# 03_in_hospital_mortality

July 19, 2021

Nicola De Cristofaro (Matr. 0522500876)    Cloud Computing Curriculum

## 1 In-Hospital Mortality

First of all what do we mean by "In-Hospital Mortality"? In-Hospital Mortality refers to a death occurred during the hospital stay.

### 1.1 1. Problem Statement

**The goal is to create a model that predicts if a patient will incur in death during its hospital stay.**

### 1.2 2. Type of model used for prediction

"In-Hospital Mortality" is a categorical attribute (YES,NO), so normally a classification model has to be used, but in this case we have a binary classifier so we could also treat the two possible output categories as numeric (YES=1, NO=0) and use a **regression model**. We will do Regression in our task.

### 1.3 3. Metrics used for validation

So, to measure performance of our model, we'll compute the root-mean-square error (RMSE). The RMSE is a commonly used measure of the differences between values predicted by a model and the values observed, where a *lower score implies better accuracy.* For example, a perfect prediction model would have an RMSE of 0.

The RMSE equation for this work is given as follows, where (n) is the number of hospital admission records, (y-hat) the prediction In-Hospital Mortality, and (y) is the actual In-Hospital Mortality.

To determine the best regression model between the subset of models that will be evaluated, the **R2 (R-squared)** score will be used.

R Square measures how much variability in dependent variable can be explained by the model. In other words, it is the proportion of the variance in the dependent variable that is predictable from the independent variables. R2 is defined as the following equation where (y__i) is an observed data point, (ŷ) is the mean of the observed data, and (f__i) the predicted model value.

Best possible R2 score is 1.0.

[136]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

We start importing our baseline dataset extracted selecting only the necessary tables from MIMIC dataset.

[137]:
```python
# Import baseline dataset constructed in data extraction and preparation phase
admits_patients_diag = pd.read_csv('admits_patients_diag.csv')

#convert dates
admits_patients_diag.admittime = pd.to_datetime(admits_patients_diag.admittime)
admits_patients_diag.dischtime = pd.to_datetime(admits_patients_diag.dischtime)
admits_patients_diag.deathtime = pd.to_datetime(admits_patients_diag.deathtime)

admits_patients_diag.head()
```

[137]:
```
   Unnamed: 0  subject_id   hadm_id           admittime           dischtime  \
0           0    14679932  21038362 2139-09-26 14:16:00 2139-09-28 11:30:00
1           1    15585972  24941086 2123-10-07 23:56:00 2123-10-12 11:22:00
2           2    15078341  23272159 2122-08-28 08:48:00 2122-08-30 12:32:00
3           3    17301855  29732723 2140-06-06 14:23:00 2140-06-08 14:25:00
4           4    17991012  24298836 2181-07-10 20:28:00 2181-07-12 15:49:00

  deathtime admission_type insurance              ethnicity  \
0       NaT       ELECTIVE     Other          OTHER/UNKNOWN
1       NaT       ELECTIVE     Other                  WHITE
2       NaT       ELECTIVE     Other  BLACK/AFRICAN AMERICAN
3       NaT       ELECTIVE     Other                  WHITE
4       NaT       ELECTIVE     Other                  WHITE

   died_at_the_hospital  … injury  mental misc  muscular  neoplasms  \
0                     0  …      2       0    0         0          0
1                     0  …      2       0    0         0          0
2                     0  …      3       0    0         0          0
3                     0  …      2       0    0         0          0
4                     0  …      2       0    0         0          0

   nervous  pregnancy  prenatal  respiratory  skin
0        0          0         0            0     0
1        0          0         0            0     0
2        0          0         0            0     0
3        0          0         1            0     0
4        0          0         0            0     0

[5 rows x 30 columns]
```

[138]:
```python
admits_patients_diag.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 335378 entries, 0 to 335377
Data columns (total 30 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   Unnamed: 0            335378 non-null  int64
 1   subject_id            335378 non-null  int64
 2   hadm_id               335378 non-null  int64
 3   admittime             335378 non-null  datetime64[ns]
 4   dischtime             335378 non-null  datetime64[ns]
 5   deathtime             5670 non-null    datetime64[ns]
 6   admission_type        335378 non-null  object
 7   insurance             335378 non-null  object
 8   ethnicity             335378 non-null  object
 9   died_at_the_hospital  335378 non-null  int64
 10  gender                335378 non-null  object
 11  anchor_age            335378 non-null  int64
 12  dod                   24651 non-null   object
 13  blood                 335378 non-null  int64
 14  circulatory           335378 non-null  int64
 15  congenital            335378 non-null  int64
 16  digestive             335378 non-null  int64
 17  endocrine             335378 non-null  int64
 18  genitourinary         335378 non-null  int64
 19  infectious            335378 non-null  int64
 20  injury                335378 non-null  int64
 21  mental                335378 non-null  int64
 22  misc                  335378 non-null  int64
 23  muscular              335378 non-null  int64
 24  neoplasms             335378 non-null  int64
 25  nervous               335378 non-null  int64
 26  pregnancy             335378 non-null  int64
 27  prenatal              335378 non-null  int64
 28  respiratory           335378 non-null  int64
 29  skin                  335378 non-null  int64
dtypes: datetime64[ns](3), int64(22), object(5)
memory usage: 76.8+ MB
```

[139]:
```
# Create LOS attribute converting timedelta type into float 'days', 86400␣
↪seconds in a day
admits_patients_diag['los'] = (admits_patients_diag['dischtime'] -␣
↪admits_patients_diag['admittime']).dt.total_seconds()/86400

# Verify LOS computation
admits_patients_diag[['admittime', 'dischtime', 'los']].head()
```

[139]:
```
             admittime           dischtime       los
0 2139-09-26 14:16:00 2139-09-28 11:30:00  1.884722
1 2123-10-07 23:56:00 2123-10-12 11:22:00  4.476389
2 2122-08-28 08:48:00 2122-08-30 12:32:00  2.155556
3 2140-06-06 14:23:00 2140-06-08 14:25:00  2.001389
4 2181-07-10 20:28:00 2181-07-12 15:49:00  1.806250
```

We have computed the length of stay fo each admission because there are some LOS less than 0 (negative), they could refer to a patients died before the admission do this kind of entries have to be dropped.

[140]:
```python
# Remove LOS with negative number
admits_patients_diag = admits_patients_diag[admits_patients_diag['los'] > 0]
admits_patients_diag.info()
```

