Predicting the Length of Stay using Machine Learning

Narendra Gude, Priya Kaur, Sai Narayana Das

I. Abstract

The length of stay (LOS) is a clinical measurement that counts the time between a patient's hospital admission and discharge. One of the most important uses of in-patient stay prediction is to provide patients with tailored treatment plans. Healthcare workers can change treatment plans to offer the best possible care by anticipating how long a patient will stay in the hospital. This method can result in improved results and shorter hospital stays, lowering the risk of hospital-acquired infections, reducing the strain on the healthcare system, and optimizing hospital operations. We used Random Forest and Decision tree to train the model. With greater performance in categorizing positive occurrences, the optimized decision tree had a test f1 score of 0.747 and the optimized random forest had a test F1 score of 0.76. For predicting unknowable data, the optimized random forest was selected.

II. INTRODUCTION

Inpatient stay prediction is a crucial task in healthcare that involves predicting the length of a patient's hospitalization based on their medical history and symptoms. Accurate predictions can help hospitals and healthcare professionals allocate resources, save costs, and improve patient outcomes. Statistical, machine learning and deep learning models are the most common techniques for predicting inpatient stay.

While statistical models are straightforward and easy to understand, they may fail to account for the non-linear relationship between input and output variables. Machine learning models are adaptable and can deal with non-linear interactions, but they need an enormous amount of processing power. Deep learning models, which have shown remarkable success in a variety of industries, including healthcare, are especially well suited for inpatient stay prediction due to their capacity to learn complicated representations and capture nonlinear interactions. However, enormous volumes of data and computer resources are required for training.

Inpatient stay prediction has significant hurdles due to data harmonization and the variety of patients' medical histories. To solve this, researchers have created data preprocessing and integration techniques such as natural language processing and data integration. Additionally, the data imbalance, in which short-duration hospitalizations outnumber long-duration hospitalizations, might lead to biased models. To balance the data and increase prediction accuracy, researchers have proposed several strategies such as oversampling and cost-sensitive learning.

In conclusion, accurate inpatient stay prediction is essential for improving patient outcomes and lowering healthcare expenses. For inpatient stay prediction, a variety of methods are available, including statistical, machine learning, and deep learning models.

III. DATA

The original data is from HealthData: Hospital Inpatient Discharges (SPARCS De-Identified)(https://healthdata.gov). The data we are using is based on this, with some modifications. Our dataset has close to 60,000 records with 14 variables such as admission type, birth weight, and one target variable (lengthofstay)[3]

The Statewide Planning and Research Cooperative System (SPARCS) Inpatient De-identified File contains discharge level detail on patient characteristics, diagnoses, treatments, services, and charges.[3]

The first attribute is "HealthServiceArea," which is a categorical variable that describes the location of the facility. "Gender" is the next characteristic, which identifies the patient's gender. The next attribute is "Race," which indicates the patient's ethnicity. The fourth characteristic is "TypeOfAdmission", which describes how the patient was admitted to the hospital. The sixth attribute is "CCSProcedureCode", which is a code that defines the type of procedure conducted on the patient. The sixth feature is "APRSSeverityOfIllnessCode", which represents the severity of the patient's disease. The seventh property is "PaymentTypology," which details the payment method used for the patient's hospital stay. The eighth property is "BirthWeight," which is the neonate's weight in grams. The ninth property is "EmergencyDepartmentIndicator," which shows whether or not the patient was seen in the emergency room. The next four qualities (10-13) describe the county and facility's typical cost and charges for hospitalization. The fourteenth characteristic is "AverageIncomeInZipCode," that provides information about the patient's zip code's average income.The goal variable is "LengthOfStay," which is the total number of patient days spent in acute and/or non-acute care. We converted the target variable 'LengthOfStay' into a binary variable by classifying stays from 0 to 3 as 0 and stays from 4 to 10 as 1,

IV. DATA PRE-PROCESSING

Healthcare data due to its multi-channel data sources comes with missing values, errors, and outliers. To ensure that our data was accurate and relevant for analysis, we performed data pre-processing.

