02_length_of_stay

July 19, 2021

Nicola De Cristofaro (Matr. 0522500876)   Cloud Computing Curriculum

1 Hospital LOS (Length-of-Stay)

First of all what is LOS? **Hospital length-of-stay (LOS)** is defined as the time between hospital admission and discharge measured in days.

1.1 1. Problem Statement

The goal is to create a model that predicts the length-of-stay for each patient at time of admission.

In order to predict hospital LOS, the MIMIC data needed to be separated into terms of: - dependent target variable (length-of-stay in this case) - and independent variables (features) to be used as inputs to the model.

1.2 2. Type of model used for prediction

Since LOS is not a categorical but continuous variable (measured in days), a **regression model** will be used for prediction.

1.3 3. Metrics used for validation

The expected outcome is that the model we use will be better at predicting hospital LOS than the industry standards of **median and average LOS**. The median LOS is simply the median LOS of past admissions to a hospital. Similarly, a second commonly used metric in healthcare is the average, or mean LOS.

So, to measure performance of our model, we'll compare the prediction model against the median and average LOS using 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 LOS, and (y) is the actual LOS.

We could say we have a successful model if its prediction results in a lower RMSE than the average or median models.

There is a multitude of regression models available for predicting LOS. 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 and a negative value means it is worse than a constant model, average or median in this case.

1.4 4. Features distribution and features engineering

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.linear_model import SGDRegressor
from sklearn.model_selection import GridSearchCV
```

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

[4]:
```
# 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()
```

[4]:
```
   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]
```

Length of stays computation The LOS is not explicitly expressed as attribute in the admission table, so we have to calculate it. As we said, LOS is defined as the time between admission and discharge from the hospital.

[5]:
```
# 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()
```

[5]:
```
            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
```

[7]:
```
# We could already have a quick insight on how LOS values are distributed
admits_patients_diag['los'].describe()
```

[7]:
```
count    335378.000000
mean          4.257902
std           7.223969
min          -0.945139
```

```
25%            1.129861
50%            2.542361
75%            4.730556
max         1191.416667
Name: los, dtype: float64
```

We noticed that the mean LOS is 4 days, but we noticed also that the min LOS calculated is a negative value, how is it possible that a LOS is negative? Let's see records associated with negative values of LOS:

[9]:
```
admits_patients_diag[admits_patients_diag['los'] < 0]
```

[9]:
```
        Unnamed: 0  subject_id   hadm_id           admittime  \
2359          2359    14556829  27223222 2160-01-04 20:43:00
3416          3416    13362952  25586069 2173-05-02 18:50:00
5927          5927    19649539  20159343 2117-05-27 22:04:00
14137        14137    13604937  22022786 2193-02-16 22:20:00
18846        18846    14316510  27404352 2177-07-26 15:33:00
…             …          …        …                …
318993      318993    17766453  27023395 2180-07-24 20:51:00
319157      319157    13659453  23252384 2145-05-20 05:08:00
322903      322903    13316652  22658929 2130-05-11 14:55:00
323506      323506    15838787  29390236 2137-07-23 16:58:00
327732      327732    13535122  21247013 2177-08-15 11:56:00

                 dischtime           deathtime admission_type insurance  \
2359   2160-01-04 01:50:00 2160-01-04 01:50:00       ELECTIVE  Medicaid
3416   2173-05-02 09:06:00 2173-05-02 09:06:00       ELECTIVE     Other
5927   2117-05-27 21:16:00                 NaT    OBSERVATION     Other
14137  2193-02-16 00:01:00 2193-02-16 00:01:00      EMERGENCY  Medicare
18846  2177-07-26 00:04:00                 NaT    OBSERVATION     Other
…                        …                   …              …         …
318993 2180-07-24 20:50:00                 NaT    OBSERVATION  Medicare
319157 2145-05-20 02:20:00                 NaT      EMERGENCY     Other
322903 2130-05-11 02:30:00 2130-05-11 22:23:00      EMERGENCY     Other
323506 2137-07-23 00:01:00 2137-07-23 23:09:00      EMERGENCY     Other
327732 2177-08-15 01:00:00 2177-08-15 01:00:00      EMERGENCY     Other

                     ethnicity  died_at_the_hospital  … mental  misc  \
2359    BLACK/AFRICAN AMERICAN                     1  …      0     0
3416                     WHITE                     1  …      0     0
5927                     WHITE                     0  …      1     0
14137                    WHITE                     1  …      0     0
18846                    WHITE                     0  …      0     1
…                          …                    …  …    …    …
318993                   WHITE                     0  …      0     1
319157                   WHITE                     0  …      1     0
322903                   WHITE                     1  …      0     1
```

```
323506                    WHITE                         1  …       0     0
327732                    ASIAN                         1  …       0     2

       muscular  neoplasms  nervous  pregnancy  prenatal  respiratory  skin  \
2359          0          0        0          0         2            0     0
3416          0          0        0          0         4            0     0
5927          0          0        0          0         0            0     0
14137         0          2        0          0         0            0     0
18846         0          0        0          0         0            0     0
…          …        …       …         …        …       …   …
318993        1          0        0          0         1            0     0
319157        0          0        0          0         0            0     0
322903        0          0        4          0         0            0     0
323506        3          0        3          0         0            1     0
327732        0          2        0          0         0            2     0

             los
2359   -0.786806
3416   -0.405556
5927   -0.033333
14137  -0.929861
18846  -0.645139
…            …
318993 -0.000694
319157 -0.116667
322903 -0.517361
323506 -0.706250
327732 -0.455556

[116 rows x 31 columns]
```

We noticed that rows with negative LOS, usually are related to a time of death before admission, so in this case there is no use to predict LOS, so we drop these rows.

[10]:
```
admits_patients_diag = admits_patients_diag[admits_patients_diag['los'] > 0]
admits_patients_diag.describe()
```

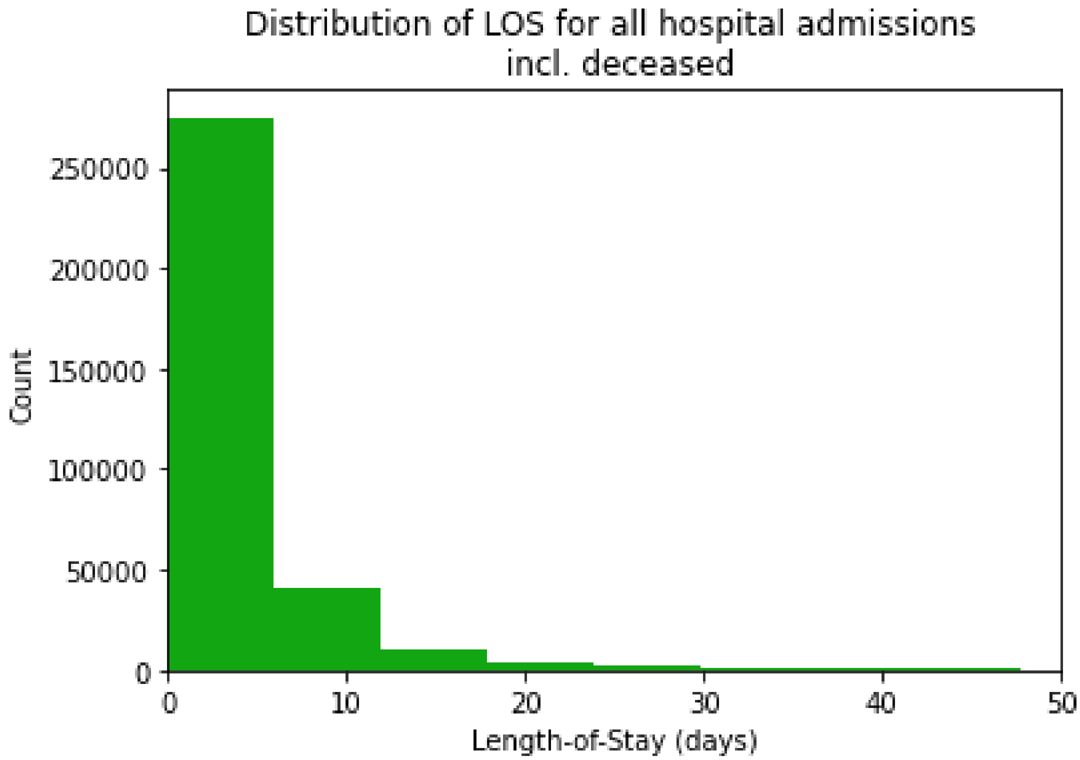
[10]:
```
          Unnamed: 0    subject_id       hadm_id  died_at_the_hospital  \
count  335257.000000  3.352570e+05  3.352570e+05         335257.000000
mean   167692.265262  1.500596e+07  2.500376e+07              0.016719
std     96815.453052  2.882620e+06  2.888682e+06              0.128215
min         0.000000  1.000002e+07  2.000002e+07              0.000000
25%     83845.000000  1.251288e+07  2.250434e+07              0.000000
50%    167697.000000  1.500984e+07  2.500010e+07              0.000000
75%    251536.000000  1.749618e+07  2.750696e+07              0.000000
max    335377.000000  1.999999e+07  2.999983e+07              1.000000

