Heart Failure Prediction Using the MIMIC III ICU database.

Author, J.D. Schonhoft Ph.D.

1) Goal:

For myself to learn more about heart failure and SQL with the MIMIC database.

2) Domain Background:

Heart failure is a condition where the heart does not pump blood efficiently. It has many underlying causes. It's one of the leading cause of mortality and most of the cost is associated with inpatient care. Predicting outcomes in this case is an important problem.

There have been a few models published. A quick search in the pubmed database includes about 11 references (a dearth given the frequency of Heart Failure). All of these studies use smaller databases, therefore using the MIMIC database should provide a good research oppurtunity.

There have been a handful of models (Random Forest, SVM, Naive Bayes etc) using a few hundred patients. In the MIMIC III database there are 10000 HF patients which represents a significant advantage.

3) Problem Statement:

I want to see if I can predict 30 day mortality after their ICU visit (whether the patient is alive or dead) for heart failure patients with available ICU records, primarlily the lab records. The lab records are primarily numeric and contain measurements from different clinical tests. An example of this would be blood oxygen content or blood hemoglobin levels.

This data can be used in several different ways. It is essentially a classification problem (i.e. predicting whether patients survived 30 days after their ICU visit). However as a side, it would also be interesting to look for different clusters within heart failure patients to understand if particular groups of people are more susceptible or the like.

4) Datasets:

I am using the MIMIC III ICU database (https://mimic.physionet.org (https://mimic.physionet.org)). The MIMIC III ICU database is a database of of over 50000+ hospital records of patients who were admitted to the critical care unit. In total about 10000 of these patients have a heart failure diagnosis. Data is stored in a SQL type database, is about 80GB in size, and contains ~10+ tables for each ICU visit that contain information such as:

- lab records - ~700 + clinical lab measurements for each patient (blood hemoglobin levels, liver enzymes, blood oxygen etc.)
- Demographics
- Doctor's notes
- Diagnosis/Condition

Here are some basic information for the HF patients specifically

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import psycopg2
%matplotlib inline

# Connect to local copy of the MIMIC III database
sqluser = 'postgres'
dbname = 'mimic'
schema_name = 'mimiciii'

# Connect to local postgres version of mimic
con = psycopg2.connect(dbname=dbname, user=sqluser)
cur = con.cursor()
cur.execute('SET search_path to ' + schema_name)

# selecting all patient ids with heart failure diagnosis using ICD9code. There a
re about 10,000 in total.
data = pd.read_sql_query("SELECT * FROM diagnoses_icd WHERE icd9_code LIKE '%428
%'",con)
HF_subject_id = data.subject_id.unique()

#Next creating dataframe of patient information with id
patient_list = ', '.join([str(i) for i in HF_subject_id])
query= 'SELECT * FROM patients WHERE subject_id IN (%s)' % patient_list

# creating pandas data frame with patient demographics. Note that dates are shif
ted to that patients cannot be identified.
HF_patients = pd.read_sql_query(query, con)
HF_patients.head()
```

Out[1]:

	row_id	subject_id	gender	dob	dod	dod_hosp	dod_ssn	expire_flag
0	2	3	M	2025-04-11	2102-06-14	NaT	2102-06-14	1
1	8	9	M	2108-01-26	2149-11-14	2149-11-14	2149-11-14	1
2	18	21	M	2047-04-04	2135-02-08	2135-02-08	2135-02-08	1
3	23	26	M	2054-05-04	2128-02-25	NaT	2128-02-25	1
4	26	30	M	1872-10-14 00:00:00	NaT	NaT	NaT	0

What is the gender distribution of Heart Failure patients diagnoses in the ICU?

In [2]:

