Prediction on Diabetes Patient's Hospital Readmission¶

Problem Statement and Objective A hospital readmission is when a patient who is discharged from the hospital, gets re-admitted again within a certain period of time. Hospital readmission rates for certain conditions are now considered an indicator of hospital quality, and also affect the cost of care adversely. For this reason, Centers for Medicare & Medicaid Services established the Hospital Readmissions Reduction Program which aims to improve quality of care for patients and reduce health care spending by applying payment penalties to hospitals that have more than expected readmission rates for certain conditions. Although diabetes is not yet included in the penalty measures, the program is regularly adding new disease conditions to the list, now totaling 6 for FY2018. In 2011, American hospitals spent over $41 billion on diabetic patients who got readmitted within 30 days of discharge. Being able to determine factors that lead to higher readmission in such patients, and correspondingly being able to predict which patients will get readmitted can help hospitals save millions of dollars while improving quality of care. So, with that background in mind, we used a medical claims dataset (description below), to answer these questions:

1. What factors are the strongest predictors of hospital readmission in diabetic patients?
2. How well can we predict hospital readmission in this dataset with limited features?

Data Set Description¶

VARIABLE NAMES: DESCRIPTION

- **Encounter ID** Unique identifier of an encounter
- **Patient number** Unique identifier of a patient
- **Race** Values: Caucasian, Asian, African American, Hispanic, and other
- **Gender** Values: male, female, and unknown/invalid
- **Age** Grouped in 10-year intervals: 0, 10), 10, 20), …, 90, 100)
- **Weight** Weight in pounds
- **Admission type** Integer identifier corresponding to 9 distinct values, for example, emergency, urgent, elective, newborn, and not available
- **Discharge disposition** Integer identifier corresponding to 29 distinct values, for example, discharged to home, expired, and not available
- **Admission source** Integer identifier corresponding to 21 distinct values, for example, physician referral, emergency room, and transfer from a hospital
- **Time in hospital** Integer number of days between admission and discharge
- **Payer code** Integer identifier corresponding to 23 distinct values, for example, Blue Cross/Blue Shield, Medicare, and self-pay Medical

- **Medical specialty** Integer identifier of a specialty of the admitting physician, corresponding to 84 distinct values, for example, cardiology, internal medicine, family/general practice, and surgeon
- **Number of lab procedures** Number of lab tests performed during the encounter
- **Number of procedures** Numeric Number of procedures (other than lab tests) performed during the encounter
- **Number of medications** Number of distinct generic names administered during the encounter
- **Number of outpatient visits** Number of outpatient visits of the patient in the year preceding the encounter
- **Number of emergency visits** Number of emergency visits of the patient in the year preceding the encounter
- **Number of inpatient visits** Number of inpatient visits of the patient in the year preceding the encounter
- **Diagnosis 1** The primary diagnosis (coded as first three digits of ICD9); 848 distinct values
- **Diagnosis 2** Secondary diagnosis (coded as first three digits of ICD9); 923 distinct values
- **Diagnosis 3** Additional secondary diagnosis (coded as first three digits of ICD9); 954 distinct values
- **Number of diagnoses** Number of diagnoses entered to the system 0%
- **Glucose serum test result** Indicates the range of the result or if the test was not taken. Values: “>200,” “>300,” “normal,” and “none” if not measured
- **A1c test result** Indicates the range of the result or if the test was not taken. Values: “>8” if the result was greater than 8%, “>7” if the result was greater than 7% but less than 8%, “normal” if the result was less than 7%, and “none” if not measured.
- **Change of medications** Indicates if there was a change in diabetic medications (either dosage or generic name). Values: “change” and “no change”
- **Diabetes medications** Indicates if there was any diabetic medication prescribed. Values: “yes” and “no”
- 24 features for medications For the generic names: **metformin, repaglinide, nateglinide, chlorpropamide, glimepiride, acetohexamide, glipizide, glyburide, tolbutamide, pioglitazone, rosiglitazone, acarbose, miglitol, troglitazone, tolazamide, examide, sitagliptin, insulin, glyburide-metformin, glipizide-metformin, glimepiride- pioglitazone, metformin-rosiglitazone, and metformin- pioglitazone**, the feature indicates whether the drug was prescribed or there was a change in the dosage. Values: “up” if the dosage was increased during the encounter, “down” if the dosage was decreased, “steady” if the dosage did not change, and “no” if the drug was not prescribed
- **Readmitted** Days to inpatient readmission. Values: “<30” if the patient was readmitted in less than 30 days, “>30” if the patient was readmitted in more than 30 days, and “No” for no record of readmission

Data Preparation & Exploration¶

In [1]:

```
#Loading libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
#loading Dataset
df = pd.read_csv("../input/diabetic_data.csv")
```

In [3]:

```
#displaying first 10 rows of data
df.head(10).T
```

Out[3]:

	0	1	2	3	4	5	6	
encounter_id	2278392	149190	64410	500364	16680	35754	55842	6376
patient_nbr	8222157	55629189	86047875	82442376	42519267	82637451	84259809	1148
race	Caucasian	Caucasian	AfricanAmerican	Caucasian	Caucasian	Caucasian	Caucasian	Cauc
gender	Female	Female	Female	Male	Male	Male	Male	Male
age	[0-10)	[10-20)	[20-30)	[30-40)	[40-50)	[50-60)	[60-70)	[70-8
weight	?	?	?	?	?	?	?	?
admission_type_id	6	1	1	1	1	2	3	1
discharge_disposition_id	25	1	1	1	1	1	1	1
admission_source_id	1	7	7	7	7	2	2	7
time_in_hospital	1	3	2	2	1	3	4	5
payer_code	?	?	?	?	?	?	?	?
medical_specialty	Pediatrics-Endocrinology	?	?	?	?	?	?	?
num_lab_procedures	41	59	11	44	51	31	70	73
num_procedures	0	0	5	1	0	6	1	0
num_medications	1	18	13	16	8	16	21	12
number_outpatient	0	0	2	0	0	0	0	0
number_emergency	0	0	0	0	0	0	0	0
number_inpatient	0	0	1	0	0	0	0	0
diag_1	250.83	276	648	8	197	414	414	428
diag_2	?	250.01	250	250.43	157	411	411	492
diag_3	?	255	V27	403	250	250	V45	250
number_diagnoses	1	9	6	7	5	9	7	8
max_glu_serum	None	None	None	None	None	None	None	None
A1Cresult	None	None	None	None	None	None	None	None
metformin	No	No	No	No	No	No	Steady	No
repaglinide	No	No	No	No	No	No	No	No
nateglinide	No	No	No	No	No	No	No	No
chlorpropamide	No	No	No	No	No	No	No	No
glimepiride	No	No	No	No	No	No	Steady	No
acetohexamide	No	No	No	No	No	No	No	No
glipizide	No	No	Steady	No	Steady	No	No	No
glyburide	No	No	No	No	No	No	No	Stead

	0	1	2	3	4	5		6	
tolbutamide	No	No	No	No	No	No	No		No
pioglitazone	No	No	No	No	No	No	No		No
rosiglitazone	No	No	No	No	No	No	No		No
acarbose	No	No	No	No	No	No	No		No
miglitol	No	No	No	No	No	No	No		No
troglitazone	No	No	No	No	No	No	No		No
tolazamide	No	No	No	No	No	No	No		No
examide	No	No	No	No	No	No	No		No
citoglipton	No	No	No	No	No	No	No		No
insulin	No	Up	No	Up	Steady	Steady	Steady		No
glyburide-metformin	No	No	No	No	No	No	No		No
glipizide-metformin	No	No	No	No	No	No	No		No
glimepiride-pioglitazone	No	No	No	No	No	No	No		No
metformin-rosiglitazone	No	No	No	No	No	No	No		No
metformin-pioglitazone	No	No	No	No	No	No	No		No
change	No	Ch	No	Ch	Ch	No	Ch		No
diabetesMed	No	Yes	Yes	Yes	Yes	Yes	Yes		Yes
readmitted	NO	>30	NO	NO	NO	>30	NO		>30

In [4]:

```
#checking shape of the dataset
df.shape
```

Out[4]:

```
(101766, 50)
```

In [5]:

```
#Checking data types of each variable
df.dtypes
```

Out[5]:

```
encounter_id                int64
patient_nbr                 int64
race                       object
gender                     object
age                        object
weight                     object
admission_type_id           int64
discharge_disposition_id    int64
admission_source_id         int64
time_in_hospital            int64
payer_code                 object
medical_specialty          object
num_lab_procedures          int64
num_procedures              int64
num_medications             int64
number_outpatient           int64
number_emergency            int64
number_inpatient            int64
diag_1                     object
diag_2                     object
diag_3                     object
number_diagnoses            int64
max_glu_serum              object
A1Cresult                  object
metformin                  object
repaglinide                object
nateglinide                object
```

```
chlorpropamide              object
glimepiride                 object
acetohexamide               object
glipizide                   object
glyburide                   object
tolbutamide                 object
pioglitazone                object
rosiglitazone               object
acarbose                    object
miglitol                    object
troglitazone                object
tolazamide                  object
examide                     object
citoglipton                 object
insulin                     object
glyburide-metformin         object
glipizide-metformin         object
glimepiride-pioglitazone    object
metformin-rosiglitazone     object
metformin-pioglitazone      object
change                      object
diabetesMed                 object
readmitted                  object
dtype: object
```

In [6]:

```
#Checking for missing values in dataset

#In the dataset missing values are represented as '?' sign

for col in df.columns:

    if df[col].dtype == object:

         print(col,df[col][df[col] == '?'].count())
```

```
race 2273
gender 0
age 0
weight 98569
payer_code 40256
medical_specialty 49949
diag_1 21
diag_2 358
diag_3 1423
max_glu_serum 0
A1Cresult 0
metformin 0
repaglinide 0
nateglinide 0
chlorpropamide 0
glimepiride 0
acetohexamide 0
glipizide 0
glyburide 0
tolbutamide 0
pioglitazone 0
rosiglitazone 0
acarbose 0
miglitol 0
troglitazone 0
tolazamide 0
examide 0
citoglipton 0
insulin 0
glyburide-metformin 0
glipizide-metformin 0
glimepiride-pioglitazone 0
```

```
metformin-rosiglitazone 0
metformin-pioglitazone 0
change 0
diabetesMed 0
readmitted 0
```

In [7]:

```
# gender was coded differently so we use a custom count for this one
print('gender', df['gender'][df['gender'] == 'Unknown/Invalid'].count())
```

```
gender 3
```

Dealing with Missing Values¶

Variable weight contains approximate 98% of the missing values so there is no significance in filling those missing values so we decided to drop these variables. Variable Payer code and medical specialty contains approximate 40% missing values so we also dropped these variables. Variables race, diag_1, diag_2, diag_3 and gender contains very less missing values as compared to other attributes which we dropped so for these attributes we also decided to drop those where missing values contains.

In [8]:

```
#dropping columns with large number of missing values
df = df.drop(['weight','payer_code','medical_specialty'], axis = 1)
```

In [9]:

```
drop_Idx = set(df[(df['diag_1'] == '?') & (df['diag_2'] == '?') & (df['diag_3'] ==
'?')].index)


drop_Idx = drop_Idx.union(set(df['diag_1'][df['diag_1'] == '?'].index))
drop_Idx = drop_Idx.union(set(df['diag_2'][df['diag_2'] == '?'].index))
drop_Idx = drop_Idx.union(set(df['diag_3'][df['diag_3'] == '?'].index))
drop_Idx = drop_Idx.union(set(df['race'][df['race'] == '?'].index))
drop_Idx = drop_Idx.union(set(df[df['discharge_disposition_id'] == 11].index))
drop_Idx = drop_Idx.union(set(df['gender'][df['gender'] == 'Unknown/Invalid'].index
))
new_Idx = list(set(df.index) - set(drop_Idx))
df = df.iloc[new_Idx]
```

variables (drugs named citoglipton and examide), all records have the same value. So essentially these cannot provide any interpretive or discriminatory information for predicting readmission so we decided to drop these two variables

In [10]:

```
df = df.drop(['citoglipton', 'examide'], axis = 1)
```

In [11]:

```
#Checking for missing values in the data
for col in df.columns:
    if df[col].dtype == object:
         print(col,df[col][df[col] == '?'].count())

print('gender', df['gender'][df['gender'] == 'Unknown/Invalid'].count())
```

```
race 0
gender 0
age 0
diag_1 0
diag_2 0
diag_3 0
max_glu_serum 0
A1Cresult 0
metformin 0
repaglinide 0
nateglinide 0
chlorpropamide 0
glimepiride 0
acetohexamide 0
glipizide 0
glyburide 0
tolbutamide 0
pioglitazone 0
rosiglitazone 0
acarbose 0
miglitol 0
troglitazone 0
tolazamide 0
insulin 0
glyburide-metformin 0
glipizide-metformin 0
glimepiride-pioglitazone 0
metformin-rosiglitazone 0
metformin-pioglitazone 0
change 0
diabetesMed 0
readmitted 0
gender 0
```

Feature Engineering¶

This is highly subjective, and partly depends on a knowledge of health care services, and making sense of the potential relationships between features. There are perhaps thousands of ways to try here. We tried some...

- Service utilization: The data contains variables for number of inpatient (admissions), emergency room visits and outpatient visits for a given patient in the previous one year. These are (crude) measures of how much hospital/clinic services a person has used in the past year. We added these three to create a new variable called service utilization (see figure below). The idea was to see which version gives us better results. Granted,

we did not apply any special weighting to the three ingredients of service utilization but we wanted to try something simple at this stage.

In [12]:

```
df['service_utilization'] = df['number_outpatient'] + df['number_emergency'] + df['
number_inpatient']
```

- Number of medication changes: The dataset contains 23 features for 23 drugs (or combos) which indicate for each of these, whether a change in that medication was made or not during the current hospital stay of patient. Medication change for diabetics upon admission has been shown by previous research to be associated with lower readmission rates. We decided to count how many changes were made in total for each patient, and declared that a new feature. The reasoning here was to both simplify the model and possibly discover a relationship with number of changes regardless of which drug was changed.

In [13]:

```
keys = ['metformin', 'repaglinide', 'nateglinide', 'chlorpropamide', 'glimepiride',
'glipizide', 'glyburide', 'pioglitazone', 'rosiglitazone', 'acarbose', 'miglitol',
'insulin', 'glyburide-metformin', 'tolazamide', 'metformin-pioglitazone','metformin
-rosiglitazone', 'glimepiride-pioglitazone', 'glipizide-metformin', 'troglitazone',
'tolbutamide', 'acetohexamide']

for col in keys:

    colname = str(col) + 'temp'

    df[colname] = df[col].apply(lambda x: 0 if (x == 'No' or x == 'Steady') else 1)

df['numchange'] = 0

for col in keys:

    colname = str(col) + 'temp'

    df['numchange'] = df['numchange'] + df[colname]

    del df[colname]


df['numchange'].value_counts()
```

Out[13]:

```
0    70142
1    24922
2     1271
3      106
4        5
Name: numchange, dtype: int64
```

In [14]:

```
# re-encoding admission type, discharge type and admission source into fewer catego
ries


df['admission_type_id'] = df['admission_type_id'].replace(2,1)
```

```
df['admission_type_id'] = df['admission_type_id'].replace(7,1)
df['admission_type_id'] = df['admission_type_id'].replace(6,5)
df['admission_type_id'] = df['admission_type_id'].replace(8,5)

df['discharge_disposition_id'] = df['discharge_disposition_id'].replace(6,1)
df['discharge_disposition_id'] = df['discharge_disposition_id'].replace(8,1)
df['discharge_disposition_id'] = df['discharge_disposition_id'].replace(9,1)
df['discharge_disposition_id'] = df['discharge_disposition_id'].replace(13,1)
df['discharge_disposition_id'] = df['discharge_disposition_id'].replace(3,2)
df['discharge_disposition_id'] = df['discharge_disposition_id'].replace(4,2)
df['discharge_disposition_id'] = df['discharge_disposition_id'].replace(5,2)
df['discharge_disposition_id'] = df['discharge_disposition_id'].replace(14,2)
df['discharge_disposition_id'] = df['discharge_disposition_id'].replace(22,2)
df['discharge_disposition_id'] = df['discharge_disposition_id'].replace(23,2)
df['discharge_disposition_id'] = df['discharge_disposition_id'].replace(24,2)
df['discharge_disposition_id'] = df['discharge_disposition_id'].replace(12,10)
df['discharge_disposition_id'] = df['discharge_disposition_id'].replace(15,10)
df['discharge_disposition_id'] = df['discharge_disposition_id'].replace(16,10)
df['discharge_disposition_id'] = df['discharge_disposition_id'].replace(17,10)
df['discharge_disposition_id'] = df['discharge_disposition_id'].replace(25,18)
df['discharge_disposition_id'] = df['discharge_disposition_id'].replace(26,18)

df['admission_source_id'] = df['admission_source_id'].replace(2,1)
df['admission_source_id'] = df['admission_source_id'].replace(3,1)
df['admission_source_id'] = df['admission_source_id'].replace(5,4)
df['admission_source_id'] = df['admission_source_id'].replace(6,4)
df['admission_source_id'] = df['admission_source_id'].replace(10,4)
df['admission_source_id'] = df['admission_source_id'].replace(22,4)
df['admission_source_id'] = df['admission_source_id'].replace(25,4)
df['admission_source_id'] = df['admission_source_id'].replace(15,9)
df['admission_source_id'] = df['admission_source_id'].replace(17,9)
df['admission_source_id'] = df['admission_source_id'].replace(20,9)
df['admission_source_id'] = df['admission_source_id'].replace(21,9)
df['admission_source_id'] = df['admission_source_id'].replace(13,11)
df['admission_source_id'] = df['admission_source_id'].replace(14,11)
```

- Encoding some variables: The original dataset used string values for gender, race, medication change, and each of the 23 drugs used. To better fit those variables into our model, we interpret the variables to numeric binary variables to reflect their nature. For example, we encoded the “ medication change ” feature from “No” (no change) and “Ch” (changed) into 0 and 1.

In [15]:

```
df['change'] = df['change'].replace('Ch', 1)
df['change'] = df['change'].replace('No', 0)
df['gender'] = df['gender'].replace('Male', 1)
df['gender'] = df['gender'].replace('Female', 0)
df['diabetesMed'] = df['diabetesMed'].replace('Yes', 1)
df['diabetesMed'] = df['diabetesMed'].replace('No', 0)
# keys is the same as before
for col in keys:
    df[col] = df[col].replace('No', 0)
    df[col] = df[col].replace('Steady', 1)
    df[col] = df[col].replace('Up', 1)
    df[col] = df[col].replace('Down', 1)
```

We also reduced both A1C test result and Glucose serum test result into categories of Normal, Abnormal and Not tested.