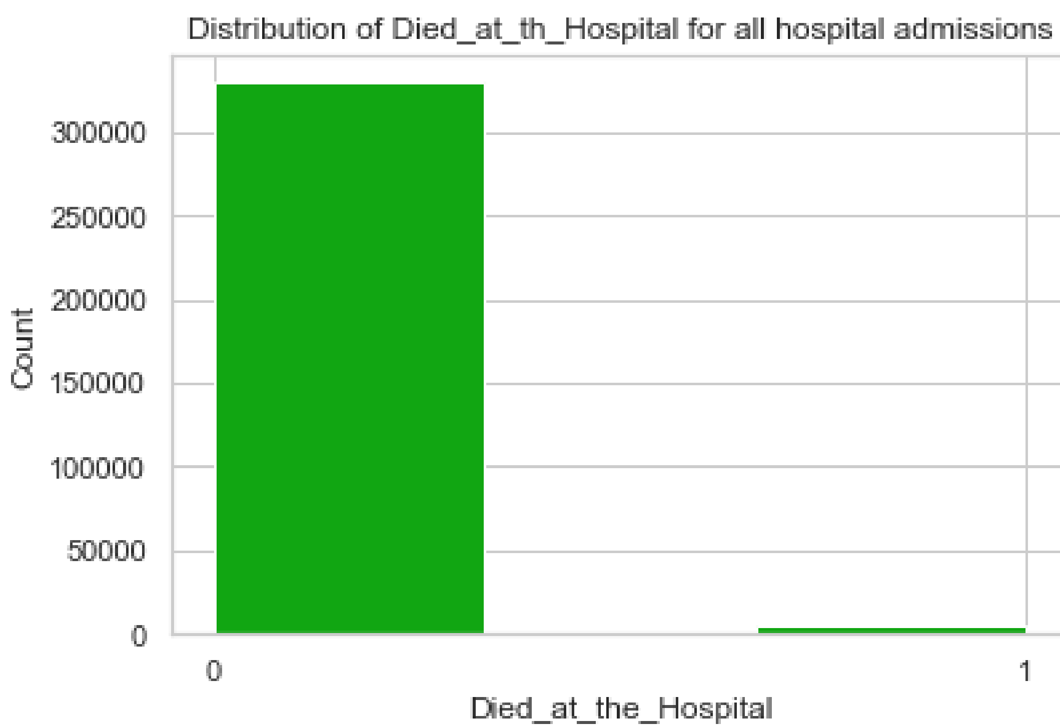
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 335257 entries, 0 to 335377
Data columns (total 31 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   Unnamed: 0             335257 non-null  int64
 1   subject_id             335257 non-null  int64
 2   hadm_id                335257 non-null  int64
 3   admittime              335257 non-null  datetime64[ns]
 4   dischtime              335257 non-null  datetime64[ns]
 5   deathtime              5605 non-null    datetime64[ns]
 6   admission_type         335257 non-null  object
 7   insurance              335257 non-null  object
 8   ethnicity              335257 non-null  object
 9   died_at_the_hospital   335257 non-null  int64
 10  gender                 335257 non-null  object
 11  anchor_age             335257 non-null  int64
 12  dod                    24581 non-null   object
 13  blood                  335257 non-null  int64
 14  circulatory            335257 non-null  int64
 15  congenital             335257 non-null  int64
 16  digestive              335257 non-null  int64
 17  endocrine              335257 non-null  int64
 18  genitourinary          335257 non-null  int64
 19  infectious             335257 non-null  int64
 20  injury                 335257 non-null  int64
 21  mental                 335257 non-null  int64
 22  misc                   335257 non-null  int64
 23  muscular               335257 non-null  int64
 24  neoplasms              335257 non-null  int64
 25  nervous                335257 non-null  int64
 26  pregnancy              335257 non-null  int64
 27  prenatal               335257 non-null  int64
```

```
 28   respiratory           335257 non-null   int64
 29   skin                  335257 non-null   int64
 30   los                   335257 non-null   float64
dtypes: datetime64[ns](3), float64(1), int64(22), object(5)
memory usage: 81.8+ MB
```

[141]:
```
admits_patients_diag.died_at_the_hospital.value_counts()
```

[141]:
```
0    329652
1      5605
Name: died_at_the_hospital, dtype: int64
```

[142]:
```
# Plot LOS Distribution
plt.hist(admits_patients_diag['died_at_the_hospital'], bins=3, color =␣
 ↪'#11a612')
plt.title('Distribution of Died_at_th_Hospital for all hospital admissions')
plt.ylabel('Count')
plt.xlabel('Died_at_the_Hospital')
plt.xticks(range(0,2))
plt.show();
```



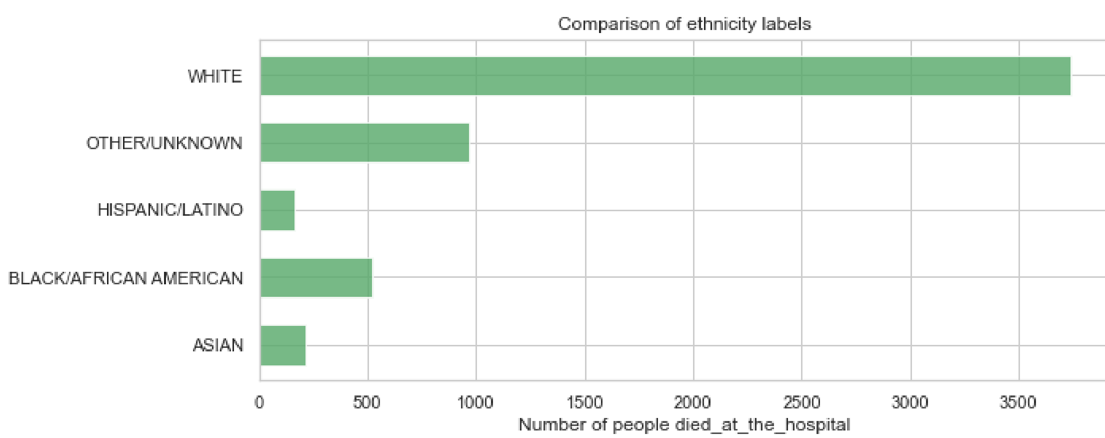

Died_at_the_Hospital = 1 means that the patients is died during the hospital stay. As we can see, luckily we have a small number of death compared to survivors.

[143]:
```
print('Number of positive samples:', (admits_patients_diag.died_at_the_hospital␣
↪== 1).sum()) # died at hospital
print('Number of negative samples:',  (admits_patients_diag.
↪died_at_the_hospital == 0).sum()) # not died at hospital
print('Total samples:', len(admits_patients_diag))
```

```
Number of positive samples: 5605
Number of negative samples: 329652
Total samples: 335257
```

### 1.3.1 Died_at_the_Hospital VS Ethnicity

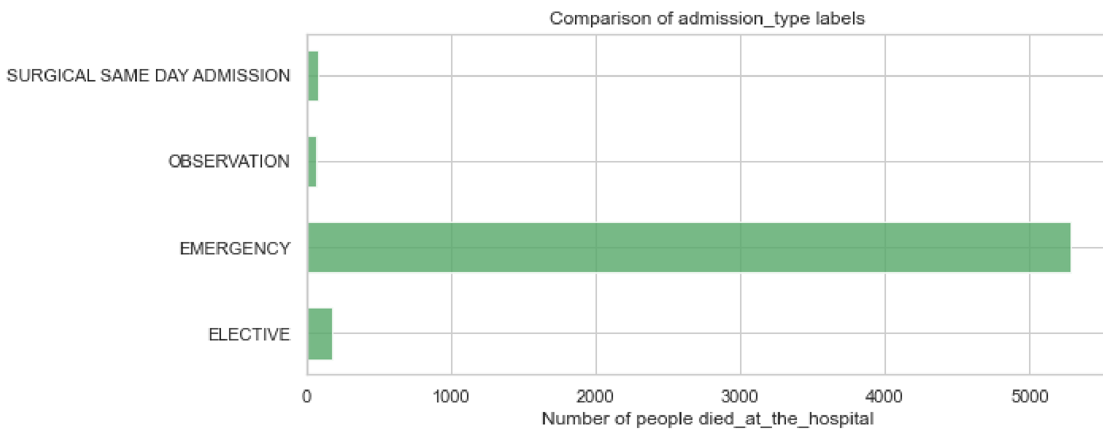
[144]:
```
# Re-usable plotting function
def plot_los_groupby(variable, size=(7,4)):
    results = admits_patients_diag[[variable, 'died_at_the_hospital']].
↪groupby(variable).sum().reset_index()
    values = list(results['died_at_the_hospital'].values)
    labels = list(results[variable].values)

    fig, ax = plt.subplots(figsize=size)
    ind = range(len(results))
    ax.barh(ind, values, align='center', height=0.6, color = '#55a868', alpha=0.
↪8)
    ax.set_yticks(ind)
    ax.set_yticklabels(labels)
    ax.set_xlabel('Number of people died_at_the_hospital')
    ax.tick_params(left=False, top=False, right=False)
    ax.set_title('Comparison of {} labels'.format(variable))
    plt.tight_layout()
    plt.show();