```
HF_patients.gender.value_counts()
```

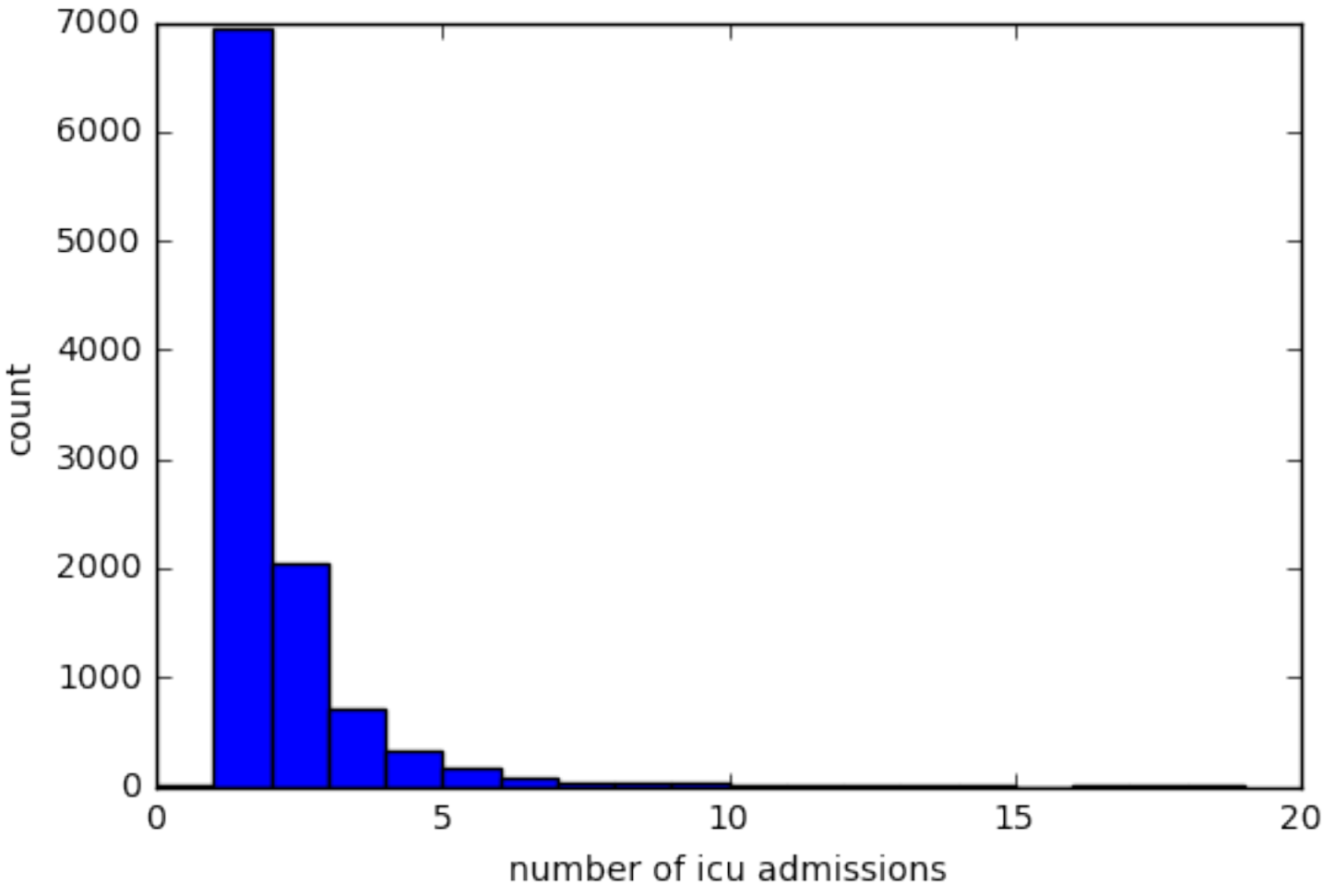
Out[2]:

```
M    5555
F    4849
Name: gender, dtype: int64
```

How many admissions entries do each of these people have

In [17]:

```
#Next creating dataframe of patient information with id
patient_list = ', '.join([str(i) for i in HF_subject_id])
query= 'SELECT * FROM admissions WHERE subject_id IN (%s)' % (patient_list)

HF_admissions = pd.read_sql_query(query, con)

plt.hist(list(HF_admissions.subject_id.value_counts()), bins = range(0,20))
plt.xlabel('number of icu admissions')
plt.ylabel('count')

from collections import Counter
admissions = Counter(HF_admissions.subject_id)
HF_patients['icu_admissions_num'] = [admissions[i] for i in HF_patients.subject_
id]
```



5) Solution:

First I will find features of the data with an exploratory analysis, e.g. by correlation etc. and then develop and ML pipeline to test mortality predictions.

6) Benchmark model:

There are several models in the litereature. A search of "Heart Failure Mortality Prediction Machine Learning" yielded 11 papers. Here is one example using a variant of Naive Bayes (AODE in WEKA), that predicted mortality with a c-statistic of 0.84 - e.g from a ROC curve. There are other models as well using RF or SVM, howerver the majority of these use patients from a very small cohort (50 - 100). I will also compare my model to these.

There is an excellent review that jsut came out: I will use this as my main point of reference:

Tripoliti EE, Papadopoulos TG, Karanasiou GS, Naka KK, Fotiadis DI. Heart Failure: Diagnosis, Severity Estimation and Prediction of Adverse Events Through Machine Learning Techniques. Computational and Structural Biotechnology Journal. 2017;15:26-47. doi:10.1016/j.csbj.2016.11.001.

7) Evaluation metrics:

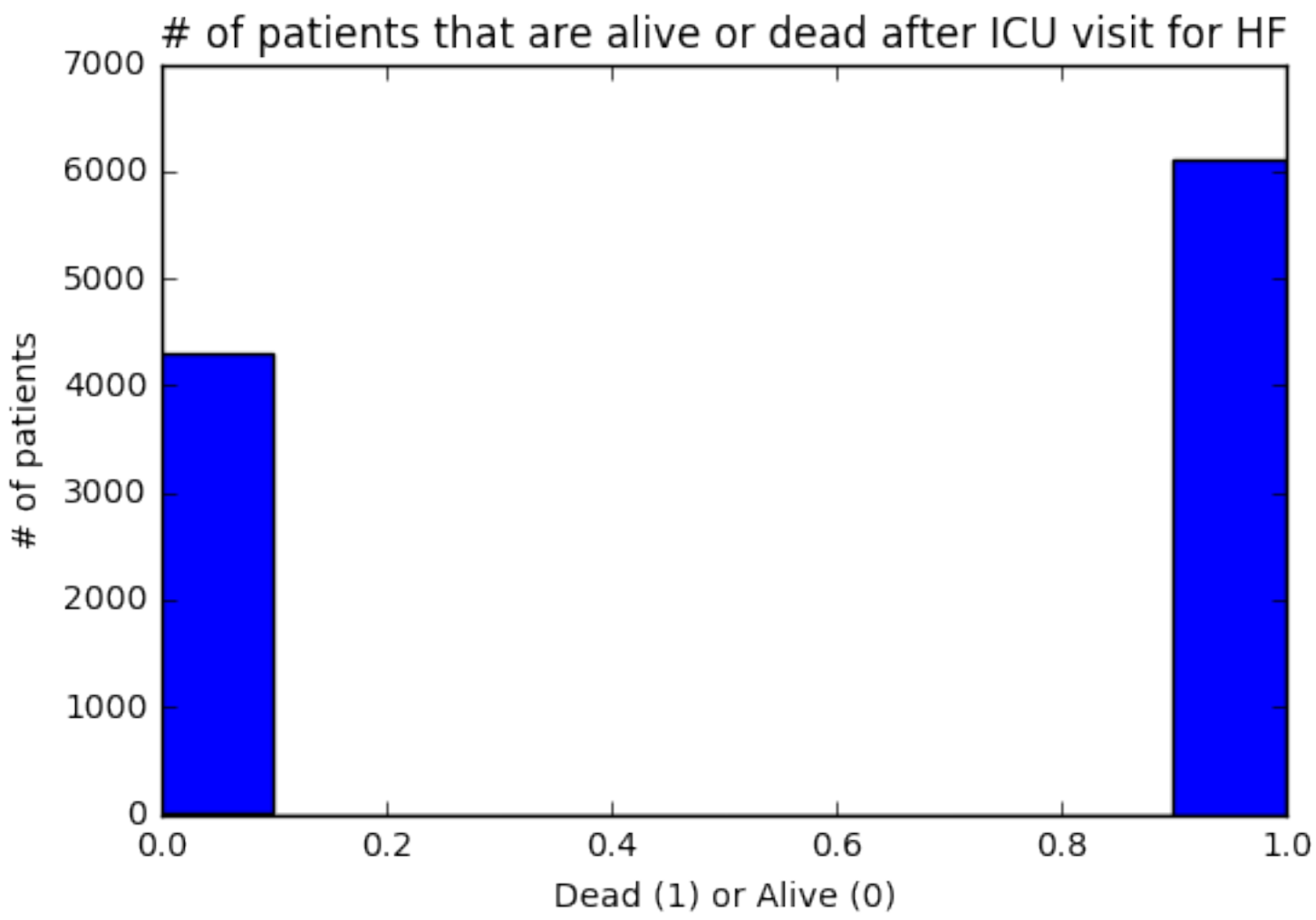
I plan to use accuracy along with a contingency table to assess how well the model works. I will use cross validation after I develop and ML classifier. I also will experiment with using different measures of mortality, such as 30-day mortality or 180 day mortality.

The dataset is relatively well balanced with about equal numbers alive and dead patients so accuracy should be a good measure of model performance.

I will also experiment with using a ROC/cstatistic type curve to evaluate performance

In [7]:

```
plt.figure()
plt.hist(HF_patients['expire_flag'])
plt.ylabel('# of patients')
plt.xlabel('Dead (1) or Alive (0)')
plt.title('# of patients that are alive or dead after ICU visit for HF')
plt.show()
```



8) project design

First I will parse the data choosing from a few different categories of measurements.

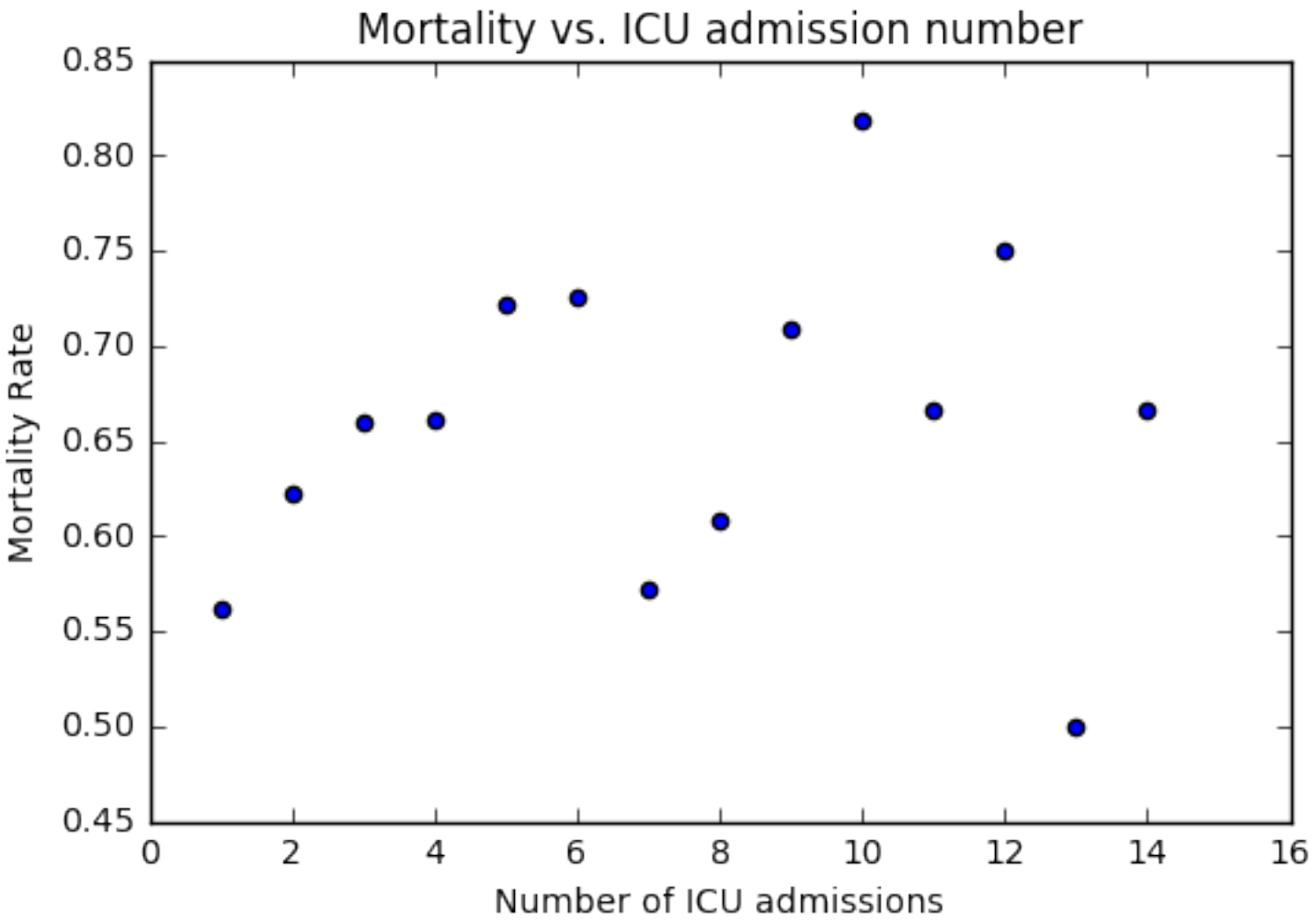
1) Lab Measurements primarily, there are about ~700 total, I'm going to read a bit and pick maybe 20-50 measurements
2) Drugs that the patient is on
3) Remove outliers
4) Develop an ML pipeline and test several different supervised learning methods (SVM, DT, etc.)
5) experiment with omitting and using patient drugs, also stratify data with different types of HF
6) perform cross validation
7) compare performance to the the naive bayes model

9) Other exploratory analysis.

Mostly for my own purpose.

In [29]:

```
plt.figure()
for i in range(1, 15):
    plt.scatter(i, HF_patients[HF_patients['icu_admissions_num'] == i]['expire_f
lag'].mean())
plt.ylabel('Mortality Rate')
plt.xlabel('Number of ICU admissions')
plt.title('Mortality vs. ICU admission number')
plt.show()
```



In [54]:

```
#selecting patients with only one ICU admission


HF_patients_1 = HF_patients[HF_patients['icu_admissions_num'] == 1]
HF_admissions_1 = HF_admissions[HF_admissions['subject_id'].isin(HF_patients_1['
subject_id'])]


# from datetime import datetime
# date_format = "%m/%d/%Y"
# a = datetime.strptime('8/18/2008', date_format)
# b = datetime.strptime('9/26/2008', date_format)
# delta = b - a
# print delta.days # that's it


HF_pat1 = pd.merge(HF_patients_1, HF_admissions_1[['subject_id','hadm_id','admit
time','dischtime']], on='subject_id')
```

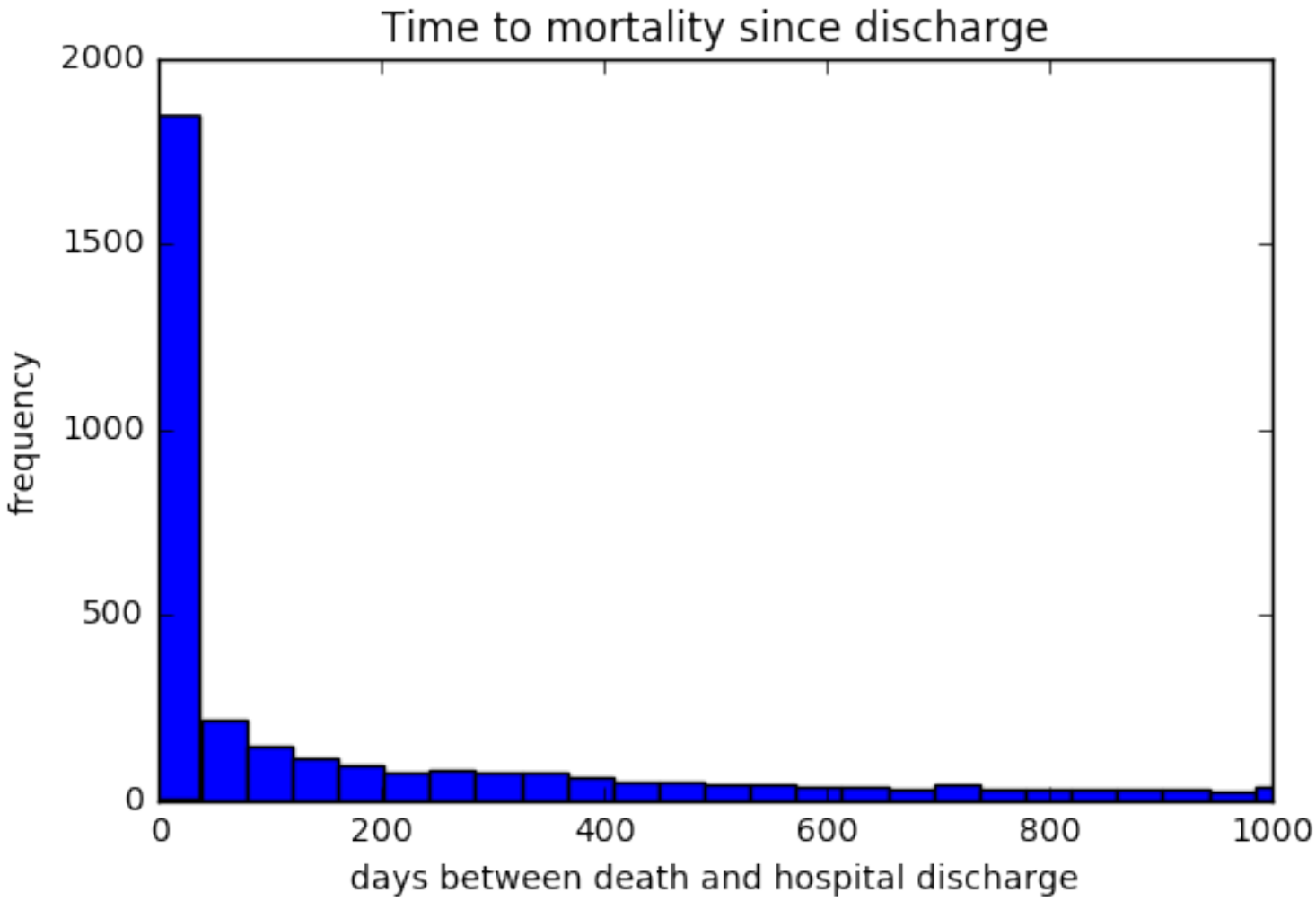
39

In [120]:

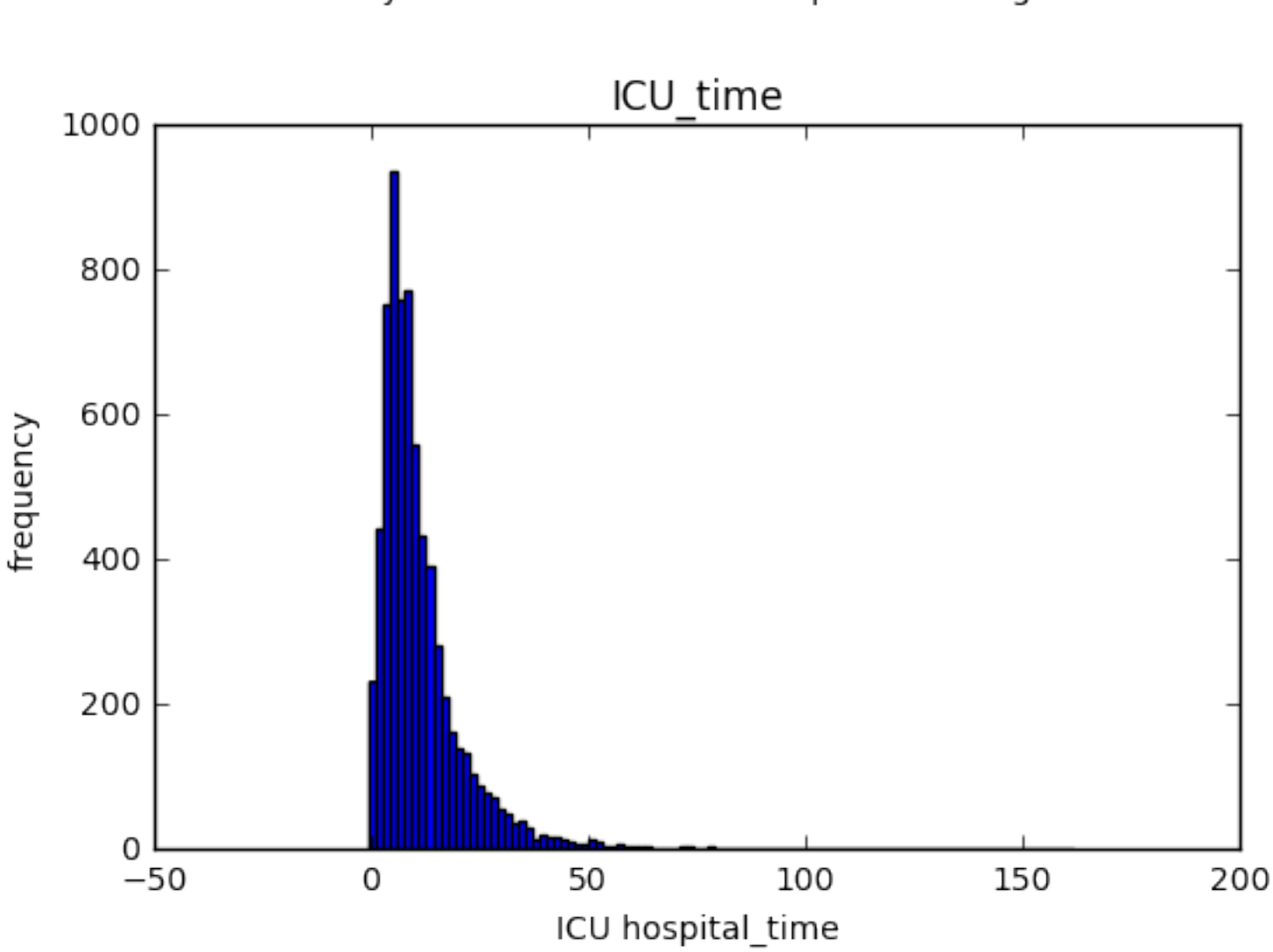
```
HF_pat1['hospital_time'] = (HF_pat1['dischtime']-HF_pat1['admittime'])/np.timede
lta64(1, 's')/3600./24.
HF_pat1['mortality_time'] = (HF_pat1['dod']-HF_pat1['dischtime'])/np.timedelta64
(1, 's')/3600./24.

print HF_pat1.shape

plt.figure()
plt.hist(HF_pat1[np.isfinite(HF_pat1['mortality_time'])]['mortality_time'], bins
= 100)
plt.xlabel('days between death and hospital discharge')
plt.ylabel('frequency')
plt.xlim([-1,1000])
plt.title('Time to mortality since discharge')


plt.figure()
plt.hist(HF_pat1[np.isfinite(HF_pat1['hospital_time'])]['hospital_time'], bins =
100)
plt.xlabel('ICU hospital_time')
plt.ylabel('frequency')
plt.title('ICU_time')
plt.show()
```

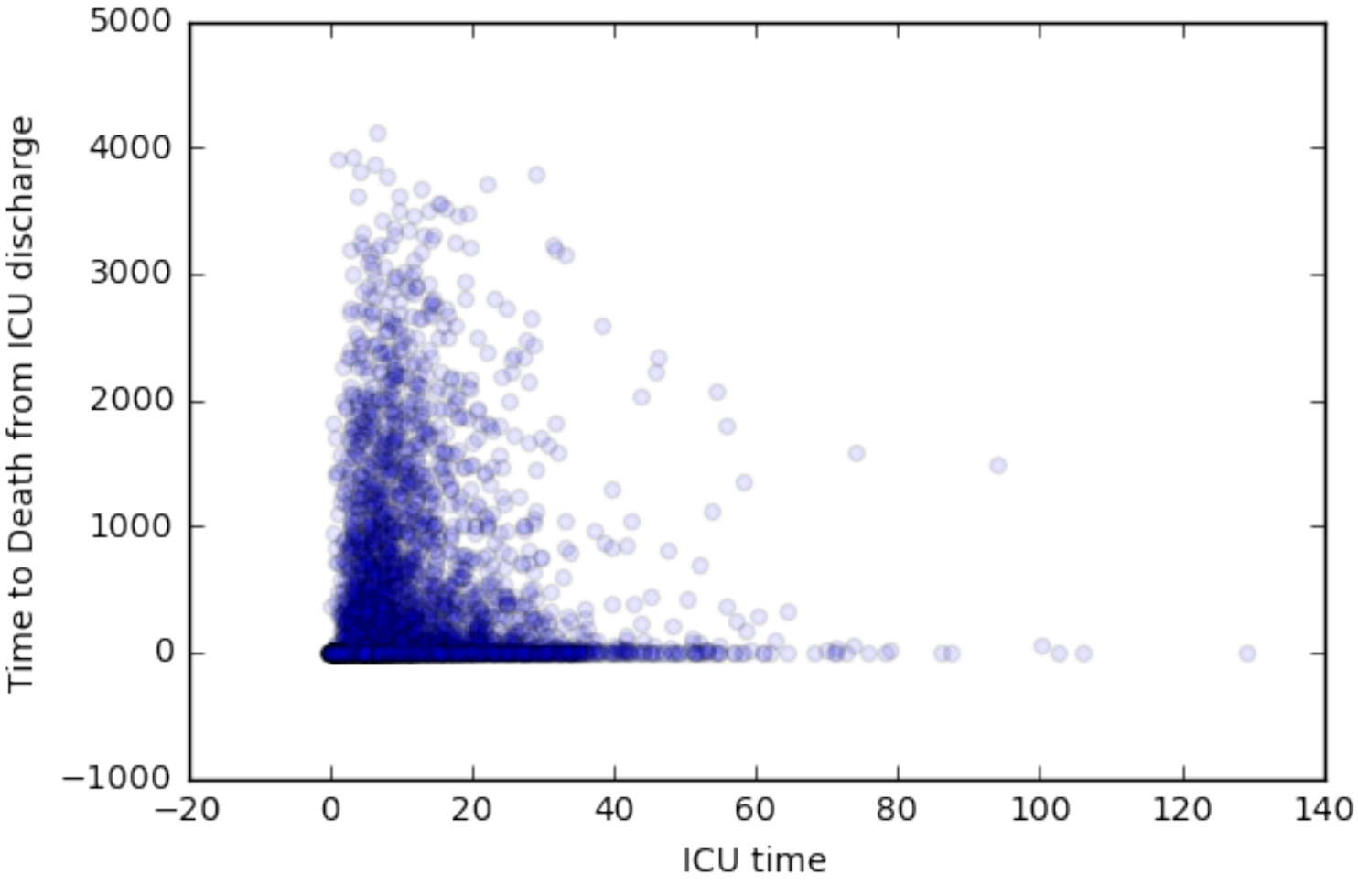
```
plt.figure()
df = HF_pat1[(HF_pat1['mortality_time'] > -1.)]
x = df['hospital_time']
y = df['mortality_time']
plt.xlabel('ICU time')
plt.ylabel('Time to Death from ICU discharge')
plt.scatter(x,y, alpha = 0.1)
plt.show()
from scipy.stats.stats import pearsonr
print pearsonr(x, y)
```

```
(6953, 14)
```


Time to mortality since discharge
2000
1500
1000
500
0
frequency
0
200
400
600
800
1000
days between death and hospital discharge

ICU time


1000
800
600
400
200
0
frequency
−50
0
50
100
150
200
ICU hospital_time




```
(-0.035292333299381265, 0.02754525301035747 1)
```

Now selecting one measure and looking at distribution - blood pH (50820)

In [8]:

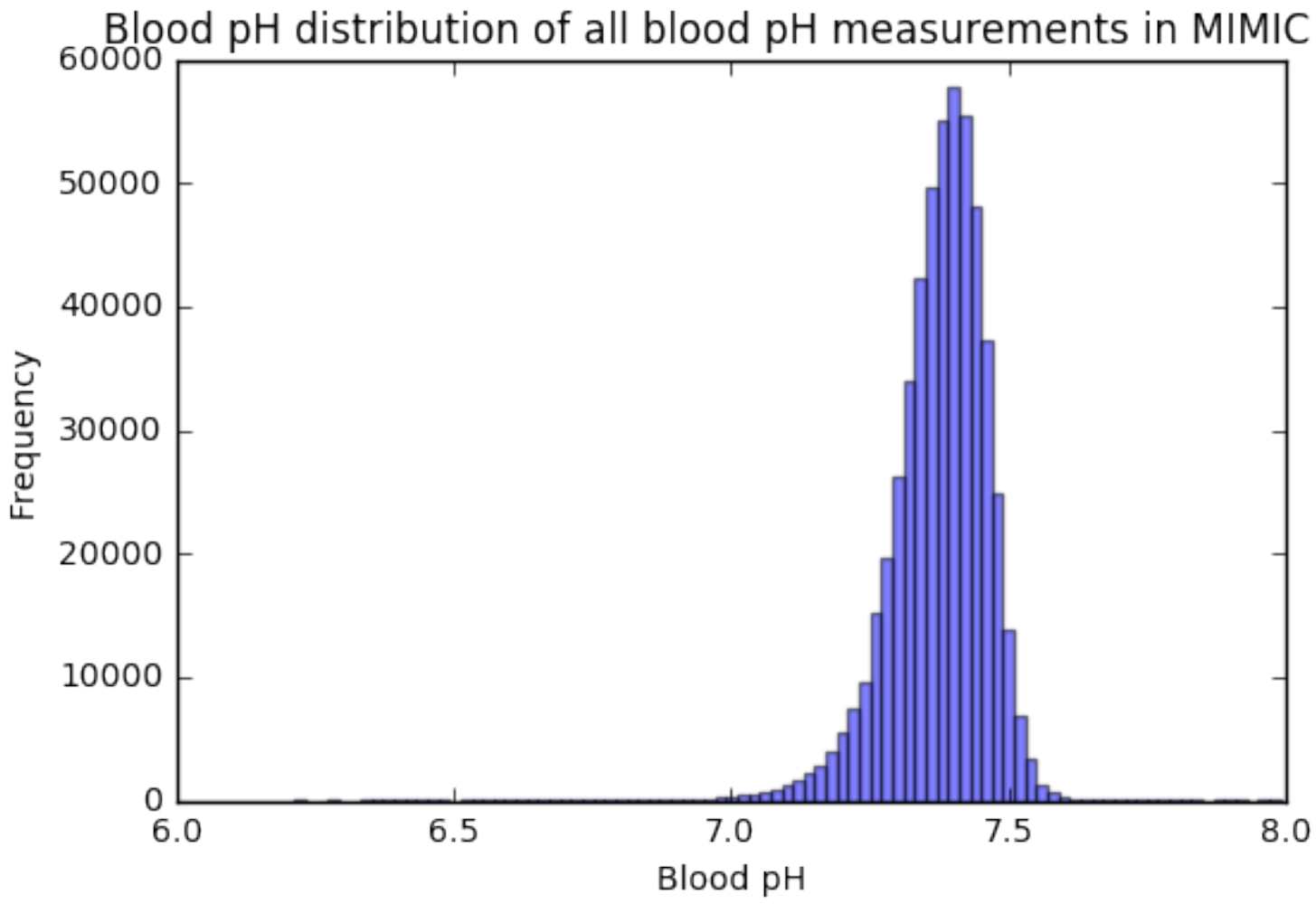
```
query = 'SELECT * FROM labevents WHERE itemid IN (%s)' % (50820)
HF_labevents_bloodph = pd.read_sql_query(query, con)
```

In [10]:

```
HF_labevents_bloodph['valuenum'].plot.hist(bins = 400, alpha = 0.5)
plt.xlim([6,8])
plt.xlabel('Blood pH')
plt.title('Blood pH distribution of all blood pH measurements in MIMIC III')
```

Out[10]:

```
<matplotlib.text.Text at 0x134dcc950>
```

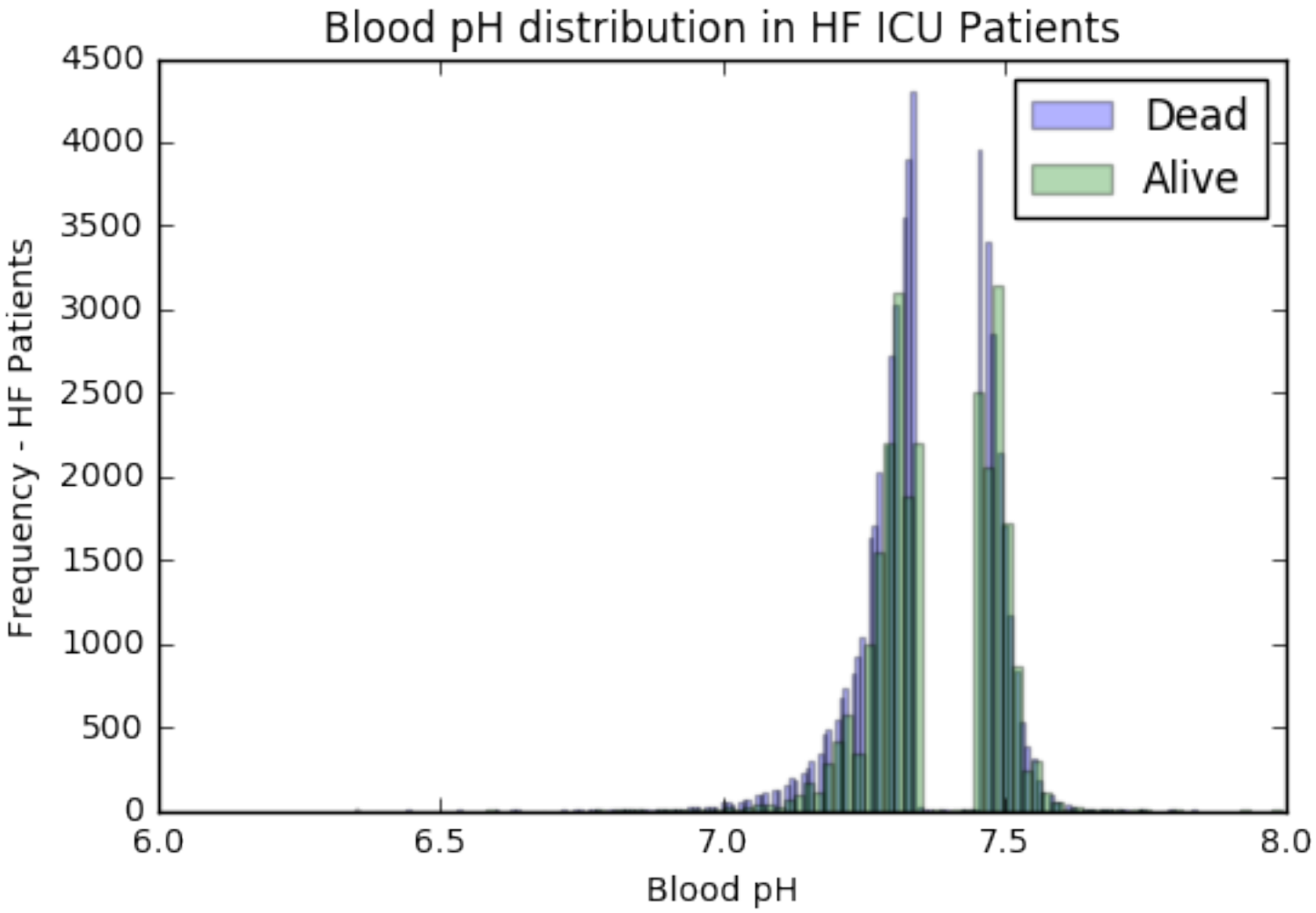


In [11]:

```
HF_mortality = dict(zip(HF_patients['subject_id'], HF_patients['expire_flag']))

Y = []
for i in HF_labevents_bloodph['subject_id']:
    try:
        Y.append(HF_mortality[i])
    except KeyError:
        Y.append(np.nan)

HF_labevents_bloodph['HF_mortality'] = Y
```

In [16]:

```
data = HF_labevents_bloodph.dropna()
# plt.scatter(data['valuenum'], [float(i) for i in data['HF_mortality']], alpha
= 0.1)
plt.figure(1)
plt.xlim([6,8])
plt.hist(data[data['HF_mortality'] == 1]['valuenum'], bins = 200, alpha = 0.3, l
abel = 'Dead')
plt.hist(data[data['HF_mortality'] == 0]['valuenum'], bins = 400, alpha = 0.3, l
abel = 'Alive')
plt.xlabel('Blood pH')
plt.ylabel('Frequency - HF Patients')
plt.title('Blood pH distribution in HF ICU Patients')
plt.legend()
plt.show()
```



Doesn't seem to much correlation with mortality and blood pH. This was jsut a test to see if I could pull out some data and work with it a bit. Now I will implement my plan as above.

In [ ]:

```

```