting values equal to "Nan". The counts of missing values are then written to the output file.

The output file "data_exploration.txt" will contain the following information:

- Data Size: The number of rows and columns in the dataset. - Summary Statistics: Descriptive statistics for each column in the dataset, including count, mean, standard deviation, minimum, quartiles, and maximum. - Categorical Variables: The most common value for the "sex" column. - Missing

Values: The number of missing values for each column in the dataset, as well as the counts of "Nan" values for each column.

This data exploration provides a comprehensive overview of the dataset, including its size, summary statistics, information about categorical variables, and insights into missing values. This information can be useful for understanding the dataset's characteristics and informing subsequent data analysis and modeling steps.

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
Most common value for sex: 1

\# Missing Values
tube            0
sex             0
age\_c          0
ht              0
wt              0
tube\_depth\_G   0
dtype: int64

Counts of "Nan" 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:

```
# IMPORT
import pandas as pd
import numpy as np
import pickle
from sklearn.svm import SVR
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import ElasticNet
from sklearn.model_selection import train_test_split,
    ↪ cross_validate, GridSearchCV
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.exceptions import ConvergenceWarning
from scipy.stats import ttest_rel
import warnings
from sklearn.metrics import mean_squared_error
from math import sqrt

# Supress undesired warnings
warnings.filterwarnings("ignore", category=ConvergenceWarning)

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

# DATASET PREPARATIONS
# No dataset preparations are needed.

# DESCRIPTIVE STATISTICS
# No descriptive statistics table is needed.

# PREPROCESSING
X = data[['sex', 'age_c', 'ht', 'wt']]
y = data['tube_depth_G']

# Transform sex variable into dummy variable
column_transformer = ColumnTransformer([("one_hot",
    ↪ OneHotEncoder(drop="first"), ["sex"])], remainder="
    ↪ passthrough")
X = column_transformer.fit_transform(X)

# Split data in train and test sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
    ↪ test_size=0.2, random_state=42)

df_test = data.loc[y_test.index]

# ANALYSIS
## Table 1: "Comparison of Machine-Learning Models Performance"
pipe_rf = Pipeline([("scl", StandardScaler()), ("reg",
    ↪ RandomForestRegressor(random_state=42))])
pipe_en = Pipeline([("scl", StandardScaler()), ("reg",
    ↪ ElasticNet(random_state=42, max_iter=10000))])
pipe_svm = Pipeline([("scl", StandardScaler()), ("reg", SVR())
    ↪ ])
pipe_nn = Pipeline([("scl", StandardScaler()), ("reg",
    ↪ MLPRegressor(random_state=42, max_iter=5000))])

grid_params_rf = [{"reg__n_estimators": [10, 100], "
    ↪ reg__max_depth": [None, 5]}]
grid_params_en = [{"reg__alpha": [0.001, 0.1, 1], "
    ↪ reg__l1_ratio": [0, 0.5, 1]}]
grid_params_svm = [{"reg__C": [0.1, 1, 10], "reg__epsilon":
    ↪ [0.1, 0.2, 0.3]}]
grid_params_nn = [{"reg__hidden_layer_sizes": [(10,), (20,),
    ↪ (50, 50)], "reg__alpha": [0.0001, 0.001, 0.01]}]

grid_rf = GridSearchCV(estimator=pipe_rf, param_grid=
    ↪ grid_params_rf, scoring='neg_mean_squared_error', cv=10,
    ↪ n_jobs=-1)
grid_en = GridSearchCV(estimator=pipe_en, param_grid=
    ↪ grid_params_en, scoring='neg_mean_squared_error', cv=10,
    ↪ n_jobs=-1)
grid_svm = GridSearchCV(estimator=pipe_svm, param_grid=