A. Cleaning Values

Cleaning data is another important step to ensure that any data that is ambiguous or not useful is excluded or changed to improve the overall quality of the dataset. After looking at our data we found two unsuitable values- PaymentTypology “Unknown” and Gender “U”. Gender "U" denoted patients whose gender was unknown and was deemed unsuitable for analysis. The payment type "Unknown" was eliminated since it is confusing and might be interpreted in a variety of ways. We replaced it with "Miscellaneous/Other".

B. Distribution of categorical attributes

The categorical variables are bar plotted to understand the count of each levels in an attribute. The variables are

- Gender
- race
- TypeOfAdmission
- CCSProcedureCode
- APRSeverityofIllnessCode
- PaymentTypology
- EmergencyDepartmentIndicator.

Interpretation: Discrepancies in EmergencyDepartmentIndicator and TypeOfAdmission. This means the model built would predict more accurately when the TypeOfAdmission is ‘newborn’ and EmergencyDepartmentIndicator is’N’ As there are more training records.

C. Outlier Detection

We included a data outlier detection step to ensure that the extreme values do not affect the data analysis. This was done using the Isolation Forest algorithm from the sklearn library. The algorithm isolates the data points by selecting a feature at random. This random feature is then explored further to randomly select a split value between maximum and minimum values. This process is done until the data points are segregated into small zones. These clustered data points are then discovered as Outliers.

D. Under Sampling

The undersampling method helps in the balancing of the target variable’s distribution, which can improve the performance of machine learning models. However, since we are deleting parts of sample, this can lead to loss of information from the majority class. As As a result, selecting a suitable sampling approach based on the problem’s specific objectives and goals is critical.

Since we converted the target variable ’LengthOfStay’ into a binary variable by classifying stays from 0 to 3 as 0 and stays from 4 to 10 as 1, there was a class imbalance with the majority of samples belonging to class 0. A histogram was drawn for each numerical variable in the dataset to determine if the class distribution was balanced, and the distribution of the categorical variables was also examined using a bar plot.

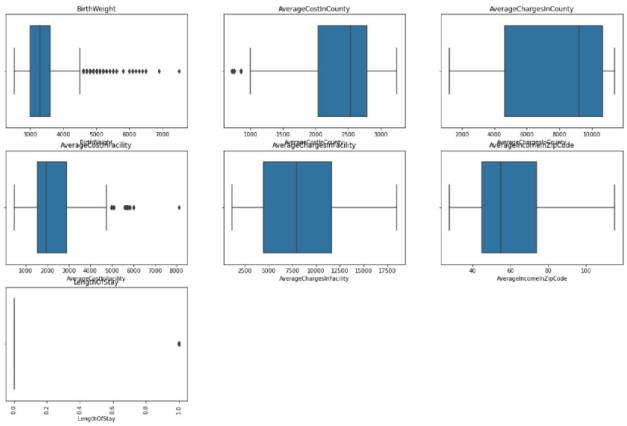

Fig. 1. Representation of the mechanism induced by traps on the average drain current.

E. Encoding

Our data contained categorical variables- Gender and PaymentTypology, which are qualitative data. This type of data cannot be used as input in machine learning. To convert it into machine learning appropriate numerical form, we used One Hot encoding technique.

This technique produced a binary vector for each category contained in the variable. For example, our data had two categories for the Gender variable: Male and Female. By following One Hot Encoding, it was divided into different columns for men and women. If the category applied to a row in the dataset, it was given a 1 in the associated column and a 0 in all other columns. Encoder for each characteristic was saved to ensure that the test data is encoded similarly to training data. This was done to make sure that the test data’s encoding is consistent with that of the training data as this is crucial for making correct predictions.

F. Data Reduction

This part of code involves removing unnecessary or irrelevant data which improves computational efficiency and reduces the risk of overfitting. We went ahead with SelectKBest‘ function from the ŝklearn.feature _ selection‘ module. The top ’k’ features with the strongest link to the target variable was chosen by this function.

This function selected the top ten features from the original dataset based on their correlation with the target variable, which is the ’Survived’ column. To assess the relationship between each feature and the target variable, the ’chi2’ scoring function is used.

The resulting reduced dataset comprised of only the top ten features that are most likely to have a substantial impact on survival prediction, which lead to increase in the machine learning model's accuracy and efficiency.