In [16]:

```
df['A1Cresult'] = df['A1Cresult'].replace('>7', 1)
df['A1Cresult'] = df['A1Cresult'].replace('>8', 1)
df['A1Cresult'] = df['A1Cresult'].replace('Norm', 0)
df['A1Cresult'] = df['A1Cresult'].replace('None', -99)
df['max_glu_serum'] = df['max_glu_serum'].replace('>200', 1)
df['max_glu_serum'] = df['max_glu_serum'].replace('>300', 1)
df['max_glu_serum'] = df['max_glu_serum'].replace('Norm', 0)
df['max_glu_serum'] = df['max_glu_serum'].replace('None', -99)
```

- Dealing with age: There are different ways to deal with this. The dataset only gives us age as 10 year categories, so we don't know the exact age of each patient. The previous study on this dataset used age categories as nominal variables, but we wanted to be able to see the effect of increasing age on readmission, even if in a crude way. To do that, we assume that age of the patient on average lies at the midpoint of the age category. For example, if the patient's age category is 20–30 years, then we assume the age = 25 years. So we converted age categories to midpoints, resulting in a numeric variable:

In [17]:

```
# code age intervals [0-10) - [90-100) from 1-10
for i in range(0,10):
    df['age'] = df['age'].replace('['+str(10*i)+'-'+str(10*(i+1))+')', i+1)
df['age'].value_counts()
```

Out[17]:

```
8     24815
7     21521
6     16546
9     16223
5      9208
4      3538
10     2594
3      1471
2       466
1        64
Name: age, dtype: int64
```

Collapsing of Multiple Encounters for same patient Some patients in the dataset had more than one encounter.We could not count them as independent encounters because that bias the results towards those patients who had multiple encounters. Thus we tried multiple techniques to collapse and consolidate multiple encounters for same patient such as:

- Considering more than 2 readmissions across multiple encounters as readmission for collapsed record.
- Considering average stay at hospital across multiple encounters.
- Considering the percentage of the medication changes across multiple encounters
- Considering the total number of the encounters to replace the encounter unique ID
- Considering the combination of diagnoses across multiple encounters as a list However, taking the features such as “diagnosis”, for instance, we did not find it not meaningful to combine multiple categorical values into an array for building data model. We then considered first encounter and last encounter separately as possible representations of multiple encounters. However, last encounters gave extremely imbalanced data for readmissions (96/4 Readmissions vs No Readmissions) and thus, we decided to use first encounters of patients with multiple encounters. This resulted in dataset being reduced to about 70,000 encounters:

In [18]:

```
df2 = df.drop_duplicates(subset= ['patient_nbr'], keep = 'first')
df2.shape
(70442, 55)
```

Out[18]:

```
(70442, 55)
```

In [19]:

```
df.head().T
```

Out[19]:

	1	2	3	4	5
encounter_id	149190	64410	500364	16680	35754
patient_nbr	55629189	86047875	82442376	42519267	82637451
race	Caucasian	AfricanAmerican	Caucasian	Caucasian	Caucasian
gender	0	0	1	1	1
age	2	3	4	5	6
admission_type_id	1	1	1	1	1

	1	2	3	4	5
discharge_disposition_id	1	1	1	1	1
admission_source_id	7	7	7	7	1
time_in_hospital	3	2	2	1	3
num_lab_procedures	59	11	44	51	31
num_procedures	0	5	1	0	6
num_medications	18	13	16	8	16
number_outpatient	0	2	0	0	0
number_emergency	0	0	0	0	0
number_inpatient	0	1	0	0	0
diag_1	276	648	8	197	414
diag_2	250.01	250	250.43	157	411
diag_3	255	V27	403	250	250
number_diagnoses	9	6	7	5	9
max_glu_serum	-99	-99	-99	-99	-99
A1Cresult	-99	-99	-99	-99	-99
metformin	0	0	0	0	0
repaglinide	0	0	0	0	0
nateglinide	0	0	0	0	0
chlorpropamide	0	0	0	0	0
glimepiride	0	0	0	0	0
acetohexamide	0	0	0	0	0
glipizide	0	1	0	1	0
glyburide	0	0	0	0	0
tolbutamide	0	0	0	0	0
pioglitazone	0	0	0	0	0
rosiglitazone	0	0	0	0	0
acarbose	0	0	0	0	0
miglitol	0	0	0	0	0
troglitazone	0	0	0	0	0
tolazamide	0	0	0	0	0
insulin	1	0	1	1	1
glyburide-metformin	0	0	0	0	0
glipizide-metformin	0	0	0	0	0
glimepiride-pioglitazone	0	0	0	0	0
metformin-rosiglitazone	0	0	0	0	0
metformin-pioglitazone	0	0	0	0	0
change	1	0	1	1	0
diabetesMed	1	1	1	1	1
readmitted	>30	NO	NO	NO	>30
service_utilization	0	3	0	0	0
numchange	1	0	1	0	0

- Encoding the outcome variable: The outcome we are looking at is whether the patient gets readmitted to the hospital within 30 days or not. The variable actually has < 30, > 30 and No Readmission categories. To reduce our problem to a binary classification, we combined the readmission after 30 days and no readmission into a single category:

In [20]:

```
df['readmitted'].value_counts()
```

Out[20]:

```
NO     50731
```

```
>30    34649
<30    11066
Name: readmitted, dtype: int64
```

In [21]:

```
df['readmitted'] = df['readmitted'].replace('>30', 0)
df['readmitted'] = df['readmitted'].replace('<30', 1)
df['readmitted'] = df['readmitted'].replace('NO', 0)
```

- Categorization of diagnoses: The dataset contained up to three diagnoses for a given patient (primary, secondary and additional). However, each of these had 700–900 unique ICD codes and it is extremely difficult to include them in the model and interpret meaningfully. Therefore, we collapsed these diagnosis codes into 9 disease categories in an almost similar fashion to that done in the original publication using this dataset. These 9 categories include Circulatory, Respiratory, Digestive, Diabetes, Injury, Musculoskeletal, Genitourinary, Neoplasms, and Others. Although we did this for primary, secondary and additional diagnoses, we eventually decided to use only the primary diagnosis in our model. Doing this in python was slightly cumbersome because, well, we are mapping the disease codes to certain category names. Below code should demonstrate this easily.

In [22]:

```
# Creating additional columns for diagnosis# Creati
df['level1_diag1'] = df['diag_1']
df['level2_diag1'] = df['diag_1']
df['level1_diag2'] = df['diag_2']
df['level2_diag2'] = df['diag_2']
df['level1_diag3'] = df['diag_3']
df['level2_diag3'] = df['diag_3']
```

In [23]:

```
df.loc[df['diag_1'].str.contains('V'), ['level1_diag1', 'level2_diag1']] = 0
df.loc[df['diag_1'].str.contains('E'), ['level1_diag1', 'level2_diag1']] = 0
df.loc[df['diag_2'].str.contains('V'), ['level1_diag2', 'level2_diag2']] = 0
df.loc[df['diag_2'].str.contains('E'), ['level1_diag2', 'level2_diag2']] = 0
df.loc[df['diag_3'].str.contains('V'), ['level1_diag3', 'level2_diag3']] = 0
df.loc[df['diag_3'].str.contains('E'), ['level1_diag3', 'level2_diag3']] = 0
df['level1_diag1'] = df['level1_diag1'].replace('?', -1)
df['level2_diag1'] = df['level2_diag1'].replace('?', -1)
df['level1_diag2'] = df['level1_diag2'].replace('?', -1)
df['level2_diag2'] = df['level2_diag2'].replace('?', -1)
df['level1_diag3'] = df['level1_diag3'].replace('?', -1)
```

```
df['level2_diag3'] = df['level2_diag3'].replace('?', -1)
```

In [24]:

```
df['level1_diag1'] = df['level1_diag1'].astype(float)

df['level2_diag1'] = df['level2_diag1'].astype(float)

df['level1_diag2'] = df['level1_diag2'].astype(float)

df['level2_diag2'] = df['level2_diag2'].astype(float)

df['level1_diag3'] = df['level1_diag3'].astype(float)

df['level2_diag3'] = df['level2_diag3'].astype(float)
```

In [25]:

```
for index, row in df.iterrows():

    if (row['level1_diag1'] >= 390 and row['level1_diag1'] < 460) or (np.floor(row[
'level1_diag1']) == 785):

        df.loc[index, 'level1_diag1'] = 1

    elif (row['level1_diag1'] >= 460 and row['level1_diag1'] < 520) or (np.floor(ro
w['level1_diag1']) == 786):

        df.loc[index, 'level1_diag1'] = 2

    elif (row['level1_diag1'] >= 520 and row['level1_diag1'] < 580) or (np.floor(ro
w['level1_diag1']) == 787):

        df.loc[index, 'level1_diag1'] = 3

    elif (np.floor(row['level1_diag1']) == 250):

        df.loc[index, 'level1_diag1'] = 4

    elif (row['level1_diag1'] >= 800 and row['level1_diag1'] < 1000):

        df.loc[index, 'level1_diag1'] = 5

    elif (row['level1_diag1'] >= 710 and row['level1_diag1'] < 740):

        df.loc[index, 'level1_diag1'] = 6

    elif (row['level1_diag1'] >= 580 and row['level1_diag1'] < 630) or (np.floor(ro
w['level1_diag1']) == 788):

        df.loc[index, 'level1_diag1'] = 7

    elif (row['level1_diag1'] >= 140 and row['level1_diag1'] < 240):

        df.loc[index, 'level1_diag1'] = 8

    else:

        df.loc[index, 'level1_diag1'] = 0


    if (row['level1_diag2'] >= 390 and row['level1_diag2'] < 460) or (np.floor(row[
'level1_diag2']) == 785):

        df.loc[index, 'level1_diag2'] = 1

    elif (row['level1_diag2'] >= 460 and row['level1_diag2'] < 520) or (np.floor(ro
w['level1_diag2']) == 786):

        df.loc[index, 'level1_diag2'] = 2

    elif (row['level1_diag2'] >= 520 and row['level1_diag2'] < 580) or (np.floor(ro
w['level1_diag2']) == 787):
```

```
        df.loc[index, 'level1_diag2'] = 3
    elif (np.floor(row['level1_diag2']) == 250):
        df.loc[index, 'level1_diag2'] = 4
    elif (row['level1_diag2'] >= 800 and row['level1_diag2'] < 1000):
        df.loc[index, 'level1_diag2'] = 5
    elif (row['level1_diag2'] >= 710 and row['level1_diag2'] < 740):
        df.loc[index, 'level1_diag2'] = 6
    elif (row['level1_diag2'] >= 580 and row['level1_diag2'] < 630) or (np.floor(ro
w['level1_diag2']) == 788):
        df.loc[index, 'level1_diag2'] = 7
    elif (row['level1_diag2'] >= 140 and row['level1_diag2'] < 240):
        df.loc[index, 'level1_diag2'] = 8
    else:
        df.loc[index, 'level1_diag2'] = 0


    if (row['level1_diag3'] >= 390 and row['level1_diag3'] < 460) or (np.floor(row[
'level1_diag3']) == 785):
        df.loc[index, 'level1_diag3'] = 1
    elif (row['level1_diag3'] >= 460 and row['level1_diag3'] < 520) or (np.floor(ro
w['level1_diag3']) == 786):
        df.loc[index, 'level1_diag3'] = 2
    elif (row['level1_diag3'] >= 520 and row['level1_diag3'] < 580) or (np.floor(ro
w['level1_diag3']) == 787):
        df.loc[index, 'level1_diag3'] = 3
    elif (np.floor(row['level1_diag3']) == 250):
        df.loc[index, 'level1_diag3'] = 4
    elif (row['level1_diag3'] >= 800 and row['level1_diag3'] < 1000):
        df.loc[index, 'level1_diag3'] = 5
    elif (row['level1_diag3'] >= 710 and row['level1_diag3'] < 740):
        df.loc[index, 'level1_diag3'] = 6
    elif (row['level1_diag3'] >= 580 and row['level1_diag3'] < 630) or (np.floor(ro
w['level1_diag3']) == 788):
        df.loc[index, 'level1_diag3'] = 7
    elif (row['level1_diag3'] >= 140 and row['level1_diag3'] < 240):
        df.loc[index, 'level1_diag3'] = 8
    else:
        df.loc[index, 'level1_diag3'] = 0
```

In [26]:

```
for index, row in df.iterrows():
    if (row['level2_diag1'] >= 390 and row['level2_diag1'] < 399):
        df.loc[index, 'level2_diag1'] = 1
    elif (row['level2_diag1'] >= 401 and row['level2_diag1'] < 415):
```

```
        df.loc[index, 'level2_diag1'] = 2
    elif (row['level2_diag1'] >= 415 and row['level2_diag1'] < 460):
        df.loc[index, 'level2_diag1'] = 3
    elif (np.floor(row['level2_diag1']) == 785):
        df.loc[index, 'level2_diag1'] = 4
    elif (row['level2_diag1'] >= 460 and row['level2_diag1'] < 489):
        df.loc[index, 'level2_diag1'] = 5
    elif (row['level2_diag1'] >= 490 and row['level2_diag1'] < 497):
        df.loc[index, 'level2_diag1'] = 6
    elif (row['level2_diag1'] >= 500 and row['level2_diag1'] < 520):
        df.loc[index, 'level2_diag1'] = 7
    elif (np.floor(row['level2_diag1']) == 786):
        df.loc[index, 'level2_diag1'] = 8
    elif (row['level2_diag1'] >= 520 and row['level2_diag1'] < 530):
        df.loc[index, 'level2_diag1'] = 9
    elif (row['level2_diag1'] >= 530 and row['level2_diag1'] < 544):
        df.loc[index, 'level2_diag1'] = 10
    elif (row['level2_diag1'] >= 550 and row['level2_diag1'] < 554):
        df.loc[index, 'level2_diag1'] = 11
    elif (row['level2_diag1'] >= 555 and row['level2_diag1'] < 580):
        df.loc[index, 'level2_diag1'] = 12
    elif (np.floor(row['level2_diag1']) == 787):
        df.loc[index, 'level2_diag1'] = 13
    elif (np.floor(row['level2_diag1']) == 250):
        df.loc[index, 'level2_diag1'] = 14
    elif (row['level2_diag1'] >= 800 and row['level2_diag1'] < 1000):
        df.loc[index, 'level2_diag1'] = 15
    elif (row['level2_diag1'] >= 710 and row['level2_diag1'] < 740):
        df.loc[index, 'level2_diag1'] = 16
    elif (row['level2_diag1'] >= 580 and row['level2_diag1'] < 630):
        df.loc[index, 'level2_diag1'] = 17
    elif (np.floor(row['level2_diag1']) == 788):
        df.loc[index, 'level2_diag1'] = 18
    elif (row['level2_diag1'] >= 140 and row['level2_diag1'] < 240):
        df.loc[index, 'level2_diag1'] = 19
    elif row['level2_diag1'] >= 240 and row['level2_diag1'] < 280 and (np.floor(row
['level2_diag1']) != 250):
        df.loc[index, 'level2_diag1'] = 20
    elif (row['level2_diag1'] >= 680 and row['level2_diag1'] < 710) or (np.floor(ro
w['level2_diag1']) == 782):
        df.loc[index, 'level2_diag1'] = 21
    elif (row['level2_diag1'] >= 290 and row['level2_diag1'] < 320):
```

```
        df.loc[index, 'level2_diag1'] = 22
    else:
        df.loc[index, 'level2_diag1'] = 0

    if (row['level2_diag2'] >= 390 and row['level2_diag2'] < 399):
        df.loc[index, 'level2_diag2'] = 1
    elif (row['level2_diag2'] >= 401 and row['level2_diag2'] < 415):
        df.loc[index, 'level2_diag2'] = 2
    elif (row['level2_diag2'] >= 415 and row['level2_diag2'] < 460):
        df.loc[index, 'level2_diag2'] = 3
    elif (np.floor(row['level2_diag2']) == 785):
        df.loc[index, 'level2_diag2'] = 4
    elif (row['level2_diag2'] >= 460 and row['level2_diag2'] < 489):
        df.loc[index, 'level2_diag2'] = 5
    elif (row['level2_diag2'] >= 490 and row['level2_diag2'] < 497):
        df.loc[index, 'level2_diag2'] = 6
    elif (row['level2_diag2'] >= 500 and row['level2_diag2'] < 520):
        df.loc[index, 'level2_diag2'] = 7
    elif (np.floor(row['level2_diag2']) == 786):
        df.loc[index, 'level2_diag2'] = 8
    elif (row['level2_diag2'] >= 520 and row['level2_diag2'] < 530):
        df.loc[index, 'level2_diag2'] = 9
    elif (row['level2_diag2'] >= 530 and row['level2_diag2'] < 544):
        df.loc[index, 'level2_diag2'] = 10
    elif (row['level2_diag2'] >= 550 and row['level2_diag2'] < 554):
        df.loc[index, 'level2_diag2'] = 11
    elif (row['level2_diag2'] >= 555 and row['level2_diag2'] < 580):
        df.loc[index, 'level2_diag2'] = 12
    elif (np.floor(row['level2_diag2']) == 787):
        df.loc[index, 'level2_diag2'] = 13
    elif (np.floor(row['level2_diag2']) == 250):
        df.loc[index, 'level2_diag2'] = 14
    elif (row['level2_diag2'] >= 800 and row['level2_diag2'] < 1000):
        df.loc[index, 'level2_diag2'] = 15
    elif (row['level2_diag2'] >= 710 and row['level2_diag2'] < 740):
        df.loc[index, 'level2_diag2'] = 16
    elif (row['level2_diag2'] >= 580 and row['level2_diag2'] < 630):
        df.loc[index, 'level2_diag2'] = 17
    elif (np.floor(row['level2_diag2']) == 788):
        df.loc[index, 'level2_diag2'] = 18
    elif (row['level2_diag2'] >= 140 and row['level2_diag2'] < 240):
```

```
        df.loc[index, 'level2_diag2'] = 19
    elif row['level2_diag2'] >= 240 and row['level2_diag2'] < 280 and (np.floor(row
['level2_diag2']) != 250):
        df.loc[index, 'level2_diag2'] = 20
    elif (row['level2_diag2'] >= 680 and row['level2_diag2'] < 710) or (np.floor(ro
w['level2_diag2']) == 782):
        df.loc[index, 'level2_diag2'] = 21
    elif (row['level2_diag2'] >= 290 and row['level2_diag2'] < 320):
        df.loc[index, 'level2_diag2'] = 22
    else:
        df.loc[index, 'level2_diag2'] = 0



    if (row['level2_diag3'] >= 390 and row['level2_diag3'] < 399):
        df.loc[index, 'level2_diag3'] = 1
    elif (row['level2_diag3'] >= 401 and row['level2_diag3'] < 415):
        df.loc[index, 'level2_diag3'] = 2
    elif (row['level2_diag3'] >= 415 and row['level2_diag3'] < 460):
        df.loc[index, 'level2_diag3'] = 3
    elif (np.floor(row['level2_diag3']) == 785):
        df.loc[index, 'level2_diag3'] = 4
    elif (row['level2_diag3'] >= 460 and row['level2_diag3'] < 489):
        df.loc[index, 'level2_diag3'] = 5
    elif (row['level2_diag3'] >= 490 and row['level2_diag3'] < 497):
        df.loc[index, 'level2_diag3'] = 6
    elif (row['level2_diag3'] >= 500 and row['level2_diag3'] < 520):
        df.loc[index, 'level2_diag3'] = 7
    elif (np.floor(row['level2_diag3']) == 786):
        df.loc[index, 'level2_diag3'] = 8
    elif (row['level2_diag3'] >= 520 and row['level2_diag3'] < 530):
        df.loc[index, 'level2_diag3'] = 9
    elif (row['level2_diag3'] >= 530 and row['level2_diag3'] < 544):
        df.loc[index, 'level2_diag3'] = 10
    elif (row['level2_diag3'] >= 550 and row['level2_diag3'] < 554):
        df.loc[index, 'level2_diag3'] = 11
    elif (row['level2_diag3'] >= 555 and row['level2_diag3'] < 580):
        df.loc[index, 'level2_diag3'] = 12
    elif (np.floor(row['level2_diag3']) == 787):
        df.loc[index, 'level2_diag3'] = 13
    elif (np.floor(row['level2_diag3']) == 250):
        df.loc[index, 'level2_diag3'] = 14
    elif (row['level2_diag3'] >= 800 and row['level2_diag3'] < 1000):
```

```
        df.loc[index, 'level2_diag3'] = 15
    elif (row['level2_diag3'] >= 710 and row['level2_diag3'] < 740):
        df.loc[index, 'level2_diag3'] = 16
    elif (row['level2_diag3'] >= 580 and row['level2_diag3'] < 630):
        df.loc[index, 'level2_diag3'] = 17
    elif (np.floor(row['level2_diag3']) == 788):
        df.loc[index, 'level2_diag3'] = 18
    elif (row['level2_diag3'] >= 140 and row['level2_diag3'] < 240):
        df.loc[index, 'level2_diag3'] = 19
    elif row['level2_diag3'] >= 240 and row['level2_diag3'] < 280 and (np.floor(row
['level2_diag3']) != 250):
        df.loc[index, 'level2_diag3'] = 20
    elif (row['level2_diag3'] >= 680 and row['level2_diag3'] < 710) or (np.floor(ro
w['level2_diag3']) == 782):
        df.loc[index, 'level2_diag3'] = 21
    elif (row['level2_diag3'] >= 290 and row['level2_diag3'] < 320):
        df.loc[index, 'level2_diag3'] = 22
    else:
        df.loc[index, 'level2_diag3'] = 0
```