          anchor_age          blood    circulatory     congenital  \
```

```
count  335257.000000  335257.000000  335257.000000  335257.000000
mean       50.373036       0.270467       1.455501       0.040459
std        25.542069       0.580229       1.970275       0.239368
min         0.000000       0.000000       0.000000       0.000000
25%        34.000000       0.000000       0.000000       0.000000
50%        54.000000       0.000000       1.000000       0.000000
75%        70.000000       0.000000       2.000000       0.000000
max        91.000000       7.000000      17.000000      11.000000

           digestive      endocrine  …         mental           misc  \
count  335257.000000  335257.000000  …  335257.000000  335257.000000
mean        0.586338       1.041777  …       0.597342       0.513469
std         1.037813       1.325580  …       1.027394       0.885857
min         0.000000       0.000000  …       0.000000       0.000000
25%         0.000000       0.000000  …       0.000000       0.000000
50%         0.000000       1.000000  …       0.000000       0.000000
75%         1.000000       2.000000  …       1.000000       1.000000
max        12.000000      12.000000  …      14.000000      13.000000

            muscular      neoplasms        nervous      pregnancy  \
count  335257.000000  335257.000000  335257.000000  335257.000000
mean        0.346093       0.221484       0.425918       0.140319
std         0.720684       0.649399       0.795117       0.774798
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.000000       0.000000       0.000000       0.000000
75%         0.000000       0.000000       1.000000       0.000000
max        10.000000      11.000000       9.000000      19.000000

            prenatal    respiratory           skin            los
count  335257.000000  335257.000000  335257.000000  335257.000000
mean        0.237507       0.332038       0.140233       4.259560
std         0.815016       0.705849       0.484663       7.224743
min         0.000000       0.000000       0.000000       0.000694
25%         0.000000       0.000000       0.000000       1.131250
50%         0.000000       0.000000       0.000000       2.543750
75%         0.000000       0.000000       0.000000       4.731944
max        17.000000       9.000000       9.000000    1191.416667

[8 rows x 23 columns]
```

Now we see how the min value for LOS is not negative anymore. To have a more informative view on the distribution of LOS values we plot those values:

[11]:
```python
# Plot LOS Distribution
plt.hist(admits_patients_diag['los'], bins=200, color = '#11a612')
plt.xlim(0, 50)
```

```
plt.title('Distribution of LOS for all hospital admissions \n incl. deceased')
plt.ylabel('Count')
plt.xlabel('Length-of-Stay (days)')
plt.tick_params(top=False, right=False)
plt.show();
```



Another thing to consider is admissions of patients who died at the hospital. This kind of admissions resulting in death will be excluded as they would bias the LOS since LOS would be shorter for this group (in data cleaning process this group will be dropped).

[12]:
```
print("{} of {} patients died at the hospital".
 ↪format(admits_patients_diag['died_at_the_hospital'].sum(),␣
 ↪admits_patients_diag['subject_id'].nunique()))
```

```
5605 of 172978 patients died at the hospital
```

We also said that we'll use the LOS mean and median for comparison and for understand the accuracy of our model. So let's compute these LOS metrics that we'll use later for model evalutaion.

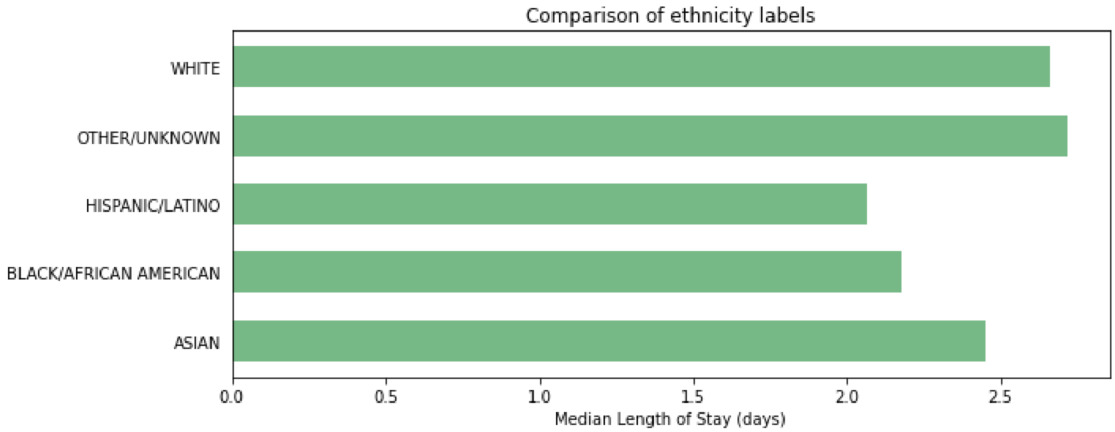
[13]:
```
# Hospital LOS metrics for later comparison
actual_mean_los = admits_patients_diag['los'].
 ↪loc[admits_patients_diag['died_at_the_hospital'] == 0].mean()
```

```
actual_median_los = admits_patients_diag['los'].
↪loc[admits_patients_diag['died_at_the_hospital'] == 0].median()