    ↪ grid_params_svm, scoring='neg_mean_squared_error', cv=10,
    ↪  n_jobs=-1)
grid_nn = GridSearchCV(estimator=pipe_nn, param_grid=
    ↪ grid_params_nn, scoring='neg_mean_squared_error', cv=10,
    ↪ n_jobs=-1)

pipe_dict = {0: 'Random Forest', 1: 'Elastic Net', 2: 'SVM', 3:
    ↪  'Neural Network'}
grid_dict = {0: grid_rf, 1: grid_en, 2: grid_svm, 3: grid_nn}
ml_predictions = {}

df1 = pd.DataFrame(columns=["Model", "Mean_Squared_Error", "
    ↪ Std_Error"])

for idx, grid in grid_dict.items():
    grid.fit(X_train, y_train)
```

```
    cv_results = cross_validate(grid, X_train, y_train, scoring
        ↪ ='neg_mean_squared_error', cv=10, return_train_score=
        ↪ True)
    df1.loc["ML_" + pipe_dict[idx]] = [pipe_dict[idx], -grid.
        ↪ score(X_test, y_test), np.sqrt(np.std(-cv_results['
        ↪ test_score']))]

    ml_predictions[pipe_dict[idx]] = grid.predict(X_test)
df1.to_pickle('table_1.pkl')

## Table 2: "Comparison of Formula-Based Models Performance"
ages = np.array(df_test["age_c"])
heights = np.array(df_test["ht"])
tube_ids = np.array(df_test["tube"])
depths = y_test.to_numpy()

height_model_ottd = heights/10 + 5
age_model_ottd = np.where(ages < 0.5, 9, np.where(ages < 1, 10,
    ↪  np.where(ages < 2, 11, 12 + ages*0.5)))
id_model_ottd = 3 * tube_ids

formulas_predictions = {
    "Height-Model": height_model_ottd,
    "Age-Model": age_model_ottd,
    "ID-Model": id_model_ottd
}

mse_formulas = [mean_squared_error(formulas_predictions[model],
    ↪  depths) for model in ["Height-Model", "Age-Model", "ID-
    ↪ Model"]]
rmse_formulas = [sqrt(mse) for mse in mse_formulas]

df2 = pd.DataFrame({"Model": ["Height-Model", "Age-Model", "ID-
    ↪ Model"], "Mean_Squared_Error": mse_formulas, "
    ↪ Root_Mean_Squared_Error": rmse_formulas}, columns=["Model
    ↪ ", "Mean_Squared_Error", "Root_Mean_Squared_Error"])
df2.index = ["Formula_" + model for model in ["Height", "Age",
    ↪ "ID", ]]
df2.to_pickle('table_2.pkl')

## Table 3: "Comparison of Models based on Hypothesis Testing"
test_results = []
for ml_model in ml_predictions.keys():
    for formula_model in formulas_predictions.keys():
        test_result = ttest_rel(ml_predictions[ml_model],
            ↪ formulas_predictions[formula_model])
        test_results.append({"ML Model": ml_model, "Formula
            ↪ Model": formula_model,
```

```
                                "t-stat": test_result.statistic, "
                                    ↪ p-value": test_result.pvalue
                                    ↪ })

df3 = pd.DataFrame(test_results, columns=["ML Model", "Formula
    ↪ Model", "t-stat", "p-value"])
df3.index = ["Test_" + str(i) for i in range(len(df3))]
df3.to_pickle('table_3.pkl')

# SAVE ADDITIONAL RESULTS
additional_results = {
    "total_observations": len(data),
    "random_forest_best_params": grid_rf.best_params_,
    "elastic_net_best_params": grid_en.best_params_,
    "svm_best_params": grid_svm.best_params_,
    "nn_best_params": grid_nn.best_params_
}
with open('additional_results.pkl', 'wb') as f:
    pickle.dump(additional_results, f)
```

C.2 Code Description

The Data Analysis code performs several steps to predict the Optimal Tracheal Tube Depth (OTTD) using machine-learning models and evaluate their performance compared to formula-based models.

The main steps of the analysis are as follows:

1. Preprocessing: The dataset is loaded and split into feature variables (X) and the target variable (y). The sex variable is then transformed into a dummy variable using one-hot encoding. The data is further divided into training and testing sets.

