

RETAIN  
(REverse Time  
Attention)

# Introduction

- Accuracy and interpretability are vital in predictive models.
- A common dilemma in predictive modeling is choosing between complex models like RNN for accuracy or more interpretable models like logistic regression.
- Medicine faces this dilemma where both accuracy and interpretability are crucial.
- Hence RETAIN, a Reverse Time Attention model for Electronic Health Records (EHR) data can be used to solve such problem.
- In this tutorial RETAIN is used for mortality prediction.

# Why RETAIN?

<https://arxiv.org/abs/1608.05745>

- RETAIN strikes a balance between predictive accuracy and interpretability, making it a valuable tool for healthcare applications.
- It can deliver accurate predictions while providing clear insights into why specific predictions were made.
- RETAIN processes data with the most recent clinical visits. This approach mimics how physicians often analyze patient records, giving more weight to recent visits.

 > cs > arXiv:1608.05745

Search...  
Help | Advanced Search

Computer Science > Machine Learning

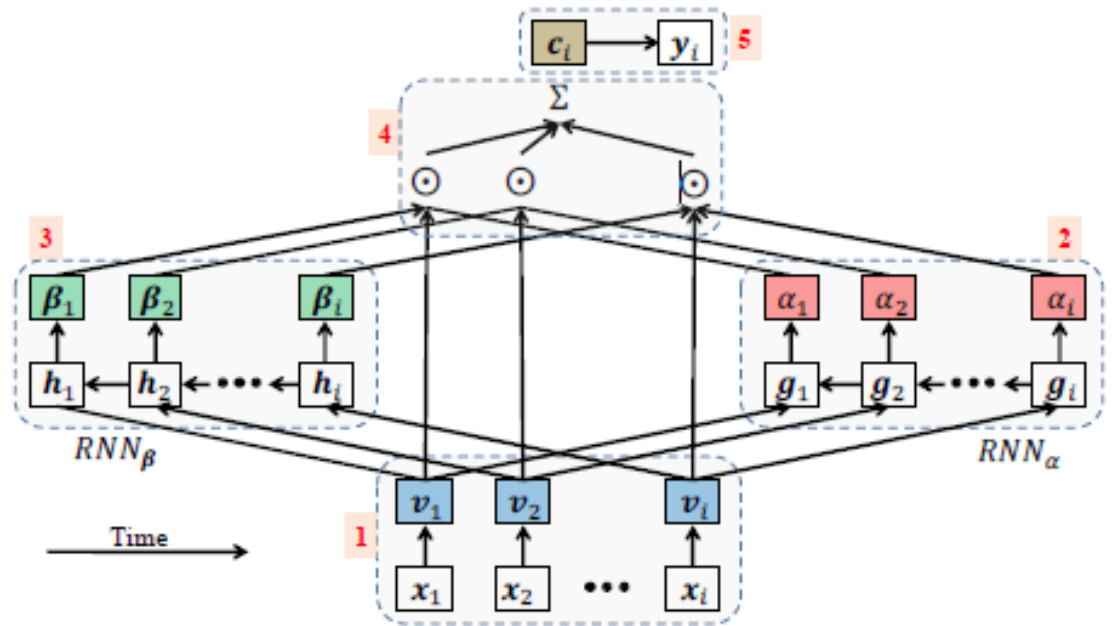
*[Submitted on 19 Aug 2016 (v1), last revised 26 Feb 2017 (this version, v4)]*

**RETAIN: An Interpretable Predictive Model for Healthcare using Reverse Time Attention Mechanism**

Edward Choi, Mohammad Taha Bahadori, Joshua A. Kulas, Andy Schuetz, Walter F. Stewart, Jimeng Sun

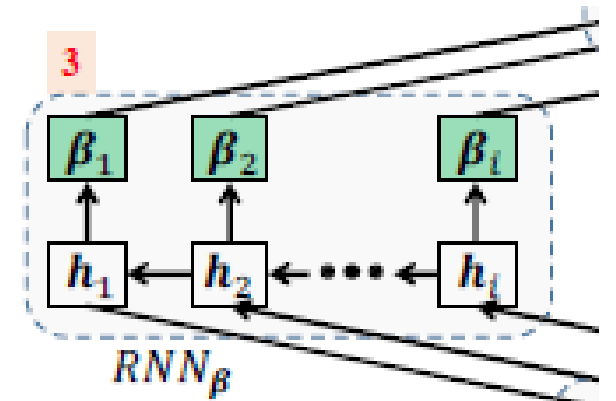
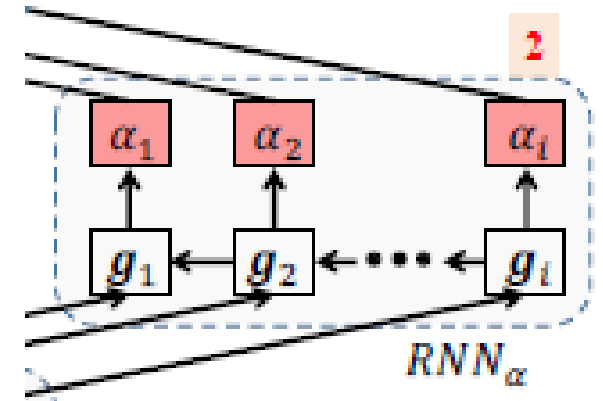
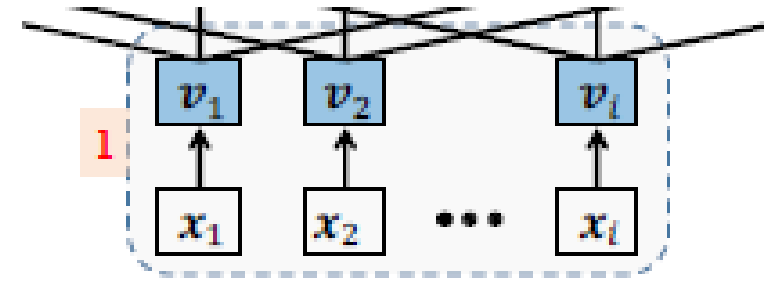
# Architecture of RETAIN

- A critical feature of RETAIN is its two-level neural attention mechanism.
- This mechanism operates at both the visit-level ( $RNN_{\alpha}$ ) and variable-level ( $RNN_{\beta}$ ), allowing the model to focus on relevant visits and specific clinical variables.



# Working of RETAIN

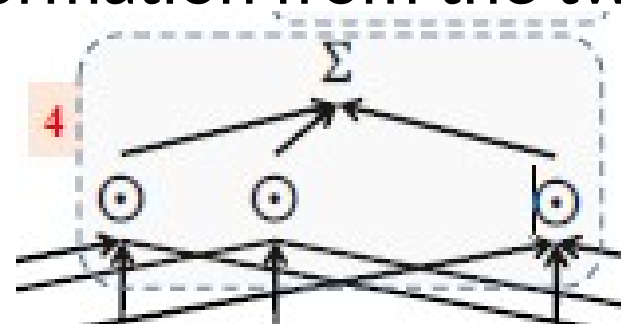
- Step 1: Visit embeddings 'v' are generated through a linear embedding.
- Step 2: The model uses one RNN to generate visit-level attention weights. This RNN processes visit embeddings and determines the importance of each visit in the patient's history.
- Step 3: A second RNN is employed to generate variable-level attention weights. This RNN operates in parallel with the visit-level RNN but focuses on individual clinical variables within each visit.



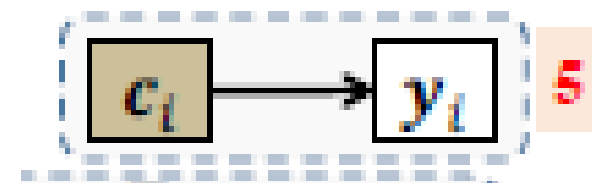
# Working of RETAIN

- Step 4: For each patient's sequence of visits, RETAIN calculates a context vector by combining the weighted information from the two RNNs and input vector.

- $\text{o/p of RNN}\alpha + \text{RNN}\beta + \text{Input vector} = \text{Context vector}$



- Step 5: Uses this context vector to make predictions (binary classification) regarding clinical outcomes. For example, it can predict whether a patient will be diagnosed with a particular disease based on their EHR data.



# Interpreting RETAIN: Understanding Variable Contributions

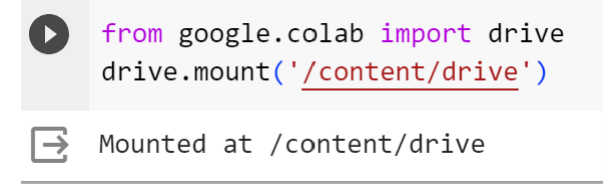
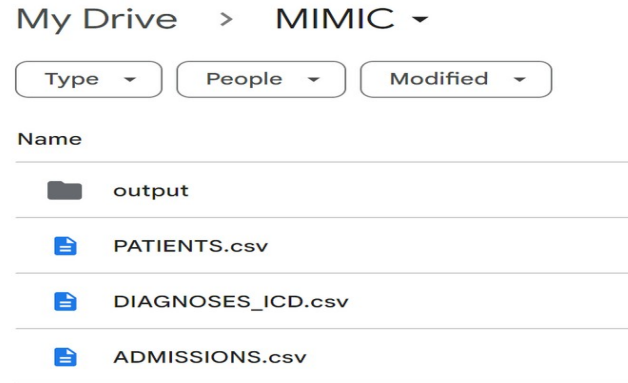
- Identifying Important Visits:

To find the visits that play a significant role in predictions, we focus on the attention weight  $\alpha$ . This weight helps us identify visits that carry more weight in the model's predictions.

- Identifying influential Variable:

A visit consists of multiple medical variables, each contributing differently to predictions. The exact contribution of each variable is determined by a combination of factors:  $v$ ,  $\alpha$  and  $\beta$  as understanding  $\alpha$  by itself tells us which visit is important, but not why.

# Retain Tutoria



- Prepare Data:

- Datasets we need

- Admissions table
    - Diagnoses\_ICD table
    - Patients table

- Store them in your google drive (MyDrive/MIMIC)
  - Your output files will go to the out\_directory which is in your google drive (MyDrive/MIMIC/output)

- Tutorial:

<https://colab.research.google.com/drive/1pzSZriI0OUefHh3MqgEybrspKIY-bw6a?authuser=1#scrollTo=HsIUPP8BCnIH>

```
[ ] # Define the paths to your CSV files in the "MIMIC" folder in your Colab environment
admission_file = "/content/drive/MyDrive/MIMIC/ADMISSIONS.csv"
diagnosis_file = "/content/drive/MyDrive/MIMIC/DIAGNOSES_ICD.csv"
patients_file = "/content/drive/MyDrive/MIMIC/PATIENTS.csv"
out_directory = "/content/drive/MyDrive/MIMIC/output"
train_proportion = 0.8 # Modify the train proportion as needed
```



# Data Formatting

- └ The primary goal is to create longitudinal patient records.
- └ It identifies patients with two or more encounters in the MIMIC-III dataset and organizes their data in a chronological order.
- └ The code takes Admissions, Diagnosis\_ICD and Patients table and output consists of pickled pandas dataframes, dictionary mapping

└ Some of the key functions include

- └ `convert_to_icd9(dx_str)`
  - 0 Maps an ICD diagnosis code to ICD9 format.
  - 0 Handles both regular and "E" diagnosis codes.
- └ `convert_to_3digit_icd9(dx_str)`
  - 0 Rolls up a diagnosis code to 3 digits.
  - 0 Accommodates both regular and "E" diagnosis codes.

```
[ ] print(convert_to_icd9('49320'))  
    print(convert_to_icd9('250'))  
    print(convert_to_icd9('E8504'))  
    print(convert_to_icd9('E950'))  
    print(convert_to_icd9('V750'))
```

```
493.20  
250  
E850.4  
E950  
V750
```

```
▶ print(convert_to_3digit_icd9('49320'))  
  print(convert_to_3digit_icd9('250'))  
  print(convert_to_3digit_icd9('E8504'))  
  print(convert_to_3digit_icd9('E950'))  
  print(convert_to_icd9('V750'))
```

```
493  
250  
E850  
E950  
V750
```

# Data Structure

- The data of one patient looks like this

```
pid: 23
2153-09-03 07:15:00
['D_414.01', 'D_411.1', 'D_424.1', 'D_V458.2', 'D_272.4', 'D_401.9', 'D_600.00', 'D_389.9']
2157-10-18 19:34:00
['D_225.2', 'D_348.5', 'D_780.39', 'D_424.1', 'D_401.9', 'D_272.0', 'D_272.4', 'D_V458.1', 'D_V457.9', 'D_V158.2']
Dead_or_alive: 0
```

- Pid: is the patient id
- Date of admission
- Diagnosed ICD9 code for that visit
- In this case patient id 23 visited 2 times and respective ICD9 codes for those visits are displayed
- Last element is whether the patient is dead or alive

# Data Structure

- The diagnosis ICD9 codes which are in string format needs to be converted into embeddings

```
[20] for i in range(len(dates[row])):  
      print(dates[row][i])  
      print(seqs[row][i])
```

```
2153-09-03 07:15:00
```

```
['D_414.01', 'D_411.1', 'D_424.1', 'D_V458.2', 'D_272.4', 'D_401.9', 'D_600.00', 'D_389.9']
```

```
2157-10-18 19:34:00
```

```
['D_225.2', 'D_348.5', 'D_780.39', 'D_424.1', 'D_401.9', 'D_272.0', 'D_272.4', 'D_V458.1', 'D_V457.9', 'D_V158.2']
```

```
[21] new_seqs[row]
```

```
[[0, 1, 2, 3, 4, 5, 6, 7], [8, 9, 10, 2, 5, 11, 4, 12, 13, 14]]
```

- it starts with zero and if any code repeats than it is assigned the previous value.
- example: D\_424.1 (encoded as 2) appears in 3rd position and in 12th position. Hence 2 is repeated twice. Similarly, D\_272.4 appears 2 times.

# Date and time format

- It calculates the time difference between a reference date (January 1, 2025) and a list of dates available. The result is stored in a variable called `to_event` and sort the data according to date.
- For each date in a patient's record, it calculates a numeric value using the formula: `date.hour * 60 + date.minute - 720`. Then stores in 'numerics'
  - `date.hour` returns the hour part of the date in minutes (e.g., 9 AM would be 540 minutes).
  - `date.minute` returns the minute part of the date.
  - Subtracting 720 represents the offset from noon (720 minutes) to obtain values that can be positive or negative.

For example, if `dates` contains something like this:

```
dates = [[datetime(2023, 10, 25, 9, 30), datetime(2023, 10, 25, 14, 15)], [datetime(2023, 10, 25, 8, 0), datetime(2023, 10, 25, 11, 45)]]
```

The `numerics` variable would be calculated as follows:

```
numerics = [[-150, 135], [-240, -15]]
```

# Input and Output data to model

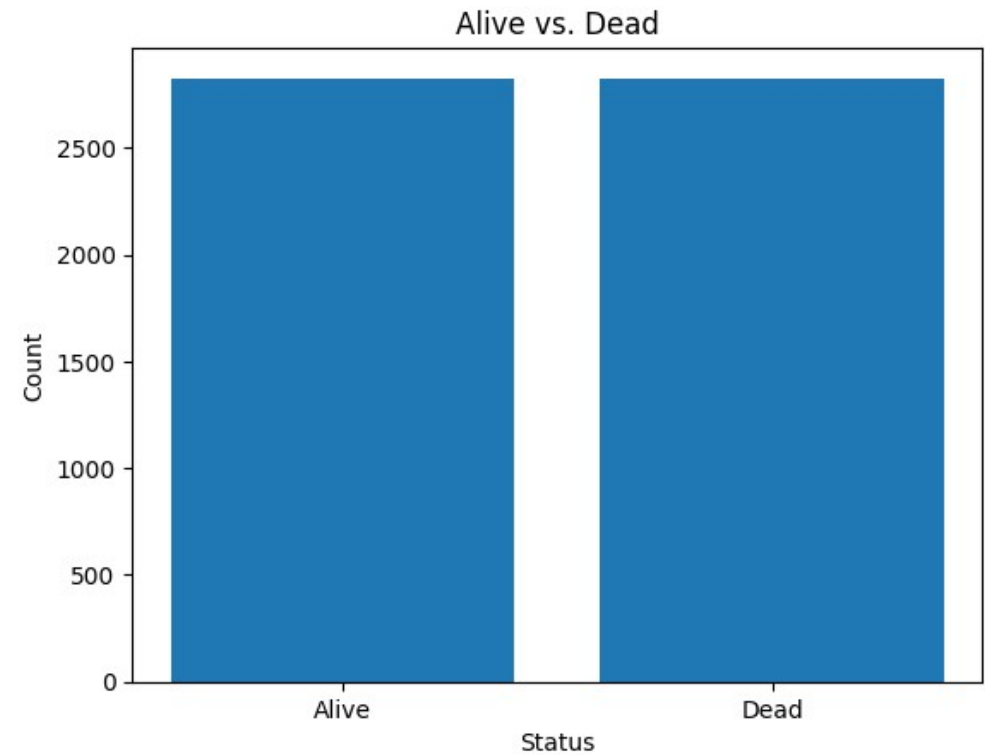
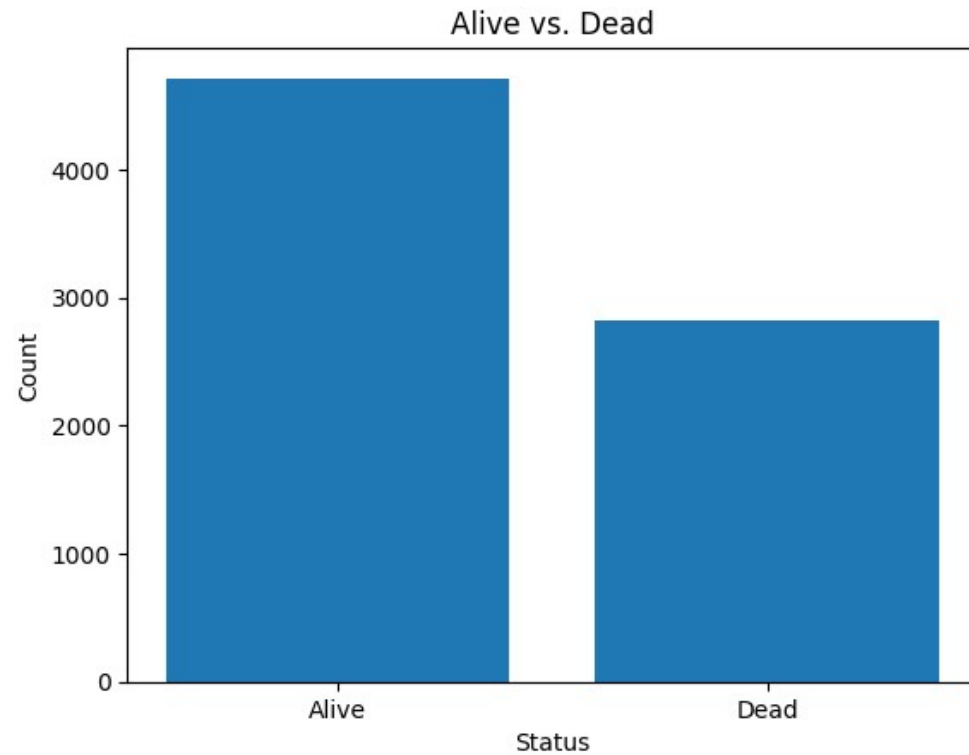
- The input data (features) and output data (labels) are given to model as shown below

	index	codes	to_event	numerics
0	0	[[0, 1, 2, 3, 4, 5, 6, 7], [8, 9, 10, 2, 5, 11...	[-46997, -48503]	[[[-285], [454]]]
1	4691	[[670], [1879, 5, 4]]	[-56002, -56038]	[[[-150], [-285]]]
2	4690	[[62, 23, 966, 20, 16, 403, 18, 54, 315, 149],...	[-51247, -52844]	[[[-95], [580]]]
3	4688	[[101, 103, 59, 1496, 1812, 105, 0, 2, 285, 6,...	[-61255, -62614]	[[[-509], [-588]]]
4	4687	[[62, 17, 23, 92, 16, 18, 0, 331, 91, 2124, 24...	[-49887, -49907]	[[[50], [306]]]

	index	target
0	0	0
1	4691	0
2	4690	0
3	4688	1
4	4687	0
















# Data imbalance

- There is a clear imbalance in class distribution
- To balance the classes, random under sampling is used



# Training and test data files

- ↗ data\_train.pkl and data\_test.pkl: Pickled dataframes for training and testing, containing patient codes and time-to-event sequences.
- ↗ data\_train\_3digit.pkl and data\_test\_3digit.pkl: Pickled dataframes for training and testing, with 3-digit diagnosis codes.
- ↗ target\_train.pkl and target\_test.pkl: Pickled dataframes containing target labels for training and testing.

 data_test_3digit.pkl	 me	4:37 PM me	203 KB
 data_test.pkl	 me	4:37 PM me	207 KB
 data_train_3digit.pkl	 me	4:37 PM me	820 KB
 data_train.pkl	 me	4:37 PM me	838 KB
 dictionary_3digit.pkl	 me	4:37 PM me	8 KB
 dictionary.pkl	 me	4:37 PM me	8 KB
 target_test.pkl	 me	4:37 PM me	36 KB
 target_train.pkl	 me	4:37 PM me	142 KB

“Pickling” is the process whereby a Python object hierarchy is converted into a byte stream, and “unpickling” is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy.

<https://docs.python.org/3/library/pickle.html#:~:text=%E2%80%9CPickling%E2%80%9D%20is%20the%20process%20whereby,back%20into%20an%20object%20hierarchy.>

# Training

Hyper parameters to consider while training

1. num\_codes: Number of medical codes.
2. numeric\_size: Size of numeric inputs (0 if none).
3. use\_time: Flag to indicate the use of time input.
4. emb\_size: Size of the embedding layer.
5. epochs: Number of training epochs.
6. n\_steps: Maximum number of visits after which data is truncated.
7. recurrent\_size: Size of the recurrent layers.
8. path\_data\_train: Path to the training data.
9. path\_data\_test: Path to the testing data.
10. path\_target\_train: Path to the training target.
11. path\_target\_test: Path to the testing target.
12. batch\_size: Batch size for training.
13. dropout\_input: Dropout rate for embedding.
14. dropout\_context: Dropout rate for context vector.
15. l2: L2 regularization value.
16. directory: Directory to save the model and log files.
17. allow\_negative: Flag to allow negative weights for embeddings/attentions.



# Freeze padding during training

- The FreezePadding class is a custom constraint used for weight constraints in neural network layers.
- This constraint is applied to the last weight in the weight matrix of certain layers to ensure that this weight is "frozen" or set to specific values, typically near 0. This is required because:
  - a. The embeddings are learned during training and by using this constraint, the model ensures that the last weight in the embeddings layer remains near 0 preventing negative embeddings.
  - b. In some applications, especially in healthcare analytics, negative weights in embeddings or attention mechanisms might not make intuitive sense. By "freezing" the last weight to be near 0, the model effectively prevents negative weights from affecting the representation of medical codes or attention scores.

# Creating Retain Mc

```
def model_create():  
    """  
    Create tensorflow DAG for training a model, and then compile/train  
    the model at the end.  
  
    :return: trained/compiled Keras model  
    :rtype: :class:`tensorflow.keras..Model`  
    """
```

- The `model_create` function is responsible for creating a Keras model using the TensorFlow library. This model is designed to mimic the RETAIN architecture and is used for training and prediction.
- The architecture of this model follows the RETAIN architecture and is suitable for binary classification tasks where the goal is to predict binary outcomes. The model uses attention mechanisms to weigh the importance of different elements in the input data. It can handle different types of input data, including codes, numerics, and time, and make predictions based on these inputs. The use of bidirectional LSTMs and attention mechanisms is a key characteristic of the RETAIN architecture, which aims to capture sequential dependencies in the data.

# Creating Retain Model

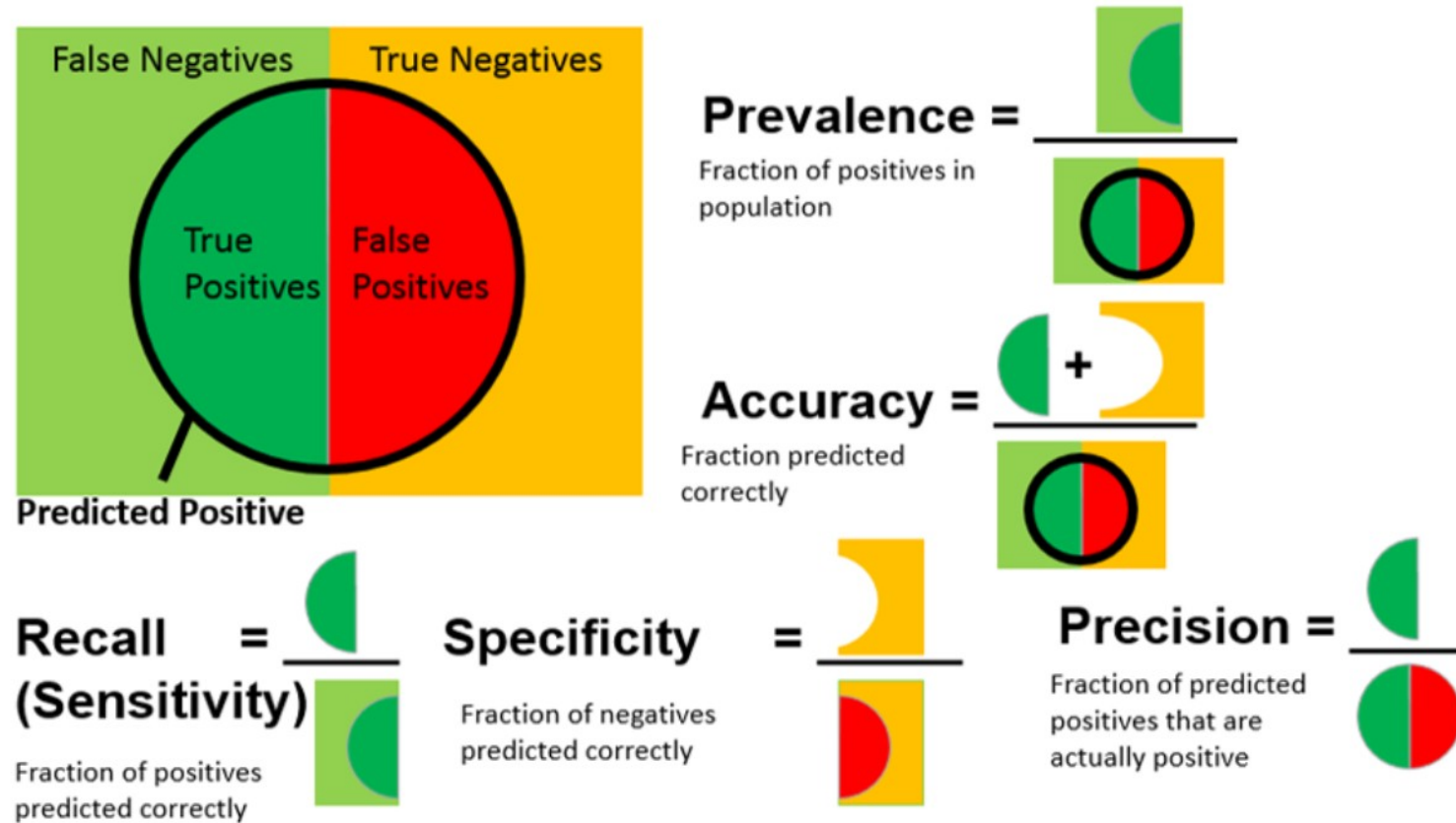
```
def create_callbacks(model, data):  
    """At the end of each epoch, determine various callback statistics (e.g. ROC-AUC)  
  
    :param model: Keras model  
    :type model: :class:`tensorflow.keras.Model`  
    :param data: Validation data - data sequences (codes, visits, numeric values) and classifier.  
    :type data: tuple( list( :class:`ndarray` ), :class:`ndarray` )  
    :param ARGS: Arguments object containing user-specified parameters  
    :type ARGS: :class:`argparse.Namespace`  
    :return: various callback objects - naming convention for saved HDF5 files, custom logging class, \  
    reduced learning rate  
    :rtype: tuple(:class:`tensorflow.keras.callbacks.ModelCheckpoint`, :class:`LogEval`, \  
    :class:`tensorflow.keras.callbacks.ReduceLROnPlateau` )  
    """
```

- The `create_callbacks` function is responsible for creating various callback objects that are used during the training of a Keras model. Callbacks are functions that can be applied at different stages during training to perform tasks such as saving model checkpoints, logging training progress, and adjusting the learning rate.
- The `LogEval` callback focuses on logging metrics such as ROC-AUC and PR-AUC, which are common evaluation metrics for binary classification tasks.

# Training Results

- ROC-AUC (Receiver Operating Characteristic - Area Under the Curve): 0.808748
  - ROC-AUC measures the model's ability to distinguish between positive and negative samples. A higher value indicates better discrimination.
- PR-AUC (Precision-Recall - Area Under the Curve): 0.719806
  - PR-AUC assesses the trade-off between precision and recall. A higher value signifies better balance between precision and recall.
- Loss: 0.5691
  - The loss is a measure of how well the model's predictions match the true labels. Lower values indicate better model performance.
- Accuracy: 0.7058
  - Accuracy represents the proportion of correctly predicted samples. In this case, the model has achieved an accuracy of 70.58% on the training data.

# Evaluation



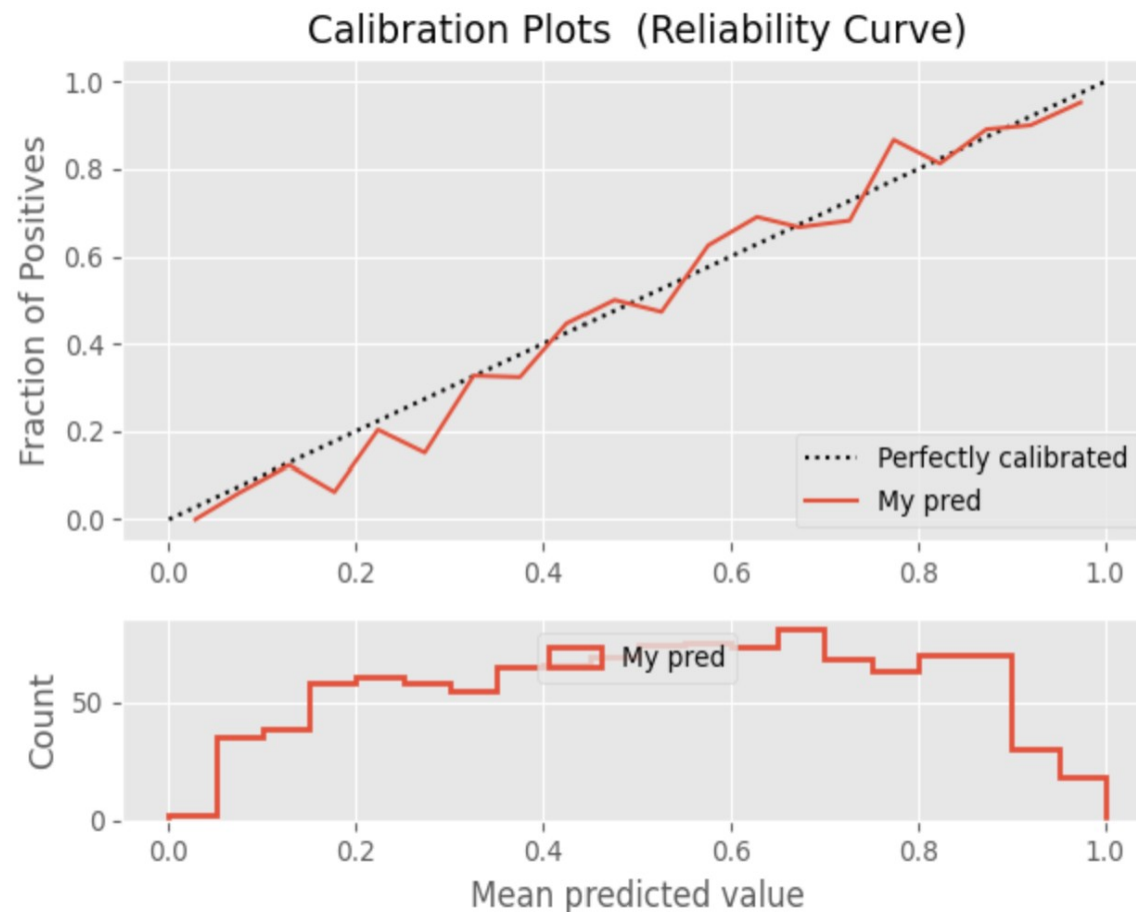
**Example: Positive = Hospitalized, Negative = Not Hospitalized**

# Precision Recall

- The `precision_recall` function is a valuable tool for assessing the performance of binary classification models, especially in situations involving imbalanced datasets or where the trade-off between precision and recall is crucial.
- It provides valuable insights into a model's ability to correctly classify positive instances while minimizing false positives, and the precision-recall curve helps visualize this trade-off.
- The purpose of this function is to compute and visualize precision-recall statistics, which can help assess the model's performance, especially in binary classification tasks.

# Test Results

- **Probability Calibration:** In many classification tasks, especially those involving binary classification, models not only make predictions but also provide probability scores. These probability scores represent the model's confidence in its predictions.
- However, these predicted probabilities are not always well-calibrated, meaning they may not accurately reflect the true likelihood of an event occurring. In other words, a model's probability score of 0.8 for "spam" may not mean that 80% of such emails are actually spam. Calibration is the process of ensuring that the predicted probabilities are in line with the actual event probabilities.
- Plot 2 provides insight into the distribution of predicted probabilities.
- In a well-calibrated model, you would typically expect the predicted probabilities to be spread evenly across the entire range of 0 to 1.



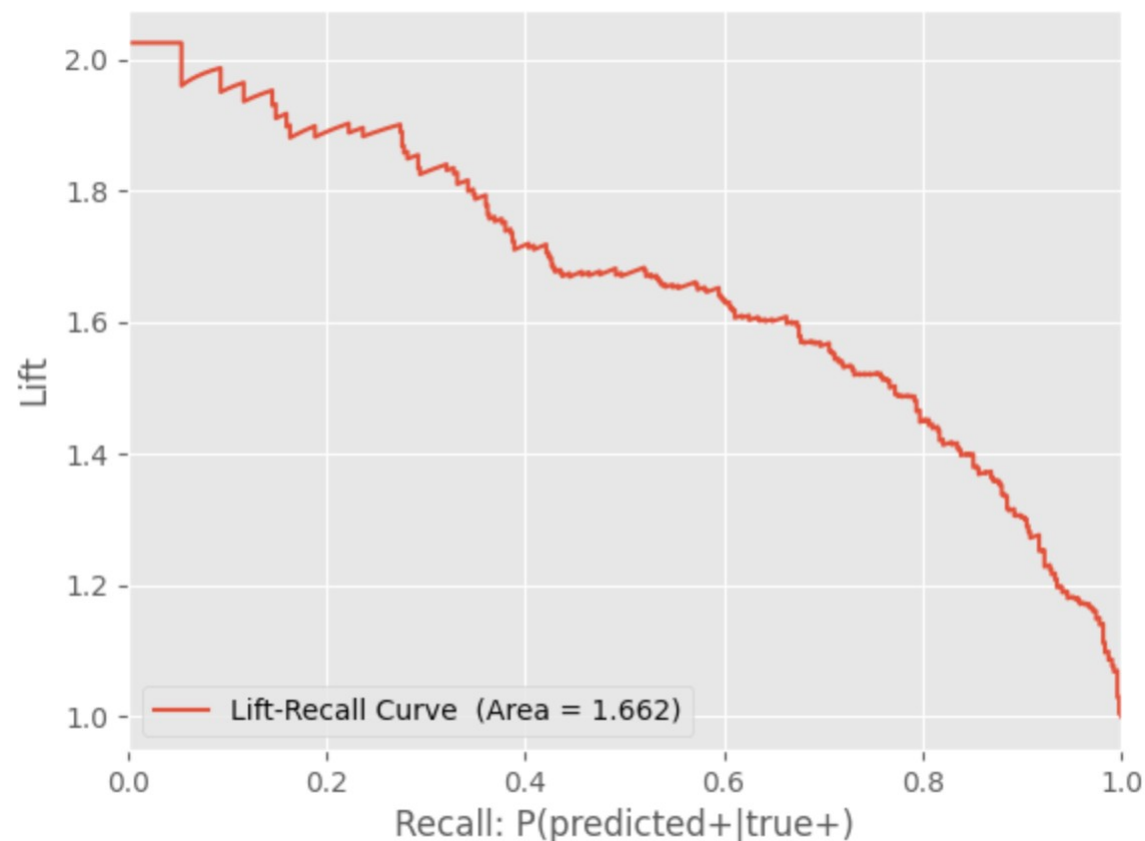
# Why Probability Calibration is Important:

- 1. Confidence Estimation:** Calibrated probabilities are useful for understanding how confident the model is in its predictions. This is important for applications where the reliability of predictions matters, such as medical diagnosis.
- 2. Threshold Selection:** Many classification tasks require setting a probability threshold for making decisions. For example, in a medical diagnosis model, a threshold might be set to determine when a patient is at high risk. Calibrated probabilities help in selecting appropriate thresholds.
- 3. Comparing Models:** Calibration makes it easier to compare the performance of different models. A well-calibrated model provides a more accurate indication of the likelihood of an event happening, making it easier to compare different models or algorithms.
- 4. Interpretability:** Calibrated probabilities are more interpretable. Users or stakeholders can better understand the confidence level of the model's predictions, which is essential for transparency and trust.
- 5. Graphical Visualization:** The probability calibration curve provides a visual representation of how well the model's predicted probabilities align with the true probabilities. This curve is useful for model evaluation and communication.



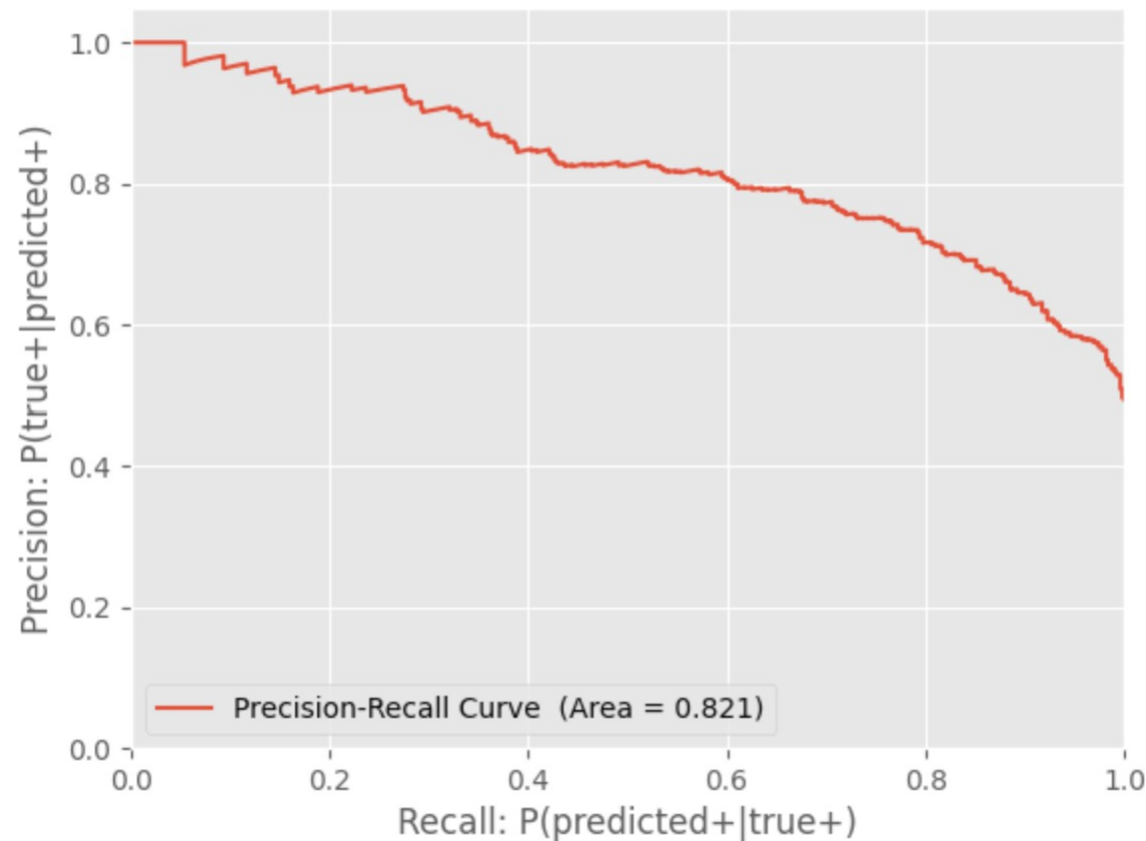
# Test Results

- Lift Recall Curve Area: 1.849
- **Lift Chart:** A lift chart is a graphical representation of the performance of a binary classification model. It helps assess the model's effectiveness in targeting a specific group, such as patients who are likely to encounter death.
- **Why Lift Chart is Important:** The lift chart provides insights into how much better the model is at identifying the target group compared to random selection. It is particularly valuable in scenarios where the cost of reaching out to a patient is high and you want to optimize the use of resources.



# Test Results

- Precision-Recall Curve Area: 0.704
- The `precision_recall` function is a valuable tool for assessing the performance of binary classification models, especially in situations involving imbalanced datasets or where the trade-off between precision and recall is crucial.
- It provides valuable insights into a model's ability to correctly classify positive instances while minimizing false positives, and the precision-recall curve helps visualize this trade-off.



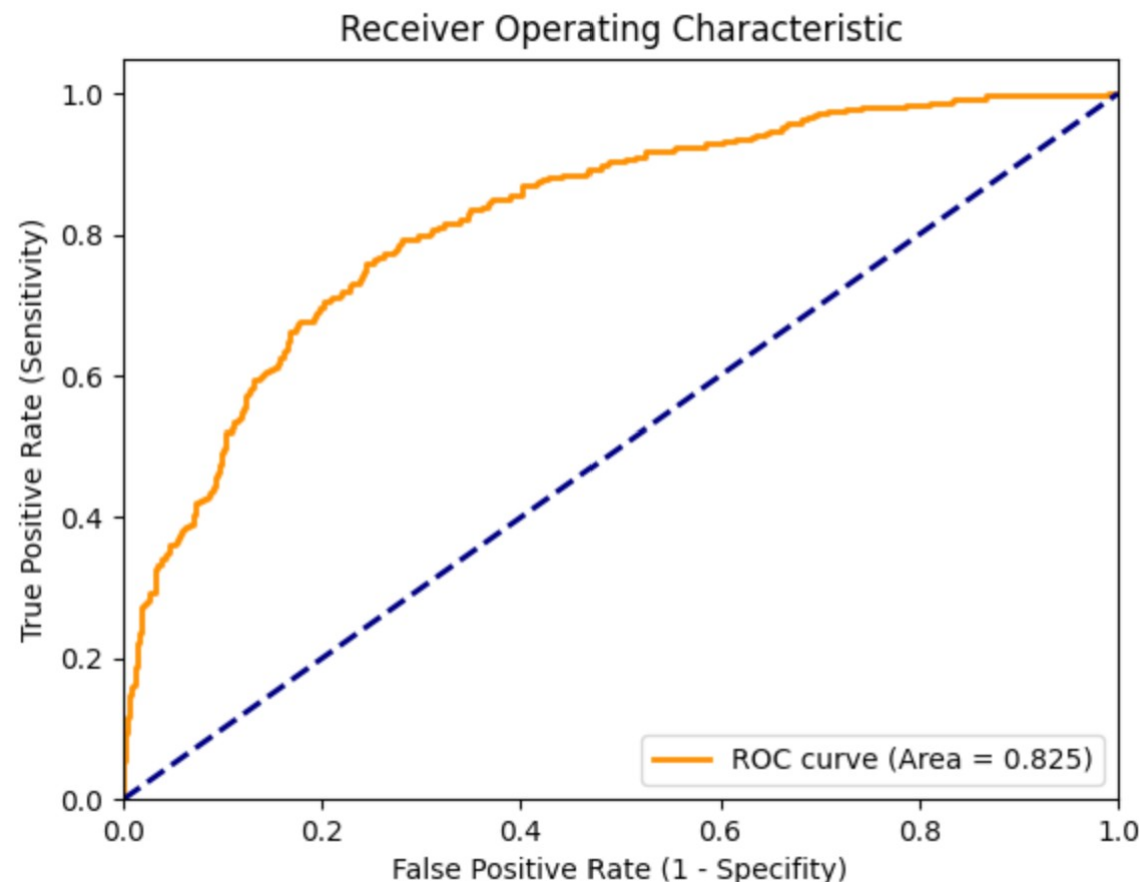
# Test Results

- **ROC Analysis:**

- ROC analysis is a technique used to evaluate the performance of a binary classification model.
- It is particularly valuable in situations where you want to assess a model's ability to distinguish between two classes (e.g., positive and negative cases) by varying the model's threshold for classification.

- **Why ROC Analysis is Important:**

- ROC analysis provides a comprehensive evaluation of a model's performance across various classification thresholds.
- It helps you understand the trade-off between true positive rate (sensitivity) and false positive rate (1 - specificity) at different threshold settings.
- ROC analysis is especially useful when dealing with imbalanced datasets or when the cost of false positives and false negatives differs.



# Interpretation

- The overall structure of the output looks like this:
- Predicts the probability of death
- The patient is diagnosed with a particular ICD code (feature) and this feature has influenced the above prediction by the given value in feature\_importance column.
- The patient was admitted some number of times given by to\_events and each of these visits importance in prediction is given by importance\_visit column.

# Inferencing on one patient (order: 1)

Input Patient Order Number. Type -1 to exit: 1

Patients probability: 0.8507523536682129

Output predictions? (y/n): y

	status	feature	importance_feature	importance_visit	to_event
2	Diagnosed	D_585.4	0.000032	0.000116	0
10	Diagnosed	D_428.0	0.000031	0.000116	0
14	Diagnosed	D_V100.5	0.000027	0.000116	0
1	Diagnosed	D_584.9	0.000024	0.000116	0
0	Diagnosed	D_403.00	0.000019	0.000116	0
9	Diagnosed	D_492.8	0.000013	0.000116	0
7	Diagnosed	D_599.7	0.000012	0.000116	0
13	Diagnosed	D_287.5	0.000007	0.000116	0
11	Diagnosed	D_600.01	0.000007	0.000116	0
3	Diagnosed	D_428.22	0.000006	0.000116	0
8	Diagnosed	D_E879.6	0.000002	0.000116	0
12	Diagnosed	D_788.20	-0.000007	0.000116	0
4	Diagnosed	D_285.1	-0.000007	0.000116	0
6	Diagnosed	D_285.21	-0.000008	0.000116	0
5	Diagnosed	D_276.7	-0.000011	0.000116	0
	status	feature	importance_feature	importance_visit	to_event
4	Diagnosed	D_276.2	0.312429	0.999884	1
2	Diagnosed	D_403.91	0.287294	0.999884	1
12	Diagnosed	D_600.00	0.280875	0.999884	1
6	Diagnosed	D_428.0	0.274460	0.999884	1
1	Diagnosed	D_585.6	0.228439	0.999884	1
3	Diagnosed	D_428.20	0.174211	0.999884	1
0	Diagnosed	D_574.20	0.075986	0.999884	1
11	Diagnosed	D_287.5	0.066994	0.999884	1
5	Diagnosed	D_428.22	0.054070	0.999884	1
8	Diagnosed	D_414.01	0.000774	0.999884	1
10	Diagnosed	D_285.21	-0.073977	0.999884	1
9	Diagnosed	D_276.7	-0.103246	0.999884	1
7	Diagnosed	D_V451.1	-0.120398	0.999884	1

# Inferencing on one patient (order: 1)

- Patient Probability: The probability of the patient being diagnosed with the condition (in this case, death) is approximately 0.85.
- Output Predictions: The following table presents the predictions and importance of features for this patient.
- The "status" column indicates that the patient has been diagnosed with the condition.
- The "feature" column lists specific encoded medical codes or diagnoses associated with the patient.
- "Importance\_feature" shows the importance of each feature in making the diagnosis prediction. Positive values suggest that the feature contributes positively to the prediction, while negative values have a negative impact.
- "Importance\_visit" indicates the importance of each visit in the patient's medical history. High values suggest that the visit has had a significant influence on the prediction.
- "to\_event" indicates the timing of the prediction relative to the patient's visits. "0" signifies that the prediction is made during first visit, while "1" suggests that the prediction was made on second visit and so on.
- In this case, it appears that the patient has been dead or alive. Some features and visits had a positive influence on the diagnosis, while others had a negative impact.
- The output includes feature-level and visit-level importance scores, highlighting the contribution of each element to the prediction.

Input Patient Order Number. Type -1 to exit: 1

Patients probability: 0.3255636692047119

Output predictions? (y/n): y

	status	feature	importance_feature	importance_visit	to_event
3	Diagnosed	D_427.31	0.000086	0.000258	0
1	Diagnosed	D_584.9	0.000055	0.000258	0
2	Diagnosed	D_486	0.000038	0.000258	0
0	Diagnosed	D_426.13	0.000030	0.000258	0
4	Diagnosed	D_427.32	0.000021	0.000258	0
6	Diagnosed	D_285.9	-0.000044	0.000258	0
5	Diagnosed	D_401.9	-0.000063	0.000258	0
	status	feature	importance_feature	importance_visit	to_event
0	Diagnosed	D_518.81	0.523924	0.999742	1
6	Diagnosed	D_286.9	0.324986	0.999742	1
7	Diagnosed	D_578.9	0.319670	0.999742	1
5	Diagnosed	D_512.8	0.282585	0.999742	1
1	Diagnosed	D_584.9	0.218181	0.999742	1
11	Diagnosed	D_V450.1	0.167004	0.999742	1
3	Diagnosed	D_428.0	0.150649	0.999742	1
4	Diagnosed	D_486	0.149106	0.999742	1
9	Diagnosed	D_585.9	0.087548	0.999742	1
2	Diagnosed	D_428.33	0.027505	0.999742	1
8	Diagnosed	D_008.45	0.016077	0.999742	1
10	Diagnosed	D_403.90	0.012669	0.999742	1

<http://www.icd9data.com/>

### ► 2015 ICD-9-CM Diagnosis Code 518.81

#### Acute respiratory failure

Input Patient Order Number. Type -1 to exit: 18

Patients probability: 0.9214457273483276

Output predictions? (y/n): y

	status	feature	importance_feature	importance_visit	to_event
7	Diagnosed	D_276.2	5.698259e-04	0.002239	0
3	Diagnosed	D_584.9	4.211321e-04	0.002239	0
1	Diagnosed	D_428.30	3.109813e-04	0.002239	0
2	Diagnosed	D_428.0	2.979059e-04	0.002239	0
0	Diagnosed	D_427.1	2.716426e-04	0.002239	0
4	Diagnosed	D_585.9	1.700847e-04	0.002239	0
14	Diagnosed	D_414.00	1.648188e-04	0.002239	0
19	Diagnosed	D_008.8	1.467526e-04	0.002239	0
17	Diagnosed	D_790.5	9.905243e-05	0.002239	0
12	Diagnosed	D_250.00	4.658242e-05	0.002239	0
11	Diagnosed	D_403.90	2.995698e-05	0.002239	0
16	Diagnosed	D_412	3.140708e-06	0.002239	0
9	Diagnosed	D_442.3	1.156144e-07	0.002239	0
5	Diagnosed	D_997.2	-6.719612e-05	0.002239	0
15	Diagnosed	D_V458.1	-8.319203e-05	0.002239	0
10	Diagnosed	D_E879.8	-8.669589e-05	0.002239	0
18	Diagnosed	D_285.21	-1.383094e-04	0.002239	0
8	Diagnosed	D_276.50	-2.285754e-04	0.002239	0
13	Diagnosed	D_V586.9	-3.701081e-04	0.002239	0
6	Diagnosed	D_411.1	-5.302868e-04	0.002239	0
	status	feature	importance_feature	importance_visit	to_event
1	Diagnosed	D_518.81	0.504702	0.997761	1
4	Diagnosed	D_585.6	0.210846	0.997761	1
2	Diagnosed	D_403.91	0.200628	0.997761	1
6	Diagnosed	D_785.59	0.169074	0.997761	1
7	Diagnosed	D_428.0	0.146782	0.997761	1
5	Diagnosed	D_427.32	0.080119	0.997761	1
0	Diagnosed	D_996.73	0.074062	0.997761	1
11	Diagnosed	D_250.00	0.023275	0.997761	1
10	Diagnosed	D_412	0.002770	0.997761	1
8	Diagnosed	D_724.00	-0.008493	0.997761	1
9	Diagnosed	D_366.8	-0.022162	0.997761	1
13	Diagnosed	D_V458.1	-0.040618	0.997761	1
15	Diagnosed	D_V457.9	-0.058622	0.997761	1
3	Diagnosed	D_285.1	-0.071716	0.997761	1
14	Diagnosed	D_V451.1	-0.097173	0.997761	1
12	Diagnosed	D_433.10	-0.136925	0.997761	1
16	Diagnosed	D_V586.7	-0.190284	0.997761	1
17	Diagnosed	D_E878.1	-0.319049	0.997761	1

Input Patient Order Number. Type -1 to exit: 22

Patients probability: 0.12459183484315872

Output predictions? (y/n): y

	status	feature	importance_feature	importance_visit	to_event
0	Diagnosed	D_571.5	0.000513	0.002515	0
3	Diagnosed	D_042	0.000254	0.002515	0
1	Diagnosed	D_456.20	0.000238	0.002515	0
8	Diagnosed	D_784.7	0.000068	0.002515	0
6	Diagnosed	D_250.00	0.000052	0.002515	0
4	Diagnosed	D_287.5	-0.000023	0.002515	0
2	Diagnosed	D_070.54	-0.000079	0.002515	0
5	Diagnosed	D_305.50	-0.000381	0.002515	0
7	Diagnosed	D_311	-0.000717	0.002515	0
	status	feature	importance_feature	importance_visit	to_event
0	Diagnosed	D_571.5	0.218271	0.997485	1
3	Diagnosed	D_486	0.145505	0.997485	1
2	Diagnosed	D_456.20	0.101105	0.997485	1
7	Diagnosed	D_250.00	0.022623	0.997485	1
4	Diagnosed	D_280.0	0.008545	0.997485	1
1	Diagnosed	D_070.51	-0.085879	0.997485	1
5	Diagnosed	D_305.00	-0.163714	0.997485	1
6	Diagnosed	D_V08	-0.191187	0.997485	1

Input Patient Order Number. Type -1 to exit: 0

Patients probability: 0.5147534608840942

Output predictions? (y/n): y

	status	feature	importance_feature	importance_visit	to_event
6	Diagnosed	D_427.31	0.000133	0.000389	0
4	Diagnosed	D_428.0	0.000060	0.000389	0
0	Diagnosed	D_427.1	0.000055	0.000389	0
9	Diagnosed	D_585.9	0.000035	0.000389	0
5	Diagnosed	D_440.20	0.000025	0.000389	0
7	Diagnosed	D_250.00	0.000009	0.000389	0
8	Diagnosed	D_403.90	0.000006	0.000389	0
1	Diagnosed	D_428.22	-0.000007	0.000389	0
2	Diagnosed	D_997.2	-0.000014	0.000389	0
3	Diagnosed	D_426.0	-0.000027	0.000389	0
	status	feature	importance_feature	importance_visit	to_event
6	Diagnosed	D_427.31	0.354912	0.999611	1
4	Diagnosed	D_428.0	0.159927	0.999611	1
7	Diagnosed	D_414.8	0.158018	0.999611	1
0	Diagnosed	D_427.1	0.147501	0.999611	1
8	Diagnosed	D_V458.2	0.139835	0.999611	1
14	Diagnosed	D_440.20	0.066481	0.999611	1
3	Diagnosed	D_250.00	0.025346	0.999611	1
11	Diagnosed	D_403.90	0.014147	0.999611	1
1	Diagnosed	D_428.23	0.012669	0.999611	1
13	Diagnosed	D_V174.1	0.006422	0.999611	1
12	Diagnosed	D_412	0.003532	0.999611	1
5	Diagnosed	D_V533.9	0.003315	0.999611	1
2	Diagnosed	D_585.3	-0.107427	0.999611	1
10	Diagnosed	D_530.81	-0.118286	0.999611	1
9	Diagnosed	D_272.4	-0.275125	0.999611	1