print(actual_mean_los)
print(actual_median_los)
```

```
4.171430522439771
2.5236111111111112
```

[ ]:
```
#### Ethnicity attribute
```

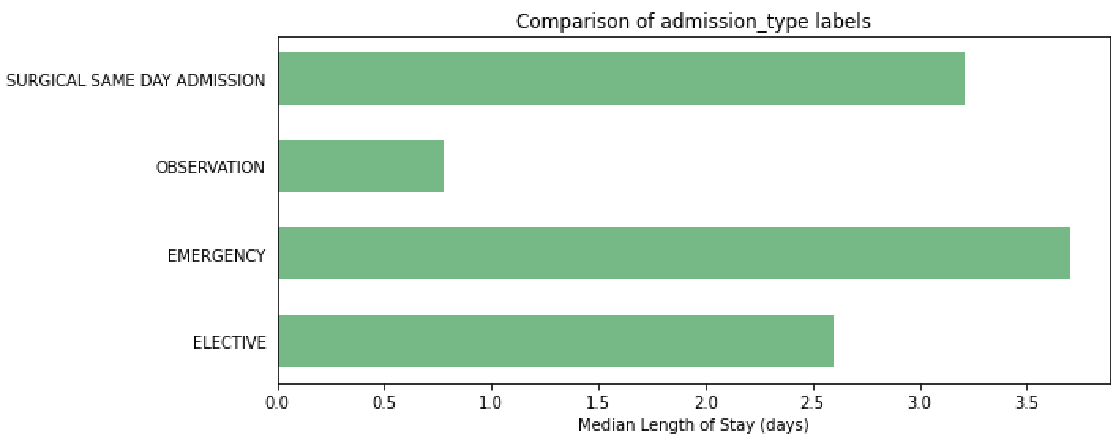
[14]:
```
# Re-usable plotting function
def plot_los_groupby(variable, size=(7,4)):
    '''
    Plot Median LOS by dataframe categorical series name
    '''
    results = admits_patients_diag[[variable, 'los']].groupby(variable).
↪median().reset_index()
    values = list(results['los'].values)
    labels = 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('Median Length of Stay (days)')
    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))
```



To notice that Hispanic/latino patients have the lowest median LOS, even if they are smaller in number in comparison to other ETHNICITY categories.

[ ]:
```
#### ADMISSION_TYPE attribute
```

[15]:
```
# Look at median LOS for groups ADMISSION_TYPE
plot_los_groupby('admission_type', size=(10,4))
```



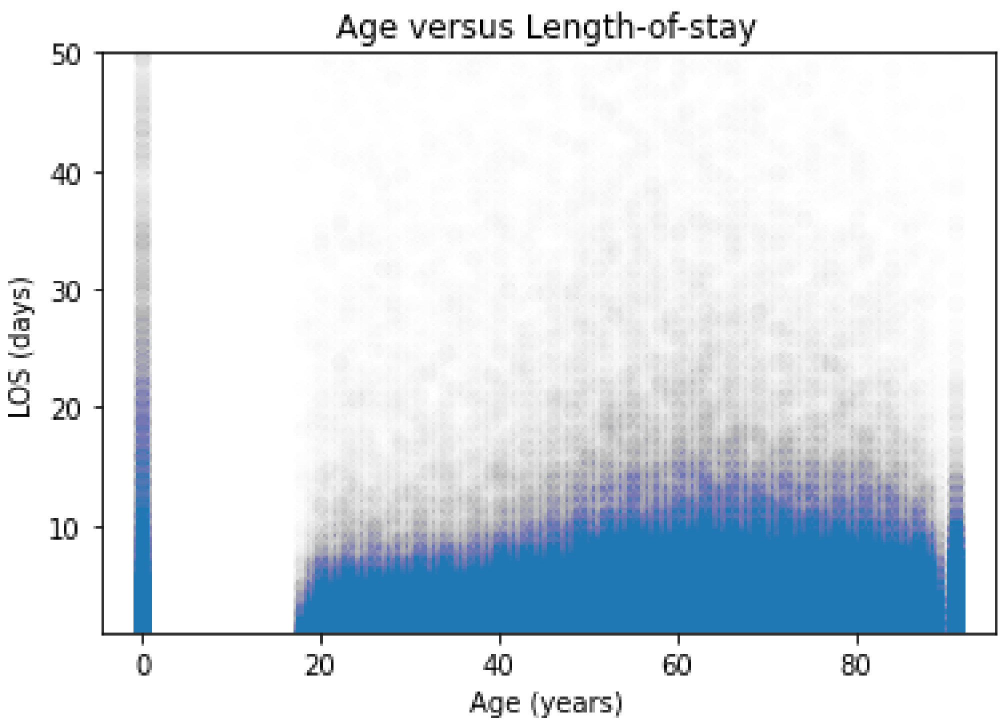
As we could expected *observation* and *elective* admissions have the lowest LOS. This is expected since these are often somewhat planned for and with the risks being understood in comparison to EMERGENCY ADMISSION_TYPE.

AGE attribute Now let's see how the LOS, our current goal, is correlated to ther age of the patients.

[16]:

```
plt.scatter(admits_patients_diag['anchor_age'], admits_patients_diag['los'],␣
 ↪alpha=0.005)
plt.ylabel('LOS (days)')
plt.xlabel('Age (years)')
plt.title('Age versus Length-of-stay')
plt.ylim(1, 50)
```

[16]: (1.0, 50.0)



The plot highlights the MIMIC groups of newborns and >89 year olds have higher LOS, and there is an increasing LOS going from 20 toward 80 years old. Because of the discrete-like distribution of data on the extremes of age, 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.

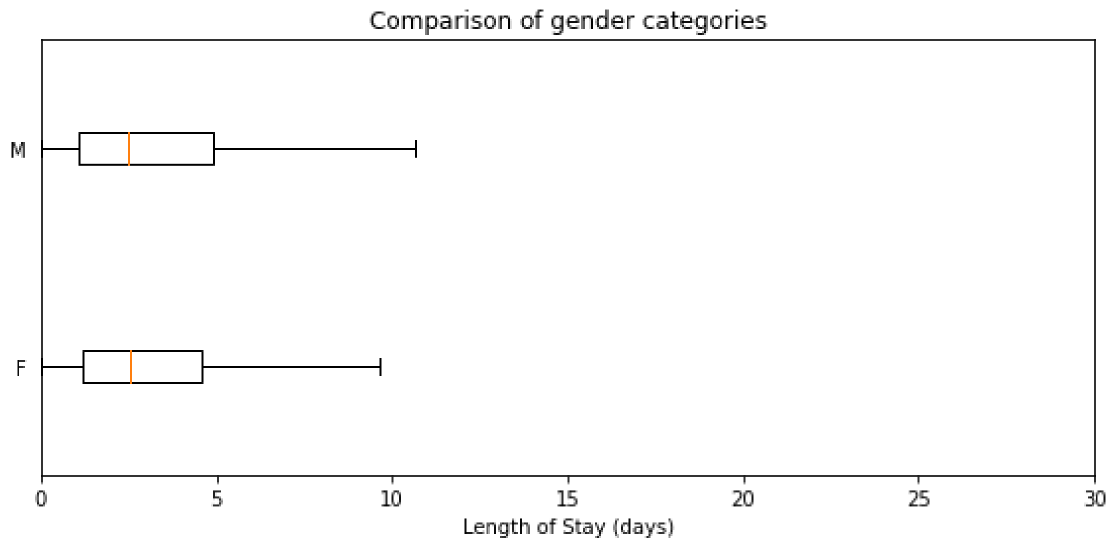
[17]:
```
age_ranges = [(0, 13), (14, 36), (37, 56), (57, 100)]
for num, cat_range in enumerate(age_ranges):
    admits_patients_diag['anchor_age'] = np.
 ↪where(admits_patients_diag['anchor_age'].between(cat_range[0],cat_range[1]),␣
 ↪num, admits_patients_diag['anchor_age'])

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()
```

[17]:
```
SENIOR          155448
MIDDLE_ADULT     88480
YOUNG_ADULT      52875
NEWBORN          38454
Name: anchor_age, dtype: int64
```

Finally, let's see the distribution of gender in patients in correlation to LOS.

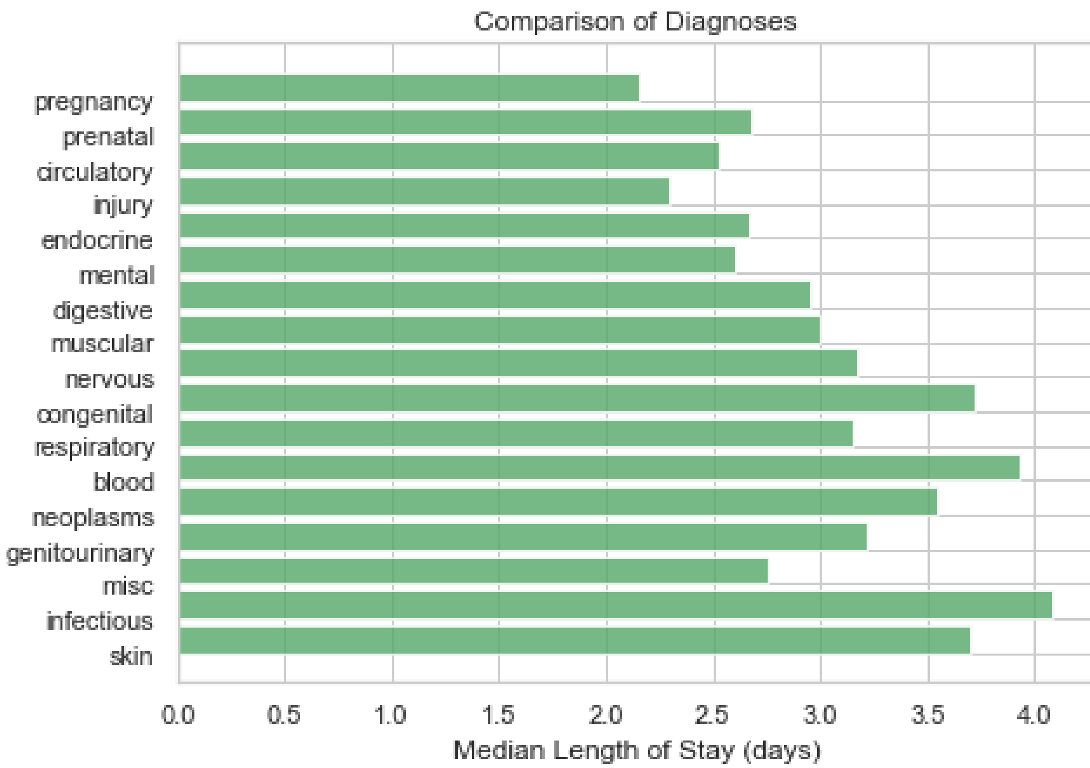
[18]:
```
# Re-usable boxplot function
def boxplot_los_groupby(variable, los_range=(-1, 30), size=(8,4)):
    '''
    Boxplot of LOS by df categorical series name
    '''
    results = admits_patients_diag[[variable, 'los']].groupby(variable).
 ↪median().reset_index()

    categories = results[variable].values.tolist()

    hist_data = []
    for cat in categories:
        hist_data.append(admits_patients_diag['los'].
 ↪loc[admits_patients_diag[variable]==cat].values)

    fig, ax = plt.subplots(figsize=size)
    ax.boxplot(hist_data, 0, '', vert=False)
    ax.set_xlim(los_range)
    ax.set_yticklabels(categories)
    ax.set_xlabel('Length of Stay (days)')
    ax.tick_params(left=False, right=False)
    ax.set_title('Comparison of {} categories'.format(variable))
    plt.tight_layout()
    plt.show();

boxplot_los_groupby('gender', los_range=(0, 30))
```



1.5 DIAGNOSIS

```
[ ]: Now, let's analyze the diagnosis in correlation to our target LOS.
```

```
[19]: # 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, 'los']].groupby(variable).
 ↪median().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('Median Length of Stay (days)')
ax.tick_params(left=False, right=False, top=False)
ax.set_title('Comparison of Diagnoses'.format(variable))
plt.show();
```



Looking at the median LOS for each ICD-9 supercategory shows an important difference between infectuous and pregnancy code groups for example. This could already means that the patients diagnosed an *infection* usually stay longer at the hospital in comparison to other categories.

1.5.1 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 increment the length of stay of patient.

[20]:
```python
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()
```

[20]:
```
   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
```

[21]:
```
icustays.groupby('first_careunit').median()
```

[21]:
```
                                                  subject_id     hadm_id  \
first_careunit
Cardiac Vascular Intensive Care Unit (CVICU)      14892840.0  24981696.0
Coronary Care Unit (CCU)                          15027761.0  24967090.5
Medical Intensive Care Unit (MICU)                15024484.0  24970198.0
Medical/Surgical Intensive Care Unit (MICU/SICU)  14995724.0  25014068.0
Neuro Intermediate                                14984524.0  25007733.0
Neuro Stepdown                                    14908467.0  25185450.0
Neuro Surgical Intensive Care Unit (Neuro SICU)   15186511.0  24976128.0
Surgical Intensive Care Unit (SICU)               14971764.0  24869088.0
Trauma SICU (TSICU)                               15021710.0  25060781.0

                                                     stay_id       los
first_careunit
Cardiac Vascular Intensive Care Unit (CVICU)      35007000.0  1.990509
Coronary Care Unit (CCU)                          34902339.0  2.011725
Medical Intensive Care Unit (MICU)                35047551.0  1.829190
Medical/Surgical Intensive Care Unit (MICU/SICU)  34964597.0  1.808738
Neuro Intermediate                                35070553.0  2.703368
Neuro Stepdown                                    34947848.0  1.843461
Neuro Surgical Intensive Care Unit (Neuro SICU)   34859288.0  3.646910
Surgical Intensive Care Unit (SICU)               35011708.0  1.974711
Trauma SICU (TSICU)                               35047227.0  1.931424
```

From this statistic we can see how, as far as LOS is concerned, a substantial difference in the median is found only between *Neuro SICU*, *Neuro Intermediate* and the other categories that we can call *Other-ICU*. The *Other_ICU* categories have a very similar median. We can therefore think of simply reducing the categories on three groups: *Neuro SICU*, *Neuro Intermediate* and *Other-ICU* (which includes all the others).

[22]:

```
icustays['first_careunit'].replace({'Cardiac Vascular Intensive Care Unit␣
↪(CVICU)': 'Other-ICU', 'Coronary Care Unit (CCU)': 'Other-ICU', 'Medical␣
↪Intensive Care Unit (MICU)': 'Other-ICU','Medical/Surgical Intensive Care␣
↪Unit (MICU/SICU)': 'Other-ICU', 'Neuro Stepdown': 'Other-ICU', 'Surgical␣
↪Intensive Care Unit (SICU)': 'Other-ICU', 'Trauma SICU (TSICU)':␣
↪'Other-ICU'}, inplace=True)

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

[22]:
```
    hadm_id                category
0  20000094             [Other-ICU]
1  20000147             [Other-ICU]
2  20000351             [Other-ICU]
3  20000397             [Other-ICU]
4  20000808  [Other-ICU, Other-ICU]
```

[23]:
```
icustays['first_careunit'].value_counts()
```

[23]:
```
Other-ICU                                         72866
Neuro Surgical Intensive Care Unit (Neuro SICU)    1851
Neuro Intermediate                                 1823
Name: first_careunit, dtype: int64
```

[24]:
```
# 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()
```

[24]:
```
   Neuro Intermediate  Neuro Surgical Intensive Care Unit (Neuro SICU)  \
0                   0                                                0
1                   0                                                0
2                   0                                                0
3                   0                                                0
4                   0                                                0

   Other-ICU   hadm_id
0          1  20000094
1          1  20000147
2          1  20000351
3          1  20000397
4          1  20000808
```

[26]:
```python
# Merge ICU data with main dataFrame
final_df = admits_patients_diag.merge(icu_item, how='outer', on='hadm_id')
final_df.head()
```

[26]:
```
   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  … neoplasms nervous pregnancy  prenatal  \
0                   0.0  …       0.0     0.0       0.0       0.0
1                   0.0  …       0.0     0.0       0.0       0.0
2                   0.0  …       0.0     0.0       0.0       0.0
3                   0.0  …       0.0     0.0       0.0       1.0
4                   0.0  …       0.0     0.0       0.0       0.0

   respiratory  skin       los  Neuro Intermediate  \
0          0.0   0.0  1.884722                 NaN
1          0.0   0.0  4.476389                 NaN
2          0.0   0.0  2.155556                 NaN
3          0.0   0.0  2.001389                 NaN
4          0.0   0.0  1.806250                 NaN

   Neuro Surgical Intensive Care Unit (Neuro SICU)  Other-ICU
0                                              NaN        NaN
1                                              NaN        NaN
2                                              NaN        NaN
3                                              NaN        NaN
4                                              NaN        NaN

[5 rows x 34 columns]
```

[27]:
```python
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 362995 entries, 0 to 362994
Data columns (total 34 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  Neuro Intermediate                                 69211 non-null   float64
 32  Neuro Surgical Intensive Care Unit (Neuro SICU)   69211 non-null   float64
 33  Other-ICU                                          69211 non-null   float64
dtypes: datetime64[ns](3), float64(24), int64(1), object(6)
memory usage: 96.9+ MB
```