# Look at median LOS for groups ETHNICITY
plot_los_groupby('ethnicity', size=(10,4))
```



We can notice that Asian and Hispanic/latino people usually do not incur in death during their hospital stays.

### 1.3.2 Died_at_the_Hospital VS Admission_Type

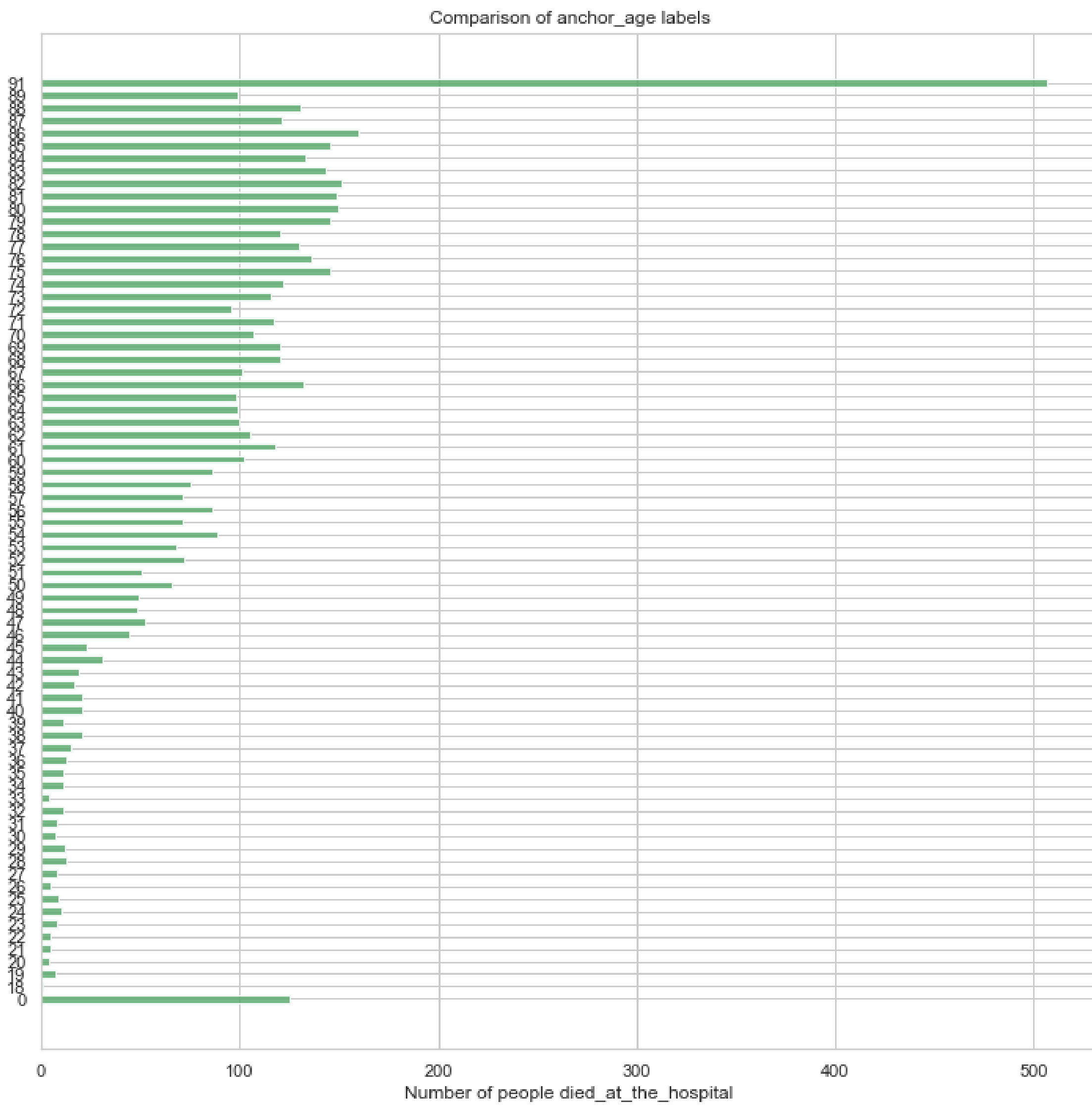
[145]:
```
plot_los_groupby('admission_type', size=(10,4))
```




As we expect the biggest number of death is for patients admitted with emergency.

### 1.3.3 Died_at_the_Hospital VS Age

[146]:
```
plot_los_groupby('anchor_age', size=(10,10))
```




Because of the discrete-like distribution of data on the extremes of age (0 and >89), it could be useful to convert all ages into the categories of **newborn, young adult, middle adult, and senior** for use in the prediction model.

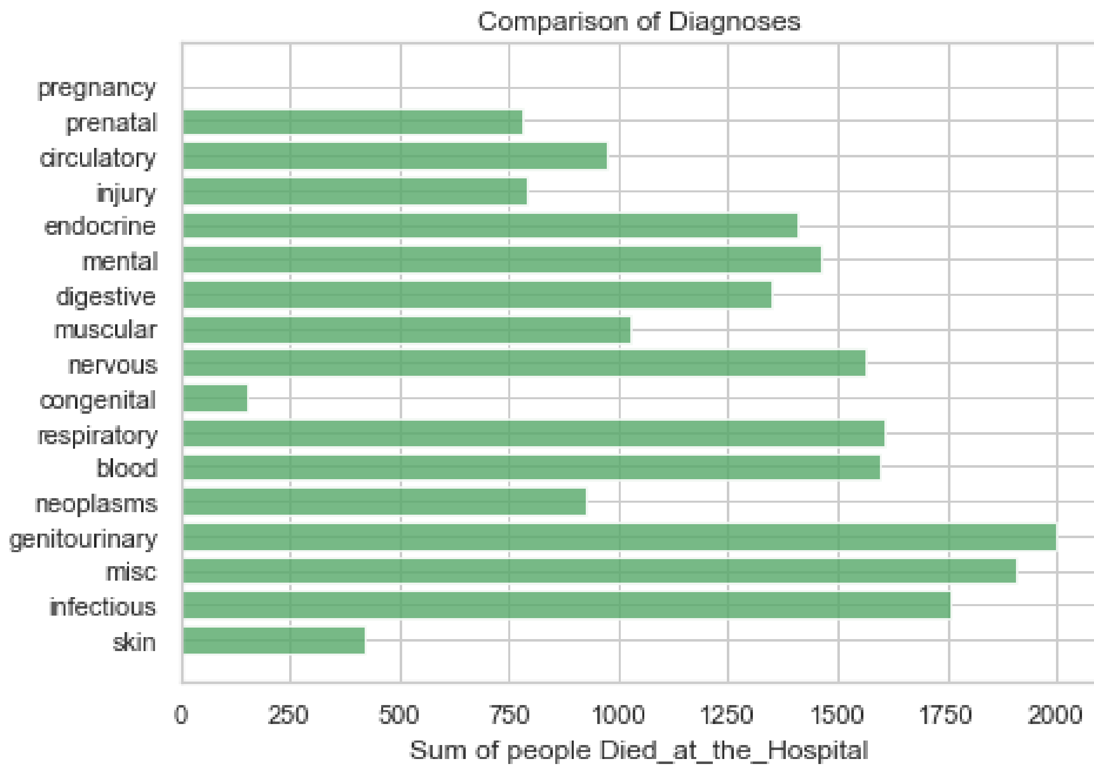
[147]:
```
age_ranges = [(0, 13), (14, 36), (37, 56), (57, 100)]
for num, cat_range in enumerate(age_ranges):
    admits_patients_diag['anchor_age'] = np.
 ↪where(admits_patients_diag['anchor_age'].between(cat_range[0],cat_range[1]),␣
 ↪num, admits_patients_diag['anchor_age'])

age_dict = {0: 'NEWBORN', 1: 'YOUNG_ADULT', 2: 'MIDDLE_ADULT', 3: 'SENIOR'}
admits_patients_diag['anchor_age'] = admits_patients_diag['anchor_age'].
 ↪replace(age_dict)
```