Data Visualization¶

Distribution of Readmission¶

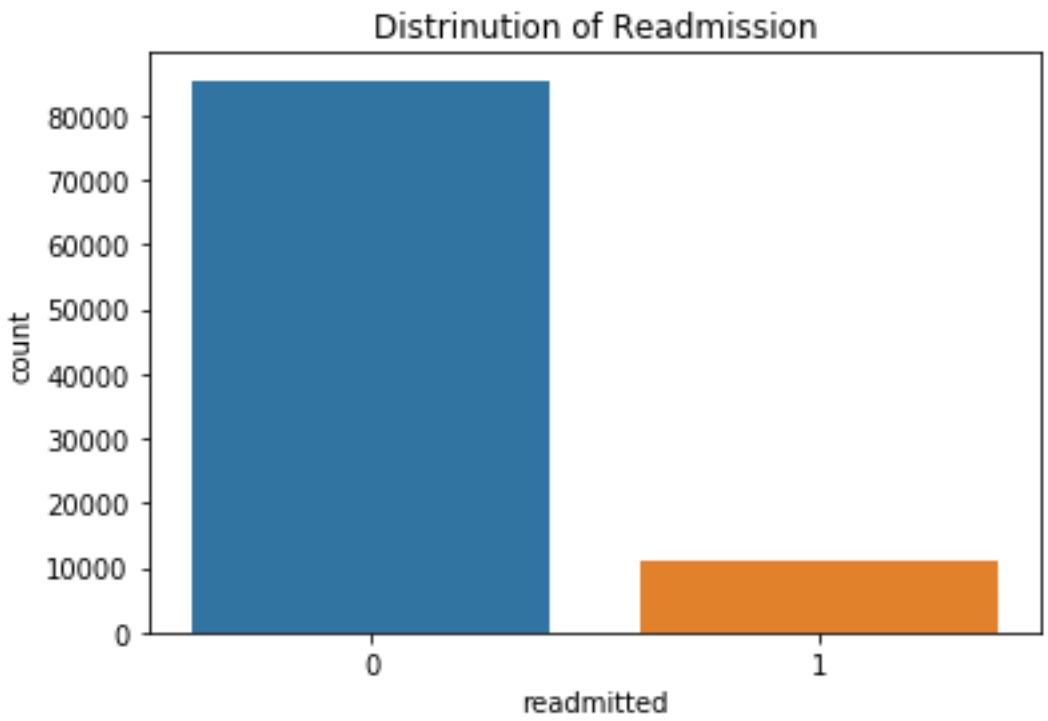
Our target variable is imbalance. Number of readmitted patient are quite less as compared to Not readmitted

In [27]:

```
# Distribution of Readmission
sns.countplot(df['readmitted']).set_title('Distrinution of Readmission')
```

Out[27]:

```
Text(0.5, 1.0, 'Distrinution of Readmission')
```



Time in Hospital and Readmission¶

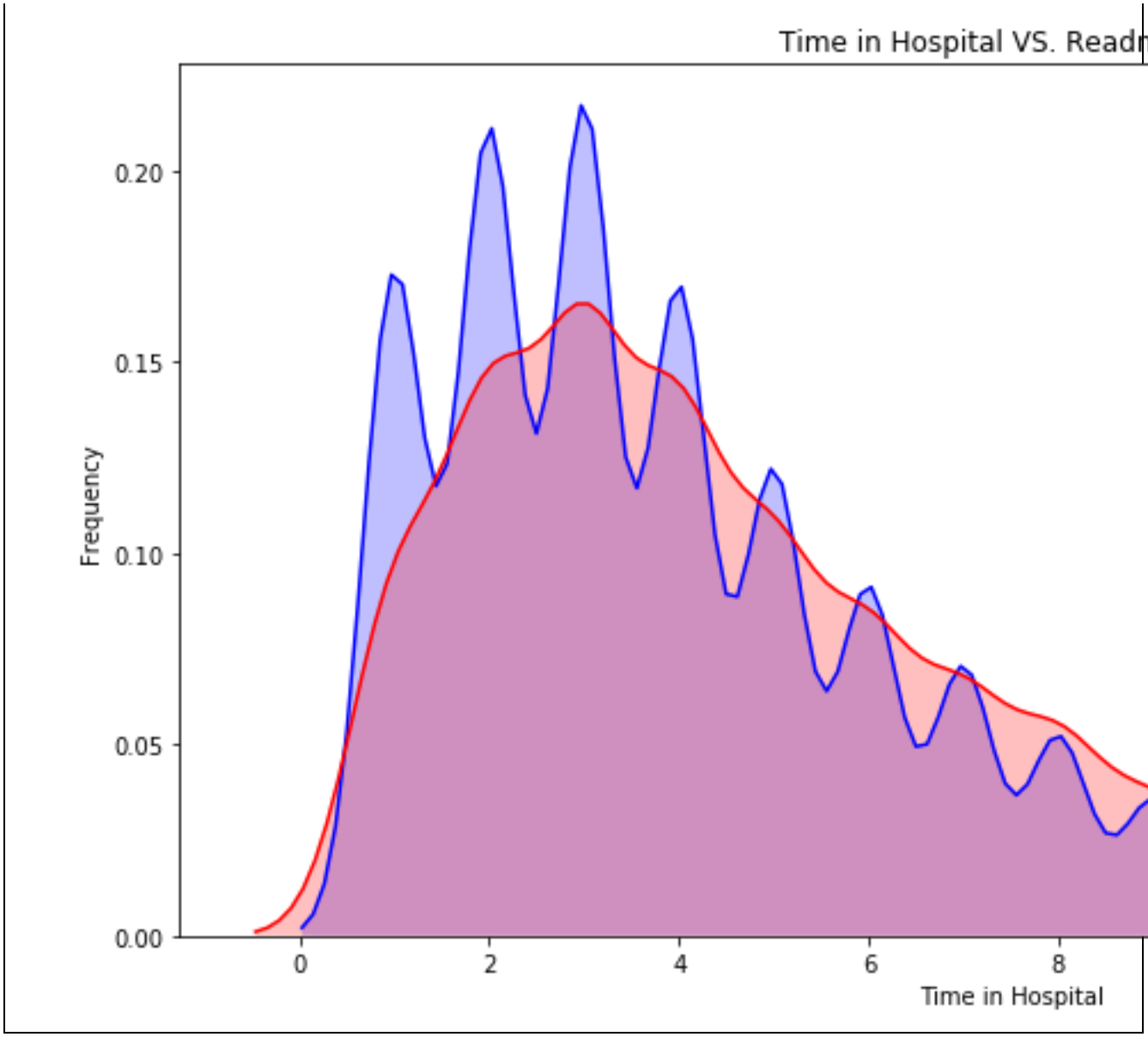
In [28]:

```
fig = plt.figure(figsize=(13,7),)

ax=sns.kdeplot(df.loc[(df['readmitted'] == 0),'time_in_hospital'] , color='b',shade
=True,label='Not Readmitted')

ax=sns.kdeplot(df.loc[(df['readmitted'] == 1),'time_in_hospital'] , color='r',shade
=True, label='Readmitted')

ax.set(xlabel='Time in Hospital', ylabel='Frequency')

plt.title('Time in Hospital VS. Readmission')
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarning: Us
ing a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tupl
e(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array
index, `arr[np.array(seq)]`, which will result either in an error or a different re
sult.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[28]:

```
Text(0.5, 1.0, 'Time in Hospital VS. Readmission')
```



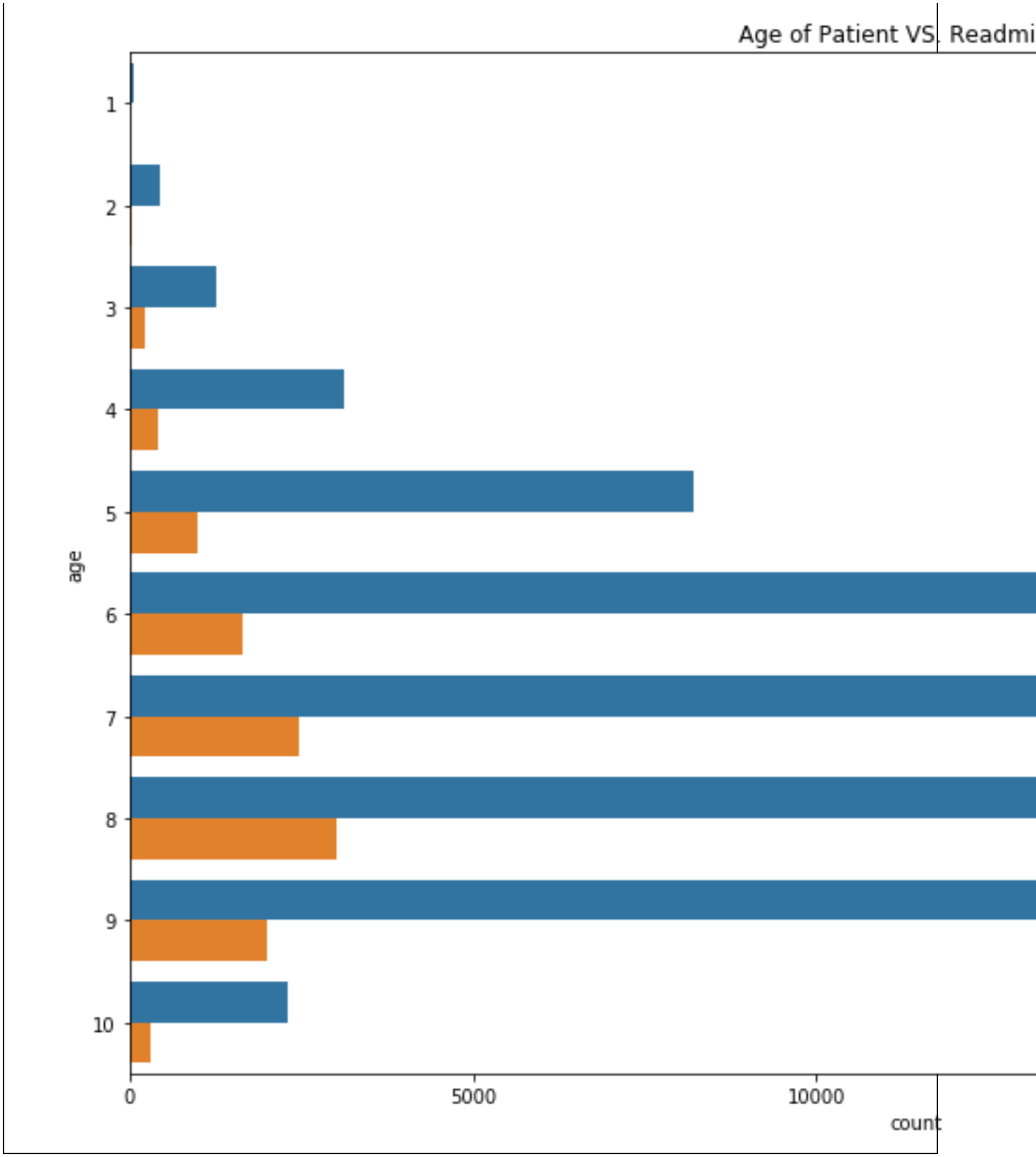
Age and Readmission¶

In [29]:

```
fig = plt.figure(figsize=(15,10))

sns.countplot(y= df['age'], hue = df['readmitted']).set_title('Age of Patient VS. Readmission')
```

Out[29]:

```
Text(0.5, 1.0, 'Age of Patient VS. Readmission')
```



Ethnicity of patient and Readmission¶

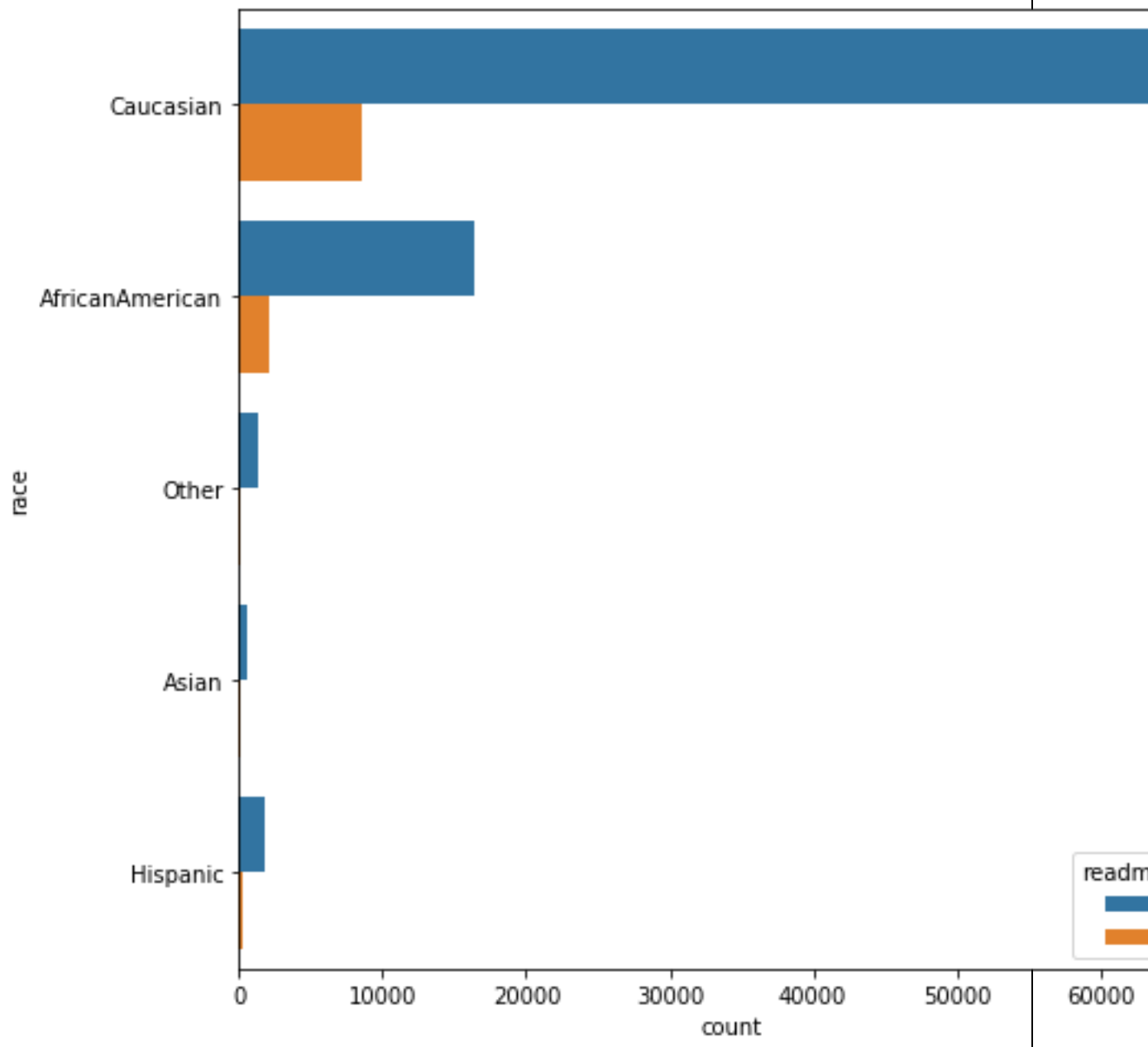
In [30]:

```
fig = plt.figure(figsize=(8,8))
```

```
sns.countplot(y = df['race'], hue = df['readmitted'])
```

Out[30]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fba49d1d0>
```



Number of medication used and Readmission¶

In [31]:

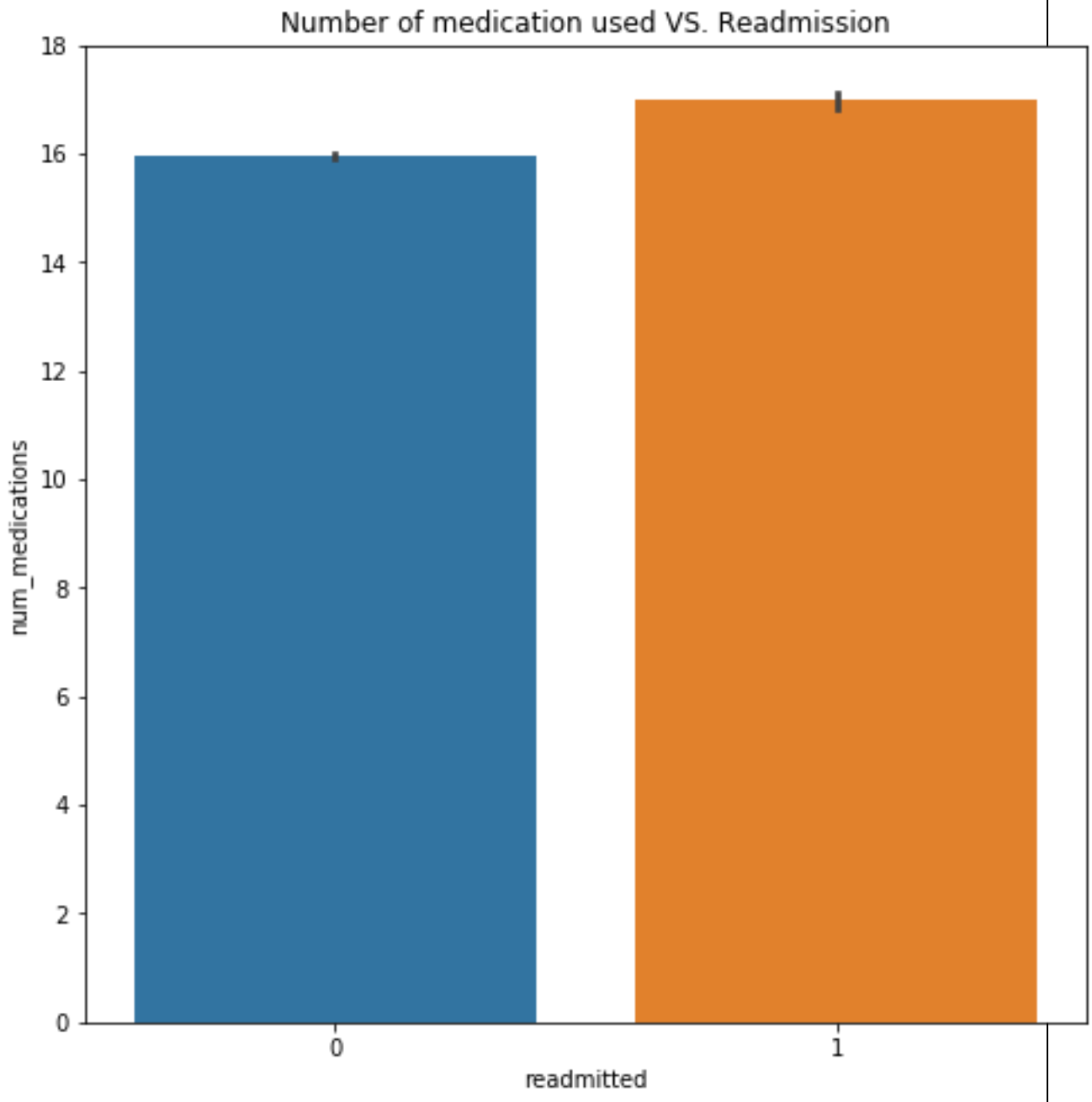
```
fig = plt.figure(figsize=(8,8))

sns.barplot(x = df['readmitted'], y = df['num_medications']).set_title("Number of m
edication used VS. Readmission")
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarning: Us
ing a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tupl
e(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array
```

```
index, `arr[np.array(seq)]`, which will result either in an error or a different re
sult.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[31]:

```
Text(0.5, 1.0, 'Number of medication used VS. Readmission')
```



Gender and Readmission¶

- Male = 1
- Female = 0

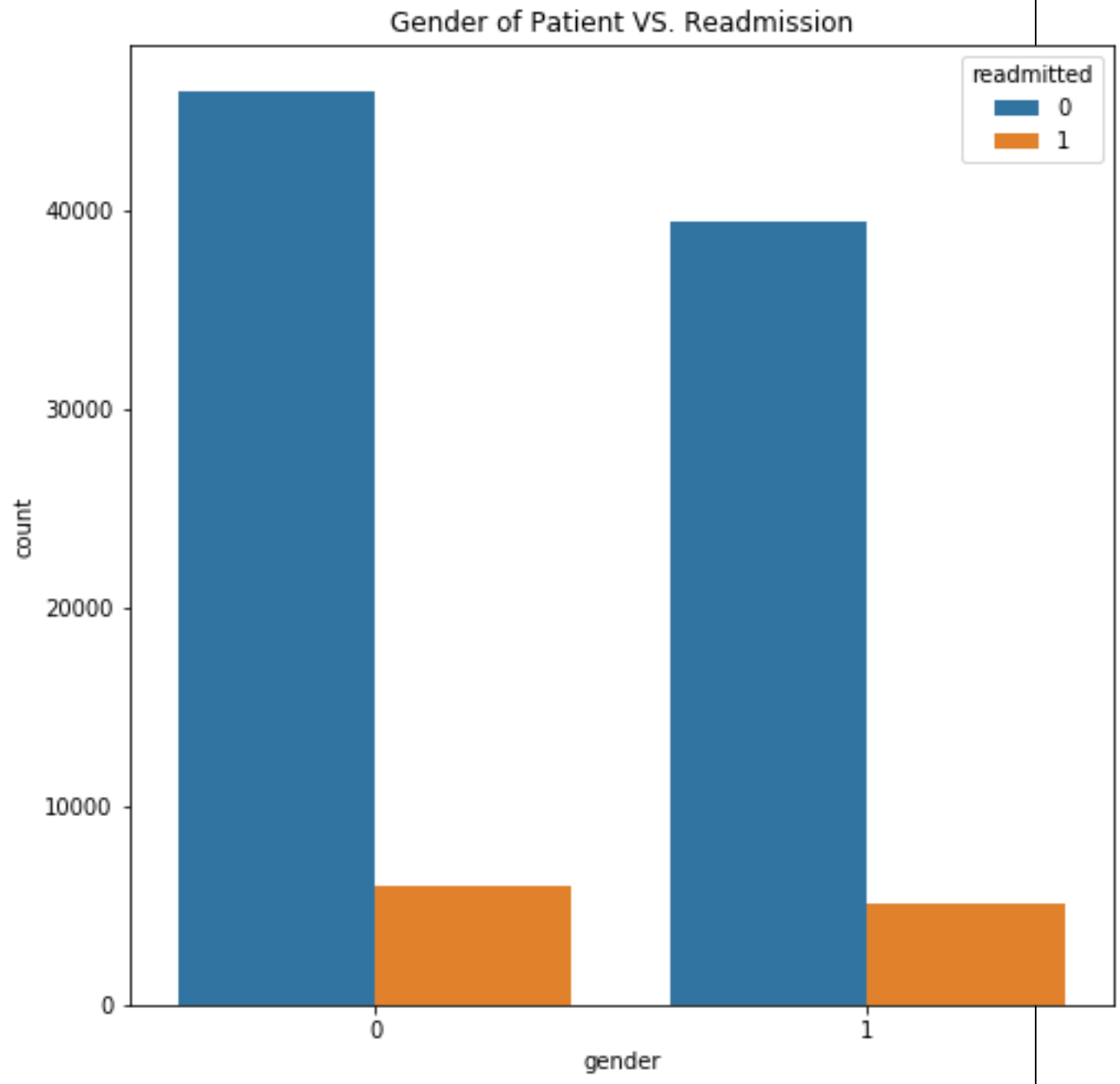
In [32]:

```
fig = plt.figure(figsize=(8,8))
```

```
sns.countplot(df['gender'], hue = df['readmitted']).set_title("Gender of Patient VS
. Readmission")
```

Out[32]:

```
Text(0.5, 1.0, 'Gender of Patient VS. Readmission')
```



Change of Medication and Readmission¶

- Change = 1
- No Change = 0

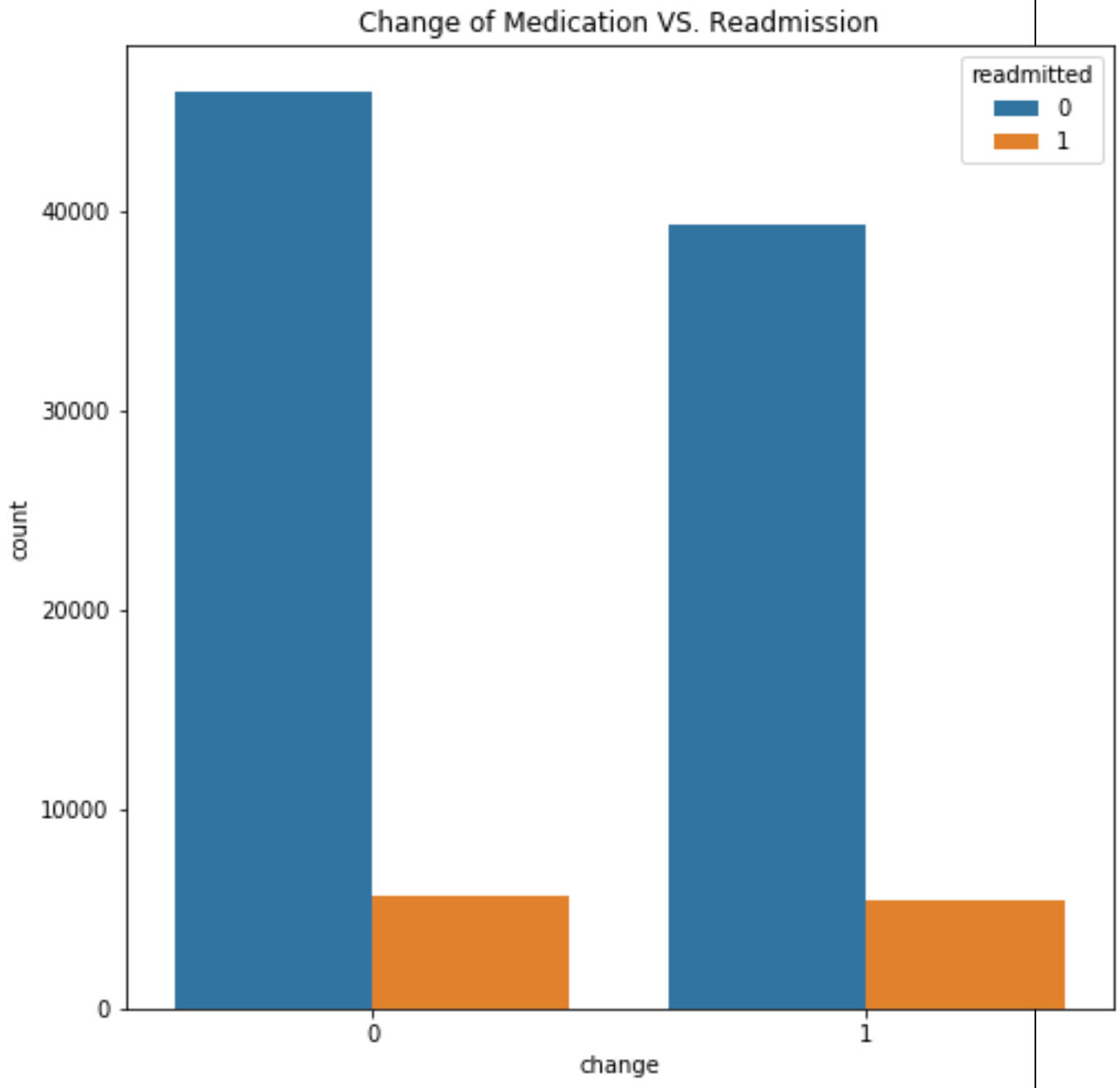
In [33]:

```
fig = plt.figure(figsize=(8,8))
```

```
sns.countplot(df['change'], hue = df['readmitted']).set_title('Change of Medication
VS. Readmission')
```

Out[33]:

```
Text(0.5, 1.0, 'Change of Medication VS. Readmission')
```



Diabetes Medication prescribed and Readmission¶

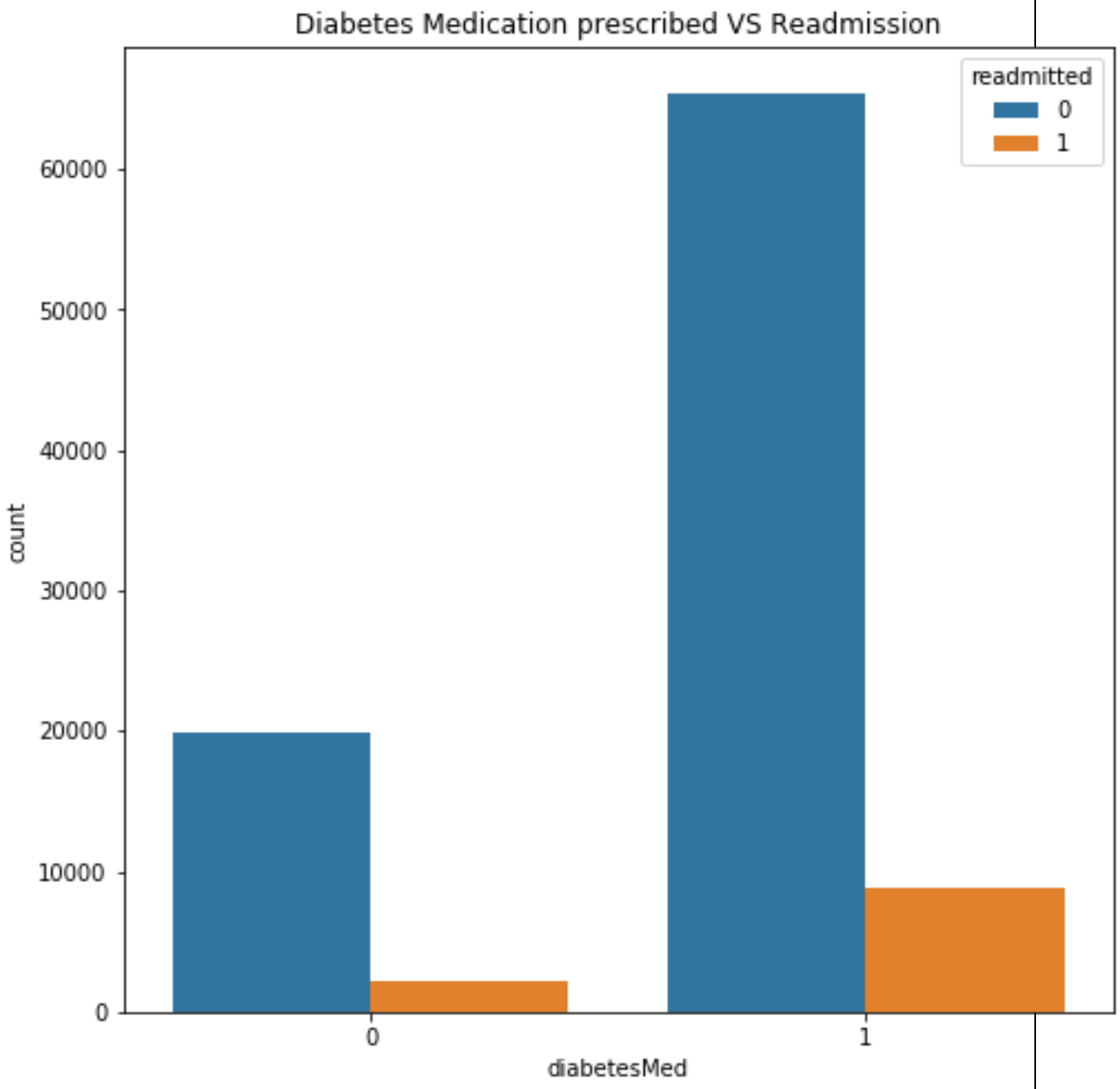
- Diabetes Medication - medications Nominal Indicates if there was any diabetic medication prescribed.
- Values: “yes” : 1 “no” : 0

In [34]:

```
fig = plt.figure(figsize=(8,8))

sns.countplot(df['diabetesMed'], hue = df['readmitted']).set_title('Diabetes Medica
tion prescribed VS Readmission')
```

Out[34]:

```
Text(0.5, 1.0, 'Diabetes Medication prescribed VS Readmission')
```



Service Utilization and Readmission¶

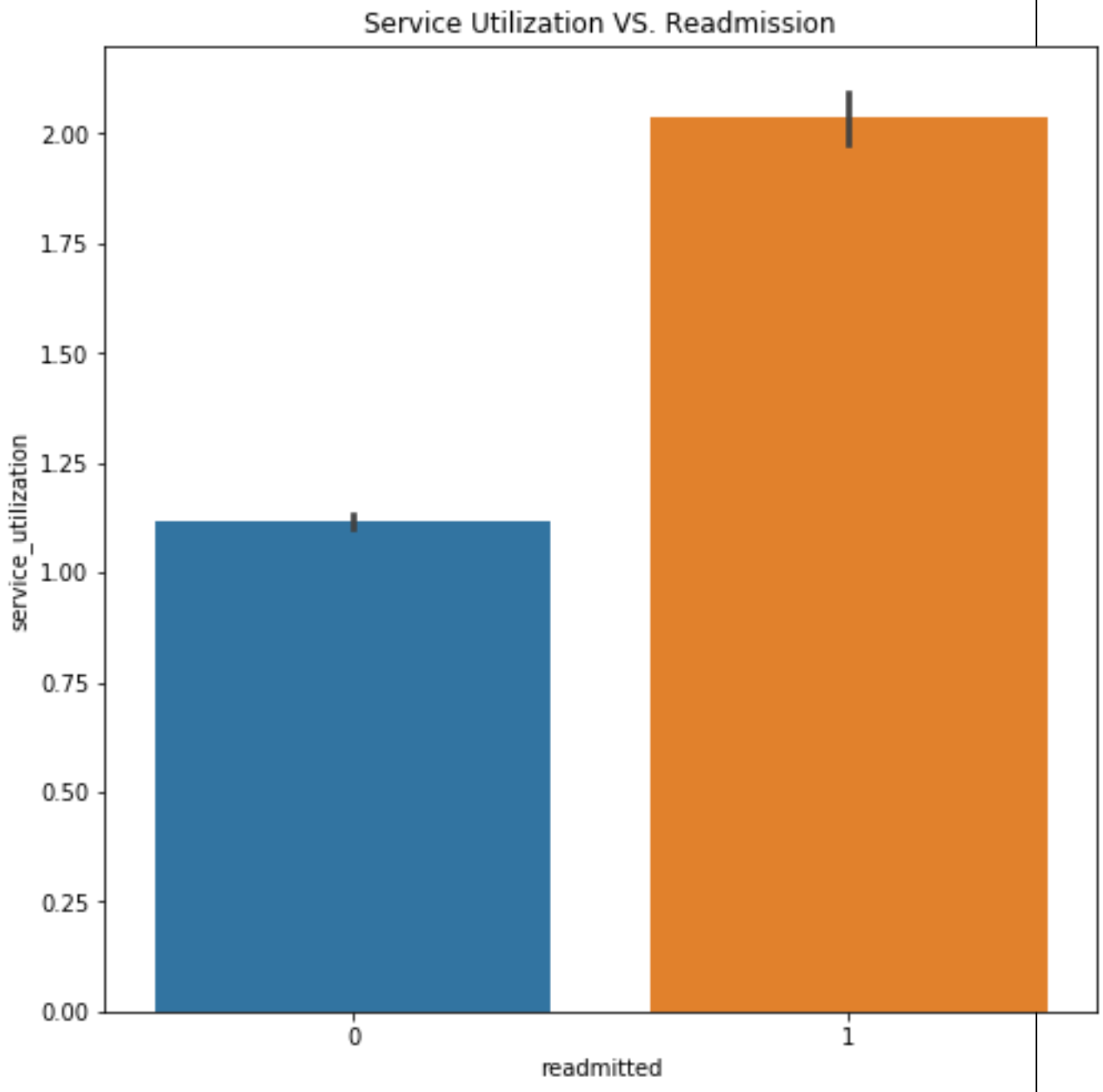
In [35]:

```
fig = plt.figure(figsize=(8,8))

sns.barplot( y = df['service_utilization'], x = df['readmitted']).set_title('Servic
e Utilization VS. Readmission')
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarning: Us
ing a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tupl
e(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array
index, `arr[np.array(seq)]`, which will result either in an error or a different re
sult.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[35]:

```
Text(0.5, 1.0, 'Service Utilization VS. Readmission')
```



Glucose serum test result and Readmission¶

Glucose Serum test - A blood glucose test is used to find out if your blood sugar levels are in the healthy range. It is often used to help diagnose and monitor diabetes.

- '>200' : 1 = indicates diabetes
- '>300' : 1 = Indicates diabetes

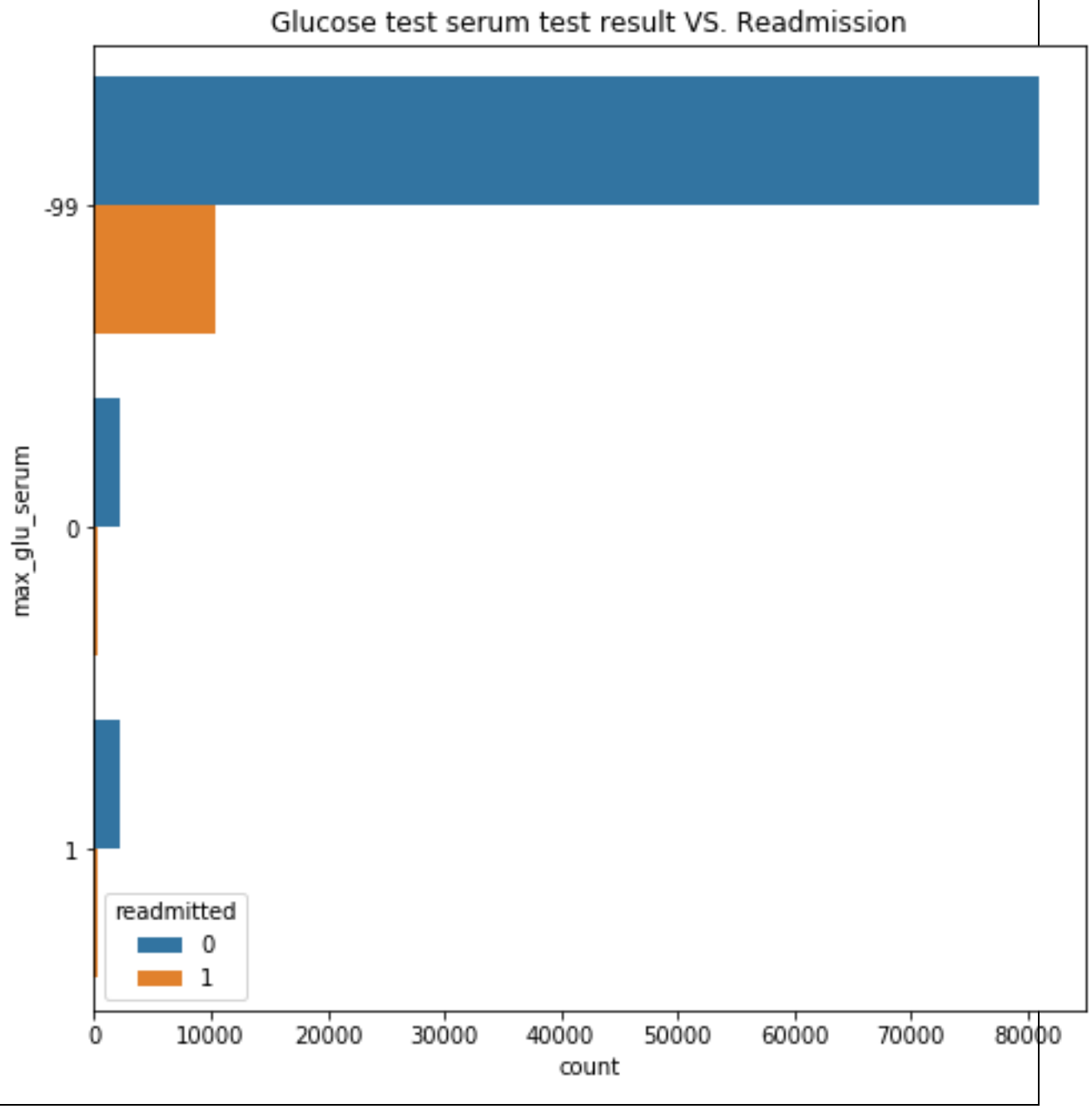
- 'Norm' : 0 = Normal
- 'None' : -99 = test was not taken

In [36]:

```
fig = plt.figure(figsize=(8,8))

sns.countplot(y = df['max_glu_serum'], hue = df['readmitted']).set_title('Glucose t
est serum test result VS. Readmission')
```

Out[36]:

```
Text(0.5, 1.0, 'Glucose test serum test result VS. Readmission')
```



A1C result and Readmission *A1C test* - The A1C test is a blood test that provides information about your average levels of blood glucose, also called blood sugar, over the past 3 months.

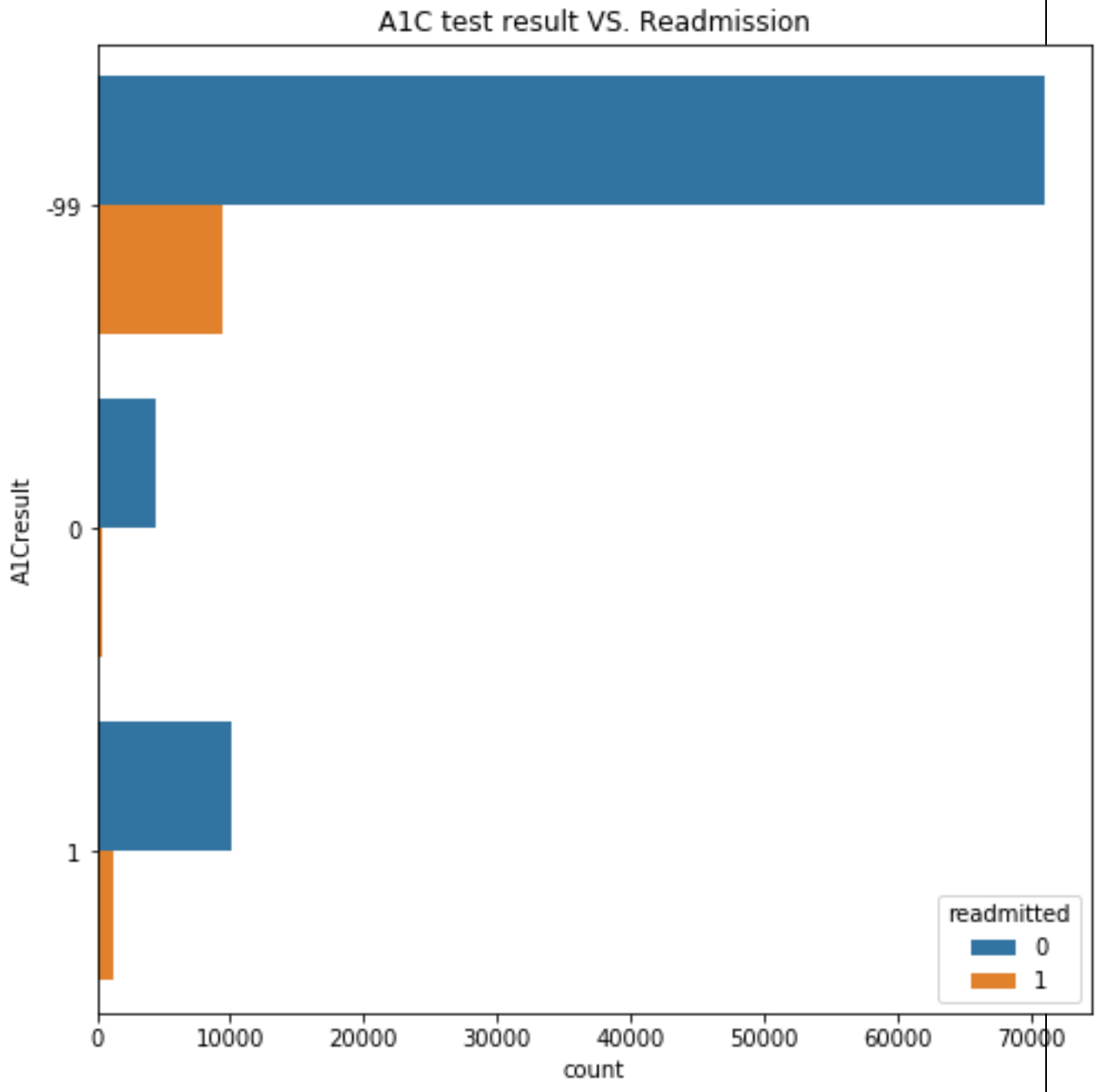
- '>7' : 1
- '>8' : 1
- Norm : 0 = Normal
- None : -99 = Test was not taken

In [37]:

```
fig = plt.figure(figsize=(8,8))

sns.countplot(y= df['A1Cresult'], hue = df['readmitted']).set_title('A1C test resul
t VS. Readmission')
```

Out[37]:

```
Text(0.5, 1.0, 'A1C test result VS. Readmission')
```



Number of lab procedure and Readmission¶

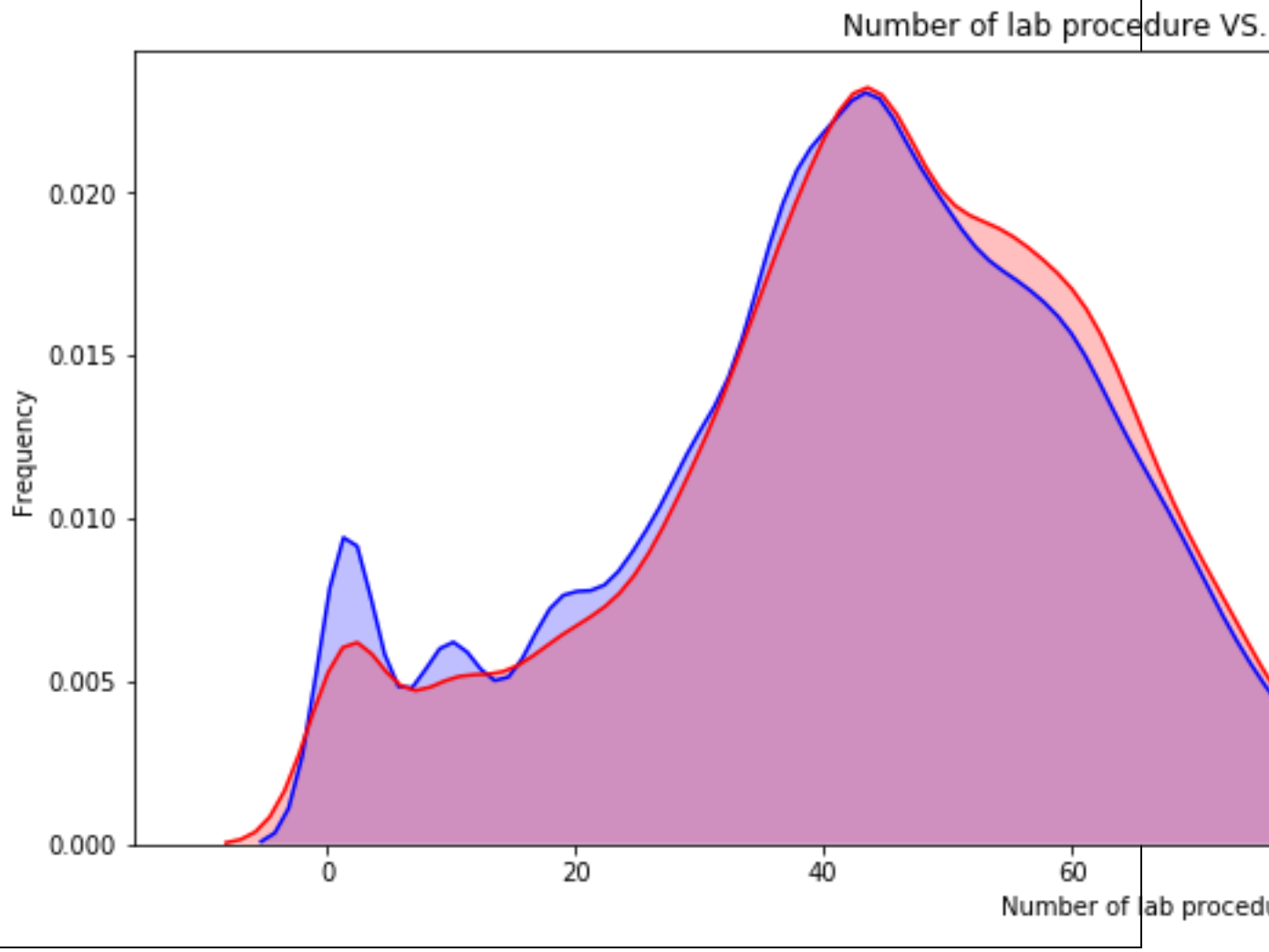
In [38]:

```
fig = plt.figure(figsize=(15,6),)

ax=sns.kdeplot(df.loc[(df['readmitted'] == 0),'num_lab_procedures'] , color='b',shade=True,label='Not readmitted')

ax=sns.kdeplot(df.loc[(df['readmitted'] == 1),'num_lab_procedures'] , color='r',shade=True, label='readmitted')

ax.set(xlabel='Number of lab procedure', ylabel='Frequency')

plt.title('Number of lab procedure VS. Readmission')
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[38]:

```
Text(0.5, 1.0, 'Number of lab procedure VS. Readmission')
```



Pre-Modeling Data Preprocessing¶

This code converts age as categorical variable to a continuous approximation by assuming mid-point of each age-category as the actual age value. This is done to avoid having to deal with age as a dummy variable in the models which makes interpretation very cumbersome. Also, since age category is not purely nominal but ordinal, we do not want to lose that information by treating it as a simple categorical variable

In [39]:

```
df['age'] = df['age'].astype('int64')

print(df.age.value_counts())

# convert age categories to mid-point values

age_dict = {1:5, 2:15, 3:25, 4:35, 5:45, 6:55, 7:65, 8:75, 9:85, 10:95}

df['age'] = df.age.map(age_dict)

print(df.age.value_counts())
```

```
8     24815
7     21521
6     16546
9     16223
5      9208
4      3538
10     2594
3      1471
2       466
1        64
Name: age, dtype: int64
75    24815
65    21521
55    16546
85    16223
45     9208
35     3538
95     2594
25     1471
15      466
5        64
Name: age, dtype: int64
```

In [40]:

```
# convert data type of nominal features in dataframe to 'object' type

i = ['encounter_id', 'patient_nbr', 'gender', 'admission_type_id', 'discharge_dispo
sition_id', 'admission_source_id',\

          'A1Cresult', 'metformin', 'repaglinide', 'nateglinide', 'chlorpropamide',
'glimepiride', 'acetohexamide', \

          'glipizide', 'glyburide', 'tolbutamide', 'pioglitazone', 'rosiglitazone',
'acarbose','miglitol', \

          'troglitazone', 'tolazamide', 'insulin', 'glyburide-metformin', 'glipizid
e-metformin', \

          'glimepiride-pioglitazone', 'metformin-rosiglitazone', 'metformin-pioglit
azone', 'change', 'diabetesMed', \

          'age', 'A1Cresult', 'max_glu_serum', 'level1_diag1', 'level1_diag2', 'lev
el1_diag3', 'level2_diag1', 'level2_diag2', 'level2_diag3']


df[i] = df[i].astype('object')
```

In [41]:

```
df.dtypes
```

Out[41]:

```
encounter_id                object
patient_nbr                 object
race                        object
gender                      object
age                         object
admission_type_id           object
discharge_disposition_id    object
admission_source_id         object
time_in_hospital             int64
num_lab_procedures           int64
num_procedures               int64
num_medications              int64
number_outpatient            int64
number_emergency             int64
number_inpatient             int64
diag_1                      object
diag_2                      object
diag_3                      object
number_diagnoses             int64
max_glu_serum               object
A1Cresult                   object
metformin                   object
repaglinide                 object
nateglinide                 object
chlorpropamide              object
glimepiride                 object
acetohexamide               object
glipizide                   object
glyburide                   object
tolbutamide                 object
pioglitazone                object
rosiglitazone               object
acarbose                    object
miglitol                    object
troglitazone                object
tolazamide                  object
insulin                     object
glyburide-metformin         object
glipizide-metformin         object
glimepiride-pioglitazone    object
metformin-rosiglitazone     object
metformin-pioglitazone      object
change                      object
diabetesMed                 object
readmitted                   int64
service_utilization          int64
numchange                    int64
level1_diag1                object
level2_diag1                object
level1_diag2                object
level2_diag2                object
level1_diag3                object
level2_diag3                object
dtype: object
```

- Number of medication used: Another possibly related factor could be the total number of medications used by the patient (which may indicate severity of their condition and/or the

intensity of care). So we created another feature by counting the medications used during the encounter (keys variable in code below is continued from above):

In [42]:

```
df['nummed'] = 0


for col in keys:
    df['nummed'] = df['nummed'] + df[col]
df['nummed'].value_counts()
```

Out[42]:

```
1    44589
0    22156
2    20901
3     7448
4     1290
5       57
6        5
Name: nummed, dtype: int64
```

In [43]:

```
# get list of only numeric features
num_col = list(set(list(df._get_numeric_data().columns))- {'readmitted'})
num_col
```

Out[43]:

```
['service_utilization',
 'number_outpatient',
 'num_procedures',
 'num_medications',
 'number_emergency',
 'time_in_hospital',
 'number_inpatient',
 'num_lab_procedures',
 'number_diagnoses',
 'numchange']
```

In [44]:

```
# Removing skewnewss and kurtosis using log transformation if it is above a thresho
ld value -  2


statdataframe = pd.DataFrame()
statdataframe['numeric_column'] = num_col
skew_before = []
skew_after = []


kurt_before = []
kurt_after = []


standard_deviation_before = []
```

```
standard_deviation_after = []

log_transform_needed = []

log_type = []

for i in num_col:
    skewval = df[i].skew()
    skew_before.append(skewval)

    kurtval = df[i].kurtosis()
    kurt_before.append(kurtval)

    sdval = df[i].std()
    standard_deviation_before.append(sdval)

    if (abs(skewval) >2) & (abs(kurtval) >2):
        log_transform_needed.append('Yes')

        if len(df[df[i] == 0])/len(df) <=0.02:
            log_type.append('log')
            skewvalnew = np.log(pd.DataFrame(df[train_data[i] > 0])[i]).skew()
            skew_after.append(skewvalnew)

            kurtvalnew = np.log(pd.DataFrame(df[train_data[i] > 0])[i]).kurtosis()
            kurt_after.append(kurtvalnew)

            sdvalnew = np.log(pd.DataFrame(df[train_data[i] > 0])[i]).std()
            standard_deviation_after.append(sdvalnew)

        else:
            log_type.append('log1p')
            skewvalnew = np.log1p(pd.DataFrame(df[df[i] >= 0])[i]).skew()
            skew_after.append(skewvalnew)

            kurtvalnew = np.log1p(pd.DataFrame(df[df[i] >= 0])[i]).kurtosis()
            kurt_after.append(kurtvalnew)

            sdvalnew = np.log1p(pd.DataFrame(df[df[i] >= 0])[i]).std()
            standard_deviation_after.append(sdvalnew)
```

```
    else:
        log_type.append('NA')
        log_transform_needed.append('No')


        skew_after.append(skewval)
        kurt_after.append(kurtval)
        standard_deviation_after.append(sdval)


statdataframe['skew_before'] = skew_before
statdataframe['kurtosis_before'] = kurt_before
statdataframe['standard_deviation_before'] = standard_deviation_before
statdataframe['log_transform_needed'] = log_transform_needed
statdataframe['log_type'] = log_type
statdataframe['skew_after'] = skew_after
statdataframe['kurtosis_after'] = kurt_after
statdataframe['standard_deviation_after'] = standard_deviation_after
```

In [45]:

```
statdataframe
```

Out[45]:

	numeric_column	skew_before	kurtosis_before	standard_deviation_before	log_transform_needed	log_type	s
0	service_utilization	5.312374	67.194018	2.315789	Yes	log1p	1
1	number_outpatient	8.767489	146.244961	1.280061	Yes	log1p	2
2	num_procedures	1.313236	0.856100	1.703183	No	NA	1
3	num_medications	1.339187	3.549325	8.072516	No	NA	1
4	number_emergency	22.695921	1165.140400	0.948089	Yes	log1p	3
5	time_in_hospital	1.127510	0.839050	2.982330	No	NA	1
6	number_inpatient	3.566269	20.044813	1.269975	Yes	log1p	1
7	num_lab_procedures	-0.240626	-0.253275	19.656782	No	NA	-(
8	number_diagnoses	-0.807741	-0.372558	1.836659	No	NA	-(
9	numchange	1.426548	1.451898	0.488614	No	NA	1

In [46]:

```
# performing the log transformation for the columns determined to be needing it abo
ve.


for i in range(len(statdataframe)):
    if statdataframe['log_transform_needed'][i] == 'Yes':
        colname = str(statdataframe['numeric_column'][i])


        if statdataframe['log_type'][i] == 'log':
            df = df[df[colname] > 0]
            df[colname + "_log"] = np.log(df[colname])
```

```
        elif statdataframe['log_type'][i] == 'log1p':
            df = df[df[colname] >= 0]
            df[colname + "_log1p"] = np.log1p(df[colname])
```

In [47]:

```
df = df.drop(['number_outpatient', 'number_inpatient', 'number_emergency','service_
utilization'], axis = 1)
```

In [48]:

```
df.shape
```

Out[48]:

```
(96446, 54)
```

In [49]:

```
# get list of only numeric features
numerics = list(set(list(df._get_numeric_data().columns))- {'readmitted'})
numerics
```

Out[49]:

```
['num_procedures',
 'num_medications',
 'time_in_hospital',
 'service_utilization_log1p',
 'number_inpatient_log1p',
 'num_lab_procedures',
 'number_diagnoses',
 'number_emergency_log1p',
 'numchange',
 'number_outpatient_log1p']
```

In [50]:

```
# show list of features that are categorical
df.encounter_id = df.encounter_id.astype('int64')
df.patient_nbr = df.patient_nbr.astype('int64')
df.diabetesMed = df.diabetesMed.astype('int64')
df.change = df.change.astype('int64')


# convert data type of nominal features in dataframe to 'object' type for aggregati
ng
i = ['metformin', 'repaglinide', 'nateglinide', 'chlorpropamide', 'glimepiride', 'a
cetohexamide', \
          'glipizide', 'glyburide', 'tolbutamide', 'pioglitazone', 'rosiglitazone',
'acarbose','miglitol', \
          'troglitazone', 'tolazamide', 'insulin', 'glyburide-metformin', 'glipizid
e-metformin', \
          'glimepiride-pioglitazone', 'metformin-rosiglitazone', 'metformin-pioglit
azone','A1Cresult']
df[i] = df[i].astype('int64')
```

```
df.dtypes
```

Out[50]:

```
encounter_id                  int64
patient_nbr                   int64
race                         object
gender                       object
age                          object
admission_type_id            object
discharge_disposition_id     object
admission_source_id          object
time_in_hospital              int64
num_lab_procedures            int64
num_procedures                int64
num_medications               int64
diag_1                       object
diag_2                       object
diag_3                       object
number_diagnoses              int64
max_glu_serum                object
A1Cresult                     int64
metformin                     int64
repaglinide                   int64
nateglinide                   int64
chlorpropamide                int64
glimepiride                   int64
acetohexamide                 int64
glipizide                     int64
glyburide                     int64
tolbutamide                   int64
pioglitazone                  int64
rosiglitazone                 int64
acarbose                      int64
miglitol                      int64
troglitazone                  int64
tolazamide                    int64
insulin                       int64
glyburide-metformin           int64
glipizide-metformin           int64
glimepiride-pioglitazone      int64
metformin-rosiglitazone       int64
metformin-pioglitazone        int64
change                        int64
diabetesMed                   int64
readmitted                    int64
numchange                     int64
level1_diag1                 object
level2_diag1                 object
level1_diag2                 object
level2_diag2                 object
level1_diag3                 object
level2_diag3                 object
nummed                       object
service_utilization_log1p   float64
number_outpatient_log1p     float64
number_emergency_log1p      float64
number_inpatient_log1p      float64
dtype: object
```

In [51]:

```
dfcopy = df.copy(deep = True)
```

In [52]:

```
df['readmitted'] = df['readmitted'].apply(lambda x: 0 if x == 2 else x)
```

In [53]:

```
# drop individual diagnosis columns that have too granular disease information
# also drop level 2 categorization (which was not comparable with any reference)
# also drop level 1 secondary and tertiary diagnoses
df.drop(['diag_1', 'diag_2', 'diag_3', 'level2_diag1', 'level1_diag2', 'level2_diag2', 'level1_diag3',
         'level2_diag3'], axis=1, inplace=True)
```

In [54]:

```
interactionterms = [('num_medications','time_in_hospital'),
('num_medications','num_procedures'),
('time_in_hospital','num_lab_procedures'),
('num_medications','num_lab_procedures'),
('num_medications','number_diagnoses'),
('age','number_diagnoses'),
('change','num_medications'),
('number_diagnoses','time_in_hospital'),
('num_medications','numchange')]
```

In [55]:

```
for inter in interactionterms:
    name = inter[0] + '|' + inter[1]
    df[name] = df[inter[0]] * df[inter[1]]
```

In [56]:

```
df[['num_medications','time_in_hospital', 'num_medications|time_in_hospital']].head()
```

Out[56]:

	num_medications	time_in_hospital	num_medications|time_in_hospital
1	18	3	54
2	13	2	26
3	16	2	32
4	8	1	8
5	16	3	48

In [57]:

```
# Feature Scaling
datf = pd.DataFrame()
datf['features'] = numerics
datf['std_dev'] = datf['features'].apply(lambda x: df[x].std())
```

```
datf['mean'] = datf['features'].apply(lambda x: df[x].mean())
```

In [58]:

```
# dropping multiple encounters while keeping either first or last encounter of thes
e patients

df2 = df.drop_duplicates(subset= ['patient_nbr'], keep = 'first')

df2.shape
```

Out[58]:

```
(67580, 55)
```

In [59]:

```
# standardize function

def standardize(raw_data):

    return ((raw_data - np.mean(raw_data, axis = 0)) / np.std(raw_data, axis = 0))
```

In [60]:

```
df2[numerics] = standardize(df2[numerics])

import scipy as sp

df2 = df2[(np.abs(sp.stats.zscore(df2[numerics])) < 3).all(axis=1)]
```

```
/opt/conda/lib/python3.6/site-packages/pandas/core/frame.py:3140: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/i
ndexing.html#indexing-view-versus-copy
  self[k1] = value[k2]
```

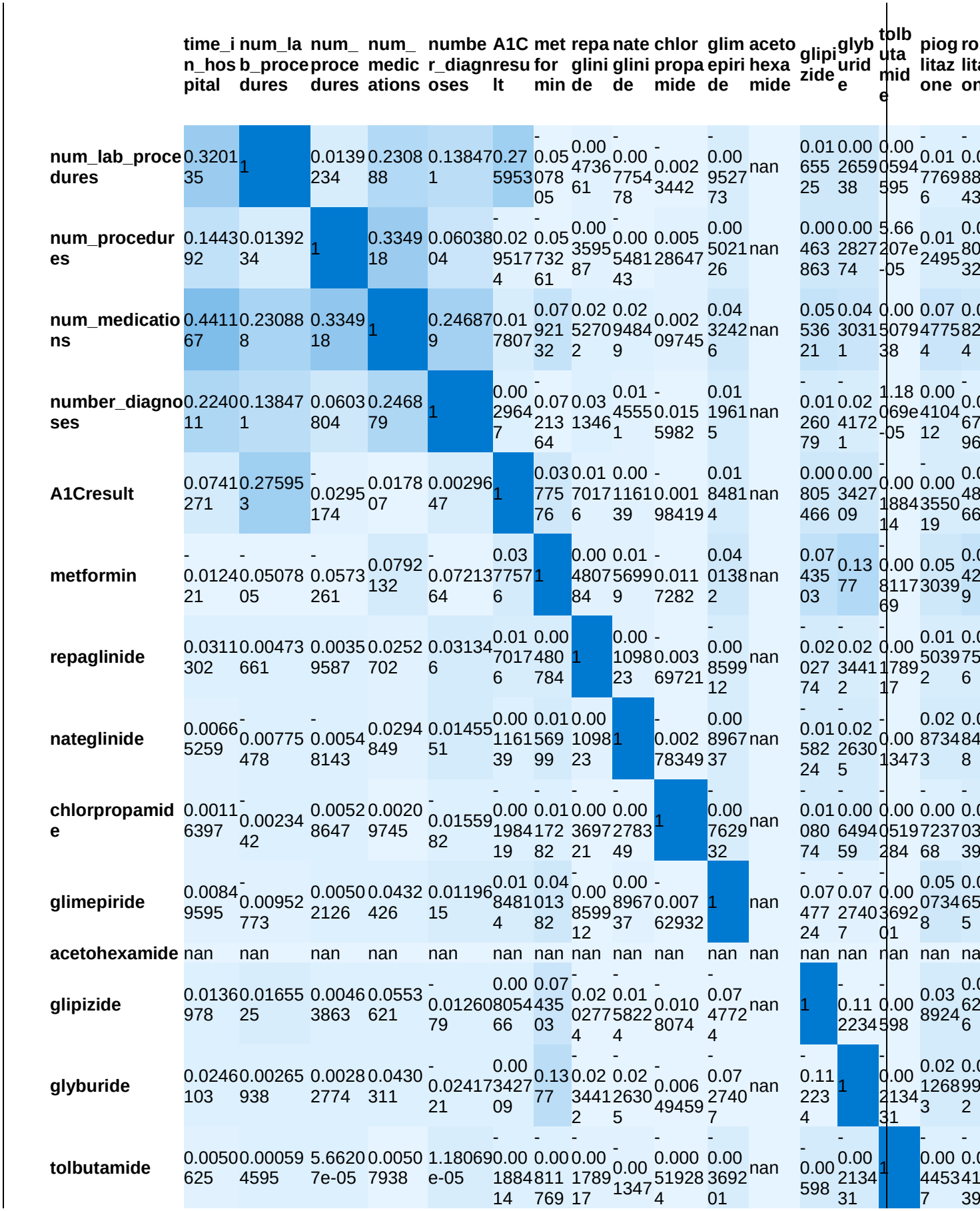
In [61]:

```
from matplotlib.colors import ListedColormap

my_cmap = ListedColormap(sns.light_palette((250, 100, 50), input="husl", n_colors=5
0).as_hex())

table = df2.drop(['patient_nbr', 'encounter_id'], axis=1).corr(method='pearson')