[30]:

```
# Replace NaNs with 0
final_df['Neuro Intermediate'].fillna(value=0, inplace=True)
final_df['Neuro Surgical Intensive Care Unit (Neuro SICU)'].fillna(value=0,␣
↪inplace=True)
final_df['Other-ICU'].fillna(value=0, inplace=True)
```

[31]:

```
final_df.head()
```

[31]:
```
   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  … neoplasms nervous pregnancy  prenatal  \
0                   0.0  …       0.0     0.0       0.0       0.0
1                   0.0  …       0.0     0.0       0.0       0.0
2                   0.0  …       0.0     0.0       0.0       0.0
3                   0.0  …       0.0     0.0       0.0       1.0
4                   0.0  …       0.0     0.0       0.0       0.0

   respiratory  skin       los  Neuro Intermediate  \
0          0.0   0.0  1.884722                 0.0
1          0.0   0.0  4.476389                 0.0
2          0.0   0.0  2.155556                 0.0
3          0.0   0.0  2.001389                 0.0
4          0.0   0.0  1.806250                 0.0

   Neuro Surgical Intensive Care Unit (Neuro SICU)  Other-ICU
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 34 columns]
```

1.6 5. Data cleaning

[32]:
```
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 362995 entries, 0 to 362994
Data columns (total 34 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  Neuro Intermediate                                 362995 non-null  float64
 32  Neuro Surgical Intensive Care Unit (Neuro SICU)    362995 non-null  float64
 33  Other-ICU                                          362995 non-null  float64
dtypes: datetime64[ns](3), float64(24), int64(1), object(6)
memory usage: 96.9+ MB
```

[34]:
```
# Remove deceased persons as they will skew LOS result
final_df = final_df[final_df['died_at_the_hospital'] == 0.0]
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 329652 entries, 0 to 335256
Data columns (total 34 columns):
 #   Column                                             Non-Null Count   Dtype
---  ------                                             --------------   -----
 0   Unnamed: 0                                         329652 non-null  float64
```

```
 1   subject_id                                     329652 non-null  float64
 2   hadm_id                                        329652 non-null  int64
 3   admittime                                      329652 non-null
datetime64[ns]
 4   dischtime                                      329652 non-null
datetime64[ns]
 5   deathtime                                      0 non-null
datetime64[ns]
 6   admission_type                                 329652 non-null  object
 7   insurance                                      329652 non-null  object
 8   ethnicity                                      329652 non-null  object
 9   died_at_the_hospital                           329652 non-null  float64
 10  gender                                         329652 non-null  object
 11  anchor_age                                     329652 non-null  object
 12  dod                                            18976 non-null   object
 13  blood                                          329652 non-null  float64
 14  circulatory                                    329652 non-null  float64
 15  congenital                                     329652 non-null  float64
 16  digestive                                      329652 non-null  float64
 17  endocrine                                      329652 non-null  float64
 18  genitourinary                                  329652 non-null  float64
 19  infectious                                     329652 non-null  float64
 20  injury                                         329652 non-null  float64
 21  mental                                         329652 non-null  float64
 22  misc                                           329652 non-null  float64
 23  muscular                                       329652 non-null  float64
 24  neoplasms                                      329652 non-null  float64
 25  nervous                                        329652 non-null  float64
 26  pregnancy                                      329652 non-null  float64
 27  prenatal                                       329652 non-null  float64
 28  respiratory                                    329652 non-null  float64
 29  skin                                           329652 non-null  float64
 30  los                                            329652 non-null  float64
 31  Neuro Intermediate                             329652 non-null  float64
 32  Neuro Surgical Intensive Care Unit (Neuro SICU) 329652 non-null  float64
 33  Other-ICU                                      329652 non-null  float64
dtypes: datetime64[ns](3), float64(24), int64(1), object(6)
memory usage: 88.0+ MB
```

[35]:
```python
# Remove LOS with negative number
final_df = final_df[final_df['los'] > 0]
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 329652 entries, 0 to 335256
Data columns (total 34 columns):
 #   Column                                         Non-Null Count   Dtype
---  ------                                         --------------   -----
```

```
 0   Unnamed: 0                                      329652 non-null  float64
 1   subject_id                                      329652 non-null  float64
 2   hadm_id                                         329652 non-null  int64
 3   admittime                                       329652 non-null
datetime64[ns]
 4   dischtime                                       329652 non-null
datetime64[ns]
 5   deathtime                                       0 non-null
datetime64[ns]
 6   admission_type                                  329652 non-null  object
 7   insurance                                       329652 non-null  object
 8   ethnicity                                       329652 non-null  object
 9   died_at_the_hospital                            329652 non-null  float64
 10  gender                                          329652 non-null  object
 11  anchor_age                                      329652 non-null  object
 12  dod                                             18976 non-null   object
 13  blood                                           329652 non-null  float64
 14  circulatory                                     329652 non-null  float64
 15  congenital                                      329652 non-null  float64
 16  digestive                                       329652 non-null  float64
 17  endocrine                                       329652 non-null  float64
 18  genitourinary                                   329652 non-null  float64
 19  infectious                                      329652 non-null  float64
 20  injury                                          329652 non-null  float64
 21  mental                                          329652 non-null  float64
 22  misc                                            329652 non-null  float64
 23  muscular                                        329652 non-null  float64
 24  neoplasms                                       329652 non-null  float64
 25  nervous                                         329652 non-null  float64
 26  pregnancy                                       329652 non-null  float64
 27  prenatal                                        329652 non-null  float64
 28  respiratory                                     329652 non-null  float64
 29  skin                                            329652 non-null  float64
 30  los                                             329652 non-null  float64
 31  Neuro Intermediate                              329652 non-null  float64
 32  Neuro Surgical Intensive Care Unit (Neuro SICU) 329652 non-null  float64
 33  Other-ICU                                       329652 non-null  float64
dtypes: datetime64[ns](3), float64(24), int64(1), object(6)
memory usage: 88.0+ MB
```

[36]:
```python
# Drop unused or no longer needed columns
final_df.drop(columns=['Unnamed: 0', 'subject_id', 'hadm_id', 'admittime', 'dischtime', 'deathtime',
                'died_at_the_hospital',  'dod'], inplace=True)

final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Int64Index: 329652 entries, 0 to 335256
Data columns (total 26 columns):
 #   Column                                      Non-Null Count   Dtype  
---  ------                                      --------------   -----  
 0   admission_type                              329652 non-null  object 
 1   insurance                                   329652 non-null  object 
 2   ethnicity                                   329652 non-null  object 
 3   gender                                      329652 non-null  object 
 4   anchor_age                                  329652 non-null  object 
 5   blood                                       329652 non-null  float64
 6   circulatory                                 329652 non-null  float64
 7   congenital                                  329652 non-null  float64
 8   digestive                                   329652 non-null  float64
 9   endocrine                                   329652 non-null  float64
 10  genitourinary                               329652 non-null  float64
 11  infectious                                  329652 non-null  float64
 12  injury                                      329652 non-null  float64
 13  mental                                      329652 non-null  float64
 14  misc                                        329652 non-null  float64
 15  muscular                                    329652 non-null  float64
 16  neoplasms                                   329652 non-null  float64
 17  nervous                                     329652 non-null  float64
 18  pregnancy                                   329652 non-null  float64
 19  prenatal                                    329652 non-null  float64
 20  respiratory                                 329652 non-null  float64
 21  skin                                        329652 non-null  float64
 22  los                                         329652 non-null  float64
 23  Neuro Intermediate                          329652 non-null  float64
 24  Neuro Surgical Intensive Care Unit (Neuro SICU)  329652 non-null  float64
 25  Other-ICU                                   329652 non-null  float64
dtypes: float64(21), object(5)
memory usage: 67.9+ MB
```