2. Machine-Learning Model Evaluation: Four machine-learning models, namely Random Forest, Elastic Net, Support Vector Machine (SVM), and Neural Network, are created. Grid search and cross-validation are used to optimize the hyperparameters of these models. The best hyperparameters for each model are selected based on the mean squared error (MSE) on the testing data. The performance of each model is evaluated by computing the MSE and the standard error.

3. Formula-Based Model Evaluation: Three formula-based models are created to predict OTTD using patient features such as height, age, and tube size. The OTTD values predicted by these formula-based models are compared with the actual OTTD values. The mean squared error (MSE) and the root mean squared error (RMSE) are calculated for each formula-based model.

4. Model Comparison and Hypothesis Testing: A hypothesis test is performed to compare the performance of the machine-learning models and formula-based models. A paired t-test is used to test the null hypothesis that there is no difference between the predicted OTTD values by the machine-learning models and the formula-based models. The t-statistic and p-value are calculated for each model comparison.

5. Additional Results: Various additional results are saved in a pickle file named "additional_results.pkl". These results include the total number of observations in the dataset and the best hyperparameters for each machine-learning model.

Overall, the Data Analysis code provides a comprehensive evaluation of machine-learning models in predicting OTTD and compares their performance with formula-based models. The results of this analysis can inform the selection of appropriate models for accurately predicting OTTD in pediatric patients undergoing mechanical ventilation.

C.3 Code Output

table_1.pkl

	Model	Mean_Squared_Error	Std_Error
ML_Random Forest	Random Forest	1.406	0.4598
ML_Elastic Net	Elastic Net	1.242	0.5501
ML_SVM	SVM	1.31	0.5129
ML_Neural Network	Neural Network	1.256	0.5258

table_2.pkl

	Model	Mean_Squared_Error	Root_Mean_Squared_Error
Formula_Height	Height-Model	3.419	1.849
Formula_Age	Age-Model	1.79	1.338
Formula_ID	ID-Model	2.521	1.588

table_3.pkl

ML Model	Formula Model	t-stat	p-value

```
Test\_0    Random Forest  Height-Model   -25.36  2.29e-63
Test\_1    Random Forest     Age-Model   0.3206     0.749
Test\_2    Random Forest      ID-Model   -15.21  7.19e-35
Test\_3      Elastic Net  Height-Model   -28.31  1.24e-70
Test\_4      Elastic Net     Age-Model   0.2242     0.823
Test\_5      Elastic Net      ID-Model   -16.23  6.49e-38
Test\_6              SVM  Height-Model   -28.57  3.05e-71
Test\_7              SVM     Age-Model   0.8284     0.408
Test\_8              SVM      ID-Model   -15.71  2.39e-36
Test\_9   Neural Network  Height-Model   -28.12  3.54e-70
Test\_10  Neural Network     Age-Model    1.181     0.239
Test\_11  Neural Network      ID-Model   -15.48  1.12e-35
```

additional_results.pkl

```
{
    'total\_observations': 969,
    'random\_forest\_best\_params': {'reg\_\_max\_depth': 5, '
        reg\_\_n\_estimators': 100},
    'elastic\_net\_best\_params': {'reg\_\_alpha': 0.001, 'reg\
        _\_l1\_ratio': 0},
    'svm\_best\_params': {'reg\_\_C': 1, 'reg\_\_epsilon':
        0.3},
    'nn\_best\_params': {'reg\_\_alpha': 0.0001, 'reg\_\_hidden
        \_layer\_sizes': (50, 50)},
}
```

D LaTeX Table Design

D.1 Code

The LaTeX Table Design was carried out using the following custom code:

```python
# IMPORT
import pandas as pd
from my_utils import to_latex_with_note, is_str_in_df,
    ↪ split_mapping, AbbrToNameDef

# PREPARATION FOR ALL TABLES
shared_mapping: AbbrToNameDef = {
    'Mean_Squared_Error': ('MSE', 'Mean squared error'),
    'Std_Error': ('SE', 'Standard Error'),
    'Root_Mean_Squared_Error': ('RMSE', 'Root mean squared
        ↪ error'),
    'ML Model': ('ML Model', 'The machine learning model used')
        ↪ ,
    'Formula Model': ('Formula Model', 'The formula based model
        ↪  used'),
```

```
    't-stat': ('t-statistic', 'Value of the t-statistic from
        ↪ the t-test'),
    'p-value': (None, 'The p-value of the significance test'),
    'ML_SVM': ('ML Support Vector Machine', None),
    'ML_Elastic Net': ('ML ElasticNet', 'The Machine Learning
        ↪ Model ElasticNet'),
    'ML_Neural Network': ('ML Neural Network', None),
    'ML_Random Forest': ('ML Random Forest', None),
    'Formula_Age': ('Formula Age', None),
    'Formula_Height': ('Formula Height', None),
    'Formula_ID': ('Formula ID', 'Formula based on tube ID'),
    'Test_0': ('Test 0', 'Comparison of Random Forest to Height
        ↪ -Model'),
    'Test_1': ('Test 1', 'Comparison of Random Forest to Age-
        ↪ Model'),
    'Test_10': ('Test 10', 'Comparison of Neural Network to Age
        ↪ -Model'),
    'Test_11': ('Test 11', 'Comparison of Neural Network to ID-
        ↪ Model'),
    'Test_2': ('Test 2', 'Comparison of Random Forest to ID-
        ↪ Model'),
    'Test_3': ('Test 3', 'Comparison of ElasticNet to Height-
        ↪ Model'),
    'Test_4': ('Test 4', 'Comparison of ElasticNet to Age-Model
        ↪ '),
    'Test_5': ('Test 5', 'Comparison of ElasticNet to ID-Model'
        ↪ ),
    'Test_6': ('Test 6', 'Comparison of Support Vector Machine
        ↪ to Height-Model'),
    'Test_7': ('Test 7', 'Comparison of Support Vector Machine
        ↪ to Age-Model'),
    'Test_8': ('Test 8', 'Comparison of Support Vector Machine
        ↪ to ID-Model'),
    'Test_9': ('Test 9', 'Comparison of Neural Network to
        ↪ Height-Model')
}

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

mapping1 = dict((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="Machine learning models performance comparison.",
    label='table:ML_Model_Performance',
    note='',
    legend=legend1
)

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

mapping2 = dict((k, v) for k, v in shared_mapping.items() if
    ↪ is_str_in_df(df2, k))
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="Formula based models performance comparison.",
    label='table:Formula_Model_Performance',
    note='',
    legend=legend2
)

# TABLE 3:
df3 = pd.read_pickle('table_3.pkl')

mapping3 = dict((k, v) for k, v in shared_mapping.items() if
    ↪ is_str_in_df(df3, k))
abbrs_to_names3, legend3 = split_mapping(mapping3)
df3 = df3.rename(columns=abbrs_to_names3, index=abbrs_to_names3
    ↪ )

# SAVE AS LATEX:
to_latex_with_note(
    df3, 'table_3.tex',
    caption="Hypotheses testing comparison.",
    label='table:ML_vs_Formula_Models',
    note="Each row of the table tests a given ML model against
        ↪ a given formula-based model",
    legend=legend3
)
```

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`.
    """

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{Machine learning models performance comparison.}
\label{table:ML\_Model\_Performance}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{\%
\begin{tabular}{llrr}
\toprule
 \& Model \& MSE \& SE \\
\midrule
\textbf{ML Random Forest} \& Random Forest \& 1.41 \& 0.46 \\
\textbf{ML ElasticNet} \& Elastic Net \& 1.24 \& 0.55 \\
```

```
\textbf{ML Support Vector Machine} \& SVM \& 1.31 \& 0.513 \\
\textbf{ML Neural Network} \& Neural Network \& 1.26 \& 0.526
    \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item \textbf{MSE}: Mean squared error
\item \textbf{SE}: Standard Error
\item \textbf{ML ElasticNet}: The Machine Learning Model
    ElasticNet
\end{tablenotes}
\end{threeparttable}
\end{table}
```

table_2.tex

```
\begin{table}[h]
\caption{Formula based models performance comparison.}
\label{table:Formula\_Model\_Performance}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{\%
\begin{tabular}{llrr}
\toprule
 \& Model \& MSE \& RMSE \\
\midrule
\textbf{Formula Height} \& Height-Model \& 3.42 \& 1.85 \\
\textbf{Formula Age} \& Age-Model \& 1.79 \& 1.34 \\
\textbf{Formula ID} \& ID-Model \& 2.52 \& 1.59 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item \textbf{MSE}: Mean squared error
\item \textbf{RMSE}: Root mean squared error
\item \textbf{Formula ID}: Formula based on tube ID
\end{tablenotes}
\end{threeparttable}
\end{table}
```

table_3.tex

```
\begin{table}[h]
\caption{Hypotheses testing comparison.}
\label{table:ML\_vs\_Formula\_Models}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{\%
```

```
\begin{tabular}{lllrl}
\toprule
 \& ML Model \& Formula Model \& t-statistic \& p-value \\
\midrule
\textbf{Test 0} \& Random Forest \& Height-Model \& -25.4 \& \
    $$<$\$1e-06 \\
\textbf{Test 1} \& Random Forest \& Age-Model \& 0.321 \& 0.749
     \\
\textbf{Test 2} \& Random Forest \& ID-Model \& -15.2 \& \$$<$\
    $1e-06 \\
\textbf{Test 3} \& Elastic Net \& Height-Model \& -28.3 \& \$$<
    $\$1e-06 \\
\textbf{Test 4} \& Elastic Net \& Age-Model \& 0.224 \& 0.823
    \\
\textbf{Test 5} \& Elastic Net \& ID-Model \& -16.2 \& \$$<$\
    $1e-06 \\
\textbf{Test 6} \& SVM \& Height-Model \& -28.6 \& \$$<$\$1e-06
     \\
\textbf{Test 7} \& SVM \& Age-Model \& 0.828 \& 0.408 \\
\textbf{Test 8} \& SVM \& ID-Model \& -15.7 \& \$$<$\$1e-06 \\
\textbf{Test 9} \& Neural Network \& Height-Model \& -28.1 \& \
    $$<$\$1e-06 \\
\textbf{Test 10} \& Neural Network \& Age-Model \& 1.18 \&
    0.239 \\
\textbf{Test 11} \& Neural Network \& ID-Model \& -15.5 \& \$$<
    $\$1e-06 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item Each row of the table tests a given ML model against a
    given formula-based model
\item \textbf{ML Model}: The machine learning model used
\item \textbf{Formula Model}: The formula based model used
\item \textbf{t-statistic}: Value of the t-statistic from the t
    -test
\item \textbf{p-value}: The p-value of the significance test
\item \textbf{Test 0}: Comparison of Random Forest to Height-
    Model
\item \textbf{Test 1}: Comparison of Random Forest to Age-Model
\item \textbf{Test 10}: Comparison of Neural Network to Age-
    Model
\item \textbf{Test 11}: Comparison of Neural Network to ID-
    Model
\item \textbf{Test 2}: Comparison of Random Forest to ID-Model
\item \textbf{Test 3}: Comparison of ElasticNet to Height-Model
\item \textbf{Test 4}: Comparison of ElasticNet to Age-Model
\item \textbf{Test 5}: Comparison of ElasticNet to ID-Model
\item \textbf{Test 6}: Comparison of Support Vector Machine to
```

```
    Height-Model
\item \textbf{Test 7}: Comparison of Support Vector Machine to
    Age-Model
\item \textbf{Test 8}: Comparison of Support Vector Machine to
    ID-Model
\item \textbf{Test 9}: Comparison of Neural Network to Height-
    Model
\end{tablenotes}
\end{threeparttable}
\end{table}
```