V. Methodology

Our code follows several steps including data cleaning, feature engineering, encoding categorical variables, and model training. We begin by importing all the necessary libraries- Pandas, Matplotlib, Sckitlearn, and Seaborn.

Since the size of the dataset is close to 60,000 records, we chose to use hold out validation split. We divided the data into train, validation and test data, The distribution was almost same in all the 3 different datasets with the size of train data being 33717, size of validation data being 11239, and size of test data being 14986.

The first step we followed through was loading the train and test data CSV files into Pandas using the read _ csv () function. Then to better understand the data structure, the head() function was used to show the first few rows of data. Each column's data types, and the number of non-null values were obtained using the info() function. This gave us a general overview of the data and helped in identifying the columns that need processing.

Our next step was data cleaning, which involved removing irrelevant and ambiguous columns from the dataset that do not add to the study. The ID column is used as the index, while the HealthServiceArea column is removed because it contains no valuable information. Any data with inaccurate or missing values, such as Gender "U" or PaymentTypology "Unknown," was eliminated.

Next, we did feature engineering. To enable categorical variable encoding we changed the CCSProcedureCode from int64 to object data type. The APRSeverityOfIllnessCode column, which is a categorical variable that reflects the severity of disease for patients, is converted to object data type.

The pair plot will be used to depict the distribution of dependent variables in respect to the target attribute. This would tell us whether or not the desired feature can be linearly separated.Only BirthWeight has a high number of outlier values.These outliers can be handled or left alone.

The outliers will not be considered for this assignment.The rationale for not treating the outliers is because there are newborn babies that are generally born underweight or overweight, need prompt treatment and maybe extending the duration of stay. As a result, if we handle these outlier numbers, they may have an effect on the length of stay. instead of preserving them.

Our data pre-processing was completed by using one-hot encoding to encode categorical variables. The categorical variables were identified and encoded to numerical values using OneHotEncoder. Encoder for each characteristic was saved to ensure that the test data is encoded similarly to training data. After that, the encoded values were concatenated with the original dataset.

After pre-processing the data, we trained the model, which uses the Random Forest Classifier algorithm to classify the data. Decision Tree and Random Forest are suitable algorithms for this type of problem because the dataset contains categorical information. These algorithms can handle both categorical and numerical data, as well as non-linear correlations between features and the target variable. Decision Trees can be used to interpret the model and learn about feature importance, whereas Random Forest may reduce the impact of overfitting and enhance accuracy.

The Random Forest Classifier algorithm's hyperparameters, such as the number of trees and the maximum depth of the trees, are tuned using GridSearchCV. The model is then trained using the preprocessed training data and used to test data to forecast the average duration of stay for patients.

Finally, the model's performance is assessed using measures like the F1-score, accuracy, and confusion matrix. The F1-score assesses the trade-off between precision and recall, whereas accuracy assesses the proportion of properly identified samples. The confusion matrix summarizes the model's performance by displaying the true positives, true negatives, false positives, and false negatives.

VI. RESULTS

The train f1 score for the base decision tree model is 0.959, while the validation f1 score is 0.771. The model, however, seems to be overfitting to the validation data based on the accuracy and f1 scores. To address this, the model is tuned using hyperparameters. The train f1 score of the optimized decision tree is 0.754, while the validation f1 score is 0.741. The optimized model has a f1 score of 0.747 on the test data. Criterion = "entropy", max depth = 6, min samples split=2, and class weight="balanced" are used in the optimized decision tree. The improved decision tree model outperforms the base decision tree model without any overfitting. As a result, we chose the decision tree to be represented by the optimized decision tree model.

The train f1 score for the Base Random Forest Model is 0.960, and the validation f1 score is 0.790. However, there is an obvious case of overfitting, like with the Base Decision Tree model, that must be handled with hyperparameter adjustment. It is unnecessary to evaluate the model using test data because it will overfit. To combat overfitting, the model is optimized by hyperparameter adjustment. On the test data, the optimized Random Forest Model has a train f1-score of 0.783, a validation f1-score of 0.762, and a f1-score of 0.76. The optimized model outperforms the base model, and no overfitting is detected in the optimized model. The

optimized model includes the following parameters: max _ depth = 10, min samples split = 2, and n estimators = 500. Therefore, the optimized model is chosen to represent the random forest.

Both the Decision Tree and Random Forest models performed well in terms of f1 score and confusion matrix after being optimized. On closer analysis, however, it became clear that the improved Random Forest model gave a higher f1 score on the test data. The confusion matrix also revealed that the Random Forest model outperformed the others in classifying positive instances, i.e., patients who stayed for longer than four days. Taking all the parameters into consideration, it was determined that the optimized Random Forest model performed better for this specific task and could be used to forecast the values for unknown data.

	Optimized Decision Tree	**Optimized Random forest**
Parameters	Criterion="entropy", maxdepth = 6 min_samples_split=2 class_weight="balanced"	Parameters: criterion="gini" max_depth = 10 min_samples_split=2n_estimators=500
F1 Score on Train Data	0.754	0.78
F1 Score on Validation Data	0.74	0.76
F1 Score on Test Data	0.74	0.76

Fig. 2. Optimize Decision Tree VS Optimize Random Forests

A. FUTURE INSIGHTS

More advanced machine learning methods and feature engineering techniques can be used in the future to improve the model's accuracy and F1 score. It can also be extrapolated to accommodate other data sources, manage missing values and outliers, and use cross-validation to evaluate the model's performance. It could also be improved in speed and memory utilization, especially for large datasets.

One way this code can be explored more is by adding new variables to the existing models. This could involve collecting and adding new patient data, such as demographic information, medical history, and vital signs, to improve the reliability and robustness of the models. Experimenting with various models such as naive bayes and neural networks, as well as feature engineering approaches, such as PCA, LDA, and feature selection algorithms may further refine the model.

VII. CONCLUSION

In conclusion, predicting inpatient stays is a critical issue in healthcare that can have a major impact on patient outcomes and healthcare costs. Statistical, machine learning, and deep learning models have all been used to predict inpatient stay, but each has advantages and limits. Additional data sources, such as electronic medical records and patient-generated data, can be added to improve model accuracy. Clinical decision assistance can also be provided by detecting high-risk patients and allowing for early intervention.

We studied two models for inpatient stay prediction- Decision Tree and Random Forest. While both models produced promising outcomes, after adjusting hyperparameters, the Random Forest model outperformed the Decision Tree model on the test data, with a higher f1 score and better positive instance classification. As a result, the improved Random Forest model was chosen to represent the inpatient stay prediction model, which can be used for tailored therapies, hospital operations optimization, and clinical decision support. Overall, predicting inpatient stay accurately is a critical part of healthcare which with future technological and data integration developments can enhance patient outcomes and efficiency.

A. Contributions

Narendra Gude: Data preprocessing

Priya Kaur: Performance Evaluation

Sai Narayan Das: Machine learning classifiers

B. References

1. Bacchi S, Gluck S, Tan Y, Chim I, Cheng J, Gilbert T, Menon DK, Jannes J, Kleinig T, Koblar S. Prediction of general medical admission length of stay with natural language processing and deep learning: a pilot study. Intern Emerg Med. 2020 Sep;15(6):989-995. doi: 10.1007/s11739-019-02265-3. Epub 2020 Jan 2. PMID: 31898204.

2. Lequertier V, Wang T, Fondrevelle J, Augusto V, Duclos A. Hospital Length of Stay Prediction Methods: A Systematic Review. Med Care. 2021 Oct 1;59(10):929-938. doi: 10.1097/MLR.0000000000001596. PMID: 34310455.

3. Hospital Inpatient Discharges (SPARCS De-Identified): 2021 | HealthData.gov," Tyler Data Insights. https://healthdata.gov/State/Hospital-Inpatient-Discharges-SPARCS-De-Identified/szqf-xu7c/data

4. K. Stone, R. Zwiggelaar, P. Jones, and N. Mac Parthaláin, "A systematic review of the prediction of hospital length of stay: Towards a unified framework," PLOS

Digital Health, vol. 1, no. 4, p. e0000017, Apr. 2022, doi: 10.1371/journal.pdig.0000017.

5.F. Jaotombo et al., “Machine-learning prediction for hospital length of stay using a French medico-administrative database,” Journal of Market Access Health Policy, vol. 11, no. 1, Nov. 2022, doi: 10.1080/20016689.2022.2149318.