[37]:
```
final_df.head()
```

[37]:
```
  admission_type insurance               ethnicity gender anchor_age  blood  \
0       ELECTIVE     Other           OTHER/UNKNOWN      F    NEWBORN    0.0   
1       ELECTIVE     Other                   WHITE      F    NEWBORN    0.0   
2       ELECTIVE     Other  BLACK/AFRICAN AMERICAN      M    NEWBORN    0.0   
3       ELECTIVE     Other                   WHITE      F    NEWBORN    0.0   
4       ELECTIVE     Other                   WHITE      M    NEWBORN    0.0   

   circulatory  congenital  digestive  endocrine  …  neoplasms  nervous  \
0          0.0         1.0        0.0        0.0  …        0.0      0.0   
1          0.0         0.0        0.0        0.0  …        0.0      0.0   
2          0.0         0.0        0.0        0.0  …        0.0      0.0   
3          0.0         0.0        0.0        0.0  …        0.0      0.0   
```

```
4          0.0          0.0         0.0         0.0  …          0.0       0.0

   pregnancy  prenatal  respiratory  skin       los  Neuro Intermediate  \
0        0.0       0.0          0.0   0.0  1.884722                 0.0
1        0.0       0.0          0.0   0.0  4.476389                 0.0
2        0.0       0.0          0.0   0.0  2.155556                 0.0
3        0.0       1.0          0.0   0.0  2.001389                 0.0
4        0.0       0.0          0.0   0.0  1.806250                 0.0

   Neuro Surgical Intensive Care Unit (Neuro SICU)  Other-ICU
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 26 columns]
```

[39]:
```python
# Convert gender into numeric boolean attribute
final_df['gender'].replace({'M': 0, 'F':1}, inplace=True)
final_df.head()
```

[39]:
```
  admission_type insurance               ethnicity  gender anchor_age  blood  \
0       ELECTIVE     Other           OTHER/UNKNOWN       1    NEWBORN    0.0
1       ELECTIVE     Other                   WHITE       1    NEWBORN    0.0
2       ELECTIVE     Other  BLACK/AFRICAN AMERICAN       0    NEWBORN    0.0
3       ELECTIVE     Other                   WHITE       1    NEWBORN    0.0
4       ELECTIVE     Other                   WHITE       0    NEWBORN    0.0

   circulatory  congenital  digestive  endocrine  …  neoplasms  nervous  \
0          0.0         1.0        0.0        0.0  …        0.0      0.0
1          0.0         0.0        0.0        0.0  …        0.0      0.0
2          0.0         0.0        0.0        0.0  …        0.0      0.0
3          0.0         0.0        0.0        0.0  …        0.0      0.0
4          0.0         0.0        0.0        0.0  …        0.0      0.0

   pregnancy  prenatal  respiratory  skin       los  Neuro Intermediate  \
0        0.0       0.0          0.0   0.0  1.884722                 0.0
1        0.0       0.0          0.0   0.0  4.476389                 0.0
2        0.0       0.0          0.0   0.0  2.155556                 0.0
3        0.0       1.0          0.0   0.0  2.001389                 0.0
4        0.0       0.0          0.0   0.0  1.806250                 0.0

   Neuro Surgical Intensive Care Unit (Neuro SICU)  Other-ICU
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 26 columns]
```

[40]:
```python
# Create dummy columns for categorical variables
prefix_cols = ['ADM', 'INS', 'ETH', 'AGE']
dummy_cols = ['admission_type', 'insurance','ethnicity', 'anchor_age']
final_df = pd.get_dummies(final_df, prefix=prefix_cols, columns=dummy_cols)
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 329652 entries, 0 to 335256
Data columns (total 38 columns):
 #   Column                                             Non-Null Count   Dtype
---  ------                                             --------------   -----
 0   gender                                             329652 non-null  int64
 1   blood                                              329652 non-null  float64
 2   circulatory                                        329652 non-null  float64
 3   congenital                                         329652 non-null  float64
 4   digestive                                          329652 non-null  float64
 5   endocrine                                          329652 non-null  float64
 6   genitourinary                                      329652 non-null  float64
 7   infectious                                         329652 non-null  float64
 8   injury                                             329652 non-null  float64
 9   mental                                             329652 non-null  float64
 10  misc                                               329652 non-null  float64
 11  muscular                                           329652 non-null  float64
 12  neoplasms                                          329652 non-null  float64
 13  nervous                                            329652 non-null  float64
 14  pregnancy                                          329652 non-null  float64
 15  prenatal                                           329652 non-null  float64
 16  respiratory                                        329652 non-null  float64
 17  skin                                               329652 non-null  float64
 18  los                                                329652 non-null  float64
 19  Neuro Intermediate                                 329652 non-null  float64
 20  Neuro Surgical Intensive Care Unit (Neuro SICU)    329652 non-null  float64
 21  Other-ICU                                          329652 non-null  float64
 22  ADM_ELECTIVE                                       329652 non-null  uint8
 23  ADM_EMERGENCY                                      329652 non-null  uint8
 24  ADM_OBSERVATION                                    329652 non-null  uint8
 25  ADM_SURGICAL SAME DAY ADMISSION                    329652 non-null  uint8
 26  INS_Medicaid                                       329652 non-null  uint8
 27  INS_Medicare                                       329652 non-null  uint8
 28  INS_Other                                          329652 non-null  uint8
 29  ETH_ASIAN                                          329652 non-null  uint8
 30  ETH_BLACK/AFRICAN AMERICAN                         329652 non-null  uint8
 31  ETH_HISPANIC/LATINO                                329652 non-null  uint8
```

```
 32  ETH_OTHER/UNKNOWN                                  329652 non-null  uint8
 33  ETH_WHITE                                          329652 non-null  uint8
 34  AGE_MIDDLE_ADULT                                   329652 non-null  uint8
 35  AGE_NEWBORN                                        329652 non-null  uint8
 36  AGE_SENIOR                                         329652 non-null  uint8
 37  AGE_YOUNG_ADULT                                    329652 non-null  uint8
dtypes: float64(21), int64(1), uint8(16)
memory usage: 62.9 MB
```

[41]:
```
# Check for any remaining NaNs
final_df.isnull().values.sum()
```

[41]:
```
0
```

The final DataFrame size resulted in 37 feature columns and 1 target column (LOS) with an entry count of 329.652

1.7 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: LOS (lenght of stay) 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.

[42]:
```
# Target Variable (Length-of-Stay-LOS)
LOS = final_df['los'].values
# Prediction Features
features = final_df.drop(columns=['los'])
```

Using the training set, we'll fit five different regression models (from the scikit-learn library) using default settings to see what the R2 score comparison looked like.

[43]:
```
# Split into training set 80% and test set 20%
X_train, X_test, y_train, y_test = train_test_split(features,
                                                    LOS,
                                                    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]))
```

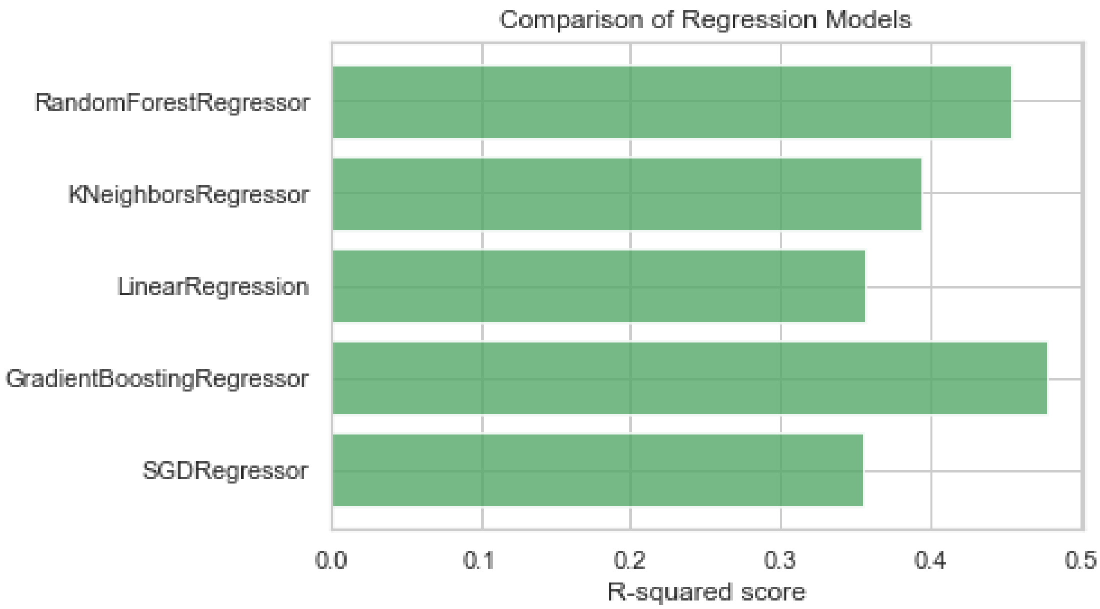
```
Training set has 263721 samples.
Testing set has 65931 samples.
```

[44]:
```
# Regression models used from scikit-learn for comparison
models = [SGDRegressor(random_state = 0),
          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))
```

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

[45]:
```
# 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.png', bbox_inches = 'tight')
```



The **GradientBoostingRegressor** has the best R2 score of ~48% so we focus on refining this particular model.

[46]:
```
# 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: 0.477324
```

1.8 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.
```

[47]:
```
# Split into train 80% and test 20%
X_train, X_test, y_train, y_test = train_test_split(features,
                                                    LOS,
                                                    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, verbose = 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 5 folds for each of 8 candidates, totalling 40 fits
0.49109741491773634
GradientBoostingRegressor(max_depth=4, n_estimators=300)
```

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=300, max_depth=4, loss = ls.

[48]:
```
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: 0.472332
```

Parameter tuning didn't improve the R2 score. This could mean that the model is overfitting the training data and can't generalize well on new data. For this reason we continue to use default parameters for GradientBoostingRegressor.

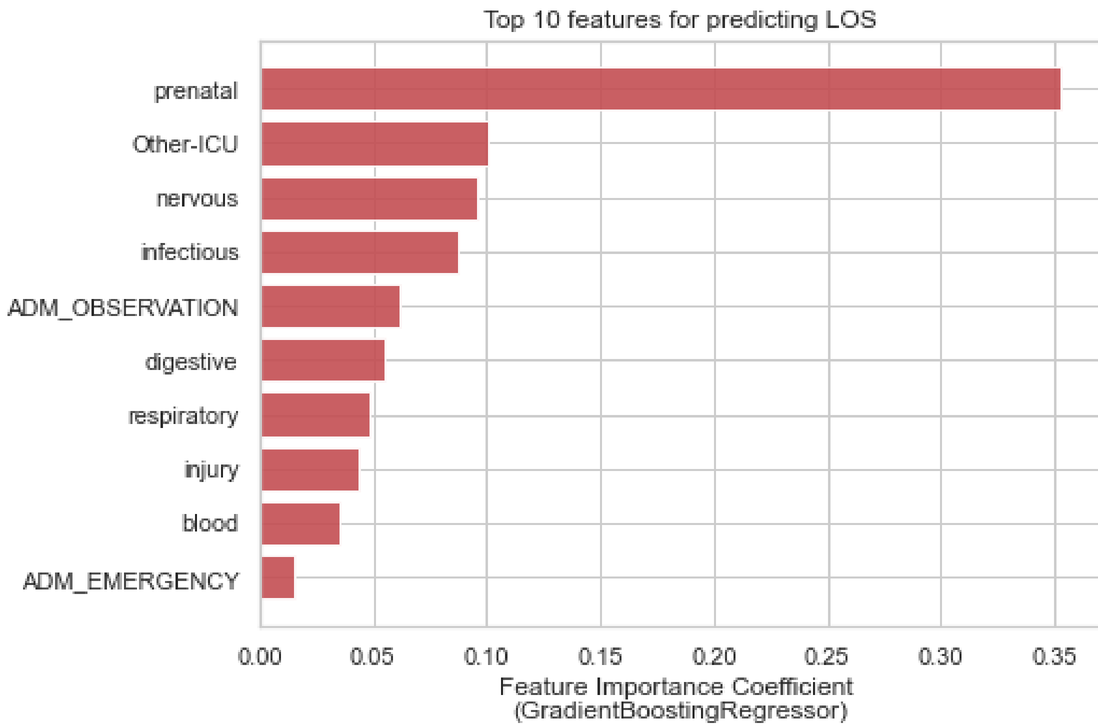
[ ]:
```
## 8. Model evaluation and result Discussion

First of al we could look at what features were most important in predicting␣
 ↪hospital length-of-stay when using the gradient boosting regression model.
```

[50]:
```
feature_imp = pd.DataFrame(reg_model_optimized.feature_importances_,
                                    index = X_train.columns,
                                    columns=['importance']).
 ↪sort_values('importance', ascending=False)

feature_imp.head(20)
```

[50]:

	importance
prenatal	0.352705
Other-ICU	0.100546
nervous	0.095366
infectious	0.087042
ADM_OBSERVATION	0.062016
digestive	0.055106
respiratory	0.048644
injury	0.043253
blood	0.035291
ADM_EMERGENCY	0.015049
neoplasms	0.014466
misc	0.011138
congenital	0.010837
skin	0.010752
ETH_ASIAN	0.008543
circulatory	0.008492
mental	0.007408
endocrine	0.005299
pregnancy	0.004724
ADM_ELECTIVE	0.004527

[57]:
```
#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_los_mimic4.png', bbox_inches = 'tight')
```



[ ]:
```
Diagnoses related to prenatal issues have the highest feature importance␣
↪coefficient followed by ICU (of general type) admission, nervous and␣
↪infectuous diagnosis. So we could say that, first of all, one of the results␣
↪is that the *ICD-9 diagnoses categories* are by far the most important␣
↪features between the features analyzed.
```

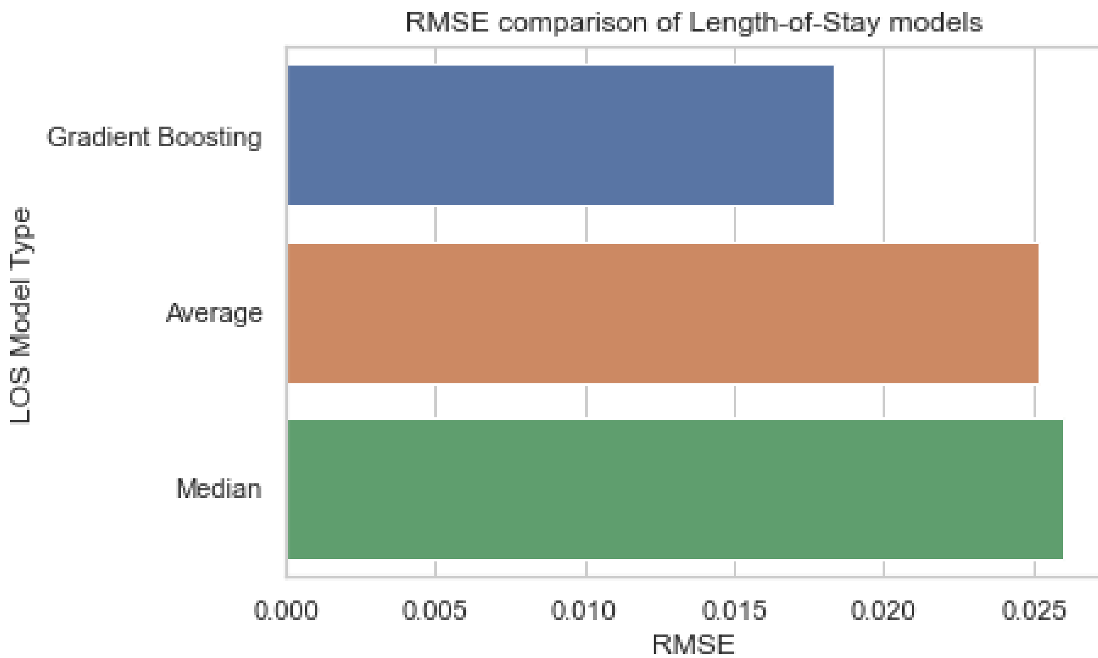
In previous metric section, we said that the RMSE would be used to compare the prediction model versus the industry-standard average and median LOS metrics.

[52]:
```
#y_test_preds = reg_model.predict(X_test)

ml_count, md_count, avg_count  = 0, 0, 0
ml_days, md_days, avg_days  = 0, 0, 0
ml_days_rms, md_days_rms, avg_days_rms  = 0, 0, 0

for i in range(y_test_preds.shape[0]):
    ml_model = abs(y_test_preds[i] - y_test[i])
    median_model = abs(actual_median_los - y_test[i])
    average_model = abs(actual_mean_los - y_test[i])

    ml_days += ml_model
    md_days += median_model
    avg_days += average_model
```

```
    ml_model_rms = (y_test_preds[i] - y_test[i])**2
    median_model_rms = (actual_median_los - y_test[i])**2
    average_model_rms = (actual_mean_los - y_test[i])**2

    ml_days_rms += ml_model_rms
    md_days_rms += median_model_rms
    avg_days_rms += average_model_rms

print("Prediction Model days {}".format(ml_days/y_test_preds.shape[0]))
print("Median Model days {}".format(md_days/y_test_preds.shape[0]))
print("Average Model days {}".format(avg_days/y_test_preds.shape[0]))

print("Prediction Model RMS {}".format((ml_days_rms**0.5)/y_test_preds.
 ↪shape[0]))
print("Median Model RMS {}".format((md_days_rms**0.5)/y_test_preds.shape[0]))
print("Average Model RMS {}".format((avg_days_rms**0.5)/y_test_preds.shape[0]))
```

```
Prediction Model days 2.2084883618218436
Median Model days 2.9238206420348725
Average Model days 3.2840947398901026
Prediction Model RMS 0.018297159327123304
Median Model RMS 0.025987491465568332
Average Model RMS 0.025188560774206607
```

[56]:
```
# RMSE plot
data = pd.DataFrame({'RMSE': [(ml_days_rms**0.5)/y_test_preds.shape[0],
                              (avg_days_rms**0.5)/y_test_preds.shape[0],
                              (md_days_rms**0.5)/y_test_preds.shape[0]],
                     'LOS Model Type': ['Gradient Boosting', 'Average',␣
 ↪'Median'] })

fig, ax = plt.subplots()
ax = sns.barplot(x='RMSE', y='LOS Model Type', data=data)
ax.set_title('RMSE comparison of Length-of-Stay models')
ax.tick_params(top=False, left=False, right=False)

fig.savefig('images/rms_comparison_los_mimic4_01.png', bbox_inches = 'tight')
```



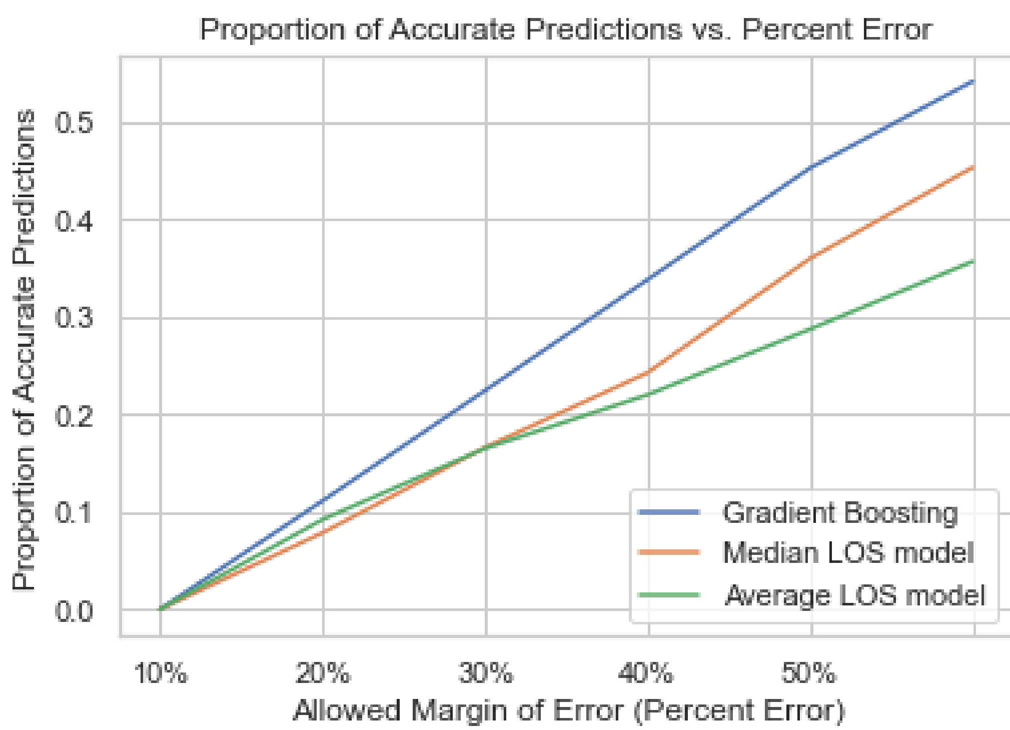
The gradient boosting model RMSE is better even if the percent difference in comparison to the constant average or median models, is not that high (as we can see from the graphic).

Another way to look at the model could be to plot the proportion of accurate predictions in the test set versus an allowed margin of error. Other studies qualify a LOS prediction as correct if it falls within a certain margin of error. Obviously, it follows that as the margin of error allowance increases, so should the proportion of accurate predictions for all models. The gradient boosting prediction model performs better than the other constant models across the margin of error range up to 50%.

[55]:
```
# Calculate Proportion of 'accurate' prediction as a function of allowed margin␣
↪of error
reg_array = []
median_array = []
average_array = []

for i in list(range(6)):
    reg_count, median_count, average_count = 0, 0, 0

    for j in range(y_test_preds.shape[0]):
        # Percent Difference
        reg_model = (y_test_preds[j] - y_test[j])/y_test[j]
        median_model = (actual_median_los - y_test[j])/y_test[j]
        average_model = (actual_mean_los - y_test[j])/y_test[j]
        if abs(reg_model) < i/10:
            reg_count += 1
```

```python
        if abs(median_model) < i/10:
            median_count += 1
        if abs(average_model) < i/10:
            average_count += 1

    reg_array.append((reg_count/y_test_preds.shape[0]))
    median_array.append((median_count/y_test_preds.shape[0]))
    average_array.append((average_count/y_test_preds.shape[0]))

# Plot proportion of 'accurate' prediction as a function of allowed margin of error
fig, ax = plt.subplots()
ax.plot(reg_array, label='Gradient Boosting')
ax.plot(median_array, label='Median LOS model')
ax.plot(average_array, label='Average LOS model')
ax.set_title('Proportion of Accurate Predictions vs. Percent Error')
ax.set_xlabel('Allowed Margin of Error (Percent Error)')
ax.set_ylabel('Proportion of Accurate Predictions')
ax.set_xticklabels(['0%', '10%', '20%', '30%', '40%', '50%'])
ax.legend(loc='lower right');
ax.tick_params(top=False, right=False)
fig.savefig('images/rms_comparison_los_mimic4_02.png', bbox_inches = 'tight')
```

```
<ipython-input-55-86a6c24f915a>:33: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_xticklabels(['0%', '10%', '20%', '30%', '40%', '50%'])
```



1.9 Conclusions for LOS (Length-of-stay)

Hospital stays cost the health system at least a big amount of money. U.S. Hospital for example spends $377.5 billion per year in the health system and recent Medicare legislation standardizes payments for procedures performed, regardless of the number of days a patient spends in the hospital.

This incentivizes hospitals to identify patients of high LOS risk at the time of admission. Once identified, patients with high LOS risk can have their treatment plan optimized to minimize LOS and lower the chance of getting a hospital-acquired condition. Another benefit is that prior knowledge of LOS can aid in logistics such as room and bed allocation planning.