In [238]:

*# Import libraries needed,*

**import** IPython

**import** pandas **as** pd

**import** numpy **as** np

**from** sklearn.preprocessing **import** MinMaxScaler

**import** matplotlib.pyplot **as** plt

**from** statistics **import** mode

In [239]:

*#Loading the data using pandas,*

dataoriginal **=** pd**.**read\_csv("C://Users//AKANKSHA//Desktop//ISB CBA//Practicum-2//dataset\_diabetes//dataset\_diabetes//diabetic\_data.csv")

print(dataoriginal**.**shape)

*#Exploring the data further,*

print(dataoriginal**.**info())

print(dataoriginal**.**describe())

(101766, 50)

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

RangeIndex: 101766 entries, 0 to 101765

Data columns (total 50 columns):

encounter\_id 101766 non-null int64

patient\_nbr 101766 non-null int64

race 101766 non-null object

gender 101766 non-null object

age 101766 non-null object

weight 101766 non-null object

admission\_type\_id 101766 non-null int64

discharge\_disposition\_id 101766 non-null int64

admission\_source\_id 101766 non-null int64

time\_in\_hospital 101766 non-null int64

payer\_code 101766 non-null object

medical\_specialty 101766 non-null object

num\_lab\_procedures 101766 non-null int64

num\_procedures 101766 non-null int64

num\_medications 101766 non-null int64

number\_outpatient 101766 non-null int64

number\_emergency 101766 non-null int64

number\_inpatient 101766 non-null int64

diag\_1 101766 non-null object

diag\_2 101766 non-null object

diag\_3 101766 non-null object

number\_diagnoses 101766 non-null int64

max\_glu\_serum 101766 non-null object

A1Cresult 101766 non-null object

metformin 101766 non-null object

repaglinide 101766 non-null object

nateglinide 101766 non-null object

chlorpropamide 101766 non-null object

glimepiride 101766 non-null object

acetohexamide 101766 non-null object

glipizide 101766 non-null object

glyburide 101766 non-null object

tolbutamide 101766 non-null object

pioglitazone 101766 non-null object

rosiglitazone 101766 non-null object

acarbose 101766 non-null object

miglitol 101766 non-null object

troglitazone 101766 non-null object

tolazamide 101766 non-null object

examide 101766 non-null object

citoglipton 101766 non-null object

insulin 101766 non-null object

glyburide-metformin 101766 non-null object

glipizide-metformin 101766 non-null object

glimepiride-pioglitazone 101766 non-null object

metformin-rosiglitazone 101766 non-null object

metformin-pioglitazone 101766 non-null object

change 101766 non-null object

diabetesMed 101766 non-null object

readmitted 101766 non-null object

dtypes: int64(13), object(37)

memory usage: 38.8+ MB

None

encounter\_id patient\_nbr admission\_type\_id \

count 1.017660e+05 1.017660e+05 101766.000000

mean 1.652016e+08 5.433040e+07 2.024006

std 1.026403e+08 3.869636e+07 1.445403

min 1.252200e+04 1.350000e+02 1.000000

25% 8.496119e+07 2.341322e+07 1.000000

50% 1.523890e+08 4.550514e+07 1.000000

75% 2.302709e+08 8.754595e+07 3.000000

max 4.438672e+08 1.895026e+08 8.000000

discharge\_disposition\_id admission\_source\_id time\_in\_hospital \

count 101766.000000 101766.000000 101766.000000

mean 3.715642 5.754437 4.395987

std 5.280166 4.064081 2.985108

min 1.000000 1.000000 1.000000

25% 1.000000 1.000000 2.000000

50% 1.000000 7.000000 4.000000

75% 4.000000 7.000000 6.000000

max 28.000000 25.000000 14.000000

num\_lab\_procedures num\_procedures num\_medications number\_outpatient \

count 101766.000000 101766.000000 101766.000000 101766.000000

mean 43.095641 1.339730 16.021844 0.369357

std 19.674362 1.705807 8.127566 1.267265

min 1.000000 0.000000 1.000000 0.000000

25% 31.000000 0.000000 10.000000 0.000000

50% 44.000000 1.000000 15.000000 0.000000

75% 57.000000 2.000000 20.000000 0.000000

max 132.000000 6.000000 81.000000 42.000000

number\_emergency number\_inpatient number\_diagnoses

count 101766.000000 101766.000000 101766.000000

mean 0.197836 0.635566 7.422607

std 0.930472 1.262863 1.933600

min 0.000000 0.000000 1.000000

25% 0.000000 0.000000 6.000000

50% 0.000000 0.000000 8.000000

75% 0.000000 1.000000 9.000000

max 76.000000 21.000000 16.000000

In [240]:

*# Let's see how the data looks like,*

dataoriginal**.**head()

Out[240]:

5 row

# Data Wrangling to make it fit for Analysis

In [241]:

*#Dealing with missing values,*

datacopy **=** dataoriginal**.**copy()

Rep **=** datacopy**.**replace('?', np**.**NaN)

nacheck **=** Rep**.**isnull()**.**sum()

nacheck

Out[241]:

encounter\_id 0

patient\_nbr 0

race 2273

gender 0

age 0

weight 98569

admission\_type\_id 0

discharge\_disposition\_id 0

admission\_source\_id 0

time\_in\_hospital 0

payer\_code 40256

medical\_specialty 49949

num\_lab\_procedures 0

num\_procedures 0

num\_medications 0

number\_outpatient 0

number\_emergency 0

number\_inpatient 0

diag\_1 21

diag\_2 358

diag\_3 1423

number\_diagnoses 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

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

dtype: int64

In [242]:

*#From the above we can see that "Weight,payer\_code and medical\_specialty" have a lot of missing values thus it is*

*#best to remove these values from the dataset.*

*#Dropping these columns,*

datacopy**=** datacopy**.**drop(['weight','payer\_code','medical\_specialty'],axis**=**1)

In [243]:

datacopy['readmitted']

Out[243]:

0 NO

1 >30

2 NO

3 NO

4 NO

5 >30

6 NO

7 >30

8 NO

9 NO

10 >30

11 <30

12 <30

13 NO

14 >30

15 NO

16 <30

17 NO

18 >30

19 NO

20 NO

21 NO

22 NO

23 >30

24 NO

25 NO

26 NO

27 >30

28 >30

29 >30

...

101736 >30

101737 NO

101738 NO

101739 NO

101740 NO

101741 NO

101742 NO

101743 NO

101744 NO

101745 NO

101746 <30

101747 >30

101748 >30

101749 NO

101750 <30

101751 NO

101752 NO

101753 NO

101754 >30

101755 >30

101756 >30

101757 NO

101758 NO

101759 NO

101760 >30

101761 >30

101762 NO

101763 NO

101764 NO

101765 NO

Name: readmitted, Length: 101766, dtype: object

In [244]:

*# We can see that there are three types of values:- 'NO','<30','>30'. So, to make it a categorical variable so that different*

*# ML techniques can be applies.*

datacopy['30readmit'] **=** np**.**where(datacopy['readmitted'] **==** 'NO', 0, 1)

In [245]:

datacopy**.**groupby('30readmit')**.**size()

Out[245]:

30readmit

0 54864

1 46902

dtype: int64

In [246]:

*# Removing the pateints from the dataset that are dead.*

datacopy **=** datacopy[((datacopy**.**discharge\_disposition\_id **!=** 11) **&**

(datacopy**.**discharge\_disposition\_id **!=** 13) **&**

(datacopy**.**discharge\_disposition\_id **!=** 14) **&**

(datacopy**.**discharge\_disposition\_id **!=** 19) **&**

(datacopy**.**discharge\_disposition\_id **!=** 20) **&**

(datacopy**.**discharge\_disposition\_id **!=** 21))]

In [247]:

datacopy**.**head()

Out[247]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2278392 | 8222157 | Caucasian | Female | [0-10) | 6 | 25 | 1 | 1 | 41 | ... | No | No | No | No | No | No | No | No | NO | 0 |
| **1** | 149190 | 55629189 | Caucasian | Female | [10-20) | 1 | 1 | 7 | 3 | 59 | ... | Up | No | No | No | No | No | Ch | Yes | >30 | 1 |
| **2** | 64410 | 86047875 | AfricanAmerican | Female | [20-30) |  | 1 |  | 2 | 11 | ... | No | No | No | No | No | No | No | Yes | NO | 0 |
| **3** | 500364 | 82442376 | Caucasian | Male | [30-40) | 1 | 1 | 7 | 2 | 44 | ... | Up | No | No | No | No | No | Ch | Yes | NO | 0 |
| **4** | 16680 | 42519267 | Caucasian | Male | [40-50) | 1 | 1 | 7 | 1 | 51 | ... | Steady | No | No | No | No | No | Ch | Yes | NO | 0 |

5 rows × 48 columns

# Performing Exploratory Data Analysis (EDA)

In [248]:

*# Let's see the relationship between different variables to understand the data and if there is a strong correlation between*

*#two variables then we can consider one of them.*

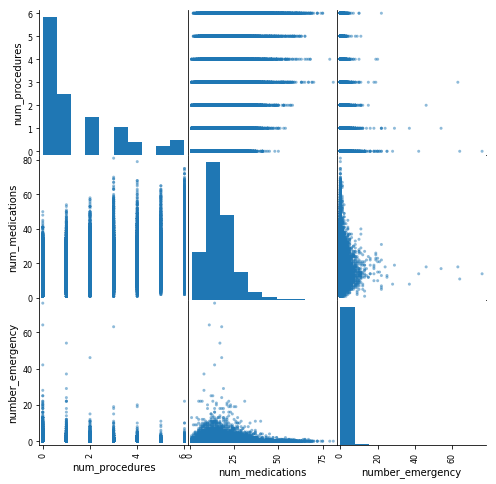
**import** seaborn **as** sns

sm **=** pd**.**scatter\_matrix(datacopy[['num\_procedures', 'num\_medications', 'number\_emergency']], figsize **=** (8, 8))

sns**.**despine()

C:\Users\AKANKSHA\Anaconda3\lib\site-packages\ipykernel\_launcher.py:4: FutureWarning: pandas.scatter\_matrix is deprecated. Use pandas.plotting.scatter\_matrix instead

after removing the cwd from sys.path.



From the above, we can see that there is no problem of multi-collinearity. We can also see that as the number\_emergency increases the num\_medication decreases.

In [249]:

*#Let's try to see how the age and number of medicines vary,*

sortage **=** datacopy**.**sort\_values(by **=** 'age')

x **=** sns**.**stripplot(x **=** "age", y **=** "num\_medications", data **=** sortage, color **=** 'red')

sns**.**despine() *#remove top and right axes*

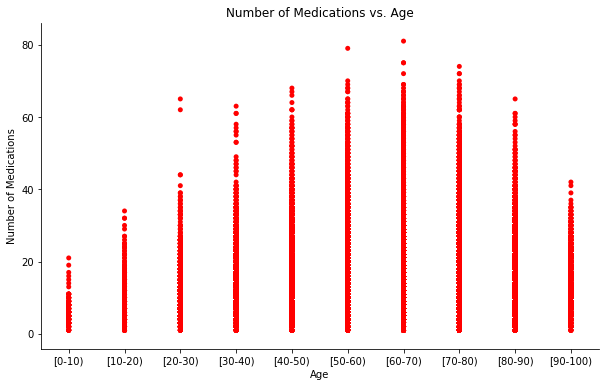
x**.**figure**.**set\_size\_inches(10, 6)

x**.**set\_xlabel('Age')

x**.**set\_ylabel('Number of Medications')

x**.**axes**.**set\_title('Number of Medications vs. Age')

plt**.**show()



In [250]:

*#Gender and Readmissions,*

plot1 **=** sns**.**countplot(x **=** 'gender', hue **=** '30readmit', data **=** datacopy)

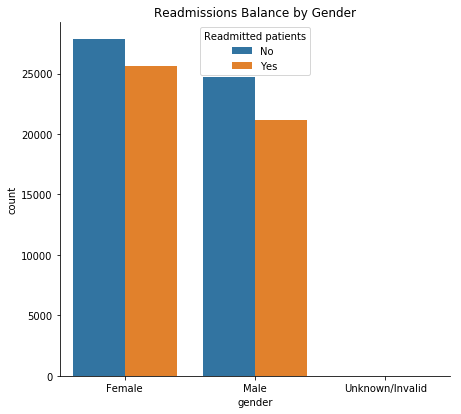
sns**.**despine()

plot1**.**figure**.**set\_size\_inches(7, 6.5)

plot1**.**legend(title **=** 'Readmitted patients', labels **=** ('No', 'Yes'))

plot1**.**axes**.**set\_title('Readmissions Balance by Gender')

plt**.**show()



In [251]:

*#Relation between age and readmission,*

b **=** datacopy**.**age**.**unique()

b**.**sort()

b\_sort **=** np**.**array(b)**.**tolist()

ageplt **=** sns**.**countplot(x **=** 'age', hue **=** '30readmit', data **=** datacopy, order **=** b\_sort)

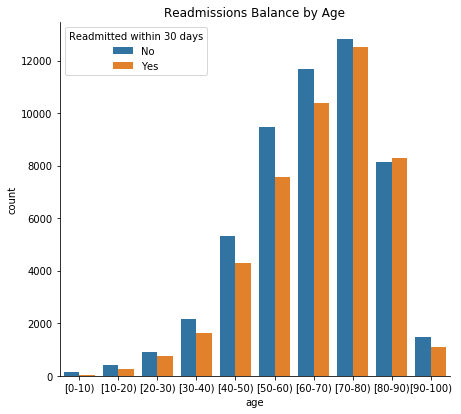
sns**.**despine()

ageplt**.**figure**.**set\_size\_inches(7, 6.5)

ageplt**.**legend(title **=** 'Readmitted within 30 days', labels **=** ('No', 'Yes'))

ageplt**.**axes**.**set\_title('Readmissions Balance by Age')

plt**.**show()



In [252]:

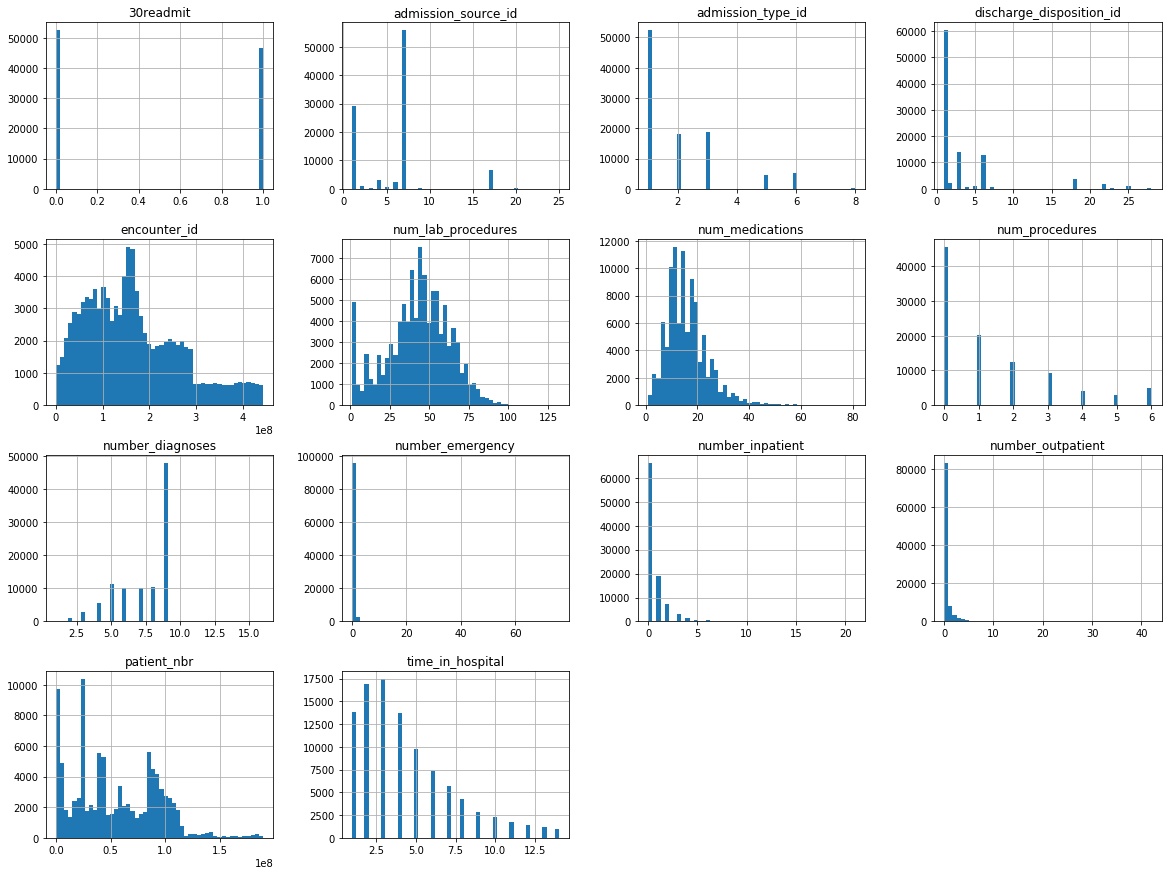
*#Plotting the numerical variables in our dataset,*

**%matplotlib** inline

**import** matplotlib.pyplot **as** plt

datacopy**.**hist(bins**=**50, figsize**=**(20,15))

plt**.**show()



In [253]:

*#Exploring the categorical variables,*

**import** seaborn **as** sns

fig, ax **=** plt**.**subplots(figsize**=**(15,10), ncols**=**2, nrows**=**2)

sns**.**countplot(x**=**"readmitted", data**=**datacopy, ax**=**ax[0][0])

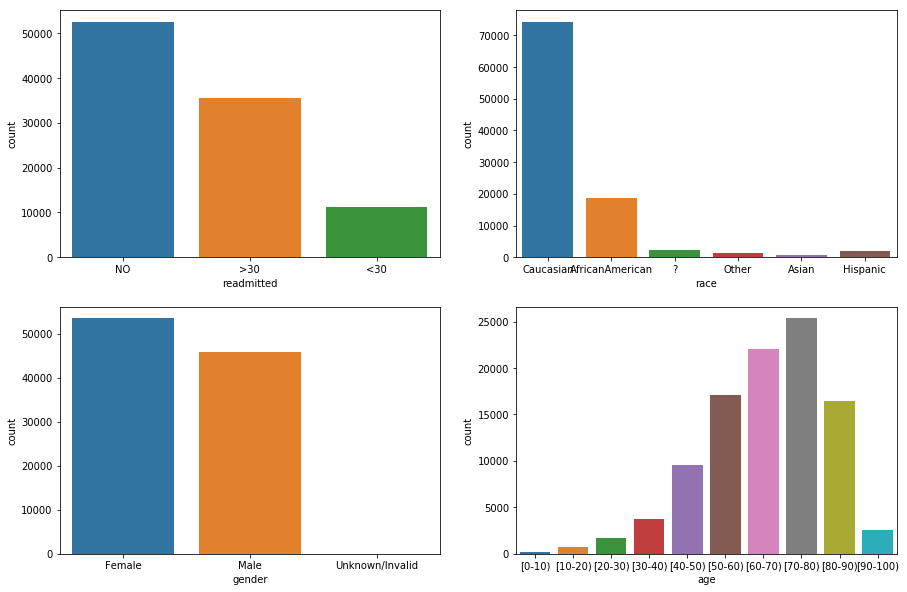
sns**.**countplot(x**=**"race", data**=**datacopy, ax**=**ax[0][1])

sns**.**countplot(x**=**"gender", data**=**datacopy, ax**=**ax[1][0])

sns**.**countplot(x**=**"age", data**=**datacopy, ax**=**ax[1][1])

Out[253]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x28b0d9c3cc0>



# Prediction using ML Models

Before applying logistic regression, we will have to make the data fit for performing logistic regression.

In [254]:

*# Cleaning the data, replacing the null values in numeric data by 0 and object data by unknown,*

numcolumn **=** datacopy**.**select\_dtypes(include **=** [np**.**number])**.**columns

objcolumn **=** datacopy**.**select\_dtypes(include **=** ['object'])**.**columns

In [255]:

*# Substituting 0 and unknown,*

datacopy[numcolumn] **=** datacopy[numcolumn]**.**fillna(0)

datacopy[objcolumn] **=** datacopy[objcolumn]**.**fillna("unknown")

In [256]:

datacopy**.**head(2)

Out[256]:

|  |
| --- |
| **0** |

2 rows × 48 columns

In [257]:

*#Encoding the data,*

**def** map\_now():

listname **=** [('infections', 139),

('neoplasms', (239 **-** 139)),

('endocrine', (279 **-** 239)),

('blood', (289 **-** 279)),

('mental', (319 **-** 289)),

('nervous', (359 **-** 319)),

('sense', (389 **-** 359)),

('circulatory', (459**-**389)),

('respiratory', (519**-**459)),

('digestive', (579 **-** 519)),

('genitourinary', (629 **-** 579)),

('pregnancy', (679 **-** 629)),

('skin', (709 **-** 679)),

('musculoskeletal', (739 **-** 709)),

('congenital', (759 **-** 739)),

('perinatal', (779 **-** 759)),

('ill-defined', (799 **-** 779)),

('injury', (999 **-** 799))]

dictcout **=** {}

count **=** 1

**for** name, num **in** listname:

**for** i **in** range(num):

dictcout**.**update({str(count): name})

count **+=** 1

**return** dictcout

**def** codemap(df, codes):

**import** pandas **as** pd

namecol **=** df**.**columns**.**tolist()

**for** col **in** namecol:

temp **=** []

**for** num **in** df[col]:

**if** ((num **is** **None**) **|** (num **in** ['unknown', '?']) **|** (pd**.**isnull(num))): temp**.**append('unknown')

**elif**(num**.**upper()[0] **==** 'V'): temp**.**append('supplemental')

**elif**(num**.**upper()[0] **==** 'E'): temp**.**append('injury')

**else**:

lkup **=** num**.**split('.')[0]

temp**.**append(codes[lkup])

df**.**loc[:, col] **=** temp

**return** df

listcol **=** ['diag\_1', 'diag\_2', 'diag\_3']

codes **=** map\_now()

datacopy[listcol] **=** codemap(datacopy[listcol], codes)

C:\Users\AKANKSHA\Anaconda3\lib\site-packages\pandas\core\indexing.py:621: SettingWithCopyWarning:

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/indexing.html#indexing-view-versus-copy

self.obj[item\_labels[indexer[info\_axis]]] = value

In [258]:

*#Now let's look at the dataset again and drop the irrelevant columns,*

datacopy**.**describe

Out[258]:

<bound method NDFrame.describe of encounter\_id patient\_nbr race gender age \

0 2278392 8222157 Caucasian Female [0-10)

1 149190 55629189 Caucasian Female [10-20)

2 64410 86047875 AfricanAmerican Female [20-30)

3 500364 82442376 Caucasian Male [30-40)

4 16680 42519267 Caucasian Male [40-50)

5 35754 82637451 Caucasian Male [50-60)

6 55842 84259809 Caucasian Male [60-70)

7 63768 114882984 Caucasian Male [70-80)

8 12522 48330783 Caucasian Female [80-90)

9 15738 63555939 Caucasian Female [90-100)

10 28236 89869032 AfricanAmerican Female [40-50)

11 36900 77391171 AfricanAmerican Male [60-70)

12 40926 85504905 Caucasian Female [40-50)

13 42570 77586282 Caucasian Male [80-90)

14 62256 49726791 AfricanAmerican Female [60-70)

15 73578 86328819 AfricanAmerican Male [60-70)

16 77076 92519352 AfricanAmerican Male [50-60)

17 84222 108662661 Caucasian Female [50-60)

18 89682 107389323 AfricanAmerican Male [70-80)

19 148530 69422211 ? Male [70-80)

20 150006 22864131 ? Female [50-60)

21 150048 21239181 ? Male [60-70)

22 182796 63000108 AfricanAmerican Female [70-80)

23 183930 107400762 Caucasian Female [80-90)

24 216156 62718876 AfricanAmerican Female [70-80)

25 221634 21861756 Other Female [50-60)

26 236316 40523301 Caucasian Male [80-90)

27 248916 115196778 Caucasian Female [50-60)

28 250872 41606064 Caucasian Male [20-30)

29 252822 18196434 Caucasian Female [80-90)

... ... ... ... ... ...

101735 443739044 106595208 Caucasian Male [70-80)

101736 443739152 90751788 Caucasian Female [60-70)

101737 443775086 125764160 Caucasian Female [40-50)

101738 443775482 95780439 Caucasian Male [70-80)

101739 443775740 30656952 AfricanAmerican Male [70-80)

101740 443778398 134647673 Caucasian Male [40-50)

101741 443787128 58160520 AfricanAmerican Male [90-100)

101742 443787512 52419276 Caucasian Male [70-80)

101744 443793992 43686936 Caucasian Female [80-90)

101745 443797076 183766055 Caucasian Male [50-60)

101746 443797298 89955270 Caucasian Male [70-80)

101747 443804570 33230016 Caucasian Female [70-80)

101748 443811536 189481478 Caucasian Female [40-50)

101749 443816024 106392411 Caucasian Female [70-80)

101750 443824292 138784172 Caucasian Female [80-90)

101751 443835140 175326800 Caucasian Male [70-80)

101752 443835512 139605341 Other Female [40-50)

101753 443841992 184875899 Other Male [40-50)

101754 443842016 183087545 Caucasian Female [70-80)

101755 443842022 188574944 Other Female [40-50)

101756 443842070 140199494 Other Female [60-70)

101757 443842136 181593374 Caucasian Female [70-80)

101758 443842340 120975314 Caucasian Female [80-90)

101759 443842778 86472243 Caucasian Male [80-90)

101760 443847176 50375628 AfricanAmerican Female [60-70)

101761 443847548 100162476 AfricanAmerican Male [70-80)

101762 443847782 74694222 AfricanAmerican Female [80-90)

101763 443854148 41088789 Caucasian Male [70-80)

101764 443857166 31693671 Caucasian Female [80-90)

101765 443867222 175429310 Caucasian Male [70-80)

admission\_type\_id discharge\_disposition\_id admission\_source\_id \

0 6 25 1

1 1 1 7

2 1 1 7

3 1 1 7

4 1 1 7

5 2 1 2

6 3 1 2

7 1 1 7

8 2 1 4

9 3 3 4

10 1 1 7

11 2 1 4

12 1 3 7

13 1 6 7

14 3 1 2

15 1 3 7

16 1 1 7

17 1 1 7

18 1 1 7

19 3 6 2

20 2 1 4

21 2 1 4

22 2 1 4

23 2 6 1

24 3 1 2

25 1 1 7

26 1 3 7

27 1 1 1

28 2 1 2

29 1 2 7

... ... ... ...

101735 2 6 7

101736 1 3 7

101737 3 1 1

101738 1 1 7

101739 1 1 7

101740 3 1 1

101741 1 3 7

101742 2 6 2

101744 1 1 7

101745 2 1 1

101746 1 1 7

101747 1 22 7

101748 1 4 7

101749 3 6 1

101750 3 1 1

101751 3 6 1

101752 3 1 1

101753 1 1 7

101754 1 1 7

101755 1 1 7

101756 1 1 7

101757 1 1 7

101758 1 1 7

101759 1 1 7

101760 1 1 7

101761 1 3 7

101762 1 4 5

101763 1 1 7

101764 2 3 7

101765 1 1 7

time\_in\_hospital num\_lab\_procedures ... insulin \

0 1 41 ... No

1 3 59 ... Up

2 2 11 ... No

3 2 44 ... Up

4 1 51 ... Steady

5 3 31 ... Steady

6 4 70 ... Steady

7 5 73 ... No

8 13 68 ... Steady

9 12 33 ... Steady

10 9 47 ... Steady

11 7 62 ... Steady

12 7 60 ... Down

13 10 55 ... Steady

14 1 49 ... Steady

15 12 75 ... Up

16 4 45 ... Steady

17 3 29 ... No

18 5 35 ... Steady

19 6 42 ... Steady

20 2 66 ... Down

21 2 36 ... Steady

22 2 47 ... No

23 11 42 ... No

24 3 19 ... Steady

25 1 33 ... No

26 6 64 ... No

27 2 25 ... Steady

28 10 53 ... Down

29 5 52 ... No

... ... ... ... ...

101735 6 46 ... Up

101736 8 54 ... Steady

101737 4 50 ... Steady

101738 1 37 ... No

101739 1 32 ... Steady

101740 1 1 ... Steady

101741 4 21 ... No

101742 4 43 ... Steady

101744 1 49 ... No

101745 3 57 ... No

101746 4 2 ... No

101747 8 51 ... Steady

101748 14 69 ... Down

101749 3 27 ... Steady

101750 3 31 ... Down

101751 13 77 ... Up

101752 3 13 ... Steady

101753 13 51 ... Down

101754 9 50 ... Steady

101755 14 73 ... Up

101756 2 46 ... Steady

101757 5 21 ... Steady

101758 5 76 ... Up

101759 1 1 ... Up

101760 6 45 ... Down

101761 3 51 ... Down

101762 5 33 ... Steady

101763 1 53 ... Down

101764 10 45 ... Up

101765 6 13 ... No

glyburide-metformin glipizide-metformin glimepiride-pioglitazone \

0 No No No

1 No No No

2 No No No

3 No No No

4 No No No

5 No No No

6 No No No

7 No No No

8 No No No

9 No No No

10 No No No

11 No No No

12 No No No

13 No No No

14 No No No

15 No No No

16 No No No

17 No No No

18 No No No

19 No No No

20 No No No

21 No No No

22 No No No

23 No No No

24 No No No

25 No No No

26 No No No

27 No No No

28 No No No

29 No No No

... ... ... ...

101735 No No No

101736 No No No

101737 No No No

101738 No No No

101739 No No No

101740 No No No

101741 No No No

101742 No No No

101744 No No No

101745 No No No

101746 No No No

101747 No No No

101748 No No No

101749 No No No

101750 No No No

101751 No No No

101752 No No No

101753 No No No

101754 No No No

101755 No No No

101756 No No No

101757 No No No

101758 No No No

101759 No No No

101760 No No No

101761 No No No

101762 No No No

101763 No No No

101764 No No No

101765 No No No

metformin-rosiglitazone metformin-pioglitazone change diabetesMed \

0 No No No No

1 No No Ch Yes

2 No No No Yes

3 No No Ch Yes

4 No No Ch Yes

5 No No No Yes

6 No No Ch Yes

7 No No No Yes

8 No No Ch Yes

9 No No Ch Yes

10 No No No Yes

11 No No Ch Yes

12 No No Ch Yes

13 No No No Yes

14 No No No Yes

15 No No Ch Yes

16 No No Ch Yes

17 No No No Yes

18 No No No Yes

19 No No Ch Yes

20 No No Ch Yes

21 No No Ch Yes

22 No No No No

23 No No No No

24 No No Ch Yes

25 No No No Yes

26 No No Ch Yes

27 No No No Yes

28 No No Ch Yes

29 No No Ch Yes

... ... ... ... ...

101735 No No Ch Yes

101736 No No No Yes

101737 No No Ch Yes

101738 No No No Yes

101739 No No No Yes

101740 No No Ch Yes

101741 No No No No

101742 No No Ch Yes

101744 No No No No

101745 No No Ch Yes

101746 No No No Yes

101747 No No No Yes

101748 No No Ch Yes

101749 No No Ch Yes

101750 No No Ch Yes

101751 No No Ch Yes

101752 No No Ch Yes

101753 No No Ch Yes

101754 No No Ch Yes

101755 No No Ch Yes

101756 No No No Yes

101757 No No No Yes

101758 No No Ch Yes

101759 No No Ch Yes

101760 No No Ch Yes

101761 No No Ch Yes

101762 No No No Yes

101763 No No Ch Yes

101764 No No Ch Yes

101765 No No No No

readmitted 30readmit

0 NO 0

1 >30 1

2 NO 0

3 NO 0

4 NO 0

5 >30 1

6 NO 0

7 >30 1

8 NO 0

9 NO 0

10 >30 1

11 <30 1

12 <30 1

13 NO 0

14 >30 1

15 NO 0

16 <30 1

17 NO 0

18 >30 1

19 NO 0

20 NO 0

21 NO 0

22 NO 0

23 >30 1

24 NO 0

25 NO 0

26 NO 0

27 >30 1

28 >30 1

29 >30 1

... ... ...

101735 NO 0

101736 >30 1

101737 NO 0

101738 NO 0

101739 NO 0

101740 NO 0

101741 NO 0

101742 NO 0

101744 NO 0

101745 NO 0

101746 <30 1

101747 >30 1

101748 >30 1

101749 NO 0

101750 <30 1

101751 NO 0

101752 NO 0

101753 NO 0

101754 >30 1

101755 >30 1

101756 >30 1

101757 NO 0

101758 NO 0

101759 NO 0

101760 >30 1

101761 >30 1

101762 NO 0

101763 NO 0

101764 NO 0

101765 NO 0

[99343 rows x 48 columns]>

In [259]:

data1 **=** datacopy**.**drop(['encounter\_id', "patient\_nbr", 'admission\_type\_id','readmitted'], axis **=**1)

In [260]:

data1**.**head(2)

Out[260]:

2 rows × 44 columns

#### Normalization of data

In [261]:

*#Normalization of the data,*

listnormal **=** ['time\_in\_hospital', 'num\_lab\_procedures', 'num\_procedures', 'num\_medications',

'number\_outpatient', 'number\_emergency', 'number\_inpatient', 'number\_diagnoses']

**from** sklearn.preprocessing **import** StandardScaler

normal **=** StandardScaler()

data1[listnormal] **=** normal**.**fit\_transform(data1[listnormal])

data1**.**describe()

Out[261]:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 99343.000000 | 99343.000000 | 9.934300e+04 | 9.934300e+04 | 9.934300e+04 | 9.934300e+04 | 9.934300e+04 | 9.934300e+04 | 9.934300e+04 | 9.934300e+04 | 99343.000000 |
| **mean** | 3.517882 | 5.731083 | 1.874346e-14 | -3.424256e-15 | 3.449070e-15 | -1.776343e-15 | 1.182191e-13 | 6.322393e-14 | -1.499301e-15 | -9.426101e-14 | 0.471256 |
| **std** | 5.184359 | 4.065029 | 1.000005e+00 | 1.000005e+00 | 1.000005e+00 | 1.000005e+00 | 1.000005e+00 | 1.000005e+00 | 1.000005e+00 | 1.000005e+00 | 0.499176 |
| **min** | 1.000000 | 1.000000 | -1.138438e+00 | -2.137026e+00 | -7.835647e-01 | -1.850439e+00 | -2.918627e-01 | -2.116216e-01 | -5.005746e-01 | -3.298145e+00 | 0.000000 |
| **25%** | 1.000000 | 1.000000 | -8.015554e-01 | -6.071886e-01 | -7.835647e-01 | -7.386238e-01 | -2.918627e-01 | -2.116216e-01 | -5.005746e-01 | -7.221572e-01 | 0.000000 |
| **50%** | 1.000000 | 7.000000 | -1.277904e-01 | 5.574070e-02 | -1.962887e-01 | -1.209485e-01 | -2.918627e-01 | -2.116216e-01 | -5.005746e-01 | 3.082380e-01 | 0.000000 |
| **75%** | 3.000000 | 7.000000 | 5.459746e-01 | 7.186700e-01 | 3.909872e-01 | 4.967267e-01 | -2.918627e-01 | -2.116216e-01 | 2.928105e-01 | 8.234355e-01 | 1.000000 |
| **max** | 28.000000 | 25.000000 | 3.241035e+00 | 4.543262e+00 | 2.740091e+00 | 8.032365e+00 | 3.290615e+01 | 8.083523e+01 | 1.616051e+01 | 4.429819e+00 | 1.000000 |

In [262]:

*#Let's store readmitted in y and rest of the columns in X,*

Y **=** data1['30readmit']

X **=** data1**.**drop(['30readmit'], axis **=**1)

X **=** pd**.**get\_dummies(X)

##### Splitting the data into training and vallidation data sets. The training data will contain 80 % of the data and validation will contain remaining 20%

In [263]:

**from** sklearn.model\_selection **import** train\_test\_split

Xtrain, Xtest, Ytrain, Ytest **=** train\_test\_split(X, Y, test\_size **=** .2,

random\_state **=** 7, stratify **=** Y)

In [264]:

print("shape of Xtrain,Xtest:",Xtrain**.**shape,Xtest**.**shape)

shape of Xtrain,Xtest: (79474, 169) (19869, 169)

# Logistic Regression

In [265]:

*# create model logistic as logistic regression using Sklearn*

**from** sklearn.linear\_model **import** LogisticRegression

logisticreg **=** LogisticRegression(tol**=**1e-7, penalty**=**'l2', C**=**0.0005)

logisticreg**.**fit(Xtrain, Ytrain)

Ylog **=** logisticreg**.**predict(Xtest)

In [266]:

*# Checking the accuracy of the model*

print(" The accuracy of the Logistic regression model:" ,logisticreg**.**score(Xtest, Ytest))

The accuracy of the Logistic regression model: 0.6277115103930746

In [267]:

*# checking the confusion matrix*

**from** sklearn.metrics **import** confusion\_matrix

print(confusion\_matrix(Ytest, Ylog))

[[8376 2130]

[5267 4096]]

In [268]:

plt**.**figure(figsize**=**(9,9))

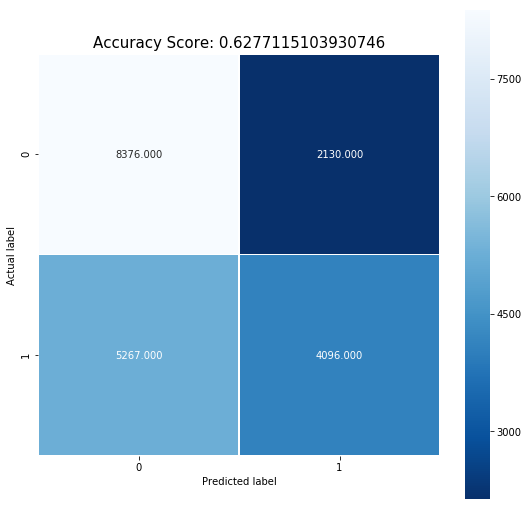
sns**.**heatmap(confusion\_matrix(Ytest, Ylog), annot**=True**, fmt**=**".3f", linewidths**=**.5, square **=** **True**, cmap **=** 'Blues\_r');

plt**.**ylabel('Actual label');

plt**.**xlabel('Predicted label');

all\_sample\_title **=** 'Accuracy Score: {0}'**.**format(logisticreg**.**score(Xtest, Ytest))

plt**.**title(all\_sample\_title, size **=** 15);



In [269]:

*# Checking the summary of classification*

**from** sklearn.metrics **import** classification\_report

print(classification\_report(Ytest, Ylog, target\_names **=** ['NO', 'YES']))

precision recall f1-score support

NO 0.61 0.80 0.69 10506

YES 0.66 0.44 0.53 9363

avg / total 0.63 0.63 0.61 19869

In [270]:

*# Checking the performance of model using ROC curve plots*

YScre **=** logisticreg**.**decision\_function(Xtest)

In [271]:

*##Computing false and true positive rates*

**from** sklearn.metrics **import** roc\_curve

fpr, tpr,\_**=**roc\_curve(logisticreg**.**predict(Xtrain),Ytrain,drop\_intermediate**=False**)

**import** matplotlib.pyplot **as** plt

plt**.**figure()

*##Creating the ROC,*

plt**.**plot(fpr, tpr, color**=**'blue',

lw**=**2, label**=**'ROC curve')

*##Finding FPR and TPR,*

plt**.**plot([0, 1], [0, 1], color**=**'red', lw**=**2, linestyle**=**'--')

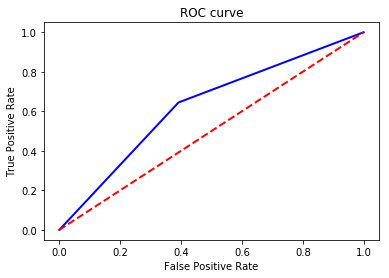
*##Splecifying the label and title,*

plt**.**xlabel('False Positive Rate')

plt**.**ylabel('True Positive Rate')

plt**.**title('ROC curve')

plt**.**show()



##### As seen from the above, the performance of our model is average, not too great. So, let's go ahead and see other models.

# Random Forest Classifier

In [35]:

**from** sklearn.ensemble **import** RandomForestClassifier

random\_forest **=** RandomForestClassifier(random\_state**=**42,n\_estimators**=**500)

random\_forest**.**fit(Xtrain, Ytrain)

Yrandforest **=** random\_forest**.**predict(Xtest)

In [36]:

*#Calculating the score,*

scorey **=**random\_forest**.**predict\_proba(Xtest)[:,1]

rfpr, rtpr, thresholds **=** roc\_curve(Ytest, scorey)

In [37]:

*#Checking the accuracy,*

print(" Accuracy of Randomeforest classification: ", random\_forest**.**score(Xtest, Ytest))

Accuracy of Randomeforest classification: 0.6362675524686698

In [38]:

*#Printing the confusion matrix,*

print(confusion\_matrix(Ytest, Yrandforest))

[[7559 2947]

[4280 5083]]

In [39]:

plt**.**figure(figsize**=**(9,9))

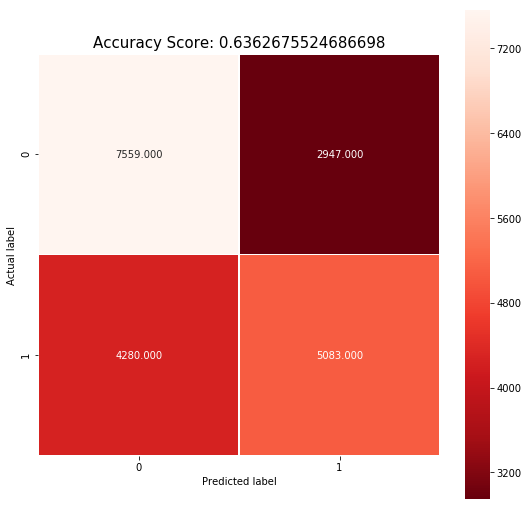
sns**.**heatmap(confusion\_matrix(Ytest, Yrandforest), annot**=True**, fmt**=**".3f", linewidths**=**.5, square **=** **True**, cmap **=** 'Reds\_r');

plt**.**ylabel('Actual label');

plt**.**xlabel('Predicted label');

all\_sample\_title **=** 'Accuracy Score: {0}'**.**format(random\_forest**.**score(Xtest, Ytest))

plt**.**title(all\_sample\_title, size **=** 15);



In [40]:

print(classification\_report(Ytest, Yrandforest, target\_names **=** ['NO', 'YES']))

precision recall f1-score support

NO 0.64 0.72 0.68 10506

YES 0.63 0.54 0.58 9363

avg / total 0.64 0.64 0.63 19869

In [293]:

*#Determining which features are most important,*

feature\_names **=** Xtrain**.**columns

feature\_imports **=** random\_forest**.**feature\_importances\_

most\_imp\_features **=** pd**.**DataFrame([f **for** f **in** zip(feature\_names,feature\_imports)], columns**=**["Feature", "Importance"])**.**nlargest(10, "Importance")

most\_imp\_features**.**sort\_values(by**=**"Importance", inplace**=True**)

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

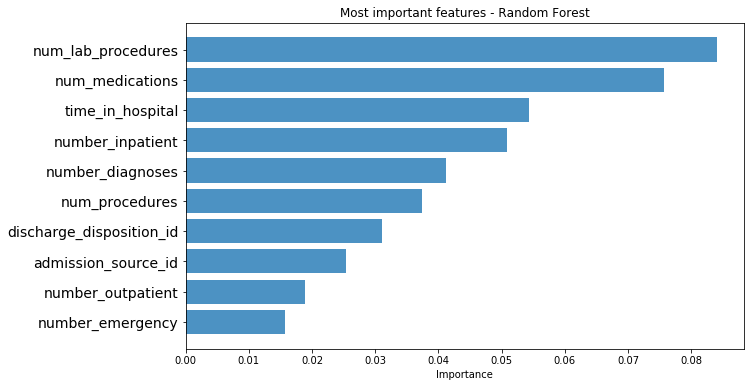
plt**.**barh(range(len(most\_imp\_features)), most\_imp\_features**.**Importance, align**=**'center', alpha**=**0.8)

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()



# AdaBoosted Classification model

In [272]:

*#Creating a AdaBoosted Classification model,*

**from** sklearn.ensemble **import** AdaBoostClassifier

adaclass **=** AdaBoostClassifier(n\_estimators **=** 20, learning\_rate **=** 0.2, random\_state **=** 123)

adaclass**.**fit(Xtrain, Ytrain)

yadaclas **=** adaclass**.**predict(Xtest)

In [273]:

print("The accurary of AdaBoosted Classification model: ", adaclass**.**score(Xtest, Ytest))

The accurary of AdaBoosted Classification model: 0.6254466757260053

In [274]:

*#Checking the confusion matrix,*

print(confusion\_matrix(Ytest, yadaclas))

[[8328 2178]

[5264 4099]]

In [275]:

plt**.**figure(figsize**=**(9,9))

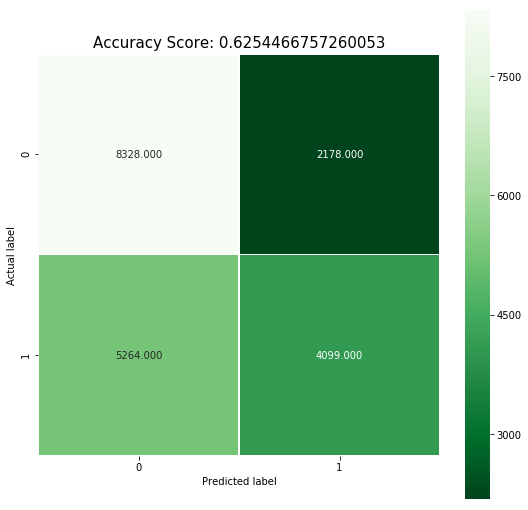
sns**.**heatmap(confusion\_matrix(Ytest, yadaclas), annot**=True**, fmt**=**".3f", linewidths**=**.5, square **=** **True**, cmap **=** 'Greens\_r');

plt**.**ylabel('Actual label');

plt**.**xlabel('Predicted label');

all\_sample\_title **=** 'Accuracy Score: {0}'**.**format(adaclass**.**score(Xtest, Ytest))

plt**.**title(all\_sample\_title, size **=** 15);



In [276]:

*#Checking recall, f1 :-*

print(classification\_report(Ytest, yadaclas, target\_names **=** ['NO', 'YES']))

precision recall f1-score support

NO 0.61 0.79 0.69 10506

YES 0.65 0.44 0.52 9363

avg / total 0.63 0.63 0.61 19869

In [277]:

*# calculate the False Positive and True Positive rate to get the ROC curve*

yadaclas **=** adaclass**.**decision\_function(Xtest)

fpr\_adaclass, tpr\_adaclass, thresholds **=** roc\_curve(Ytest, yadaclas)

##### The accuracy can be improved by tuning the model, so let's do that.

# Hyperparameters Tunning for AdaBoosted

In [47]:

*# Doing parameter tuning,*

**from** sklearn.ensemble **import** AdaBoostClassifier

**from** sklearn.model\_selection **import** GridSearchCV

ada\_boost **=** AdaBoostClassifier(n\_estimators **=** 20, learning\_rate **=** 0.2, random\_state **=** 123)

gridparam **=**{

'n\_estimators': [100, 200,500],

'learning\_rate': [0.2,0.5,1.0],

},

adagrid **=** GridSearchCV(ada\_boost, cv**=**3, n\_jobs**=**3, param\_grid**=**gridparam)

adagrid**.**fit(Xtrain, Ytrain)

Out[47]:

GridSearchCV(cv=3, error\_score='raise',

estimator=AdaBoostClassifier(algorithm='SAMME.R', base\_estimator=None,

learning\_rate=0.2, n\_estimators=20, random\_state=123),

fit\_params=None, iid=True, n\_jobs=3,

param\_grid=({'n\_estimators': [100, 200, 500], 'learning\_rate': [0.2, 0.5, 1.0]},),

pre\_dispatch='2\*n\_jobs', refit=True, return\_train\_score='warn',

scoring=None, verbose=0)

In [48]:

*#The accuracy of the model with the best parameters*

adagrid**.**score(Xtest, Ytest)

Out[48]:

0.6380794202023252

In [49]:

gridyada **=** adagrid**.**predict(Xtest)

print(classification\_report(Ytest, gridyada, target\_names **=** ['NO', 'YES']))

precision recall f1-score support

NO 0.63 0.76 0.69 10506

YES 0.65 0.50 0.57 9363

avg / total 0.64 0.64 0.63 19869

In [54]:

ygridadascore **=** adagrid**.**decision\_function(Xtest)

fpr\_adamod, tpr\_adamod, thresholds\_grid\_ada **=** roc\_curve(Ytest, ygridadascore)

In [ ]:

# Decision tree

In [278]:

**import** numpy **as** np

**import** pandas **as** pd

**from** sklearn.cross\_validation **import** train\_test\_split

**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn.metrics **import** accuracy\_score

**from** sklearn **import** tree

In [279]:

clfgini **=** DecisionTreeClassifier(criterion **=** "gini", random\_state **=** 100,

max\_depth**=**3, min\_samples\_leaf**=**5)

clfgini**.**fit(Xtrain, Ytrain)

Out[279]:

DecisionTreeClassifier(class\_weight=None, criterion='gini', max\_depth=3,

max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=5, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, presort=False, random\_state=100,

splitter='best')

In [280]:

ypreddt **=** clfgini**.**predict(Xtest)

ypreddt

Out[280]:

array([1, 1, 0, ..., 1, 0, 1])

In [281]:

print( "Accuracy is ", accuracy\_score(Ytest,ypreddt)**\***100)

Accuracy is 62.358447833308176

# Comparing the models

In [77]:

fpr\_log, tpr\_log, thresholds **=** roc\_curve(Ytest, YScre)*#logistic regression*

fpr\_rf, tpr\_rf, thresholds **=** roc\_curve(Ytest, Yrandforest)*#random forest classifier*

fpr\_adaclf, tpr\_adaclf, thresholds **=** roc\_curve(Ytest, yadaclas)*#Ada boost classifier*

fpr\_adamod, tpr\_adamod, thresholds **=** roc\_curve(Ytest,ygridadascore )*#Hyperparameters Tunning for AdaBoosted*

fpr\_dt, tpr\_dt, thresholds **=** roc\_curve(Ytest,ypreddt )

In [78]:

*#compare the ROC curve between different models*

plt**.**figure(figsize**=**(8,8))

plt**.**plot(fpr\_log, tpr\_log, label**=**'Logistic regression')

plt**.**plot(fpr\_adaclf, tpr\_adaclf, label**=**'Adaboost Classifier')

plt**.**plot(fpr\_rf, tpr\_rf, label**=**'Randomforest Classifier')

plt**.**plot(fpr\_adamod, tpr\_adamod, label**=**'Adaboost with the best Pars')

plt**.**plot(fpr\_dt, tpr\_dt, label**=**'Decision Tree')

plt**.**plot([0, 1], [0, 1], linestyle**=**'--', lw**=**2, color**=**'r',

label**=**'random', alpha**=**.8)

plt**.**xlim([0,1])

plt**.**ylim([0,1])

plt**.**xticks(np**.**arange(0,1.1,0.1))

plt**.**yticks(np**.**arange(0,1.1,0.1))

plt**.**grid()

plt**.**legend()

plt**.**axes()**.**set\_aspect('equal')

plt**.**xlabel('False Positive Rate')

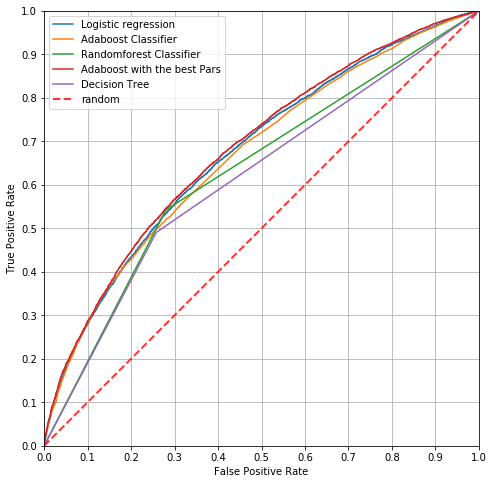
plt**.**ylabel('True Positive Rate')

C:\Users\AKANKSHA\Anaconda3\lib\site-packages\matplotlib\cbook\deprecation.py:106: MatplotlibDeprecationWarning: Adding an axes using the same arguments as a previous axes currently reuses the earlier instance. In a future version, a new instance will always be created and returned. Meanwhile, this warning can be suppressed, and the future behavior ensured, by passing a unique label to each axes instance.

warnings.warn(message, mplDeprecation, stacklevel=1)

Out[78]:

Text(0,0.5,'True Positive Rate')



In [79]:

print(" Accuracy of Logistic regression model:" ,logisticreg**.**score(Xtest, Ytest)**\***100)

print("Accuracy of Random forest classification: ", random\_forest**.**score(Xtest, Ytest)**\***100)

print("Accuracy of AdaBoosted Classification model: ", adaclass**.**score(Xtest, Ytest)**\***100)

print("Accuracy of Hyperparameter Tuning AdaBoosted Classification model: ", adagrid**.**score(Xtest, Ytest)**\***100)

print( "Accuracy of Decision Tree ", accuracy\_score(Ytest,ypreddt)**\***100)

Accuracy of Logistic regression model: 62.77115103930746

Accuracy of Random forest classification: 63.626755246866985

Accuracy of AdaBoosted Classification model: 62.54466757260053

Accuracy of Hyperparameter Tuning AdaBoosted Classification model: 63.807942020232524

Accuracy of Decision Tree 62.358447833308176