In-hospital Intensive Care Unit Mortality Prediction Model

COMPUTING FOR DATA SCIENCES

GROUP 6:
MANASWI VELIGATLA (24), NEETI POKHARNA (27),
ROBIN SINGH (36), SAURABH RAWAL (42)
Impact

- Problem Statement
- Methodology
- Challenges and Steps to overcome
- Implementation
- Program structure
- Results
- Improvement steps
- References
CONCEPT

- Using Artificial intelligence and predictive analytics in hospitals
- Huge amount of data generated in hospitals

Concerns

- High Reliability is required
- Highly domain knowledge centric field - reflected in methodology also
IMPACT

- Saves LIFE

- Focus resources on and only-on patients who need

- Data backed decision making for Doctors
Numbers

Expenditure on healthcare in India – 50 Billion USD

Number of Doctors – 7 lakhs

Average cost per survivor from ICU – Rs. 17,000

Nearly 40% of the people admitted to ICU have to borrow money or sell assets

Source:

*http://www.ijccm.org/article.asp?issn=0972-5229;year=2008;volume=12;issue=2;spage=55;epage=61;aulast=Jayaram
Impact

Problem Statement

Methodology

Challenges and Steps to overcome

Implementation

Program structure

Results

Improvement steps

References
Predict risk of death (Mortality) in patients admitted in Intensive Care Unit (ICU) in a hospital.

5990 (simulated) patient records where each patient record had following variables:

- **ID**: a unique identifier for each patient
- **Age**
- **6 Vitals**: Blood Pressure, Heart Rate, Respiration Rate, Oxygen Saturation, Temperature
- **25 Labs**: like Albumin, WBC Count, Hematocrit, Urine Output, etc.
- **Timestamps**: measurement time relative to first measurement for patient (First, timestamp 0)
- **ICU flag**: indicates whether a patient is in ICU or not at a given time
- **Mortality label**: indicates whether a patient survived or died (the label or outcome variable) at the end of hospital stay
<table>
<thead>
<tr>
<th>Patient ID</th>
<th>Age</th>
<th>Time Stamp</th>
<th>ICU Flag</th>
<th>Vital lab measurement (6 Col)</th>
<th>Labs measurements (25 Col)</th>
<th>Mortality Label (Only in train dataset)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1</td>
<td></td>
<td>T2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1</td>
<td></td>
<td>T3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>80</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>P2</td>
<td></td>
<td>T5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td></td>
<td>T2</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>P2</td>
<td></td>
<td>T7</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Constraints

- Prediction only for patients in ICU
- Prediction for all time stamps of the patient
- Only history data of patient for prediction
- Overall prediction – at least one 1 for final prediction 1
Performance Metrics

- Final Score
  - Sensitivity
  - Specificity
  - Median Prediction time
Performance Metrics

<table>
<thead>
<tr>
<th>Actual Outcome</th>
<th>Prediction</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dead</td>
<td>Dead</td>
<td>True Positive (TP)</td>
</tr>
<tr>
<td>Dead</td>
<td>Alive</td>
<td>False Negative (FN)</td>
</tr>
<tr>
<td>Alive</td>
<td>Dead</td>
<td>False Positive (FP)</td>
</tr>
<tr>
<td>Alive</td>
<td>Alive</td>
<td>True Negative (TN)</td>
</tr>
</tbody>
</table>

Sensitivity = \( \frac{TP}{TP + FN} \)

Specificity = \( \frac{TN}{TN + FP} \)
Problem Discussion - Metrics

Median Prediction Time:

Only for true positives:

<table>
<thead>
<tr>
<th>Patient ID</th>
<th>Time Stamp</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2000</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>5893</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>6137</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>7889</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>9578</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>10345</td>
<td>0</td>
</tr>
</tbody>
</table>
Score =

\[
100 \times \{ \\
0.75 \times \text{Sensitivity} \\
+ \\
0.2 \times \text{Median Prediction time clipped at 72} \\
+ \\
0.05 \times (\text{Specificity} - 0.99)
\}
\]
Impact

Problem Statement

**Methodology**

Challenges and Steps to overcome

Implementation

Program structure

Results

Improvement steps

References
Methodology

Train data → Train Data modified features → Selective Train data with labels

Test Data → Test Data Modified Features → Classifier

Input → Model → Output

Classifier → Test data predictions
Project Stages

- Test Data
- Validation Data
- Train Data

1% data by volume

60% - 40% split

Combined train and validation

Strategy -> Model 1 -> Model 2 -> Model 3 -> Submission
Impact

Problem Statement

Methodology

Challenges and Steps to overcome

Implementation

Program structure

Results

Improvement steps

References
Challenges

1. Healthcare Variables
2. Missing Values (more than 95% values missing)
3. Train data label assignment
4. Large Data Size (approx. 6 Lakh rows)
5. Minimum score on two-of-the-three metrics
6. Limited attempts submission on test dataset
Healthcare Variables

- Non – linear relation to mortality
- Effective in combinations (e.g. Oxygen Saturation, Carbon Dioxide)
- Depends highly on person to person (e.g. smokers and non-smokers)
- Mortality v/s Morbidity
- Exhaustive Coverage of all mortality reasons is difficult

Overcoming
- Consulted doctors
- Literature review
- Verified using Rpart
Missing Values

- More than 95% data missing
- Data missing for different time stamps for the same patient

For every patient-timestamp{
    for every feature{
        if current value is missing{
            fill with worst value in last 24 hours
        }
        else : fill with worst value since ICU entry
        else : fill with worst value since hospital entry
        else : fill with the normal value for the feature
    }
}
Train Data Label Assignment

- Mortality label only given for patients not patient-timestamp combination
- Aggressive v/s Conservative model

<table>
<thead>
<tr>
<th>Case</th>
<th>Label Assigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient who ultimately died</td>
<td></td>
</tr>
<tr>
<td>Combination of best value of features from Non-ICU Data</td>
<td>0</td>
</tr>
<tr>
<td>Combination of Worst value of features from ICU Data</td>
<td>1</td>
</tr>
<tr>
<td>Patient who was alive after ICU</td>
<td></td>
</tr>
<tr>
<td>Combination of best value of features from Non-ICU Data</td>
<td>0</td>
</tr>
<tr>
<td>Combination of Worst value of features from ICU Data</td>
<td>0</td>
</tr>
</tbody>
</table>
Large Data Size

- Approx. 6 Lakh rows
- Approx. Feature development time on test set – 35 Hours on PC
- Multiple data slicing involved

Overcome

- Used small but representative dataset while coding (approx. 1% of full dataset)
- Distributed Feature development task on different computers
Minimum score on metrics

- Minimum specificity = 0.99 & Minimum Median Prediction Time – 5 hours
- Specificity v/s Sensitivity tradeoff
- Specificity v/s Median Prediction Time Tradeoff
- Low sensitivity leading to high run-run variation in Median Prediction Time

Overcome

- Vary train data label weights
- Conducted many runs to get the optimum score model parameters
Limited attempts submission on test dataset

- Only 3 submission per team on test data
- High run-to-run variation in metrics
- Model invalid if the minimum metric not achieved

Overcome

- Using Model parameter values which resulted in lower run-to-run variation
- Using conservative parameter values to reduce risk and hence compromising on the final score
- Impact
- Problem Statement
- Methodology
- Challenges and Steps to overcome

**Implementation**

- Program structure
- Results
- Improvement steps
- References
Implementation

- Language – Python, R

- Packages - numpy, pandas, Scikit-learn, os, csv, rpart, e1071

- Some important functions - merge, subset, rpart, crossValidation, RandomForestClassifier, KNeighborsClassifier, svm
Impact
Problem Statement
Methodology
Challenges and Steps to overcome
Implementation

Program structure
Results
Improvement steps
References
Program Structure for train data

For every patient in training data
{
  if patient died
  {
    Extracting modified feature from non-icu data of the current patient
    Extracting modified features from icu data of the current patient
  }
  Else
  {
    Extracting modified feature from non-icu data of the current patient
    Extracting modified features from icu data of the current patient
  }
}
Program Structure for test data

For every patient timestamp in test data
{
  if patient in ICU
  {
    Creating modified feature for the current patient timestamp using his/her historical data
  }
}

Train classifier using the extracted modified feature matrix

Predicting mortality for every timestamp (in test data) when patient is in ICU
```python
def score_mean_blood_pressure(systolic_bp, diastolic_bp):
    if not pd.isnull(systolic_bp):
        if not pd.isnull(diastolic_bp):
            mean_bp = (systolic_bp + diastolic_bp) / 2
            if mean_bp <= 39:
                return 23
            elif mean_bp > 39 and mean_bp < 60:
                return 15
            elif mean_bp >= 60 and mean_bp < 70:
                return 7
            elif mean_bp >= 70 and mean_bp < 80:
                return 6
            elif mean_bp >= 80 and mean_bp < 100:
                return 0
            elif mean_bp >= 100 and mean_bp < 120:
                return 4
            elif mean_bp >= 120 and mean_bp < 130:
                return 7
            elif mean_bp >= 130 and mean_bp < 140:
                return 9
            elif mean_bp >= 140:
                return 10
    # Mean BP modified feature for Non-ICU of current patient if he/she finally died
    pressure_data = data_sub[['V1', 'V2']]
    score_list = []
    for index, row in pressure_data.iterrows():
        systolic_bp = float(row['V1'])
        diastolic_bp = float(row['V2'])
        if not pd.isnull(systolic_bp):
            if not pd.isnull(diastolic_bp):
                mean_bp = score_mean_blood_pressure(systolic_bp, diastolic_bp)
                score_list.append(mean_bp)
        else:
            mean_bp_non_icu_score = 0
    if not score_list:
        mean_bp_non_icu_score = 0
    else:
        mean_bp_non_icu_score = min(score_list)
```
# Mean BP modified feature
systolic_current_value = float(row['V1'])
diastolic_current_value = float(row['V2'])
if ((pd.isnull(systolic_current_value)) or (pd.isnull(diastolic_current_value))):
    if timestamp > 0:
        data_sub = val_df[(val_df.ID == patient_id) & (val_df.TIME < timestamp)]
        bp_data_history = data_sub[['V1', 'V2']]
        if timestamp < (3600*24):
            score_list = []
            for index, row_1 in bp_data_history.iterrows():
                systolic_bp = float(row_1['V1'])
                diastolic_bp = float(row_1['V2'])
                if (pd.isnull(systolic_bp)) or (pd.isnull(diastolic_bp)):
                    score_list.append(0)
                else:
                    mean_bp = score_mean_blood_pressure(systolic_bp, diastolic_bp)
                    score_list.append(mean_bp)
                if not score_list:
                    mean_bp_score = 0
                else:
                    mean_bp_score = max(score_list)
        else:
            timestamp_less_24 = timestamp - (3600*24)
            data_sub = val_df[(val_df.ID == patient_id) & (val_df.TIME < timestamp) & (val_df.TIME > timestamp_less_24)]
            bp_data_history = data_sub[['V1', 'V2']]
            score_list = []
            for index, row_2 in bp_data_history.iterrows():
                systolic_bp = float(row_2['V1'])
                diastolic_bp = float(row_2['V2'])
                if (pd.isnull(systolic_bp)) or (pd.isnull(diastolic_bp)):
                    score_list.append(0)
                else:
                    mean_bp = score_mean_blood_pressure(systolic_bp, diastolic_bp)
                    score_list.append(mean_bp)
                if not score_list:
                    mean_bp_score = 0
                else:
                    mean_bp_score = max(score_list)
else:
    mean_bp_score = 0
else:
    mean_bp_score = score_mean_blood_pressure(systolic_current_value, diastolic_current_value)
Impact

Problem Statement

Methodology

Challenges and Steps to overcome

Implementation

Program structure

**Results**

Improvement steps

References
Results

- Better results with Random Forest Classifier than KNN and SVM
Improvement Steps

- Different classifiers; tweaking depth and sample weight

- KNN - lower run to run variance Vs Random forest - higher median prediction time

- Added train and validation data as training data for prediction on test data
Impact
Problem Statement
Methodology
Challenges and Steps to overcome
Implementation
Program structure
Results
Improvement steps

References
References

- Literature
  - Published approaches from Physionet challenge 2012
  - National Centre for Biotechnology Information (NCBI)
  - Journal of intensive care

- Doctors consulted
  - Dr. Priyanka Singh
  - Dr. Tejaswi
  - Dr. Ram Kiran

- www.stackoverflow.com
- Hackerrank Discussion forum
- Discussions with classmates (Pradeep Mooda, Avinash Kumar)
- Lord Google 😊
Questions?