table.style.background_gradient(cmap=my_cmap, axis = 0)
```

```
/opt/conda/lib/python3.6/site-packages/matplotlib/colors.py:512: RuntimeWarning: in
valid value encountered in less
  xa[xa < 0] = -1
```

Out[61]:

	time_in_hospital	num_lab_procedures	num_procedures	num_medications	number_diagnoses	A1Cresult	metformin	repaglinide	nateglinide	chlorpropamide	glimepiride	acetohexamide	glipizide	glyburide	tolbutamide	pioglitazone	ro lit on
time_in_hospital	1	0.320135	0.144392	0.441167	0.224011	0.074127	-0.012421	0.031132	0.006659	0.00116397	0.008495	nan	0.013697	0.024613	0.005062	-0.004371	-0.0 0



	time_in_hospital	num_lab_procedures	num_procedures	num_medications	number_diagnoses	A1Cresult	metformin	repaglinide	nateglinide	chlorpropamide	glimepiride	acetohexamide	glipizide	glyburide	tolbutamide	pioglitazone	ro…
pioglitazone	-0.00437611	-0.0177696	0.012495	0.0747754	0.004100412	-0.0035519	0.05303	0.01503982	0.02873423	-0.00723768	0.050734	nan	0.0389224	0.021268	-0.0044537	1	-0.0…
rosiglitazone	-0.00081331	-0.0088943	0.00801632	0.0582544	-0.00677482566	0.00422	0.09239	0.017575	0.018498	-0.00038039	0.036555	nan	0.0362716	0.029915	0.00419239	-0.06779	1
acarbose	0.000816241	-0.00192971	-0.00183317	0.0139269	0.00682689	0.00082	0.00179	0.02423	0.00362	-0.00164947	0.015442	nan	0.0244253	0.01780	-0.0007	0.01	0.0…
miglitol	-0.00220634	-0.0036	0.0010088	0.0030	-0.00120699	-0.00	0.01	0.02	0.01	-0.00	0.01	nan	2.09	-0.00	0.00	-0.00	0.0
troglitazone	0.00569745	0.00519974	-0.00573509	0.0042	0.005220184	-0.00	0.00	0.00	0.00	-0.00	0.00	nan	-0.00	-0.00	-0.00	-0.00	0.0
tolazamide	-0.00665848	-0.00100684	-0.00028	-0.0015	-0.0104194	-0.00	0.00	-0.00	-0.00	-0.00	-0.00	nan	-0.00	-0.00	-0.00	-0.00	0.0
insulin	0.09730714	0.100137	0.0024	0.194	0.090	0.09	-0.02	0.00	0.00	-0.018	0.00	nan	-0.02	-0.07	-0.00	0.01	0.0
glyburide-metformin	-0.00030	-0.01424	-0.0091	0.0087	-0.00243533	-0.00	0.02	0.00	-0.00	-0.00	-0.00	nan	-0.02	-0.01	-0.00	0.03	0.0
glipizide-metformin	0.00233415	-0.00379084	-0.0033	0.0042	0.00174275	0.00	0.00	0.00	-0.00	-0.00	-0.00	nan	-0.00	0.00	-0.00	0.00	-0.0
glimepiride-pioglitazone	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	na
metformin-rosiglitazone	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	na
metformin-pioglitazone	0.00279377	-0.00398099	-0.0009	0.0028	-0.00702432	-0.00	0.00	-0.00	-0.00	-0.00	-0.00	nan	-0.00	-0.00	-6.48	0.01	-0.0
change	0.09440059	0.0645598	-0.0161973	0.233256	0.0423337	0.09	0.35	0.07	0.05	-0.00587703	0.147047	nan	0.20	0.19	-0.00	0.21	0.2
diabetesMed	0.05980106	0.0374758	-0.0173448	0.190055	0.0225025	0.07	0.29	0.06	0.04	0.0190128	0.135177	nan	0.21	0.20	0.00	0.16	0.1
readmitted	0.05510478	0.0256438	0.00208551	0.0387684	0.03963-02	-0.00	-0.01	0.01	-0.00	-0.00447192	-0.00	nan	0.00	0.00	-0.00	-0.00	-0.0

	time_in_hospital	num_lab_procedures	num_procedures	num_medications	number_diagnoses	A1Cresult	metformin	repaglinide	nateglinide	chlorpropamide	glimepiride	acetohexamide	glipizide	glyburide	tolbutamide	pioglitazone	ro lita on
						525765	01449		309871		405575				103866	391913	0179
numchange	0.139607	0.111876	-0.010996	0.193801	0.065820	0.111597	0.0196183	0.0316958	0.0043866	-0.00913918	0.0209341	nan	0.0032080	-0.0001529	-0.0006041	0.0237585	0.0676
service_utilization_log1p	0.00686184	-0.00178267	-0.0305084	0.0450339	0.086091	-0.050729	0.0083782	0.0096222	0.0106214	-0.00329107	0.0046563	nan	0.0043218	0.0028434	-0.0033433	0.0183025	0.0203
number_outpatient_log1p	-0.0276142	-0.0412072	-0.0172952	0.0315759	0.0579024	-0.0326161	0.0263852	0.0044300	0.0124786	-0.00466847	0.0003389	nan	0.0051196	0.0127531	-0.0020737	0.0139669	0.0178
number_emergency_log1p	-0.00182889	0.0067835	-0.026767	0.0133994	0.050317	-0.011517	-0.0020265	0.0090656	0.0066456	-0.00772182	0.0048499	nan	0.0021471	-0.0137833	-0.0037378	0.0172995	0.0325
number_inpatient_log1p	0.0514963	0.0491532	-0.0126986	0.0348021	0.0515976	-0.043999	-0.0187815	0.0107275	-0.0010749	0.00465557	0.0046016	nan	0.0010038	-0.0017052	-0.0010575	-0.0029046	-0.0191
num_medications|time_in_hospital	0.859364	0.328828	0.2637	0.752726	0.244135	0.062719	0.0180484	0.0314325	0.0150395	0.00275431	0.0203083	nan	0.0275866	0.0325263	0.0006270	0.0248944	0.054
num_medications|num_procedures	0.270542	0.114754	0.877783	0.599321	0.111045	0.0016956	-0.0122985	0.0086090	0.0012881	0.00956255	0.0132681	nan	0.0176849	0.0178769	0.0040163	0.0276996	0.026
time_in_hospital|num_lab_procedures	0.853015	0.674595	0.138598	0.439898	0.219112	0.179762	-0.0356974	0.0250283	-0.0011295	-0.00288621	0.0017135	nan	0.0168332	0.0161035	0.0046982	-0.0124606	-0.056
num_medications|num_lab_procedures	0.502133	0.732844	0.24444	0.772237	0.25072	0.178755	0.0080366	0.0215169	0.0126763	-0.00277266	0.0214087	nan	0.0414828	0.0235942	0.0042641	0.0312011	0.0646
num_medications|number_diagnoses	0.456259	0.258944	0.292403	0.898603	0.599241	0.0282212	0.0294402	0.0351256	0.0293022	-0.00714166	0.0387338	nan	0.0365288	0.0217994	0.0051049	0.0518631	0.0343
change|num_medications	0.231169	0.135242	0.109633	0.530894	0.123807	0.0782336	0.3006672	0.0693623	0.0526539	-0.00511084	0.130815	nan	0.1817777	0.165521	0.0007650	0.1976476	0.181
number_diagnoses|time_in_hospital	0.938944	0.322335	0.153126	0.459725	0.484368	0.061568	-0.0341624	0.0397097	0.0112965	-0.0045613	0.0099439	nan	0.0052989	0.0130817	0.0038351	-0.0029231	-0.0178
num_medications|numchange	0.229845	0.155882	0.0750897	0.392592	0.120178	0.0995713	0.0261156	0.0318125	0.0078441	-0.00804966	0.0260481	nan	0.0105357	0.0017084	-0.0048672	0.0362543	0.046

In [62]:

```
df2['level1_diag1'] = df2['level1_diag1'].astype('object')

df_pd = pd.get_dummies(df2, columns=['gender', 'admission_type_id', 'discharge_disp
osition_id',

                                     'admission_source_id', 'max_glu_serum', 'A1Cr
esult', 'level1_diag1'], drop_first = True)

just_dummies = pd.get_dummies(df_pd['race'])

df_pd = pd.concat([df_pd, just_dummies], axis=1)

df_pd.drop(['race'], inplace=True, axis=1)
```

In [63]:

```
non_num_cols = ['race', 'gender', 'admission_type_id', 'discharge_disposition_id',
'admission_source_id',

                'max_glu_serum', 'A1Cresult', 'level1_diag1' ]
```

In [64]:

```
num_cols = list(set(list(df._get_numeric_data().columns))- {'readmitted', 'change'}
)

num_cols
```

Out[64]:

```
['num_medications|time_in_hospital',
 'num_medications|num_procedures',
 'time_in_hospital',
 'num_medications|numchange',
 'metformin',
 'encounter_id',
 'insulin',
 'service_utilization_log1p',
 'repaglinide',
 'acetohexamide',
 'pioglitazone',
 'time_in_hospital|num_lab_procedures',
 'numchange',
 'nateglinide',
 'glipizide-metformin',
 'num_procedures',
 'num_medications',
 'A1Cresult',
 'number_inpatient_log1p',
 'num_lab_procedures',
 'number_diagnoses',
 'glyburide-metformin',
 'rosiglitazone',
 'number_diagnoses|time_in_hospital',
 'glimepiride-pioglitazone',
 'num_medications|number_diagnoses',
 'num_medications|num_lab_procedures',
 'number_emergency_log1p',
 'patient_nbr',
 'acarbose',
 'miglitol',
 'metformin-pioglitazone',
 'tolbutamide',
 'glipizide',
 'glyburide',
```

```
 'number_outpatient_log1p',
 'glimepiride',
 'metformin-rosiglitazone',
 'chlorpropamide',
 'troglitazone',
 'diabetesMed',
 'tolazamide',
 'change|num_medications']
```

In [65]:

```
new_non_num_cols = []
for i in non_num_cols:
    for j in df_pd.columns:
        if i in j:
            new_non_num_cols.append(j)
```

In [66]:

```
new_non_num_cols
```

Out[66]:

```
['gender_1',
 'admission_type_id_3',
 'admission_type_id_4',
 'admission_type_id_5',
 'discharge_disposition_id_2',
 'discharge_disposition_id_7',
 'discharge_disposition_id_10',
 'discharge_disposition_id_18',
 'discharge_disposition_id_19',
 'discharge_disposition_id_20',
 'discharge_disposition_id_27',
 'discharge_disposition_id_28',
 'admission_source_id_4',
 'admission_source_id_7',
 'admission_source_id_8',
 'admission_source_id_9',
 'admission_source_id_11',
 'max_glu_serum_0',
 'max_glu_serum_1',
 'A1Cresult_0',
 'A1Cresult_1',
 'level1_diag1_1.0',
 'level1_diag1_2.0',
 'level1_diag1_3.0',
 'level1_diag1_4.0',
 'level1_diag1_5.0',
 'level1_diag1_6.0',
 'level1_diag1_7.0',
 'level1_diag1_8.0']
```

In [67]:

```
l = []
for feature in list(df_pd.columns):
    if '|' in feature:
        l.append(feature)
l
```

Out[67]:

```
['num_medications|time_in_hospital',
 'num_medications|num_procedures',
 'time_in_hospital|num_lab_procedures',
 'num_medications|num_lab_procedures',
 'num_medications|number_diagnoses',
 'age|number_diagnoses',
 'change|num_medications',
 'number_diagnoses|time_in_hospital',
 'num_medications|numchange']
```

In [68]:

```
df_pd.head().T
```

Out[68]:

	1	2	3	4	5
encounter_id	149190	64410	500364	16680	35754
patient_nbr	55629189	86047875	82442376	42519267	82637451
age	15	25	35	45	55
time_in_hospital	-0.444872	-0.784109	-0.784109	-1.12335	-0.444872
num_lab_procedures	0.803605	-1.60206	0.0518349	0.402661	-0.5997
num_procedures	-0.816784	2.02679	-0.24807	-0.816784	2.5955
num_medications	0.270081	-0.335198	0.0279693	-0.940477	0.0279693
number_diagnoses	0.881754	-0.704844	-0.175978	-1.23371	0.881754
metformin	0	0	0	0	0
repaglinide	0	0	0	0	0
nateglinide	0	0	0	0	0
chlorpropamide	0	0	0	0	0
glimepiride	0	0	0	0	0
acetohexamide	0	0	0	0	0
glipizide	0	1	0	1	0
glyburide	0	0	0	0	0
tolbutamide	0	0	0	0	0
pioglitazone	0	0	0	0	0
rosiglitazone	0	0	0	0	0
acarbose	0	0	0	0	0
miglitol	0	0	0	0	0
troglitazone	0	0	0	0	0
tolazamide	0	0	0	0	0
insulin	1	0	1	1	1
glyburide-metformin	0	0	0	0	0
glipizide-metformin	0	0	0	0	0
glimepiride-pioglitazone	0	0	0	0	0
metformin-rosiglitazone	0	0	0	0	0
metformin-pioglitazone	0	0	0	0	0
change	1	0	1	1	0
...	...	...	...	...	...
discharge_disposition_id_2	0	0	0	0	0
discharge_disposition_id_7	0	0	0	0	0
discharge_disposition_id_10	0	0	0	0	0
discharge_disposition_id_18	0	0	0	0	0
discharge_disposition_id_19	0	0	0	0	0
discharge_disposition_id_20	0	0	0	0	0
discharge_disposition_id_27	0	0	0	0	0
discharge_disposition_id_28	0	0	0	0	0
admission_source_id_4	0	0	0	0	0
admission_source_id_7	1	1	1	1	0

	1	2	3	4	5
admission_source_id_8	0	0	0	0	0
admission_source_id_9	0	0	0	0	0
admission_source_id_11	0	0	0	0	0
max_glu_serum_0	0	0	0	0	0
max_glu_serum_1	0	0	0	0	0
A1Cresult_0	0	0	0	0	0
A1Cresult_1	0	0	0	0	0
level1_diag1_1.0	0	0	0	0	1
level1_diag1_2.0	0	0	0	0	0
level1_diag1_3.0	0	0	0	0	0
level1_diag1_4.0	0	0	0	0	0
level1_diag1_5.0	0	0	0	0	0
level1_diag1_6.0	0	0	0	0	0
level1_diag1_7.0	0	0	0	0	0
level1_diag1_8.0	0	0	0	1	0
AfricanAmerican	0	1	0	0	0
Asian	0	0	0	0	0
Caucasian	1	0	1	1	1
Hispanic	0	0	0	0	0
Other	0	0	0	0	0

81 rows × 5 columns

Modeling¶

In [69]:

```
feature_set = ['age', 'time_in_hospital', 'num_procedures', 'num_medications', 'number_outpatient_log1p',
                'number_emergency_log1p', 'number_inpatient_log1p', 'number_diagnoses', 'metformin',
                'repaglinide', 'nateglinide', 'chlorpropamide', 'glimepiride', 'glipizide', 'glyburide',
                'pioglitazone', 'rosiglitazone', 'acarbose', 'tolazamide', 'insulin', 'glyburide-metformin',
                'AfricanAmerican', 'Asian', 'Caucasian', 'Hispanic', 'Other', 'gender_1',
                'admission_type_id_3', 'admission_type_id_5', 'discharge_disposition_id_2', 'discharge_disposition_id_7',
                'discharge_disposition_id_10', 'discharge_disposition_id_18', 'admission_source_id_4',
                'admission_source_id_7', 'admission_source_id_9', 'max_glu_serum_0', 'max_glu_serum_1', 'A1Cresult_0',
                'A1Cresult_1', 'num_medications|time_in_hospital', 'num_medications|num_procedures',
                'time_in_hospital|num_lab_procedures', 'num_medications|num_lab_procedures', 'num_medications|number_diagnoses',
                'age|number_diagnoses', 'change|num_medications', 'number_diagnoses|time_in_hospital',
                'num_medications|numchange', 'level1_diag1_1.0', 'level1_diag1_2.0', 'level1_diag1_3.0', 'level1_diag1_4.0',
```

```
                'level1_diag1_5.0','level1_diag1_6.0', 'level1_diag1_7.0', 'level1
_diag1_8.0']
```

In [70]:

```
X = df_pd[feature_set]

y = df_pd['readmitted']
```

Logistic Regression¶

In [71]:

```
df_pd['readmitted'].value_counts()
```

Out[71]:

```
0    54635
1     5071
Name: readmitted, dtype: int64
```

In [72]:

```
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import cross_val_score

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_st
ate=0)

logit = LogisticRegression(fit_intercept=True, penalty='l1')

logit.fit(X_train, y_train)
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:433: Future
Warning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to sil
ence this warning.
  FutureWarning)
```

Out[72]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [73]:

```
logit_pred = logit.predict(X_test)

pd.crosstab(pd.Series(y_test, name = 'Actual'), pd.Series(logit_pred, name = 'Predi
ct'), margins = True)
```

Out[73]:

Predict	0	All
Actual		
0	1466	1466
1	167	167
All	1633	1633

In [74]:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score
print("Accuracy is {0:.2f}".format(accuracy_score(y_test, logit_pred)))
print("Precision is {0:.2f}".format(precision_score(y_test, logit_pred)))
print("Recall is {0:.2f}".format(recall_score(y_test, logit_pred)))
```

```
Accuracy is 0.91
Precision is 0.00
Recall is 0.00
/opt/conda/lib/python3.6/site-packages/sklearn/metrics/classification.py:1143: Unde
finedMetricWarning: Precision is ill-defined and being set to 0.0 due to no predict
ed samples.
  'precision', 'predicted', average, warn_for)
```

Since our target variable is having class imbalance problem, So will use SMOTE technique to resolve it

In [75]:

```
from imblearn.over_sampling import SMOTE
from collections import Counter
print('Original dataset shape {}'.format(Counter(y_train)))
sm = SMOTE(random_state=20)
train_input_new, train_output_new = sm.fit_sample(X_train, y_train)
print('New dataset shape {}'.format(Counter(train_output_new)))
```

```
Using TensorFlow backend.
Original dataset shape Counter({0: 43711, 1: 4053})
New dataset shape Counter({0: 43711, 1: 43711})
```

In [76]:

```
train_input_new = pd.DataFrame(train_input_new, columns = list(X.columns))
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
X_train, X_test, y_train, y_test = train_test_split(train_input_new, train_output_n
ew, test_size=0.20, random_state=0)
logit = LogisticRegression(fit_intercept=True, penalty='l1')
logit.fit(X_train, y_train)
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:433: Future
Warning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to sil
ence this warning.
  FutureWarning)
```

Out[76]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [77]:

```
logit_pred = logit.predict(X_test)
```

```
pd.crosstab(pd.Series(y_test, name = 'Actual'), pd.Series(logit_pred, name = 'Predi
ct'), margins = True)
```

Out[77]:

Predict	0	1	All
Actual			
0	5805	2901	8706
1	3895	4884	8779
All	9700	7785	17485

In [78]:

```
print("Accuracy is {0:.2f}".format(accuracy_score(y_test, logit_pred)))
print("Precision is {0:.2f}".format(precision_score(y_test, logit_pred)))
print("Recall is {0:.2f}".format(recall_score(y_test, logit_pred)))


accuracy_logit = accuracy_score(y_test, logit_pred)
precision_logit = precision_score(y_test, logit_pred)
recall_logit = recall_score(y_test, logit_pred)
```

```
Accuracy is 0.61
Precision is 0.63
Recall is 0.56
```

Decision Tree¶

In [79]:

```
feature_set_no_int = ['age', 'time_in_hospital', 'num_procedures', 'num_medications
', 'number_outpatient_log1p',
                  'number_emergency_log1p', 'number_inpatient_log1p', 'number_diagno
ses', 'metformin',
                  'repaglinide', 'nateglinide', 'chlorpropamide', 'glimepiride', 'gl
ipizide',
                  'glyburide', 'pioglitazone', 'rosiglitazone', 'acarbose',
                  'tolazamide', 'insulin', 'glyburide-metformin',
                  'AfricanAmerican', 'Asian', 'Caucasian',
                  'Hispanic', 'Other', 'gender_1',
                  'admission_type_id_3', 'admission_type_id_5',
                  'discharge_disposition_id_2', 'discharge_disposition_id_7',
                  'discharge_disposition_id_10', 'discharge_disposition_id_18',
                  'admission_source_id_4', 'admission_source_id_7',
                  'admission_source_id_9', 'max_glu_serum_0',
                  'max_glu_serum_1', 'A1Cresult_0', 'A1Cresult_1',
                  'level1_diag1_1.0',
                  'level1_diag1_2.0',
                  'level1_diag1_3.0',
                  'level1_diag1_4.0',
```

```
                  'level1_diag1_5.0',
                  'level1_diag1_6.0',
                  'level1_diag1_7.0',
                  'level1_diag1_8.0']
```

In [80]:

```
X = df_pd[feature_set_no_int]
y = df_pd['readmitted']
df_pd['readmitted'].value_counts()
```

Out[80]:

```
0    54635
1     5071
Name: readmitted, dtype: int64
```

In [81]:

```
print('Original dataset shape {}'.format(Counter(y)))
smt = SMOTE(random_state=20)
train_input_new, train_output_new = smt.fit_sample(X, y)
print('New dataset shape {}'.format(Counter(train_output_new)))
train_input_new = pd.DataFrame(train_input_new, columns = list(X.columns))
X_train, X_test, y_train, y_test = train_test_split(train_input_new, train_output_new, test_size=0.20, random_state=0)
```

```
Original dataset shape Counter({0: 54635, 1: 5071})
New dataset shape Counter({0: 54635, 1: 54635})
```

