Project: 1

Prediction of mortality of ICU patients based on random forest algorithm

Background

Because the basic causes of critical illness are different, critical patients are scattered to different medical specialties, which makes the lack of unity for critical illness. ICU (Intensive Care Unit) is a clinical base of a professional team specialized in critical care medicine and specialized in critical care treatment. It is a centralized management unit for critical patients and high-risk patients after surgery from various clinical departments. The birth of ICU has directly promoted the development of critical care medicine. It pays more attention to the characteristics of patients in critical care and the common threats and damages they face, so that many patients who previously thought that they could not be cured can survive or extend their survival time.

Develop the population mortality prediction in the intensive care unit (ICU) and the in-hospital mortality prediction method for specific patients. The information collected at the time of admission to the ICU will be used to predict which patients can survive during the hospitalization and which patients cannot.

Technology

Programming language: python

Import: pandas, imbearn, sklearn, pyecharts, matplotlib, math, numpy

Analysis: sequence classification

Introduction of Algorithm

In machine learning, random forest is a classifier containing multiple decision trees, and its output category is determined by the mode of the category output by individual trees.

Leo Breiman and Adele Cutler developed algorithms to deduce random forests. "Random Forests" is their trademark. This term is derived from the random decision forests proposed by Tin Kam Ho of Bell Laboratories in 1995. This method combines Breimans' "Bootstrap aggregating" idea and Ho's "random subspace method" to build a set of decision trees.

The advantages of random forest include:

1) For many kinds of data, it can produce high accuracy classifiers.

2) It can handle a large number of input variables;

3) It can evaluate the importance of variables when determining categories.

4) When building a forest, it can produce an unbiased estimate of the generalized error internally.

5) It contains a good method to estimate the lost data, and if a large part of the data is lost, the accuracy can still be maintained.

6) It provides an experimental method to detect variable interactions.

7) For unbalanced classification data sets, it can balance errors.

8) It calculates the proximity in each example, which is very useful for data mining, outlier detection and data visualization.

9) Use the above. It can be extended to unmarked data, which usually uses unsupervised clustering. It can also detect deviators and view data.

10) The learning process is very fast

11) Significant effect on high latitude data

Data source

https://www.physionet.org/content/challenge-2012/1.0.0/

Project process

1: Data acquisition:

The original data is a .txt file. Each file is used for the various data collected by a patient since entering the hospital.

132539.txt
132540.txt
132541.txt
132543.txt
132545.txt
132547.txt
132548.txt
132551.txt
132554.txt
132555.txt
132556.txt
132567.txt

```
Time,Parameter,Value
00:00,RecordID,132539
00:00,Age,54
00:00,Gender,0
00:00,Height,-1
00:00,ICUType,4
00:00,Weight,-1
00:07,GCS,15
00:07,HR,73
00:07,NIDiasABP,65
00:07,NIMAP,92.33
00:07,NISysABP,147
00:07,RespRate,19
00:07,Temp,35.1
00:07,Urine,900
00:37,HR,77
00:37,NIDiasABP,58
00:37,NIMAP,91
00:37,NISysABP,157
00:37,RespRate,19
```

- *RecordID* (defined as above)
- *SAPS-I score* (Le Gall et al., 1984)
- *SOFA score* (Ferreira et al., 2001)
- *Length of stay* (days)
- *Survival* (days)
- *In-hospital death* (0: survivor, or 1: died in-hospital)

2: Preliminary arrangement of data:

Read the .txt file through the open function loop and format the data after slicing, segmentation and statistics to form a set_ a_ data.csv、set_ b_ data.csv、set_ c_ Data.csv

Three tables with the same column name, 12000 data in total.

This data is sequence data, and features are extracted based on six basic statistical methods: mean value, variance, extreme value (maximum, minimum), initial value and current value,

Feature extraction: For RecordID, Age, Gender, Height, ICUType and Weight, these six fields are read directly,

The remaining 37 fields:

- *Albumin* (g/dL)
- *ALP* [Alkaline phosphatase (IU/L)]
- *ALT* [Alanine transaminase (IU/L)]
- *AST* [Aspartate transaminase (IU/L)]
- *Bilirubin* (mg/dL)
- *BUN* [Blood urea nitrogen (mg/dL)]
- *Cholesterol* (mg/dL)
- *Creatinine* [Serum creatinine (mg/dL)]
- *DiasABP* [Invasive diastolic arterial blood pressure (mmHg)]
- *FiO2* [Fractional inspired O_2 (0-1)]
- *GCS* [Glasgow Coma Score (3-15)]
- *Glucose* [Serum glucose (mg/dL)]
- *HCO3* [Serum bicarbonate (mmol/L)]
- *HCT* [Hematocrit (%)]
- *HR* [Heart rate (bpm)]
- *K* [Serum potassium (mEq/L)]
- *Lactate* (mmol/L)
- *Mg* [Serum magnesium (mmol/L)]
- *MAP* [Invasive mean arterial blood pressure (mmHg)]
- *MechVent* [Mechanical ventilation respiration (0:false, or 1:true)]
- *Na* [Serum sodium (mEq/L)]
- *NIDiasABP* [Non-invasive diastolic arterial blood pressure (mmHg)]
- *NIMAP* [Non-invasive mean arterial blood pressure (mmHg)]
- *NISysABP* [Non-invasive systolic arterial blood pressure (mmHg)]
- *PaCO2* [partial pressure of arterial CO_2 (mmHg)]
- *PaO2* [Partial pressure of arterial O_2 (mmHg)]
- *pH* [Arterial pH (0-14)]
- *Platelets* (cells/nL)
- *RespRate* [Respiration rate (bpm)]
- *SaO2* [O_2 saturation in hemoglobin (%)]
- *SysABP* [Invasive systolic arterial blood pressure (mmHg)]
- *Temp* [Temperature (°C)]
- *TropI* [Troponin-I (μg/L)]
- *TropT* [Troponin-T (μg/L)]
- *Urine* [Urine output (mL)]
- *WBC* [White blood cell count (cells/nL)]
- *Weight* (kg)*

In addition to the weight, each field is subdivided into 6 features, and the total number of features is 222.

(The data processing and feature extraction code is in the open_file.py file).

set_a_data.csv 使用Microsoft Excel打开

RecordID	lab	Age	Gender	Height	ICUType	Weight	GCS_jun	GCS_fang	GCS_da	GCS_xiao	GCS_chu	GCS_dang	HR_jun	HR_fang	HR_da	HR_xiao	HR_chu	HR_dang	NIDiasABP_jun	NIDiasABP_fang	NIDiasABP_da	NIDiasABP_xiao	NIDiasABP_chu	NIDiasABP_dang	NIMAP_jun	NI
132539.0	0	54.0	0.0	-1.0	4.0	-1.0	14.923076923076923	0.0710059171597633	15.0	14.0	15.0	15.0	70.8108108108108	72.04528853177501	86.0	58.0	73.0	86.0	50.14705882352941	61.77249134948096	67.0	39.0	65.0	55.0	71.55911764705883	83
132540.0	0	76.0	1.0	175.3	2.0	76.0	13.333333333333334	9.955555555555556	15.0	3.0	3.0	15.0	80.79411764705883	44.75173010380623	90.0	65.0	88.0	65.0	56.714285714285715	91.34693877551021	67.0	38.0	38.0	49.0	75.30857142857143	11
132541.0	0	44.0	0.0	-1.0	3.0	56.7	5.923076923076923	1.3017751479289943	8.0	5.0	7.0	5.0	83.75925925925925	130.6272290809328	113.0	57.0	100.0	71.0	79.0	51.3684210526315 8	95.0	66.0	84.0	84.0	96.7513157894737	53
132543.0	0	68.0	1.0	180.3	3.0	84.6	14.944444444444445	0.05246913580246914	15.0	14.0	15.0	15.0	70.98333333333333	58.88305555555555	88.0	57.0	79.0	79.0	65.05172413793103	44.393876337693214	81.0	45.0	63.0	81.0	83.88551724137932	38
132545.0	0	88.0	0.0	-1.0	3.0	-1.0	15.0	0.0	15.0	15.0	15.0	15.0	74.95833333333333	54.41493055555555	94.0	65.0	93.0	68.0	45.72093023255814	152.8058409951325	96.0	26.0	41.0	42.0	74.9465116279069 7	85
132547.0	0	64.0	1.0	180.3	1.0	114.0	8.666666666666666	0.8888888888888887	11.0	7.0	7.0	8.0	88.53191489361703	63.10004526935264	101.0	71.0	78.0	92.0	70.5	342.25	89.0	52.0	89.0	52.0	81.985	41
132548.0	0	68.0	0.0	162.6	3.0	87.0	15.0	0.0	15.0	15.0	15.0	15.0	68.33898305084746	43.37661591496697	88.0	50.0	73.0	60.0	72.0	298.85714285714283	88.0	31.0	88.0	77.0	102.14714285714285	54
132551.0	1	78.0	0.0	162.6	3.0	48.4	11.846153846153847	8.899408284023668	15.0	8.0	15.0	9.0	70.94520547945206	152.02439482079188	111.0	55.0	111.0	58.0	30.697674418604652	135.32720389399677	56.0	14.0	51.0	19.0	55.177906976744175	11
132554.0	0	64.0	0.0	-1.0	3.0	60.7	15.0	0.0	15.0	15.0	15.0	15.0	127.23913043478261	22.52977315689981	137.0	115.0	127.0	122.0	64.47826086956522	83.33648393194707	92.0	47.0	71.0	53.0	84.47739130434783	77
132555.0	0	74.0	1.0	175.3	2.0	66.1	14.083333333333334	3.4097222222222228	15.0	10.0	10.0	15.0	85.1896551724138	64.0847205707491	99.0	67.0	67.0	78.0	53.0	16.0	57.0	49.0	57.0	49.0	75.67	49
132556.0	0	64.0	0.0	-1.0	3.0	65.0	15.0	0.0	15.0	15.0	15.0	15.0	110.5625	693.37109375	161.0	57.0	101.0	91.0	48.166666666666664	21.338888888888892	60.0	32.0	48.0	45.0	57.28813559322034	22
132567.0	0	71.0	0.0	157.5	2.0	56.0	14.181818181818182	3.057851239669422	15.0	10.0	10.0	15.0	95.22727272727273	53.02410468319559	112.0	82.0	84.0	95.0	43.43333333333333	38.845555555555556	57.0	32.0	45.0	48.0	64.66733333333335	46
132568.0	0	66.0	0.0	157.5	3.0	84.5	15.0	0.0	15.0	15.0	15.0	15.0	88.5	34.06818181818182	100.0	77.0	88.0	93.0	56.8974358974359	109.83563445101906	79.0	36.0	62.0	48.0	71.21621621621621	86
132570.0	0	84.0	1.0	170.2	1.0	102.6	14.6	0.24	15.0	14.0	15.0	15.0	70.60975609756098	143.0672218917311	106.0	55.0	70.0	73.0	48.675	125.06937500000001	66.0	0.0	61.0	47.0	72.38461538461539	10
132573.0	0	77.0	1.0	162.6	1.0	90.1	15.0	0.0	15.0	15.0	15.0	15.0	73.66071428571429	19.045599489795922	84.0	64.0	80.0	68.0	48.55357142857143	60.81855867346938	82.0	37.0	49.0	44.0	82.83821428571427	81
132575.0	0	78.0	1.0	167.6	2.0	63.0	10.857142857142858	27.55102040816328	15.0	3.0	3.0	15.0	84.49253731343283	146.24994430830918	119.0	65.0	73.0	106.0								
132577.0	0	65.0	1.0	-1.0	3.0	66.3	12.125	6.109375	15.0	9.0	9.0	15.0	84.47916666666667	25.666232638888886	96.0	71.0	94.0	88.0	74.9090909090909	201.53719008264463	98.0	48.0	62.0	65.0	99.65181818181819	24
132582.0	0	84.0	1.0	182.9	3.0	82.5	14.846153846153847	0.13017751479289943	15.0	14.0	15.0	15.0	94.07142857142857	48.53061224489795	108.0	77.0	101.0	83.0	43.870370370370374	48.594307270233195	60.0	24.0	57.0	51.0	66.07407407407408	46
132584.0	0	78.0	0.0	-1.0	3.0	72.8	9.916666666666666	4.743055555555556	11.0	3.0	3.0	11.0	84.125	290.359375	124.0	53.0	124.0	73.0	64.29411764705883	518.2076124567475	129.0	44.0	75.0	52.0	84.74235294117646	57
132585.0	0	40.0	0.0	165.1	2.0	84.7	13.785714285714286	10.596938775510207	15.0	3.0	3.0	15.0	80.45614035087719	67.09018159433671	101.0	66.0	79.0	92.0	41.57142857142857	38.81632653061225	50.0	30.0	38.0	50.0	57.285714285714285	22
132588.0	1	48.0	0.0	154.9	3.0	42.3	15.0	0.0	15.0	15.0	15.0	15.0	93.85	160.6775	115.0	76.0	115.0	78.0	50.083333333333336	77.85416666666669	68.0	31.0	47.0	46.0	64.14861111111112	62
132590.0	0	58.0	1.0	188.0	2.0	98.0	12.0	14.181818181818182	15.0	3.0	3.0	15.0	97.54098360655738	90.51061542596075	131.0	84.0	119.0	88.0	55.0	55.0	55.0	55.0	55.0	55.0	75.67	75
132591.0	0	81.0	1.0	-1.0	3.0	63.7	15.0	0.0	15.0	15.0	15.0	15.0	70.13725490196079	103.60861207227991	109.0	59.0	62.0	61.0	55.745098039215684	54.0330642060746	70.0	28.0	62.0	54.0	74.1435294117647	39
132592.0	0	35.0	0.0	-1.0	3.0	71.8	15.0	0.0	15.0	15.0	15.0	15.0	97.07142857142857	80.78061224489795	115.0	77.0	112.0	82.0	60.609756097560975	105.11600237953597	85.0	43.0	43.0	55.0	80.92560975609756	10
132595.0	0	26.0	0.0	-1.0	3.0	-1.0																				
132597.0	0	66.0	0.0	137.2	3.0	82.0	15.0	0.0	15.0	15.0	15.0	15.0	68.79245283018868	139.78711285154859	143.0	56.0	76.0	65.0	39.34615384615385	48.45710059171598	60.0	24.0	37.0	25.0	60.49307692307692	57
132598.0	1	80.0	0.0	-1.0	4.0	60.0	7.173913043478261	0.3175803402646504	8.0	6.0	6.0	8.0	78.70886075949367	216.96587085402984	158.0	49.0	51.0	72.0	66.7674418604651 1	247.66684694429424	115.0	43.0	69.0	61.0	89.03767441860465	28
132599.0	0	53.0	0.0	177.8	4.0	73.5	8.416666666666666	2.909722222222222	14.0	7.0	8.0	14.0	85.39285714285714	97.6670918367347	104.0	67.0	98.0	94.0	63.0	18.5	69.0	57.0	69.0	64.0	79.5	18
132601.0	0	74.0	1.0	177.8	2.0	75.9	12.538461538461538	18.55621301775148	15.0	3.0	3.0	15.0	99.24193548387096	37.957596253902175	109.0	87.0	103.0	95.0	66.0	66.0	66.0	66.0	66.0	66.0	93.0	93
132602.0	1	80.0	1.0	180.3	3.0	70.0	15.0	0.0	15.0	15.0	15.0	15.0	74.45283018867924	288.3987184051263	144.0	59.0	67.0	85.0	66.38	60.915600000000005	82.0	47.0	72.0	61.0	80.79166666666667	12
132605.0	1	90.0	0.0	-1.0	3.0	55.0	7.416666666666667	0.2430555555555555	8.0	7.0	8.0	7.0	72.12280701754386	65.96737457679285	98.0	58.0	72.0	98.0	36.57142857142857	174.35204081632654	75.0	22.0	66.0	72.0	64.095	24
132610.0	0	72.0	1.0	172.9	3.0	72.26	14.9	0.09	15.0	14.0	15.0	15.0	74.11627906976744	12.753921038399135	84.0	70.0	75.0	78.0	39.0	12.666666666666666	42.0	34.0	42.0	34.0	57.333333333333336	1.4
132612.0	0	52.0	1.0	-1.0	4.0	109.0	9.222222222222221	4.950617283950617	11.0	3.0	11.0	9.0	74.61111111111111	257.42283950617286	126.0	57.0	114.0	73.0	53.0	74.0	64.0	43.0	43.0	52.0	73.55666666666667	11
132614.0	0	77.0	1.0	162.6	1.0	59.0	15.0	0.0	15.0	15.0	15.0	15.0	70.78947368421052	1.1662049861495845	74.0	70.0	73.0	70.0	39.42857142857143	29.102040816326532	49.0	32.0	41.0	36.0	57.42714285714285	29
132615.0	0	46.0	0.0	152.4	3.0	88.6	8.11111111111111	10.76543209876543	11.0	3.0	3.0	11.0	70.05797101449275	101.38794370930475	107.0	55.0	80.0	55.0	43.13333333333333	69.44888888888887	61.0	30.0	61.0	36.0	63.53200000000001	88
132617.0	1	77.0	1.0	170.2	1.0	75.0	15.0	0.0	15.0	15.0	15.0	15.0	82.74285714285715	83.90530612244898	100.0	66.0	66.0	79.0	40.0	46.56410256410256	61.0	19.0	56.0	37.0	58.59846153846154	53
132618.0	0	72.0	0.0	152.4	4.0	69.1	11.714285714285714	3.727891156462585	15.0	10.0	14.0	11.0	58.28	35.041599999999995	74.0	49.0	56.0	64.0	47.10344827586207	29.12722948870392	57.0	36.0	36.0	43.0	63.64344827586207	32
132622.0	1	71.0	0.0	160.0	3.0	79.0	14.454545454545455	0.42975206611570244	15.0	13.0	15.0	15.0	77.92156862745098	174.22914263744715	104.0	58.0	95.0	80.0	55.46	125.4484	85.0	30.0	68.0	50.0	69.4468	11
132623.0	0	24.0	1.0	182.9	2.0	78.0	13.222222222222221	14.617283950617283	15.0	3.0	3.0	15.0	110.0952380952381	27.752834467120177	121.0	96.0	101.0	96.0	63.0	39.68421052631579	76.0	49.0	68.0	62.0	81.77263157894737	30
132632.0	0	49.0	0.0	-1.0	3.0	162.2	13.384615384615385	5.3136094674556205	15.0	8.0	13.0	15.0	72.16981132075472	22.10323958704165	81.0	62.0	73.0	75.0	58.625	199.15104166666666	82.0	16.0	82.0	60.0	72.56521739130434	10
132634.0	0	82.0	0.0	-1.0	4.0	-1.0	14.909090909090908	0.08264462809917357	15.0	14.0	15.0	15.0	104.31818181818181	51.171487603305785	123.0	84.0	116.0	88.0	46.395348837209305	42.192536506219575	67.0	33.0	33.0	50.0	62.69767441860465	65
132635.0	0	66.0	0.0	167.6	2.0	70.1	14.583333333333334	1.9097222222222223	15.0	10.0	10.0	15.0	81.0126582278481	48.189713186989266	98.0	69.0	84.0	74.0	44.0	0.0	44.0	44.0	44.0	44.0	61.67	0.
132636.0	0	42.0	0.0	-1.0	4.0	114.2	8.666666666666666	2.9999999999999996	11.0	3.0	3.0	10.0	67.51063829787235	28.334993209597105	84.0	57.0	58.0	71.0	46.75609756097561	34.08685306365258	60.0	32.0	32.0	41.0	69.60219512195123	43
132637.0	0	78.0	0.0	170.2	2.0	56.0	12.818181818181818	21.421487603305785	15.0	3.0	3.0	15.0	86.66666666666667	110.76190476190474	114.0	46.0	95.0	80.0								
132639.0	0	73.0	1.0	180.3	2.0	96.3	12.166666666666666	10.305555555555555	15.0	5.0	5.0	13.0	75.64864864864865	107.25493060628195	113.0	63.0	80.0	85.0	38.6	83.03999999999999	56.0	29.0	45.0	56.0	64.5	93
132642.0	0	89.0	0.0	-1.0	3.0	-1.0	14.333333333333334	0.3888888888888889	15.0	13.0	14.0	14.0	120.70491803278688	237.9457135178715	163.0	97.0	131.0	124.0	75.8	126.66000000000001	117.0	54.0	59.0	68.0	90.66101694915254	11
132644.0	0	71.0	0.0	149.9	4.0	64.5	14.454545454545455	2.066115702479338	15.0	10.0	15.0	15.0	105.35135135135135	58.87655222790358	134.0	93.0	112.0	105.0								
132647.0	0	27.0	0.0	-1.0	3.0	-1.0																				
132648.0	0	87.0	0.0	157.5	2.0	66.0	8.823529411764707	0.1453287197231834	9.0	8.0	8.0	9.0	78.63333333333334	37.83222222222222	95.0	65.0	65.0	71.0								
132650.0	0	83.0	0.0	-1.0	3.0	56.0	10.454545454545455	15.884297520661153	15.0	3.0	15.0	11.0	68.87931034482759	511.3475029726516	122.0	46.0	122.0	56.0	50.707317073170735	162.0118976799524	91.0	33.0	86.0	39.0	62.41463414634146	14
132653.0	0	78.0	1.0	177.8	2.0	91.3	10.5	26.392857142857142	15.0	3.0	3.0	15.0	88.38888888888889	56.48765432098766	105.0	74.0	92.0	84.0	59.333333333333336	27.222222222222225	66.0	52.0	63.0	55.0	74.83333333333333	20
132658.0	0	81.0	1.0	185.4	2.0	105.4	9.461538461538462	7.786982248520711	11.0	3.0	3.0	11.0	88.47945205479452	8.523550384687562	90.0	75.0	82.0	90.0	56.75	55.6875	64.0	47.0	64.0	47.0	74.5	72

Missing value processing: remove the feature data with missing ratio greater than 20%, and finally get 109 features, as shown below:

```
data1 = pd.read_csv('set_a_data.csv')
data2 = pd.read_csv('set_b_data.csv')
data3 = pd.read_csv('set_b_data.csv')
data = pd.concat([data1, data2,data3])
data_list = data.dropna(thresh=9600,axis=1)
print(data_list.isnull().sum())
print(data_list.head())
print(data_list)
```

```
RecordID      0
lab           0
Age           0
Gender        0
Height        0
           ...
WBC_fang    205
WBC_da      205
WBC_xiao    205
WBC_chu     205
WBC_dang    205
Length: 109, dtype: int64
```

```
Length: 109, dtype: int64
   RecordID  lab   Age  Gender  ...  WBC_da  WBC_xiao  WBC_chu  WBC_dang
0  132539.0    0  54.0     0.0  ...    11.2       9.4     11.2       9.4
1  132540.0    0  76.0     1.0  ...    13.3       7.4      7.4      13.3
2  132541.0    0  44.0     0.0  ...     6.2       3.7      4.2       6.2
3  132543.0    0  68.0     1.0  ...    11.5       7.9     11.5       7.9
4  132545.0    0  88.0     0.0  ...     4.8       3.8      3.8       4.8

[5 rows x 109 columns]
```

```
      RecordID  lab   Age  Gender  ...  WBC_da  WBC_xiao  WBC_chu  WBC_dang
0     132539.0    0  54.0     0.0  ...    11.2       9.4     11.2       9.4
1     132540.0    0  76.0     1.0  ...    13.3       7.4      7.4      13.3
2     132541.0    0  44.0     0.0  ...     6.2       3.7      4.2       6.2
3     132543.0    0  68.0     1.0  ...    11.5       7.9     11.5       7.9
4     132545.0    0  88.0     0.0  ...     4.8       3.8      3.8       4.8
...        ...  ...   ...     ...  ...     ...       ...      ...       ...
3995  152849.0    0  78.0     1.0  ...    20.0      14.5     20.0      15.8
3996  152851.0    0  90.0     1.0  ...    41.8      18.0     27.4      21.4
3997  152858.0    0  70.0     0.0  ...    15.1      13.1     14.8      15.1
3998  152862.0    0  49.0     0.0  ...    16.6      13.6     13.6      16.6
3999  152864.0    0  82.0     0.0  ...    17.3      10.3     17.3      10.3

[12000 rows x 109 columns]
```

Missing value padding:

```
for i in data_list.isnull().any().items():
        name = str(i[0])
        val    = str(i[1])
        if 'True'==val:
              data_list[name] = data_list[name].fillna(data_list[name].median())
    print(data_list.isnull().any())
```

Fill in the missing values of each column in the way of median, and the results show.

```
RecordID    False
lab         False
Age         False
Gender      False
Height      False
            ...
WBC_fang    False
WBC_da      False
WBC_xiao    False
WBC_chu     False
WBC_dang    False
Length: 109, dtype: bool
```

3. Analysis of correlation coefficient for each feature:

```
def xiangguanxishu(X, Y):
    '''计算相关系数'''
    XY = X * Y
    X2 = X ** 2
    Y2 = Y ** 2
    n = len(XY)
    numerator = n * XY.sum() - X.sum() * Y.sum()   # 分子
    denominator = math.sqrt(n * X2.sum() - X.sum() ** 2) * math.sqrt(n * Y2.sum() - Y.sum() ** 2)   # 分母
    if denominator == 0:
        return 'NaN'
    rhoXY = numerator / denominator
    return rhoXY
```

Print results:

```
['Age', 0.13048980922783202]
['Gender', -0.015418852639407147]
['Height', -0.028738795253975824]
['ICUType', 0.0702577782808679]
……………………………………………………
['WBC_xiao', 0.05710239848080716]
['WBC_chu', 0.03335311353058349]
['WBC_dang', 0.09195582784538718]
```

4. Data processing:

According to statistics, in the data sample, the tag value survivors (lab=0) account for about 85%, and the death (lab=1) account for about 15%. According to the binary classification algorithm, it belongs to the unbalanced data, and the final prediction result will deviate greatly.

Solution: Reduce sampling method。

```
cc = ClusterCentroids()
X_resampled, y_resampled = cc.fit_resample(data_txt, data_lab.values.reshape([-1]))
data_train_X, data_train_Y = X_resampled, y_resampled
```

The final data is divided into training set (iris_x_train, iris_y_train), test set (iris_x_test, iris_y_test) and verification set (iris_x_yan, iris_y_yan) according to the two-eighth rule. The following is the division of data code block and display.

```
x_train_all, iris_x_test, y_train_all, iris_y_test = train_test_split(data_train_X, data_train_Y, random_state=7, test_size=0.2)
iris_x_train, iris_x_yan, iris_y_train, iris_y_yan = train_test_split(x_train_all, y_train_all, random_state=11, test_size=0.2)
print(iris_x_test)
print(iris_x_train)
print(iris_x_yan)
```

Test set:

```
           Age  Gender      Height  ...   WBC_xiao    WBC_chu   WBC_dang
1395  52.20000    0.60   34.260000  ...  31.220000  34.860000  31.860000
1062  42.00000    0.00  165.100000  ...   9.700000   9.700000  10.500000
440   70.75000    0.50   62.875000  ...   5.856250   8.187500   6.643750
1417  72.09375    0.50   47.540625  ...  11.053125  13.496875  12.053125
3333  64.00000    0.00  157.500000  ...   8.100000   8.100000  10.700000
...        ...     ...         ...  ...        ...        ...        ...
962   44.00000    1.00   -1.000000  ...   3.533333   6.100000   3.666667
3102  88.00000    1.00  182.900000  ...   6.400000   6.400000  13.400000
2478  78.00000    1.00  182.900000  ...  14.000000  14.000000  33.500000
1601  75.50000    1.00   -1.000000  ...   5.400000   7.100000   5.550000
429   65.00000    0.25  107.700000  ...  10.825000  15.400000  13.675000

[676 rows x 107 columns]
```

Training set:

```
            Age    Gender      Height  ...   WBC_xiao    WBC_chu   WBC_dang
1460  76.116279  0.395349   59.862791  ...  10.102326  12.653488  11.241860
1857  89.000000  1.000000   -1.000000  ...   9.900000  10.000000   9.900000
3016  80.000000  1.000000   -1.000000  ...   8.500000  11.700000   9.000000
18    64.000000  1.000000   -1.000000  ...   8.800000  10.100000  12.700000
2206  87.000000  1.000000   -1.000000  ...   7.300000  11.300000   7.300000
...         ...       ...         ...  ...        ...        ...        ...
1215  67.000000  0.000000   -1.000000  ...   2.600000   2.600000   5.900000
760   58.857143  0.857143  151.885714  ...   7.785714   9.700000   9.685714
3295  63.000000  0.000000   -1.000000  ...   8.800000   9.700000   8.800000
1595  62.444444  0.666667  171.588889  ...   7.944444   9.933333   9.622222
466   69.000000  1.000000   -1.000000  ...  11.300000  17.000000  11.300000

[2163 rows x 107 columns]
```

Validation set:

```
              Age    Gender      Height  ...    WBC_xiao  WBC_chu   WBC_dang
1044    65.666667  0.555556   37.044444  ...  13.333333    17.40  13.833333
3052    80.000000  1.000000  175.300000  ...   3.900000    15.00  13.700000
1565    37.500000  1.000000   -1.000000  ...  10.900000    14.45  10.900000
2989    66.000000  1.000000  170.200000  ...   3.900000     3.90   9.800000
1169    70.000000  0.666667  141.216667  ...   8.733333     9.30  10.766667
...           ...       ...         ...  ...        ...      ...        ...
2739    76.000000  1.000000  182.900000  ...  15.200000    15.20  16.100000
2063    55.000000  1.000000   -1.000000  ...  28.200000    33.60  28.200000
2808    73.000000  1.000000  160.000000  ...  31.300000    31.30  60.200000
2380    76.000000  0.000000   -1.000000  ...  12.500000    12.50  16.300000
2656    50.000000  0.000000  167.600000  ...   9.400000    16.10  15.400000

[541 rows x 107 columns]
```

5. For data after down sampling:

feature standardization, maximum and minimum value standardization, converted value range (0,1)

```
from sklearn.preprocessing import MinMaxScaler
min_max_scaler = MinMaxScaler(copy=True, feature_range=(0, 1))
new_X_train = iris_x_train
new_X_test = iris_x_test
new_X_yan = iris_x_yan
from sklearn.preprocessing import Normalizer
normalizer = Normalizer(copy=True, norm='l2').fit(new_X_train)
new_X_train = normalizer.transform(new_X_train)
new_X_test = normalizer.transform(new_X_test)
new_X_yan = normalizer.transform(new_X_yan)
```

6. Model Building:

Model 1:

MLPClassifier (multilayer perceptron classifier) and try to use this algorithm to build the model. The parameters are set to three hidden layers, the first layer is 100 neurons, the second layer is 50 neurons, the third layer is 50 neurons, clf=MLPClassifier (solver='lbfgs', alpha=1e-5, hidden _layer_sizes=(100, 50, 50), random_ State=1) # 3 hidden layers.

```
# 拟合（模型训练）
print(iris_x_train)
clf.fit(iris_x_train, iris_y_train)
iris_y_predict = clf.predict(iris_x_test)
score = clf.score(iris_x_test, iris_y_test, sample_weight=None)
print('iris_y_predict=')
print(iris_y_predict)
```

```
ddd = []
for i in iris_y_predict:
      ddd.append(i)
print(ddd)
print('iris_y_test=')
print(iris_y_test)
print('Accuracy:', score)
```

Finally, the accuracy rate is obtained through the test set test:

Accuracy: 0.5754437869822485.

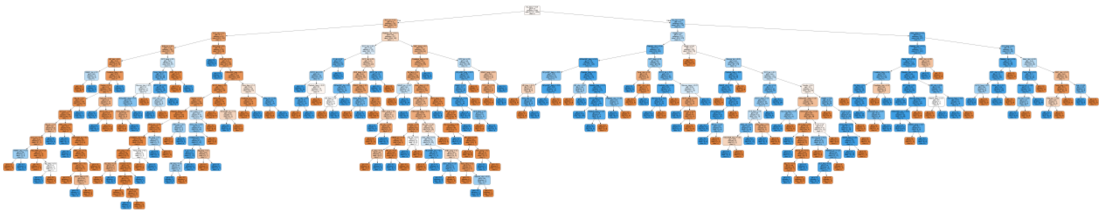
Model 2 (final):

Random forest algorithm: It is very suitable for multi-dimensional data and has a good effect on dealing with binary classification problems.

```
RandomForest Classifier model under sklearn
clf1 = RandomForestClassifier(n_estimators=200)
clf1.fit(new_X_train, iris_y_train)
```

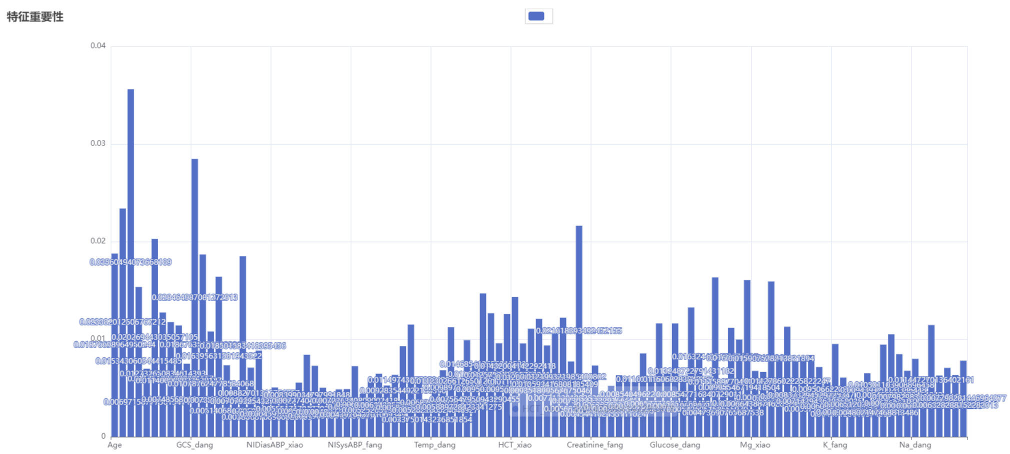
Set the number of decision trees to 200 (the best effect)

The following is a visual diagram of each decision tree (200 complete under the project, only one is shown here).



7. Model results

Visualization of feature importance:

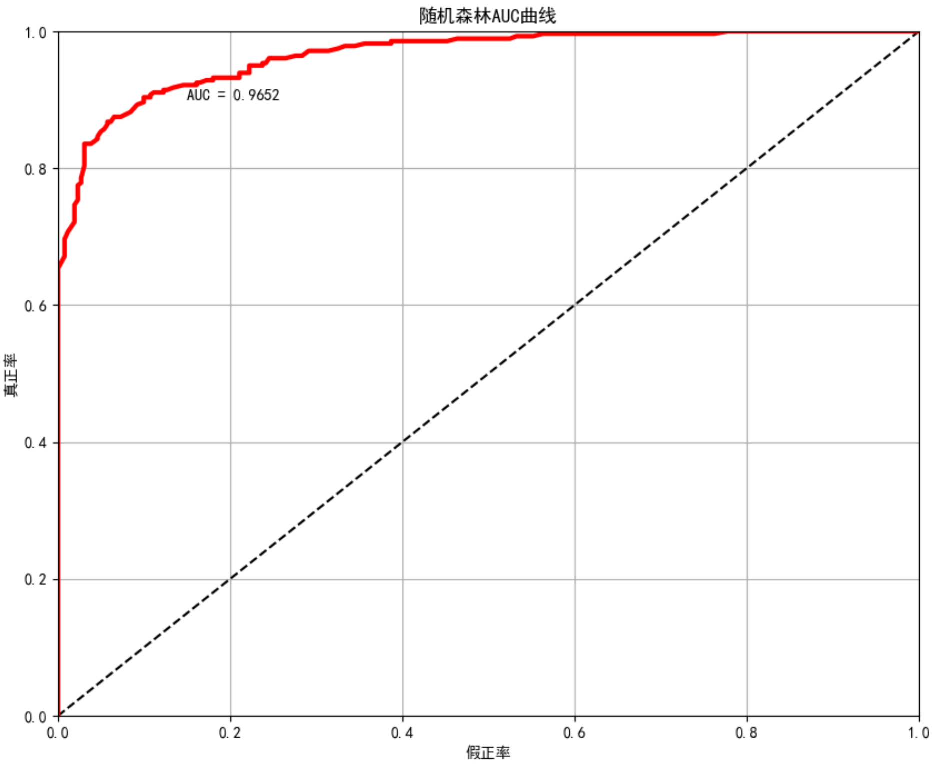


The impact of features on the whole model can be clearly analyzed from the figure (file all.html), and some features with low impact can be eliminated.

```
Predicted value: (1: death, 0: survivor)
[0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1,
```

1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1]
True value: (1: death, 0: survivor) [0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0]

Use the validation set to generate the AUC visualization:



Evaluation index F1: 0.8958333333334
Accuracy: 0.8958333333334
Response rate: 0.8958333333334

Conclusion

The research and analysis of this medical data, for predicting the mortality rate of patients in ICU wards, adopts the random forest algorithm to build the model, and the accuracy rate of the results in the test set reaches 89.5%, which has certain medical reference value. The performance of the model in the validation set, draw the AUC curve, and get AUC=0.9652 (the closer the value is, the better).

Improvements:

For data processing and feature engineering, the extraction and segmentation of features are not detailed enough, this data can also mine deeper features, which need to be optimized.