```
admits_patients_diag.anchor_age.value_counts()
```

[147]:
```
SENIOR          155448
MIDDLE_ADULT     88480
YOUNG_ADULT      52875
NEWBORN          38454
Name: anchor_age, dtype: int64
```

Now, let's analyze the diagnosis in correlation to our target LOS.

[148]:
```
import seaborn as sns

# Look at the median LOS by diagnosis category
diag_cat_list = ['skin', 'infectious',  'misc', 'genitourinary', 'neoplasms',␣
 ↪'blood', 'respiratory',
                  'congenital','nervous', 'muscular', 'digestive', 'mental',␣
 ↪'endocrine', 'injury',
                 'circulatory', 'prenatal',  'pregnancy']

results = []
for variable in diag_cat_list:
    results.append(admits_patients_diag[[variable, 'died_at_the_hospital']].
 ↪groupby(variable).sum().reset_index().values[1][1])


sns.set(style="whitegrid")
fig, ax = plt.subplots(figsize=(7,5))
ind = range(len(results))
ax.barh(ind, results, color = '#55a868', alpha=0.8)
ax.set_yticks(ind)
ax.set_yticklabels(diag_cat_list)
ax.set_xlabel('Sum of people Died_at_the_Hospital')
ax.tick_params(left=False, right=False, top=False)
ax.set_title('Comparison of Diagnoses'.format(variable))
plt.show();
```




We can have interesting insights from this plot: - the biggest number of deaths in hospital is for people diagnosed in the category **genitourinary** - the lowest number of deaths in hospital (0 in the subset considered) is for people in **pregnancy**

### 1.3.4 ICUSTAYS table data extraction

The data in the ICUSTAYS table could be useful because indicates if a patient during an admission was in an ICU (Intensive Care Unit). This of course could be a factor that could determine the death of the patient at the hospital.

[149]:
```
mimic4_path = '../../mimic-iv-1.0/'

# read icustays table
def read_icustays_table(mimic4_path):
    icustays = pd.read_csv(mimic4_path + 'icu/icustays.csv')
    return icustays

icustays = read_icustays_table(mimic4_path)
icustays.head()
```

[149]:
```
   subject_id   hadm_id   stay_id       first_careunit        last_careunit  \
0    17867402  24528534  31793211  Trauma SICU (TSICU)  Trauma SICU (TSICU)
1    14435996  28960964  31983544  Trauma SICU (TSICU)  Trauma SICU (TSICU)
```

```
2    17609946  27385897  33183475  Trauma SICU (TSICU)  Trauma SICU (TSICU)
3    18966770  23483021  34131444  Trauma SICU (TSICU)  Trauma SICU (TSICU)
4    12776735  20817525  34547665       Neuro Stepdown       Neuro Stepdown

                intime             outtime       los
0  2154-03-03 04:11:00  2154-03-04 18:16:56  1.587454
1  2150-06-19 17:57:00  2150-06-22 18:33:54  3.025625
2  2138-02-05 18:54:00  2138-02-15 12:42:05  9.741725
3  2123-10-25 10:35:00  2123-10-25 18:59:47  0.350544
4  2200-07-12 00:33:00  2200-07-13 16:44:40  1.674769
```

[150]:
```
icustays['category'] = icustays['first_careunit']
icu_list = icustays.groupby('hadm_id')['category'].apply(list).reset_index()
icu_list.head()
```

[150]:
```
    hadm_id                                           category
0  20000094                        [Coronary Care Unit (CCU)]
1  20000147                        [Coronary Care Unit (CCU)]
2  20000351  [Medical/Surgical Intensive Care Unit (MICU/SI…
3  20000397                        [Coronary Care Unit (CCU)]
4  20000808  [Surgical Intensive Care Unit (SICU), Surgical…
```

[151]:
```
icustays['first_careunit'].value_counts()
```

[151]:
```
Medical Intensive Care Unit (MICU)                  16729
Medical/Surgical Intensive Care Unit (MICU/SICU)    13421
Cardiac Vascular Intensive Care Unit (CVICU)        12169
Surgical Intensive Care Unit (SICU)                 11765
Trauma SICU (TSICU)                                  9165
Coronary Care Unit (CCU)                             8746
Neuro Surgical Intensive Care Unit (Neuro SICU)      1851
Neuro Intermediate                                   1823
Neuro Stepdown                                        871
Name: first_careunit, dtype: int64
```

[152]:
```
# Create admission-ICU matrix
icu_item = pd.get_dummies(icu_list['category'].apply(pd.Series).stack()).
 ↪sum(level=0)
icu_item[icu_item >= 1] = 1
icu_item = icu_item.join(icu_list['hadm_id'], how="outer")
icu_item.head()
```

[152]:
```
   Cardiac Vascular Intensive Care Unit (CVICU)  Coronary Care Unit (CCU)  \
0                                             0                         1
1                                             0                         1
2                                             0                         0
3                                             0                         1
```

```
4                                                    0                              0

   Medical Intensive Care Unit (MICU)  \
0                                   0
1                                   0
2                                   0
3                                   0
4                                   0

   Medical/Surgical Intensive Care Unit (MICU/SICU)  Neuro Intermediate  \
0                                                 0                   0
1                                                 0                   0
2                                                 1                   0
3                                                 0                   0
4                                                 0                   0

   Neuro Stepdown  Neuro Surgical Intensive Care Unit (Neuro SICU)  \
0               0                                                0
1               0                                                0
2               0                                                0
3               0                                                0
4               0                                                0

   Surgical Intensive Care Unit (SICU)  Trauma SICU (TSICU)   hadm_id
0                                    0                    0  20000094
1                                    0                    0  20000147
2                                    0                    0  20000351
3                                    0                    0  20000397
4                                    1                    0  20000808
```

[153]:
```python
# Merge ICU data with main dataFrame
final_df = admits_patients_diag.merge(icu_item, how='outer', on='hadm_id')
final_df.head()
```

[153]:
```
   Unnamed: 0  subject_id   hadm_id           admittime           dischtime  \
0         0.0  14679932.0  21038362 2139-09-26 14:16:00 2139-09-28 11:30:00
1         1.0  15585972.0  24941086 2123-10-07 23:56:00 2123-10-12 11:22:00
2         2.0  15078341.0  23272159 2122-08-28 08:48:00 2122-08-30 12:32:00
3         3.0  17301855.0  29732723 2140-06-06 14:23:00 2140-06-08 14:25:00
4         4.0  17991012.0  24298836 2181-07-10 20:28:00 2181-07-12 15:49:00

  deathtime admission_type insurance               ethnicity  \
0       NaT       ELECTIVE     Other           OTHER/UNKNOWN
1       NaT       ELECTIVE     Other                   WHITE
2       NaT       ELECTIVE     Other  BLACK/AFRICAN AMERICAN
3       NaT       ELECTIVE     Other                   WHITE
4       NaT       ELECTIVE     Other                   WHITE
```

```
   died_at_the_hospital  …       los  \
0                   0.0  …  1.884722
1                   0.0  …  4.476389
2                   0.0  …  2.155556
3                   0.0  …  2.001389
4                   0.0  …  1.806250

  Cardiac Vascular Intensive Care Unit (CVICU) Coronary Care Unit (CCU)  \
0                                          NaN                      NaN
1                                          NaN                      NaN
2                                          NaN                      NaN
3                                          NaN                      NaN
4                                          NaN                      NaN

   Medical Intensive Care Unit (MICU)  \
0                                 NaN
1                                 NaN
2                                 NaN
3                                 NaN
4                                 NaN

   Medical/Surgical Intensive Care Unit (MICU/SICU)  Neuro Intermediate  \
0                                               NaN                 NaN
1                                               NaN                 NaN
2                                               NaN                 NaN
3                                               NaN                 NaN
4                                               NaN                 NaN

   Neuro Stepdown  Neuro Surgical Intensive Care Unit (Neuro SICU)  \
0             NaN                                              NaN
1             NaN                                              NaN
2             NaN                                              NaN
3             NaN                                              NaN
4             NaN                                              NaN

   Surgical Intensive Care Unit (SICU)  Trauma SICU (TSICU)
0                                  NaN                  NaN
1                                  NaN                  NaN
2                                  NaN                  NaN
3                                  NaN                  NaN
4                                  NaN                  NaN

[5 rows x 40 columns]
```

[154]:

```python
# Replace NaNs with 0
final_df['Cardiac Vascular Intensive Care Unit (CVICU)'].fillna(value=0,␣
 ↪inplace=True)
final_df['Coronary Care Unit (CCU)'].fillna(value=0, inplace=True)
final_df['Medical Intensive Care Unit (MICU)'].fillna(value=0, inplace=True)
final_df['Medical/Surgical Intensive Care Unit (MICU/SICU)'].fillna(value=0,␣
 ↪inplace=True)
final_df['Neuro Intermediate'].fillna(value=0, inplace=True)
final_df['Neuro Stepdown'].fillna(value=0, inplace=True)
final_df['Neuro Surgical Intensive Care Unit (Neuro SICU)'].fillna(value=0,␣
 ↪inplace=True)
final_df['Surgical Intensive Care Unit (SICU)'].fillna(value=0, inplace=True)
final_df['Trauma SICU (TSICU)'].fillna(value=0, inplace=True)
```

[155]:
```python
final_df.head()
```

[155]:
```
   Unnamed: 0  subject_id   hadm_id           admittime           dischtime  \
0         0.0  14679932.0  21038362 2139-09-26 14:16:00 2139-09-28 11:30:00
1         1.0  15585972.0  24941086 2123-10-07 23:56:00 2123-10-12 11:22:00
2         2.0  15078341.0  23272159 2122-08-28 08:48:00 2122-08-30 12:32:00
3         3.0  17301855.0  29732723 2140-06-06 14:23:00 2140-06-08 14:25:00
4         4.0  17991012.0  24298836 2181-07-10 20:28:00 2181-07-12 15:49:00

  deathtime admission_type insurance               ethnicity  \
0       NaT       ELECTIVE     Other           OTHER/UNKNOWN
1       NaT       ELECTIVE     Other                   WHITE
2       NaT       ELECTIVE     Other  BLACK/AFRICAN AMERICAN
3       NaT       ELECTIVE     Other                   WHITE
4       NaT       ELECTIVE     Other                   WHITE

   died_at_the_hospital  …       los  \
0                   0.0  …  1.884722
1                   0.0  …  4.476389
2                   0.0  …  2.155556
3                   0.0  …  2.001389
4                   0.0  …  1.806250

  Cardiac Vascular Intensive Care Unit (CVICU) Coronary Care Unit (CCU)  \
0                                          0.0                      0.0
1                                          0.0                      0.0
2                                          0.0                      0.0
3                                          0.0                      0.0
4                                          0.0                      0.0

   Medical Intensive Care Unit (MICU)  \
0                                 0.0
1                                 0.0
```

```
2                                  0.0
3                                  0.0
4                                  0.0

   Medical/Surgical Intensive Care Unit (MICU/SICU)  Neuro Intermediate  \
0                                               0.0                 0.0
1                                               0.0                 0.0
2                                               0.0                 0.0
3                                               0.0                 0.0
4                                               0.0                 0.0

   Neuro Stepdown  Neuro Surgical Intensive Care Unit (Neuro SICU)  \
0             0.0                                              0.0
1             0.0                                              0.0
2             0.0                                              0.0
3             0.0                                              0.0
4             0.0                                              0.0

   Surgical Intensive Care Unit (SICU)  Trauma SICU (TSICU)
0                                  0.0                  0.0
1                                  0.0                  0.0
2                                  0.0                  0.0
3                                  0.0                  0.0
4                                  0.0                  0.0

[5 rows x 40 columns]
```

## 1.4 5.Data Cleaning

[156]:
```
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 362995 entries, 0 to 362994
Data columns (total 40 columns):
 #   Column                                             Non-Null Count   Dtype
---  ------                                             --------------   -----
 0   Unnamed: 0                                         335257 non-null  float64
 1   subject_id                                         335257 non-null  float64
 2   hadm_id                                            362995 non-null  int64
 3   admittime                                          335257 non-null
datetime64[ns]
 4   dischtime                                          335257 non-null
datetime64[ns]
 5   deathtime                                          5605 non-null
datetime64[ns]
 6   admission_type                                     335257 non-null  object
 7   insurance                                          335257 non-null  object
 8   ethnicity                                          335257 non-null  object
```

```
 9   died_at_the_hospital                                335257 non-null  float64
 10  gender                                              335257 non-null  object
 11  anchor_age                                          335257 non-null  object
 12  dod                                                 24581 non-null   object
 13  blood                                               335257 non-null  float64
 14  circulatory                                         335257 non-null  float64
 15  congenital                                          335257 non-null  float64
 16  digestive                                           335257 non-null  float64
 17  endocrine                                           335257 non-null  float64
 18  genitourinary                                       335257 non-null  float64
 19  infectious                                          335257 non-null  float64
 20  injury                                              335257 non-null  float64
 21  mental                                              335257 non-null  float64
 22  misc                                                335257 non-null  float64
 23  muscular                                            335257 non-null  float64
 24  neoplasms                                           335257 non-null  float64
 25  nervous                                             335257 non-null  float64
 26  pregnancy                                           335257 non-null  float64
 27  prenatal                                            335257 non-null  float64
 28  respiratory                                         335257 non-null  float64
 29  skin                                                335257 non-null  float64
 30  los                                                 335257 non-null  float64
 31  Cardiac Vascular Intensive Care Unit (CVICU)        362995 non-null  float64
 32  Coronary Care Unit (CCU)                            362995 non-null  float64
 33  Medical Intensive Care Unit (MICU)                  362995 non-null  float64
 34  Medical/Surgical Intensive Care Unit (MICU/SICU)    362995 non-null  float64
 35  Neuro Intermediate                                  362995 non-null  float64
 36  Neuro Stepdown                                      362995 non-null  float64
 37  Neuro Surgical Intensive Care Unit (Neuro SICU)     362995 non-null  float64
 38  Surgical Intensive Care Unit (SICU)                 362995 non-null  float64
 39  Trauma SICU (TSICU)                                 362995 non-null  float64
dtypes: datetime64[ns](3), float64(30), int64(1), object(6)
memory usage: 113.5+ MB
```

[157]:
```
# Drop unused or no longer needed columns
final_df.drop(columns=['Unnamed: 0', 'subject_id', 'hadm_id', 'admittime',␣
 ↪'dischtime', 'deathtime','dod', 'los'], inplace=True)

final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 362995 entries, 0 to 362994
Data columns (total 32 columns):
 #   Column                                              Non-Null Count   Dtype
---  ------                                              --------------   -----
 0   admission_type                                      335257 non-null  object
 1   insurance                                           335257 non-null  object
 2   ethnicity                                           335257 non-null  object
```

```
 3   died_at_the_hospital                               335257 non-null  float64
 4   gender                                             335257 non-null  object
 5   anchor_age                                         335257 non-null  object
 6   blood                                              335257 non-null  float64
 7   circulatory                                        335257 non-null  float64
 8   congenital                                         335257 non-null  float64
 9   digestive                                          335257 non-null  float64
 10  endocrine                                          335257 non-null  float64
 11  genitourinary                                      335257 non-null  float64
 12  infectious                                         335257 non-null  float64
 13  injury                                             335257 non-null  float64
 14  mental                                             335257 non-null  float64
 15  misc                                               335257 non-null  float64
 16  muscular                                           335257 non-null  float64
 17  neoplasms                                          335257 non-null  float64
 18  nervous                                            335257 non-null  float64
 19  pregnancy                                          335257 non-null  float64
 20  prenatal                                           335257 non-null  float64
 21  respiratory                                        335257 non-null  float64
 22  skin                                               335257 non-null  float64
 23  Cardiac Vascular Intensive Care Unit (CVICU)       362995 non-null  float64
 24  Coronary Care Unit (CCU)                           362995 non-null  float64
 25  Medical Intensive Care Unit (MICU)                 362995 non-null  float64
 26  Medical/Surgical Intensive Care Unit (MICU/SICU)   362995 non-null  float64
 27  Neuro Intermediate                                 362995 non-null  float64
 28  Neuro Stepdown                                     362995 non-null  float64
 29  Neuro Surgical Intensive Care Unit (Neuro SICU)    362995 non-null  float64
 30  Surgical Intensive Care Unit (SICU)                362995 non-null  float64
 31  Trauma SICU (TSICU)                                362995 non-null  float64
dtypes: float64(27), object(5)
memory usage: 91.4+ MB
```

[160]:
```
# Create dummy columns for categorical variables
prefix_cols = ['ADM', 'INS', 'ETH', 'AGE']
dummy_cols = ['admission_type', 'insurance','ethnicity', 'anchor_age']
df_cleaned = pd.get_dummies(final_df, prefix=prefix_cols, columns=dummy_cols)
df_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 362995 entries, 0 to 362994
Data columns (total 44 columns):
 #   Column                                             Non-Null Count   Dtype
---  ------                                             --------------   -----
 0   died_at_the_hospital                               335257 non-null  float64
 1   gender                                             335257 non-null  object
 2   blood                                              335257 non-null  float64
 3   circulatory                                        335257 non-null  float64
 4   congenital                                         335257 non-null  float64
```

```
 5   digestive                                        335257 non-null  float64
 6   endocrine                                        335257 non-null  float64
 7   genitourinary                                    335257 non-null  float64
 8   infectious                                       335257 non-null  float64
 9   injury                                           335257 non-null  float64
 10  mental                                           335257 non-null  float64
 11  misc                                             335257 non-null  float64
 12  muscular                                         335257 non-null  float64
 13  neoplasms                                        335257 non-null  float64
 14  nervous                                          335257 non-null  float64
 15  pregnancy                                        335257 non-null  float64
 16  prenatal                                         335257 non-null  float64
 17  respiratory                                      335257 non-null  float64
 18  skin                                             335257 non-null  float64
 19  Cardiac Vascular Intensive Care Unit (CVICU)     362995 non-null  float64
 20  Coronary Care Unit (CCU)                         362995 non-null  float64
 21  Medical Intensive Care Unit (MICU)               362995 non-null  float64
 22  Medical/Surgical Intensive Care Unit (MICU/SICU) 362995 non-null  float64
 23  Neuro Intermediate                               362995 non-null  float64
 24  Neuro Stepdown                                   362995 non-null  float64
 25  Neuro Surgical Intensive Care Unit (Neuro SICU)  362995 non-null  float64
 26  Surgical Intensive Care Unit (SICU)              362995 non-null  float64
 27  Trauma SICU (TSICU)                              362995 non-null  float64
 28  ADM_ELECTIVE                                     362995 non-null  uint8
 29  ADM_EMERGENCY                                    362995 non-null  uint8
 30  ADM_OBSERVATION                                  362995 non-null  uint8
 31  ADM_SURGICAL SAME DAY ADMISSION                  362995 non-null  uint8
 32  INS_Medicaid                                     362995 non-null  uint8
 33  INS_Medicare                                     362995 non-null  uint8
 34  INS_Other                                        362995 non-null  uint8
 35  ETH_ASIAN                                        362995 non-null  uint8
 36  ETH_BLACK/AFRICAN AMERICAN                       362995 non-null  uint8
 37  ETH_HISPANIC/LATINO                              362995 non-null  uint8
 38  ETH_OTHER/UNKNOWN                                362995 non-null  uint8
 39  ETH_WHITE                                        362995 non-null  uint8
 40  AGE_MIDDLE_ADULT                                 362995 non-null  uint8
 41  AGE_NEWBORN                                      362995 non-null  uint8
 42  AGE_SENIOR                                       362995 non-null  uint8
 43  AGE_YOUNG_ADULT                                  362995 non-null  uint8
dtypes: float64(27), object(1), uint8(16)
memory usage: 85.9+ MB
```

[172]:
```
# Drop rows that contain NaN values
df_cleaned.dropna(axis=0, inplace=True)
```

[175]:
```
# Check for any remaining NaNs
df_cleaned.isnull().values.sum()
```

[175]: 0

[173]:
```
df_cleaned.isnull().sum()
```

[173]:
```
died_at_the_hospital                                  0
gender                                                0
blood                                                 0
circulatory                                           0
congenital                                            0
digestive                                             0
endocrine                                             0
genitourinary                                         0
infectious                                            0
injury                                                0
mental                                                0
misc                                                  0
muscular                                              0
neoplasms                                             0
nervous                                               0
pregnancy                                             0
prenatal                                              0
respiratory                                           0
skin                                                  0
Cardiac Vascular Intensive Care Unit (CVICU)          0
Coronary Care Unit (CCU)                              0
Medical Intensive Care Unit (MICU)                    0
Medical/Surgical Intensive Care Unit (MICU/SICU)      0
Neuro Intermediate                                    0
Neuro Stepdown                                        0
Neuro Surgical Intensive Care Unit (Neuro SICU)       0
Surgical Intensive Care Unit (SICU)                   0
Trauma SICU (TSICU)                                   0
ADM_ELECTIVE                                          0
ADM_EMERGENCY                                         0
ADM_OBSERVATION                                       0
ADM_SURGICAL SAME DAY ADMISSION                       0
INS_Medicaid                                          0
INS_Medicare                                          0
INS_Other                                             0
ETH_ASIAN                                             0
ETH_BLACK/AFRICAN AMERICAN                            0
ETH_HISPANIC/LATINO                                   0
ETH_OTHER/UNKNOWN                                     0
ETH_WHITE                                             0
AGE_MIDDLE_ADULT                                      0
AGE_NEWBORN                                           0
AGE_SENIOR                                            0
```

```
AGE_YOUNG_ADULT                                         0
dtype: int64
```

[176]:
```
# Convert gender into numeric boolean attribute
df_cleaned['gender'].replace({'M': 0, 'F':1}, inplace=True)
df_cleaned.head()
```

[176]:
```
   died_at_the_hospital  gender  blood  circulatory  congenital  digestive  \
0                   0.0       1    0.0          0.0         1.0        0.0
1                   0.0       1    0.0          0.0         0.0        0.0
2                   0.0       0    0.0          0.0         0.0        0.0
3                   0.0       1    0.0          0.0         0.0        0.0
4                   0.0       0    0.0          0.0         0.0        0.0

   endocrine  genitourinary  infectious  injury  …  INS_Other  ETH_ASIAN  \
0        0.0            0.0         0.0     2.0  …          1          0
1        0.0            0.0         0.0     2.0  …          1          0
2        0.0            0.0         0.0     3.0  …          1          0
3        0.0            0.0         0.0     2.0  …          1          0
4        0.0            0.0         0.0     2.0  …          1          0

   ETH_BLACK/AFRICAN AMERICAN  ETH_HISPANIC/LATINO  ETH_OTHER/UNKNOWN  \
0                           0                    0                  1
1                           0                    0                  0
2                           1                    0                  0
3                           0                    0                  0
4                           0                    0                  0

   ETH_WHITE  AGE_MIDDLE_ADULT  AGE_NEWBORN  AGE_SENIOR  AGE_YOUNG_ADULT
0          0                 0            1           0                0
1          1                 0            1           0                0
2          0                 0            1           0                0
3          1                 0            1           0                0
4          1                 0            1           0                0

[5 rows x 44 columns]
```

[177]:
```
df_cleaned = df_cleaned.astype(int)
```

[178]:
```
# Check for any remaining NaNs
df_cleaned.isnull().values.sum()
```

[178]:
```
0
```

## 1.5 6. Prediction Model

We use a **Supervised Learning ML model**. First of all what is it? Supervised learning is defined by its use of labeled datasets to train algorithms that to classify data or predict outcomes accurately.

It uses a training set to teach models to yield the desired output. This training dataset includes inputs and correct outputs, which allow the model to learn over time. The algorithm measures its accuracy through the loss function, adjusting until the error has been sufficiently minimized.

**Why do we choose it?** Because in our case we have the corret output for each dataset entry: "Died_at_the_Hospital" (Yes or No) and we want to create a model that predicts this output for new entries, in other words that it "generalize well".

We will implement the supervised learning prediction model using the **Scikit-Learn** machine learning library.

To implement the prediction model, our dataset is splitted into training and test sets at an 80:20 ratio using the scikit-learn *train_test_split* function.

**Why split in training and test set?** Because to detect a machine learning model behavior, we need to use observations that aren't used in the training process. Otherwise, the evaluation of the model would be biased as a matter of fact when we build a predictive model, we want the model to work well on data that the model has never seen, so that's the reason why we use a training set to train the model and a test set to evaulate the model accuaracy.

Searching on the Internet for the best train-test ratio, the first answer is 80:20. This means we use 80% of the observations for training and the rest for testing. This approach is taken in this case. zability)

[179]:
```
# Target Variable (died_at_the_hospital)
HOSP_MORT = df_cleaned['died_at_the_hospital'].values
# Prediction Features
features = df_cleaned.drop(columns=['died_at_the_hospital'])
```

[183]:
```
from sklearn.model_selection import train_test_split

# Split into training set 80% and test set 20%
X_train, X_test, y_train, y_test = train_test_split(features,
                                                    HOSP_MORT,
                                                    test_size = .20,
                                                    random_state = 0)

# Show the results of the split
print("Training set has {} samples.".format(X_train.shape[0]))
print("Testing set has {} samples.".format(X_test.shape[0]))
```

```
Training set has 268205 samples.
Testing set has 67052 samples.
```

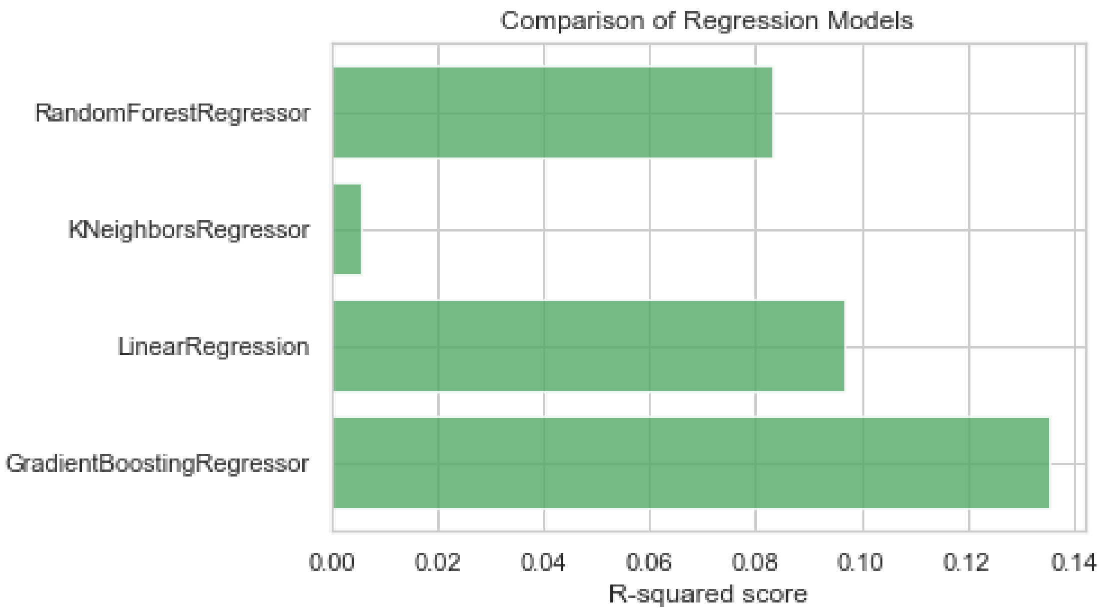
Using the training set, we'll four five different classification models (from the scikit-learn library) using default settings.

[186]:
```
from sklearn.metrics import r2_score
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
```

```python
from sklearn.ensemble import GradientBoostingRegressor

# Regression models used from scikit-learn for comparison
models = [GradientBoostingRegressor(random_state = 0),
          LinearRegression(),
          KNeighborsRegressor(),
          RandomForestRegressor(random_state = 0)]

results = {}

for model in models:
    # Instantiate and fit Regressor Model
    reg_model = model
    reg_model.fit(X_train, y_train)

    # Make predictions with model
    y_test_preds = reg_model.predict(X_test)

    # Grab model name and store results associated with model
    name = str(model).split("(")[0]

    results[name] = r2_score(y_test, y_test_preds)
    print('{} done.'.format(name))
```

```
GradientBoostingRegressor done.
LinearRegression done.
KNeighborsRegressor done.
RandomForestRegressor done.
```

[203]:
```python
# R2 score results
fig, ax = plt.subplots()
ind = range(len(results))
ax.barh(ind, list(results.values()), align='center',
        color = '#55a868', alpha=0.8)
ax.set_yticks(ind)
ax.set_yticklabels(results.keys())
ax.set_xlabel('R-squared score')
ax.tick_params(left=False, top=False, right=False)
ax.set_title('Comparison of Regression Models')
fig.savefig('images/compare_models_dinh_mimic4.png', bbox_inches = 'tight')
```




[192]:
```
# GradientBoostingRegressor will be used as the In-Hospital mortality␣
↪prediction model
# GradientBoostingRegressor will be used as the LOS prediction model
reg_model = GradientBoostingRegressor(random_state=0)
reg_model.fit(X_train, y_train)
y_test_preds = reg_model.predict(X_test)
r2_not_refined = r2_score(y_test, y_test_preds)
print("R2 score is: {:2f}".format(r2_not_refined))
```

```
R2 score is: 0.135291
```

The GradientBoostingRegressor has the best R2 score of ~13% so we focus on refining this particular model.

## 1.6 7. Parameter Tuning

To refine the GradientBoostingRegressor model, **GridSearchCV** function from scikit-learn is used to test out various permutations of parameters such as *n_estimators, max_depth, and loss.* It helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we could select the best parameters from the listed hyperparameters.

[197]:
```
from sklearn.model_selection import GridSearchCV

# Split into train 80% and test 20%
X_train, X_test, y_train, y_test = train_test_split(features,
                                                    HOSP_MORT,
                                                    test_size = .20,
                                                    random_state = 42)
```

```
# Set the parameters by cross-validation
#tuned_parameters = [{'n_estimators': [100, 200, 300],
#                     'max_depth' : [2, 3, 4],
#                     'loss': ['ls', 'lad', 'huber']}]
tuned_parameters = [{'n_estimators': [200, 300],
                     'max_depth' : [3, 4],
                     'loss': ['ls', 'lad']}]

# create and fit a ridge regression model, testing each alpha
reg_model = GradientBoostingRegressor()
grid = GridSearchCV(reg_model, tuned_parameters, cv=3, verbose = 1, n_jobs = -1)
grid.fit(X_train, y_train)
reg_model_optimized = grid.best_estimator_

# summarize the results of the grid search
print(grid.best_score_)
print(grid.best_estimator_)
```

```
Fitting 3 folds for each of 8 candidates, totalling 24 fits
0.14951616540043464
GradientBoostingRegressor(max_depth=4, n_estimators=200)
```

**Tuned Paramters** - *n_estimators*: The number of boosting stages to perform. - *max_depth*: maximum depth of the individual regression estimators. The maximum depth limits the number of nodes in the tree. - *loss*: loss function to be optimized. 'ls' refers to least squares regression. 'lad' (least absolute deviation) is a highly robust loss function solely based on order information of the input variables. 'huber' is a combination of the two.

The best estimator result from GridSearchCV was n_estimators=200, max_depth=4, loss = ls.

[198]:
```
y_test_preds = reg_model_optimized.predict(X_test)
r2_optimized = r2_score(y_test, y_test_preds)
print("Optimized R2 score is: {:2f}".format(r2_optimized))
```

```
Optimized R2 score is: 0.143552
```

[199]:
```
print('Parameter tuning improved R2 score by {:.4f}'.
 ↪format(r2_optimized-r2_not_refined))
```

```
Parameter tuning improved R2 score by 0.0083
```

### 1.7 8. Model evaluation and result Discussion

We could look at what features were most important in predicting in-hospital mortality when using the gradient boosting regression model.

[200]:
```
feature_imp = pd.DataFrame(reg_model_optimized.feature_importances_,
                           index = X_train.columns,
```

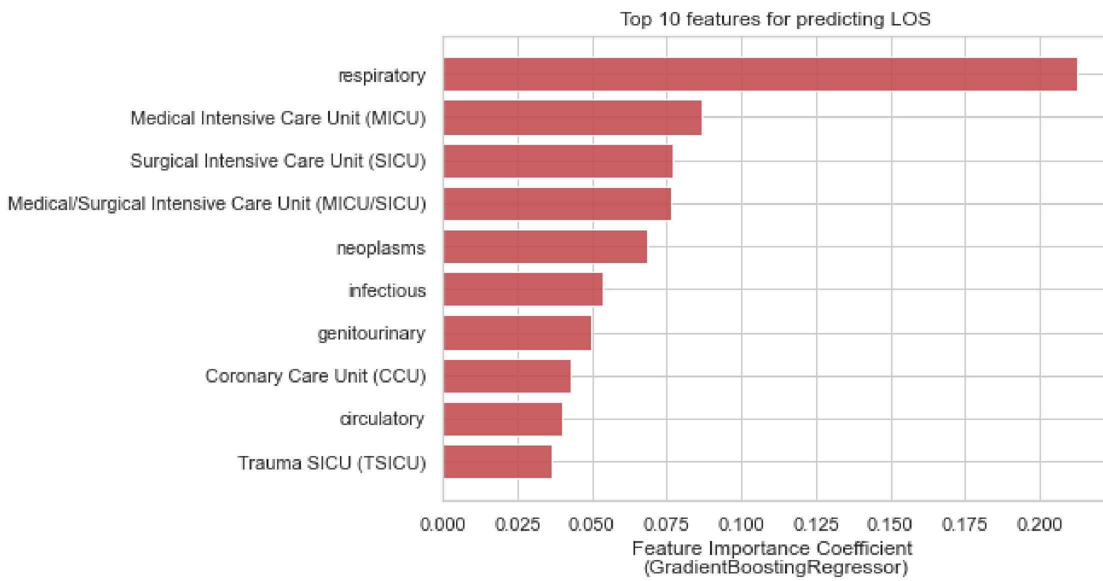
```
                                              columns=['importance']).
↪sort_values('importance', ascending=False)

feature_imp.head(20)
```

[200]:

```
                                                  importance
respiratory                                         0.212500
Medical Intensive Care Unit (MICU)                  0.086784
Surgical Intensive Care Unit (SICU)                 0.076962
Medical/Surgical Intensive Care Unit (MICU/SICU)    0.076263
neoplasms                                           0.068504
infectious                                          0.053728
genitourinary                                       0.049491
Coronary Care Unit (CCU)                            0.042526
circulatory                                         0.040123
Trauma SICU (TSICU)                                 0.036257
misc                                                0.033244
digestive                                           0.028575
blood                                               0.024942
injury                                              0.022731
ETH_OTHER/UNKNOWN                                   0.021645
nervous                                             0.019831
AGE_SENIOR                                          0.019813
ADM_EMERGENCY                                       0.015731
mental                                              0.011016
endocrine                                           0.008235
```

[202]:

```
#Let's plot the top-10 feature importance
feature_imp.index[0:10].tolist()
# Plot feature importance
fig, ax = plt.subplots(figsize=(7, 5))
ind = range(0,10)
ax.barh(ind, feature_imp['importance'].values[0:10],
        align='center', color='#c44e52', alpha=0.9)
ax.set_yticks(ind)
ax.set_yticklabels(feature_imp.index[0:10].tolist())
ax.tick_params(left=False, top=False, right=False)
ax.set_title("Top 10 features for predicting LOS")
ax.set_xlabel('Feature Importance Coefficient \n(GradientBoostingRegressor)')
plt.gca().invert_yaxis()
fig.savefig('images/feature_importance_dinh_mimic4.png', bbox_inches = 'tight')
```




- We could say that, first of all, one of the results is that the *ICD-9 diagnoses categories* and the admission to various type of ICU, are by far the most important features between the features analyzed.
- We could notice how the diagnose belongig to **respiratory** category is the most important feature in determining **in-hospital-mortality** followe by the admission of the patient to MICU, SICU or both.

Compute now other metric used for validation RMSE.

```
[ ]: for i in range(y_test_preds.shape[0]):
         ml_model = abs(y_test_preds[i] - y_test[i])

         ml_model_rms = (y_test_preds[i] - y_test[i])**2

     print("Prediction Model RMS {}".format((ml_model_rms**0.5)/y_test_preds.
      ↪shape[0]))
```

### 1.8 Conclusions for In-Hospital Mortality

The development of methods for prediction of mortality rates in populations has been motivated primarily by the need to compare the efficacy of medications, care guidelines, surgery, and other interventions when, as is common, it is necessary to control for differences in diagnoses, age, and other factors.

The prediction model achieved should be a “friend” of the doctor and not a substitute. In this case the model could help the doctor to adopt certain behaviors or to carry out different interventions depending on the expected probability of mortality in hospital.

For example, for a patient who has a high probability of death during his hospital stay, the doctor may consider adopting some surgical interventions right away without waiting for further tests and therefore reduce the expected probability of death.

Future Developments in this area could be analyzing other features of mimic dataset or also other datsets, for example the notes of doctors using Natural Language Processing and trying to have insights.