In [82]:

```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(max_depth=28, criterion = "entropy", min_samples_split=10)
dtree.fit(X_train, y_train)
```

Out[82]:

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=28,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=10,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

In [83]:

```
dtree_pred = dtree.predict(X_test)
pd.crosstab(pd.Series(y_test, name = 'Actual'), pd.Series(dtree_pred, name = 'Predict'), margins = True)
```

Out[83]:

Predict	0	1	All
Actual			
0	10244	692	10936

Predict	0	1	All
Actual			
1	1130	9788	10918
All	11374	10480	21854

In [84]:

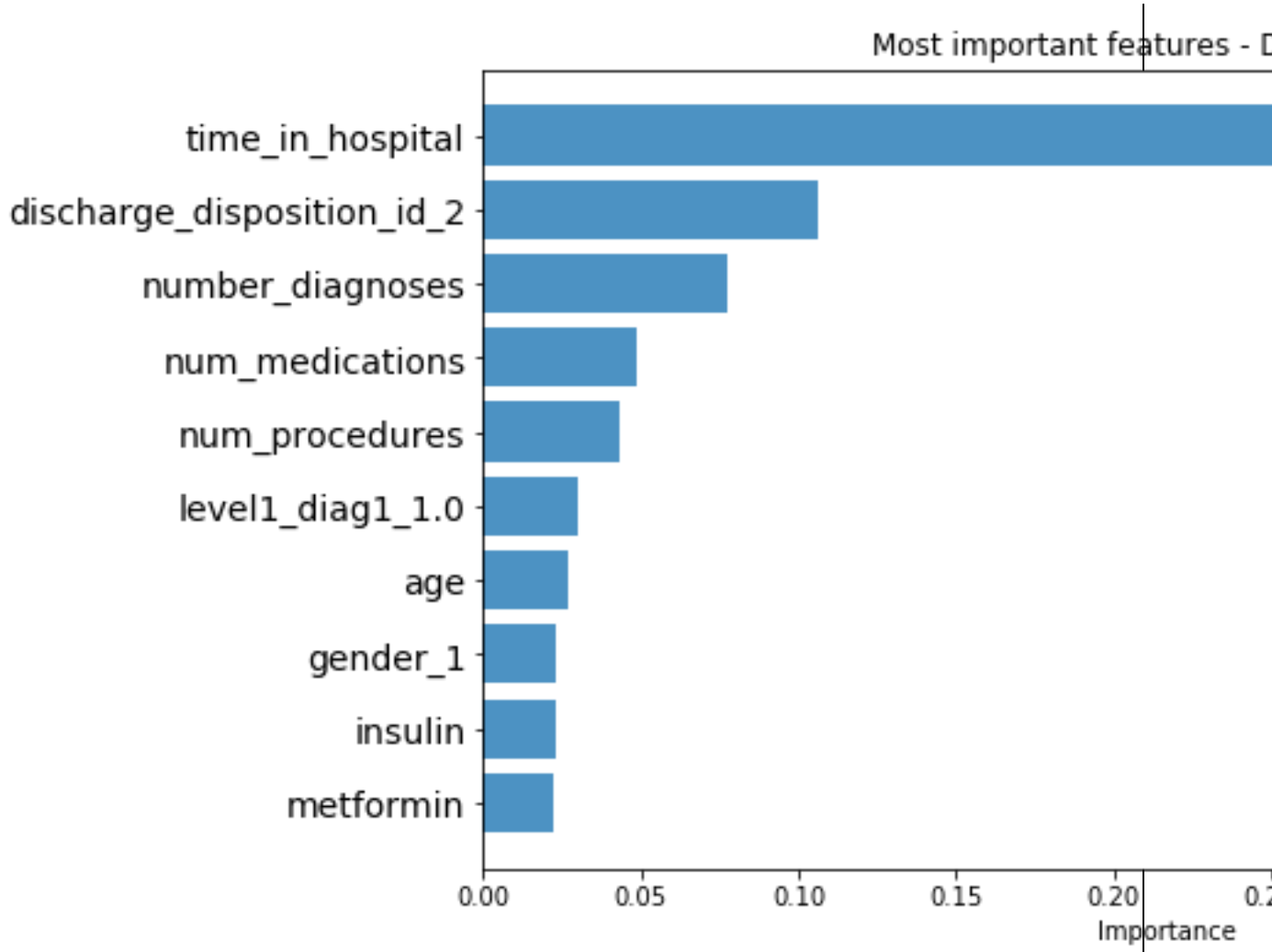
```
print("Accuracy is {0:.2f}".format(accuracy_score(y_test, dtree_pred)))

print("Precision is {0:.2f}".format(precision_score(y_test, dtree_pred)))

print("Recall is {0:.2f}".format(recall_score(y_test, dtree_pred)))


accuracy_dtree = accuracy_score(y_test, dtree_pred)

precision_dtree = precision_score(y_test, dtree_pred)

recall_dtree = recall_score(y_test, dtree_pred)
```

```
Accuracy is 0.92
Precision is 0.93
Recall is 0.90
```

In [85]:

```
# Create list of top most features based on importance

feature_names = X_train.columns

feature_imports = dtree.feature_importances_

most_imp_features = pd.DataFrame([f for f in zip(feature_names,feature_imports)], c
olumns=["Feature", "Importance"]).nlargest(10, "Importance")

most_imp_features.sort_values(by="Importance", inplace=True)

print(most_imp_features)

plt.figure(figsize=(10,6))

plt.barh(range(len(most_imp_features)), most_imp_features.Importance, align='center
', alpha=0.8)

plt.yticks(range(len(most_imp_features)), most_imp_features.Feature, fontsize=14)

plt.xlabel('Importance')

plt.title('Most important features - Decision Tree')

plt.show()
```

```
                        Feature  Importance
8                     metformin    0.022040
19                      insulin    0.022958
26                     gender_1    0.023195
0                           age    0.026882
40             level1_diag1_1.0    0.029614
2                num_procedures    0.043294
3               num_medications    0.048912
7              number_diagnoses    0.077183
29  discharge_disposition_id_2    0.106147
1              time_in_hospital    0.413369
```



Random Forest¶

In [86]:

```
X = df_pd[feature_set_no_int]
y = df_pd['readmitted']


print('Original dataset shape {}'.format(Counter(y)))
smt = SMOTE(random_state=20)
train_input_new, train_output_new = smt.fit_sample(X, y)
print('New dataset shape {}'.format(Counter(train_output_new)))
train_input_new = pd.DataFrame(train_input_new, columns = list(X.columns))
X_train, X_test, y_train, y_test = train_test_split(train_input_new, train_output_n
ew, test_size=0.20, random_state=0)
```

```
Original dataset shape Counter({0: 54635, 1: 5071})
New dataset shape Counter({0: 54635, 1: 54635})
```

In [87]:

```
from sklearn.ensemble import RandomForestClassifier

rm = RandomForestClassifier(n_estimators = 10, max_depth=25, criterion = "gini", mi
n_samples_split=10)

rm.fit(X_train, y_train)
```

Out[87]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=25, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=10,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [88]:

```
rm_prd = rm.predict(X_test)

pd.crosstab(pd.Series(y_test, name = 'Actual'), pd.Series(rm_prd, name = 'Predict')
, margins = True)
```

Out[88]:

Predict	0	1	All
Actual			
0	10795	141	10936
1	1141	9777	10918
All	11936	9918	21854

In [89]:

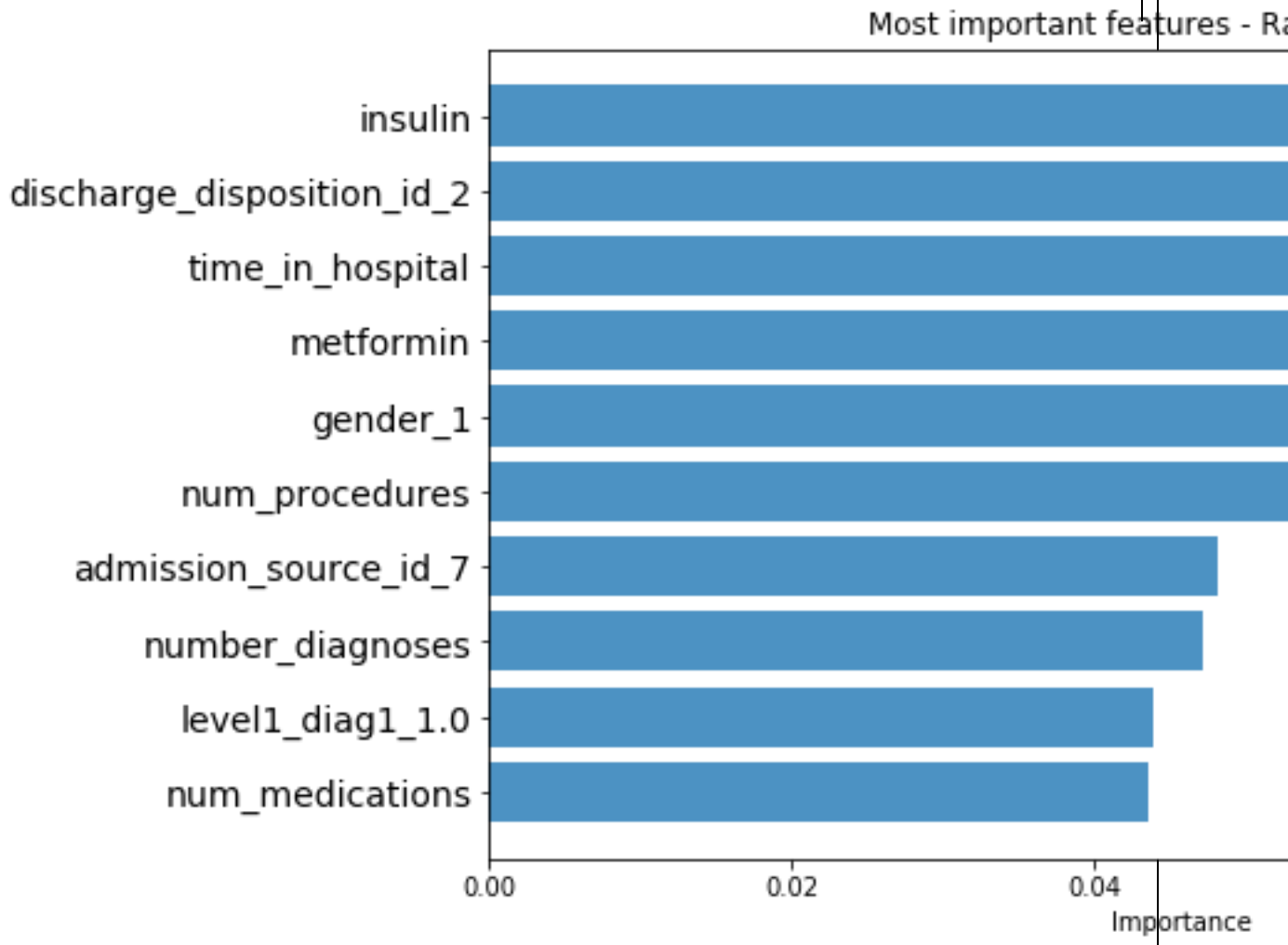
```
print("Accuracy is {0:.2f}".format(accuracy_score(y_test, rm_prd)))

print("Precision is {0:.2f}".format(precision_score(y_test, rm_prd)))

print("Recall is {0:.2f}".format(recall_score(y_test, rm_prd)))


accuracy_rm = accuracy_score(y_test, rm_prd)

precision_rm = precision_score(y_test, rm_prd)

recall_rm = recall_score(y_test, rm_prd)
```

```
Accuracy is 0.94
Precision is 0.99
Recall is 0.90
```

In [90]:

```
# Create list of top most features based on importance

feature_names = X_train.columns

feature_imports = rm.feature_importances_

most_imp_features = pd.DataFrame([f for f in zip(feature_names,feature_imports)], c
olumns=["Feature", "Importance"]).nlargest(10, "Importance")

most_imp_features.sort_values(by="Importance", inplace=True)

plt.figure(figsize=(10,6))

plt.barh(range(len(most_imp_features)), most_imp_features.Importance, align='center
', alpha=0.8)

plt.yticks(range(len(most_imp_features)), most_imp_features.Feature, fontsize=14)
```

```
plt.xlabel('Importance')
plt.title('Most important features - Random Forest ')
plt.show()
```



Model Comparision¶

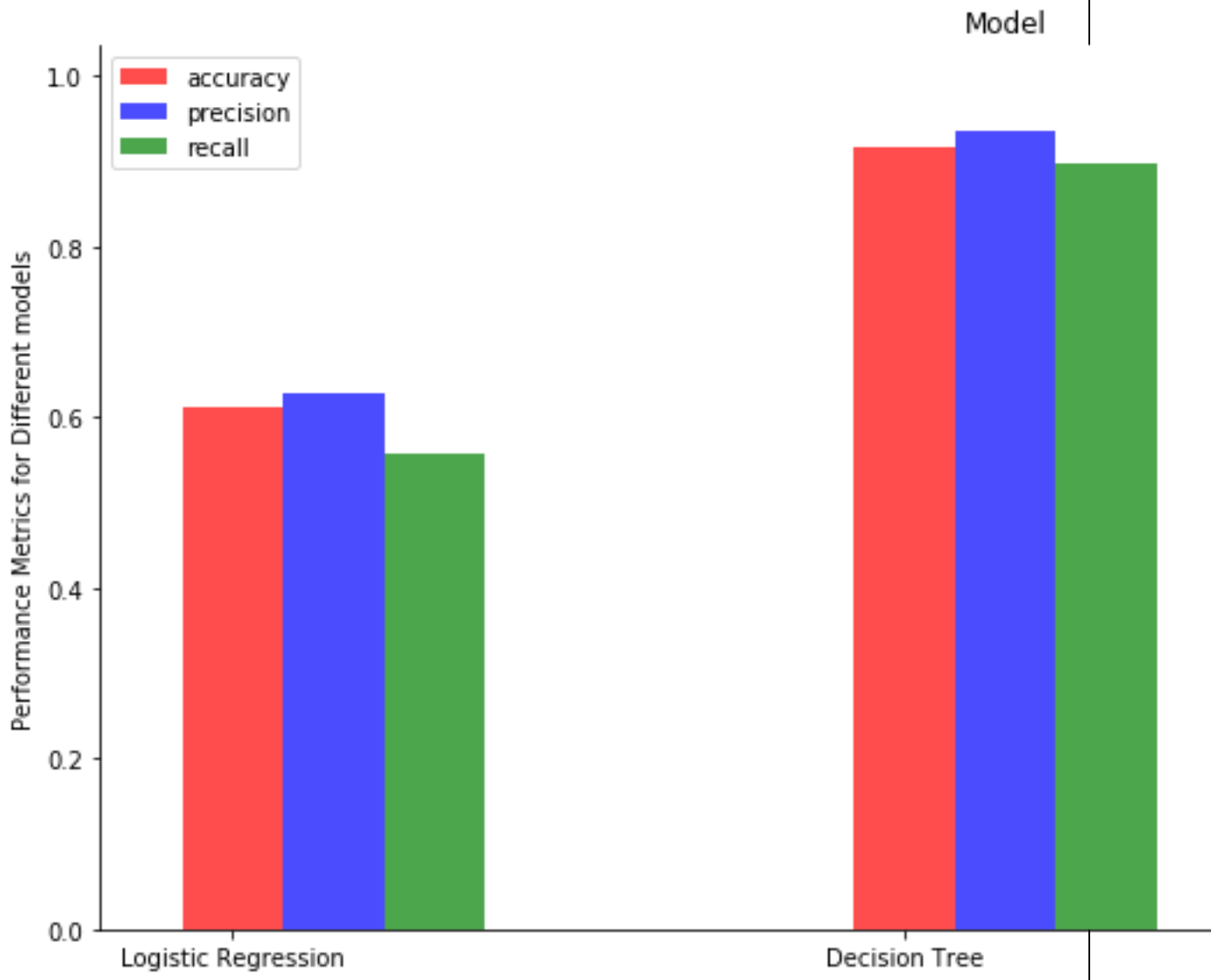
In [91]:

```
plt.figure(figsize=(14, 7))
ax = plt.subplot(111)

models = ['Logistic Regression', 'Decision Tree', 'Random Forests']
values = [accuracy_logit, accuracy_dtree, accuracy_rm]
model = np.arange(len(models))

plt.bar(model, values, align='center', width = 0.15, alpha=0.7, color = 'red', label= 'accuracy')
plt.xticks(model, models)
```

```
ax = plt.subplot(111)


models = ['Logistic Regression', 'Decision Tree', 'Random Forests']

values = [precision_logit, precision_dtree, precision_rm]

model = np.arange(len(models))


plt.bar(model+0.15, values, align='center', width = 0.15, alpha=0.7, color = 'blue'
, label = 'precision')

plt.xticks(model, models)




ax = plt.subplot(111)


models = ['Logistic Regression', 'Decision Tree', 'Random Forests' ]

values = [recall_logit, recall_dtree, recall_rm, ]

model = np.arange(len(models))


plt.bar(model+0.3, values, align='center', width = 0.15, alpha=0.7, color = 'green'
, label = 'recall')

plt.xticks(model, models)




plt.ylabel('Performance Metrics for Different models')

plt.title('Model')


# removing the axis on the top and right of the plot window

ax.spines['right'].set_visible(False)

ax.spines['top'].set_visible(False)

ax.legend()


plt.show()
```

```
/opt/conda/lib/python3.6/site-packages/matplotlib/figure.py:98: MatplotlibDeprecati
onWarning:
Adding an axes using the same arguments as a previous axes currently reuses the ear
lier instance.  In a future version, a new instance will always be created and retu
rned.  Meanwhile, this warning can be suppressed, and the future behavior ensured,
by passing a unique label to each axes instance.
  "Adding an axes using the same arguments as a previous axes "
/opt/conda/lib/python3.6/site-packages/matplotlib/figure.py:98: MatplotlibDeprecati
onWarning:
```

```
Adding an axes using the same arguments as a previous axes currently reuses the ear
lier instance.  In a future version, a new instance will always be created and retu
rned.  Meanwhile, this warning can be suppressed, and the future behavior ensured,
by passing a unique label to each axes instance.
  "Adding an axes using the same arguments as a previous axes "
```



In [92]: