Patient Survival Prediction Analysis Report

Author

Marco Narcisi

Executive Summary

This report presents the development and evaluation of a deep learning model designed to predict patient survival during hospital stays. Using a dataset that includes patient demographics, health conditions, and clinical measurements, we trained a neural network with the aim to assist healthcare providers in identifying high-risk patients and inform clinical decision-making.

Main Objective

The primary objective of this analysis is to leverage deep learning to predict hospital patient survival with high accuracy and recall. By employing complex models capable of extracting patterns from the data, we aim to provide valuable insights that could lead to improved patient outcomes and hospital care efficiency.

Data Description

The dataset comprises 91,713 patient records from a hospital, each with 85 attributes. These include basic demographics like age and gender, clinical measurements, and information regarding the patients' medical history and hospital stay. The target variable is 'hospital_death', a binary indicator of whether the patient survived the hospital stay.

Data Exploration and Preprocessing

Initial exploratory data analysis (EDA) revealed a class imbalance with fewer instances of patient deaths compared to survivals. The dataset also contained missing values and featured a mix of numerical and categorical variables. To prepare the data for modeling, we imputed missing values, standardized numerical features, and one-hot encoded categorical variables through a robust preprocessing pipeline.

Model Development and Training

We selected the neural network architecture for its ability to model complex relationships in high-dimensional data. Three variations of deep learning models were trained to explore the trade-offs in complexity and performance:

- Model 1: A simpler network with fewer layers.
- Model 2: A baseline model with moderate complexity.
- Model 3: A more complex network with additional layers and units.

Model 2 was selected as the final model due to its balance of performance and complexity, obtaining an accuracy of 92.78% on the test data.

Model Evaluation

Beyond accuracy, we examined precision, recall, and F1-scores, considering the class imbalance in the dataset. Model 2 showed strong performance for the majority class (survival) but was less effective in identifying non-survivals, with an F1-score of 42% for the minority class.

To visualize model performance, the following plots could be included in the report:

- Histograms of Age and BMI distributions.
- Count plots of Gender and Ethnicity.
- Bar plots of feature importance.
- The ROC curve and Precision-Recall curves for Model 2.

(Instructions on generating plots not provided here due to environmental constraints but would typically be done using libraries like Matplotlib and Seaborn in Python.)

Key Findings and Insights

The baseline neural network model is capable of identifying patient survival with high accuracy. However, it underperforms in correctly identifying patients who will not survive, which is critical in a healthcare setting.

Future Work

Future analyses should focus on improving recall for non-survivors, perhaps by modifying the model's decision threshold or employing resampling techniques to balance class distribution. Exploring model interpretability with tools like SHAP could also yield insights into the model's decision patterns and potentially uncover important predictive features that the model may be currently undervaluing.

Appendix

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping

# Load the dataset
df = pd.read_csv('patient_data.csv')

# Preprocessing steps

# Select numerical and categorical columns
num_cols = df.select_dtypes(include=['int64', 'float64']).columns.tolist()
cat_cols = df.select_dtypes(include=['object']).columns.tolist()

# Exclude non-predictor columns and the target variable from the list of numerical
columns
num_cols = [col for col in num_cols if col not in ['encounter_id', 'patient_id',
'hospital_id', 'hospital_death']]

# Define imputers for missing values
num_imputer = SimpleImputer(strategy='median')
cat_imputer = SimpleImputer(strategy='most_frequent')

# Define transformer pipelines
num_transformer = Pipeline(steps=[
    ('imputer', num_imputer),
    ('scaler', StandardScaler())
])

cat_transformer = Pipeline(steps=[
    ('imputer', cat_imputer),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

# Combine preprocessing for numerical and categorical data
preprocessor = ColumnTransformer(transformers=[
    ('num', num_transformer, num_cols),
    ('cat', cat_transformer, cat_cols)
])

# Preprocess the data
X = df.drop('hospital_death', axis=1)  # Predictor variables
y = df['hospital_death']  # Target variable
```

```
X_preprocessed = preprocessor.fit_transform(X)
y = y.values  # Ensure y is a numpy array

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_preprocessed, y, test_size=0.2,
random_state=42)

# Neural network model
model = Sequential([
    Dense(64, activation='relu', input_dim=X_train.shape[1]),
    Dropout(rate=0.2),
    Dense(32, activation='relu'),
    Dropout(rate=0.2),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Early stopping callback
early_stopping = EarlyStopping(patience=5, restore_best_weights=True)

# Model fitting
history = model.fit(
    X_train,
    y_train,
    validation_split=0.2,
    batch_size=128,
    epochs=30,
    callbacks=[early_stopping],
    verbose=1
)

# Model evaluation
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=1)
print(f'Test Loss: {test_loss}')
print(f'Test Accuracy: {test_accuracy}')

# Predictions for evaluation metrics
y_pred = (model.predict(X_test) > 0.5).astype('int32')

# Performance metrics
print(classification_report(y_test, y_pred))

# Confusion matrix
conf_mat = confusion_matrix(y_test, y_pred)
print(conf_mat)
```

Conclusion

This analysis demonstrates the potential of deep learning in health informatics. While the developed model shows promise, especially in its ability to predict patient survival, its current limitations in predicting non-survival cases necessitate further investigation and model refinement. The report suggests a strategic plan for continued model development aimed at improving predictions for all patient outcomes, ultimately supporting healthcare professionals in delivering timely and effective patient care.

Data Source

The datasets used in this analysis were sourced from the following providers under their respective licenses:

- Data provided by Mitisha Agarwal, available at Kaggle: Patient Survival Prediction
- License: Data files @ Original Authors.