## Practicum 2

## Predict of Diabetes hospital readmissions

###### Team:

###### Satwick Bodduna (11810077)

###### V .Venkata Sai Kushwanth Reddy (11810002)

#Importing requried packages

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import warnings

from sklearn import metrics

from sklearn import preprocessing

#Importing the dataset

diabetic\_data\_temp=pd.read\_csv("diabetic\_data.csv")

print("diabetic\_data\_temp shape",diabetic\_data\_temp.shape) #Checking the shape of the Data

Orginal\_Shape=diabetic\_data\_temp.shape[0] #Storing shape of the data into the variable for furthe use

print(diabetic\_data\_temp.info()) #Checking the Datatypes

diabetic\_data\_temp shape (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

#Checking the data

diabetic\_data\_temp.head()

|  | **encounter\_id** | **patient\_nbr** | **race** | **gender** | **age** | **weight** | **admission\_type\_id** | **discharge\_disposition\_id** | **admission\_source\_id** | **time\_in\_hospital** | **...** | **citoglipton** | **insulin** | **glyburide-metformin** | **glipizide-metformin** | **glimepiride-pioglitazone** | **metformin-rosiglitazone** | **metformin-pioglitazone** | **change** | **diabetesMed** | **readmitted** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2278392 | 8222157 | Caucasian | Female | [0-10) | ? | 6 | 25 | 1 | 1 | ... | No | No | No | No | No | No | No | No | No | NO |
| **1** | 149190 | 55629189 | Caucasian | Female | [10-20) | ? | 1 | 1 | 7 | 3 | ... | No | Up | No | No | No | No | No | Ch | Yes | >30 |
| **2** | 64410 | 86047875 | AfricanAmerican | Female | [20-30) | ? | 1 | 1 | 7 | 2 | ... | No | No | No | No | No | No | No | No | Yes | NO |
| **3** | 500364 | 82442376 | Caucasian | Male | [30-40) | ? | 1 | 1 | 7 | 2 | ... | No | Up | No | No | No | No | No | Ch | Yes | NO |
| **4** | 16680 | 42519267 | Caucasian | Male | [40-50) | ? | 1 | 1 | 7 | 1 | ... | No | Steady | No | No | No | No | No | Ch | Yes | NO |

5 rows × 50 columns

#Checking the col's

diabetic\_data\_temp.columns

Index(['encounter\_id', 'patient\_nbr', 'race', 'gender', 'age', 'weight',

'admission\_type\_id', 'discharge\_disposition\_id', 'admission\_source\_id',

'time\_in\_hospital', 'payer\_code', 'medical\_specialty',

'num\_lab\_procedures', 'num\_procedures', 'num\_medications',

'number\_outpatient', 'number\_emergency', 'number\_inpatient', 'diag\_1',

'diag\_2', 'diag\_3', 'number\_diagnoses', 'max\_glu\_serum', 'A1Cresult',

'metformin', 'repaglinide', 'nateglinide', 'chlorpropamide',

'glimepiride', 'acetohexamide', 'glipizide', 'glyburide', 'tolbutamide',

'pioglitazone', 'rosiglitazone', 'acarbose', 'miglitol', 'troglitazone',

'tolazamide', 'examide', 'citoglipton', 'insulin',

'glyburide-metformin', 'glipizide-metformin',

'glimepiride-pioglitazone', 'metformin-rosiglitazone',

'metformin-pioglitazone', 'change', 'diabetesMed', 'readmitted'],

dtype='object')

#Taking a copy of the data into new data frame further processing

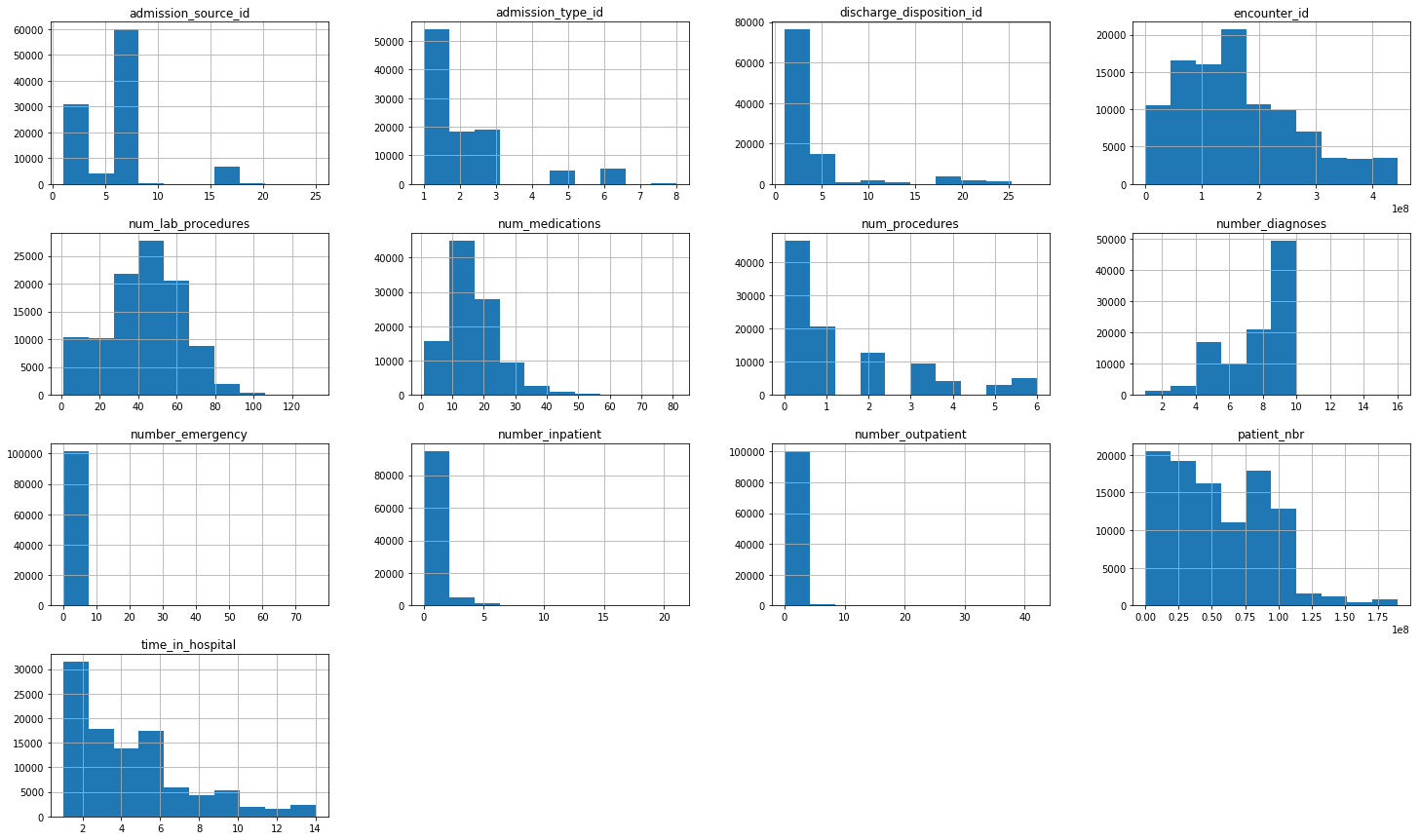
diabetic\_data=diabetic\_data\_temp.copy()

# Data Analysis (EDA)

#Exploring histograms of numerical columns

diabetic\_data\_temp.hist(bins=10, figsize=(25,15))

plt.show()

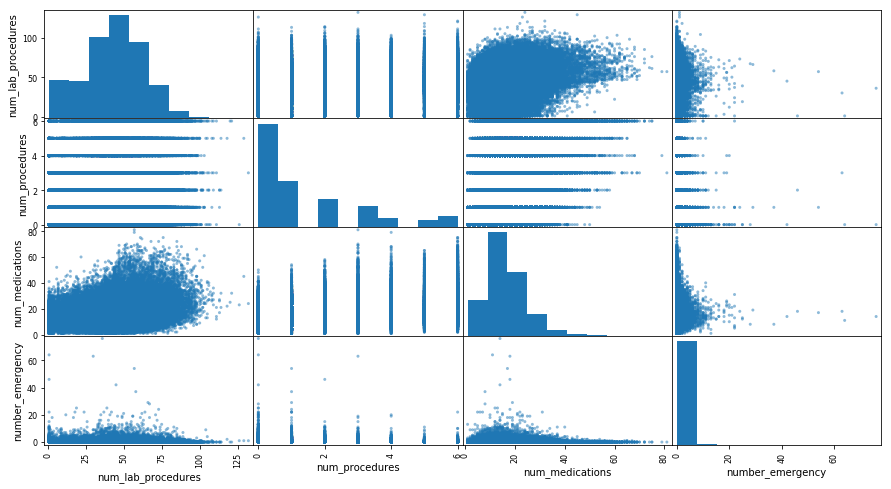
****

# Verifying relation between counting columns in the data

pd.scatter\_matrix(diabetic\_data\_temp[['num\_lab\_procedures','num\_procedures', 'num\_medications', 'number\_emergency']], figsize = (15, 8))

plt.show()

C:\Users\satwi\Anaconda3\lib\site-packages\ipykernel\_launcher.py:2: FutureWarning: pandas.scatter\_matrix is deprecated, use pandas.plotting.scatter\_matrix instead

****

#Exploring ditribution of variables in the datset

fig, ax = plt.subplots(figsize=(20,15), ncols=2, nrows=2)

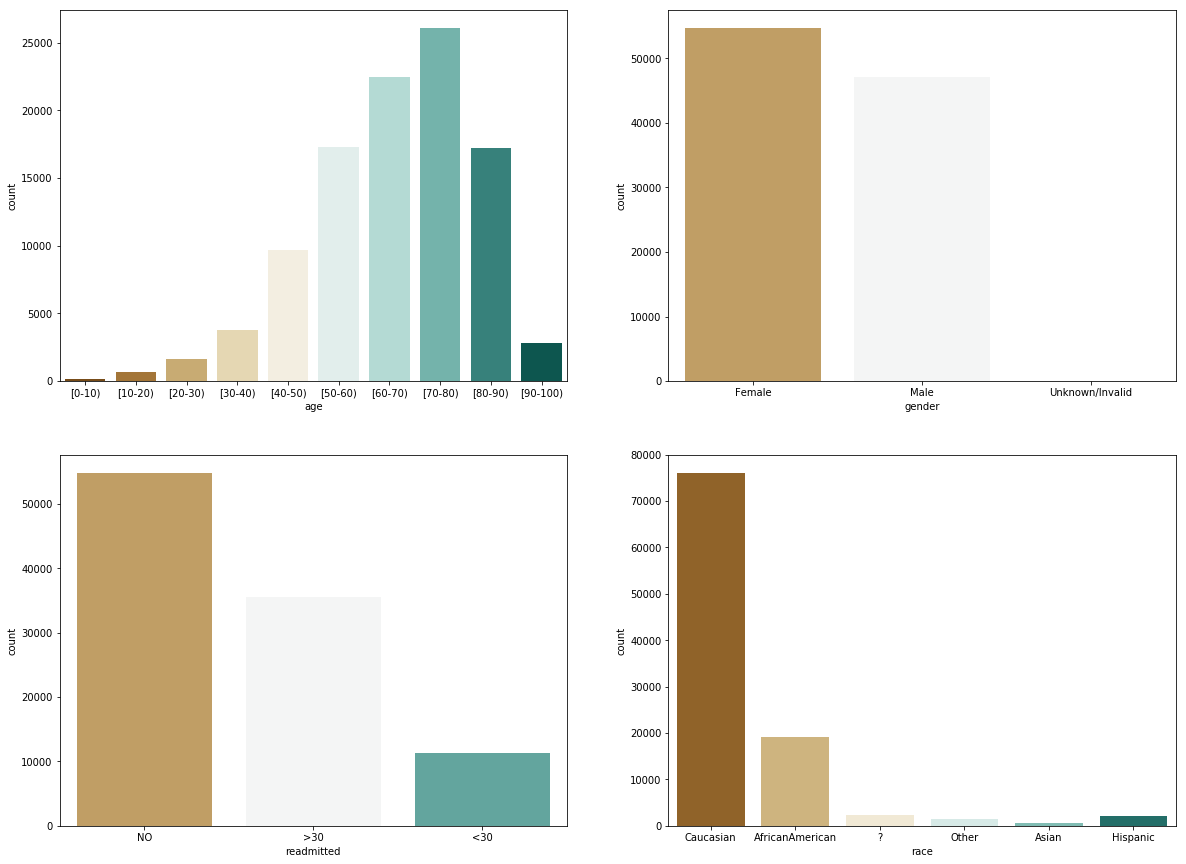
sns.countplot(x="age", data=diabetic\_data\_temp, ax=ax[0][0],palette="BrBG")

sns.countplot(x="gender", data=diabetic\_data\_temp, ax=ax[0][1],palette="BrBG")

sns.countplot(x="readmitted", data=diabetic\_data\_temp, ax=ax[1][0],palette="BrBG")

sns.countplot(x="race", data=diabetic\_data\_temp, ax=ax[1][1],palette="BrBG")

plt.show()

****

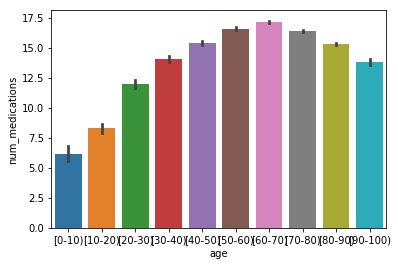
#frequency distribution of medications across age groups

ax = sns.barplot(x="age", y="num\_medications", data=diabetic\_data\_temp)

plt.show()

C:\Users\satwi\Anaconda3\lib\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

****

### Data Cleaning

#Checking for number of question marks in all the coloms

for i in diabetic\_data.columns:

if diabetic\_data[i].dtype =='object':

x=(i,diabetic\_data[i][diabetic\_data[i]=='?'].count())

if x[1]>0:#printing which has values >0

print(i,round(((x[1])/len(diabetic\_data))\*100,2))

race 2.23

weight 96.86

payer\_code 39.56

medical\_specialty 49.08

diag\_1 0.02

diag\_2 0.35

diag\_3 1.4

#From Above information dropping the col's which have more than 2 %

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

#Checking number of unique attribute in field

for i in diabetic\_data.columns:

x=len(diabetic\_data[i].unique())

if x<=1:

print(i,x)

examide 1

citoglipton 1

#from above information we have 2 cols with one unique record so it doent give any information for traning the model so

#we can drop those fields from the datafram

diabetic\_data=diabetic\_data.drop(['examide','citoglipton'],axis=1)

#Dropping rows which atleast dont have any of the diagnostics

droppingrowindex=set(diabetic\_data[(diabetic\_data['diag\_1']=='?')&(diabetic\_data['diag\_2']=='?')&(diabetic\_data['diag\_3']=='?')].index)

index=list(set(diabetic\_data.index)-set(droppingrowindex)) #checking the indexes

diabetic\_data=diabetic\_data.loc[index]

#Checking the unqiue records of gender for dropping unwanted records

diabetic\_data['gender'].unique()

array(['Female', 'Male', 'Unknown/Invalid'], dtype=object)

#Dropping Gender which has 'Unknown/Invalid 'values

droppingrowindex\_Gender=set(diabetic\_data[(diabetic\_data['gender']=='Unknown/Invalid')].index)

reindex=list(set(diabetic\_data.index)-set(droppingrowindex\_Gender))

diabetic\_data=diabetic\_data.loc[reindex]

# Removing the pateints from the dataset that are dead by checking the mapping sheet which they have provided.

diabetic\_data = diabetic\_data[((diabetic\_data.discharge\_disposition\_id != 21) &

(diabetic\_data.discharge\_disposition\_id != 19) &

(diabetic\_data.discharge\_disposition\_id != 14) &

(diabetic\_data.discharge\_disposition\_id != 13) &

(diabetic\_data.discharge\_disposition\_id != 20) &

(diabetic\_data.discharge\_disposition\_id != 11))]

#Dropping the race which has nulls in the data

diabetic\_data = diabetic\_data[diabetic\_data['race'] != '?']

diabetic\_data.head()

|  | **encounter\_id** | **patient\_nbr** | **race** | **gender** | **age** | **admission\_type\_id** | **discharge\_disposition\_id** | **admission\_source\_id** | **time\_in\_hospital** | **num\_lab\_procedures** | **...** | **tolazamide** | **insulin** | **glyburide-metformin** | **glipizide-metformin** | **glimepiride-pioglitazone** | **metformin-rosiglitazone** | **metformin-pioglitazone** | **change** | **diabetesMed** | **readmitted** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2278392 | 8222157 | Caucasian | Female | [0-10) | 6 | 25 | 1 | 1 | 41 | ... | No | No | No | No | No | No | No | No | No | NO |
| **1** | 149190 | 55629189 | Caucasian | Female | [10-20) | 1 | 1 | 7 | 3 | 59 | ... | No | Up | No | No | No | No | No | Ch | Yes | >30 |
| **2** | 64410 | 86047875 | AfricanAmerican | Female | [20-30) | 1 | 1 | 7 | 2 | 11 | ... | No | No | No | No | No | No | No | No | Yes | NO |
| **3** | 500364 | 82442376 | Caucasian | Male | [30-40) | 1 | 1 | 7 | 2 | 44 | ... | No | Up | No | No | No | No | No | Ch | Yes | NO |
| **4** | 16680 | 42519267 | Caucasian | Male | [40-50) | 1 | 1 | 7 | 1 | 51 | ... | No | Steady | No | No | No | No | No | Ch | Yes | NO |

5 rows × 45 columns

#Number of records from from the orginal dataset

Orginal\_Shape-diabetic\_data.shape[0]

4658

#### Feature Enginering

#Mapping the Diseases codes with ICD (International Classification of Diseases)Groups

def icd\_code():

diseaseclassifications = [('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 diseaseclassifications:

for i in range(num):

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

count += 1

return dictcout

def map\_icd(df, icd):

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(icd[lkup])

df.loc[:, col] = temp

return df

updatedcol = ['diag\_1', 'diag\_2', 'diag\_3']

icd = icd\_code()

diabetic\_data[updatedcol] = map\_icd(diabetic\_data[updatedcol], icd)

C:\Users\satwi\Anaconda3\lib\site-packages\pandas\core\indexing.py:630: 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

diabetic\_data.head()

|  | **encounter\_id** | **patient\_nbr** | **race** | **gender** | **age** | **admission\_type\_id** | **discharge\_disposition\_id** | **admission\_source\_id** | **time\_in\_hospital** | **num\_lab\_procedures** | **...** | **tolazamide** | **insulin** | **glyburide-metformin** | **glipizide-metformin** | **glimepiride-pioglitazone** | **metformin-rosiglitazone** | **metformin-pioglitazone** | **change** | **diabetesMed** | **readmitted** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2278392 | 8222157 | Caucasian | Female | [0-10) | 6 | 25 | 1 | 1 | 41 | ... | No | No | No | No | No | No | No | No | No | NO |
| **1** | 149190 | 55629189 | Caucasian | Female | [10-20) | 1 | 1 | 7 | 3 | 59 | ... | No | Up | No | No | No | No | No | Ch | Yes | >30 |
| **2** | 64410 | 86047875 | AfricanAmerican | Female | [20-30) | 1 | 1 | 7 | 2 | 11 | ... | No | No | No | No | No | No | No | No | Yes | NO |
| **3** | 500364 | 82442376 | Caucasian | Male | [30-40) | 1 | 1 | 7 | 2 | 44 | ... | No | Up | No | No | No | No | No | Ch | Yes | NO |
| **4** | 16680 | 42519267 | Caucasian | Male | [40-50) | 1 | 1 | 7 | 1 | 51 | ... | No | Steady | No | No | No | No | No | Ch | Yes | NO |

5 rows × 45 columns

###### Converting Categorical values into Factors

keys = ['rosiglitazone','glimepiride-pioglitazone','tolazamide','metformin','tolbutamide', 'glipizide', 'glyburide','repaglinide', 'nateglinide', 'chlorpropamide', 'glimepiride',

'pioglitazone', 'acarbose', 'miglitol', 'insulin', 'glyburide-metformin',

'metformin-pioglitazone','metformin-rosiglitazone', 'glipizide-metformin',

'troglitazone', 'acetohexamide']

for col in keys:

diabetic\_data[col] = diabetic\_data[col].replace('No', 0)

diabetic\_data[col] = diabetic\_data[col].replace('Steady', 1)

diabetic\_data[col] = diabetic\_data[col].replace('Up', 1)

diabetic\_data[col] = diabetic\_data[col].replace('Down', 1)

diabetic\_data['change'] = diabetic\_data['change'].replace('Ch', 1)

diabetic\_data['change'] = diabetic\_data['change'].replace('No', 0)

diabetic\_data['gender'] = diabetic\_data['gender'].replace('Male', 1)

diabetic\_data['gender'] = diabetic\_data['gender'].replace('Female', 0)

diabetic\_data['diabetesMed'] = diabetic\_data['diabetesMed'].replace('Yes', 1)

diabetic\_data['diabetesMed'] = diabetic\_data['diabetesMed'].replace('No', 0)

diabetic\_data['admission\_type\_id'] = diabetic\_data['admission\_type\_id'].replace(2,1)

diabetic\_data['admission\_type\_id'] = diabetic\_data['admission\_type\_id'].replace(7,1)

diabetic\_data['admission\_type\_id'] = diabetic\_data['admission\_type\_id'].replace(6,5)

diabetic\_data['admission\_type\_id'] = diabetic\_data['admission\_type\_id'].replace(8,5)

diabetic\_data['max\_glu\_serum'] = diabetic\_data['max\_glu\_serum'].replace('>200', 1)

diabetic\_data['max\_glu\_serum'] = diabetic\_data['max\_glu\_serum'].replace('>300', 1)

diabetic\_data['max\_glu\_serum'] = diabetic\_data['max\_glu\_serum'].replace('Norm', 0)

diabetic\_data['max\_glu\_serum'] = diabetic\_data['max\_glu\_serum'].replace('None', 0)

diabetic\_data['A1Cresult'] = diabetic\_data['A1Cresult'].replace('>7', 1)

diabetic\_data['A1Cresult'] = diabetic\_data['A1Cresult'].replace('>8', 1)

diabetic\_data['A1Cresult'] = diabetic\_data['A1Cresult'].replace('Norm', 0)

diabetic\_data['A1Cresult'] = diabetic\_data['A1Cresult'].replace('None', 0)

diabetic\_data['readmitted'] = diabetic\_data['readmitted'].replace('<30', 1)

diabetic\_data['readmitted'] = diabetic\_data['readmitted'].replace('>30', 0)

diabetic\_data['readmitted'] = diabetic\_data['readmitted'].replace('NO', 0)

age\_dict = {'[0-10)':5, '[10-20)':15, '[20-30)':25, '[30-40)':35, '[40-50)':45, '[50-60)':55, '[60-70)':65, '[70-80)':75, '[80-90)':85, '[90-100)':95}

diabetic\_data['age'] = diabetic\_data.age.map(age\_dict)

diabetic\_data['age'] = diabetic\_data['age'].astype('int64')

diabetic\_data.head()

|  | **encounter\_id** | **patient\_nbr** | **race** | **gender** | **age** | **admission\_type\_id** | **discharge\_disposition\_id** | **admission\_source\_id** | **time\_in\_hospital** | **num\_lab\_procedures** | **...** | **tolazamide** | **insulin** | **glyburide-metformin** | **glipizide-metformin** | **glimepiride-pioglitazone** | **metformin-rosiglitazone** | **metformin-pioglitazone** | **change** | **diabetesMed** | **readmitted** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2278392 | 8222157 | Caucasian | 0 | 5 | 5 | 25 | 1 | 1 | 41 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 149190 | 55629189 | Caucasian | 0 | 15 | 1 | 1 | 7 | 3 | 59 | ... | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| **2** | 64410 | 86047875 | AfricanAmerican | 0 | 25 | 1 | 1 | 7 | 2 | 11 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| **3** | 500364 | 82442376 | Caucasian | 1 | 35 | 1 | 1 | 7 | 2 | 44 | ... | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| **4** | 16680 | 42519267 | Caucasian | 1 | 45 | 1 | 1 | 7 | 1 | 51 | ... | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

5 rows × 45 columns

### Data Tranformation on Numerical Variables

#### Checking the data distribution for making transformation on the data

diabetic\_data\_temp=diabetic\_data

diabetic\_data\_temp.columns

Index(['encounter\_id', 'patient\_nbr', 'race', 'gender', 'age',

'admission\_type\_id', 'discharge\_disposition\_id', 'admission\_source\_id',

'time\_in\_hospital', 'num\_lab\_procedures', 'num\_procedures',

'num\_medications', 'number\_outpatient', 'number\_emergency',

'number\_inpatient', 'diag\_1', 'diag\_2', 'diag\_3', 'number\_diagnoses',

'max\_glu\_serum', 'A1Cresult', 'metformin', 'repaglinide', 'nateglinide',

'chlorpropamide', 'glimepiride', 'acetohexamide', 'glipizide',

'glyburide', 'tolbutamide', 'pioglitazone', 'rosiglitazone', 'acarbose',

'miglitol', 'troglitazone', 'tolazamide', 'insulin',

'glyburide-metformin', 'glipizide-metformin',

'glimepiride-pioglitazone', 'metformin-rosiglitazone',

'metformin-pioglitazone', 'change', 'diabetesMed', 'readmitted'],

dtype='object')

fig = plt.figure(figsize = (20,20))

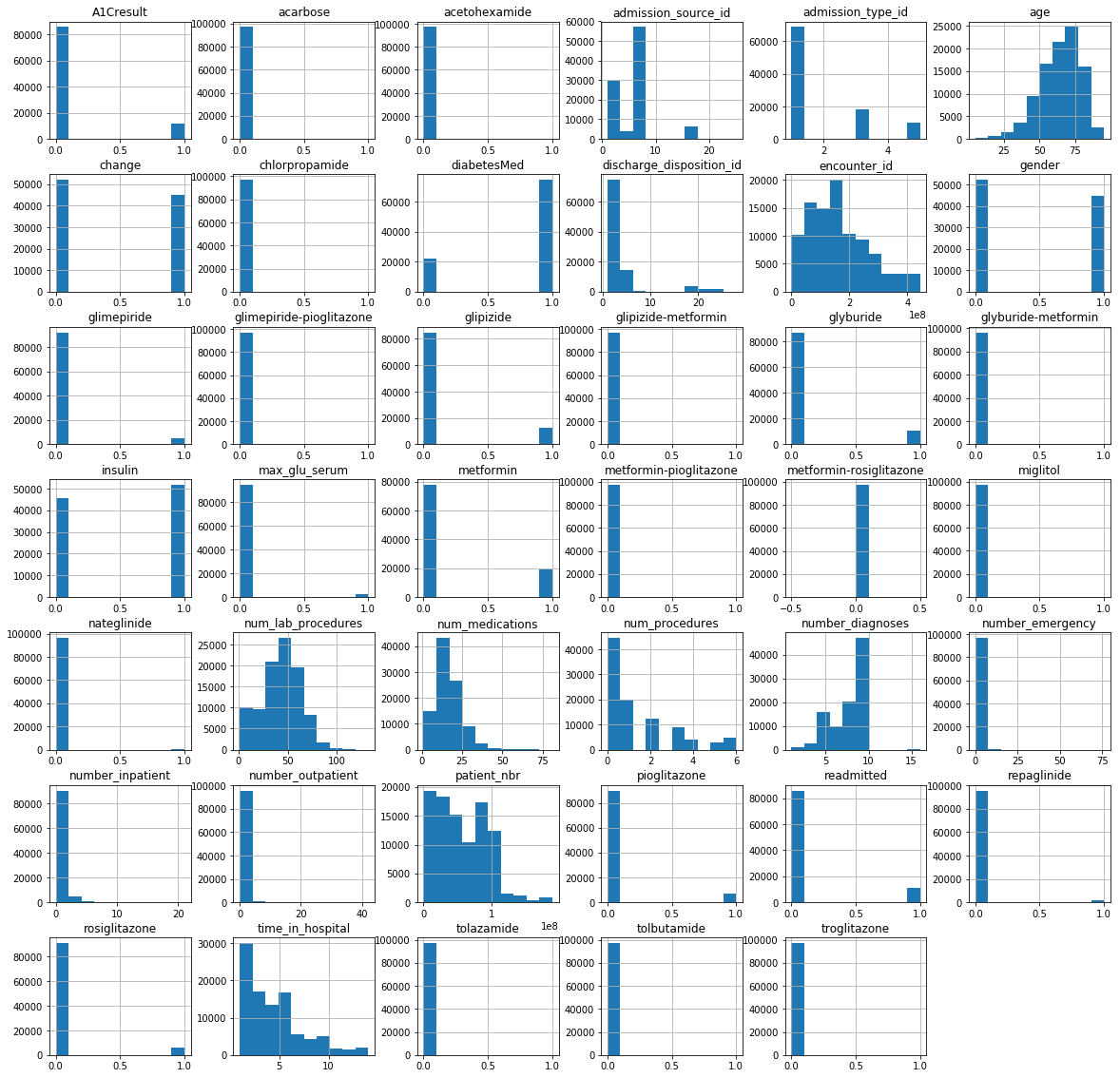
ax = fig.gca()

diabetic\_data\_temp.hist(ax = ax)

plt.show()

C:\Users\satwi\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3267: UserWarning: To output multiple subplots, the figure containing the passed axes is being cleared

exec(code\_obj, self.user\_global\_ns, self.user\_ns)

****

#Applying log transformation on

#number\_emergency

#num\_medications

#number\_inpatient

#number\_outpatient

#applying log on continues and highly skewed and with high magnitude variables

diabetic\_data['log\_number\_emergency']=np.log1p(diabetic\_data\_temp['number\_emergency'])

diabetic\_data['log\_num\_medications']=np.log1p(diabetic\_data\_temp['num\_medications'])

diabetic\_data['log\_number\_inpatient']=np.log1p(diabetic\_data\_temp['number\_inpatient'])

diabetic\_data['log\_number\_outpatient']=np.log1p(diabetic\_data\_temp['number\_outpatient'])

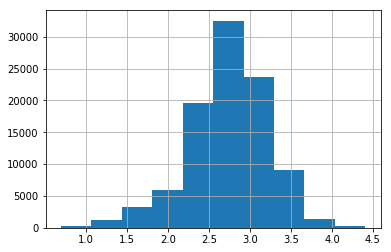
#Dropping the coloum after changing them into log

diabetic\_data=diabetic\_data.drop(['number\_emergency','num\_medications','number\_inpatient','number\_outpatient'],axis=1)

#Checking transformed coloum

diabetic\_data['log\_num\_medications'].hist()

<matplotlib.axes.\_subplots.AxesSubplot at 0x208a719f1d0>

****

#Standardizing the numerical data as we used the log transforamtion

def standardize(raw\_data):

return ((raw\_data - np.mean(raw\_data, axis = 0)) / np.std(raw\_data, axis = 0))

#diabetic\_data.info()=='int64'

(diabetic\_data.select\_dtypes(include=['int64'])).columns

Index(['encounter\_id', 'patient\_nbr', 'gender', 'age', 'admission\_type\_id',

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

'num\_lab\_procedures', 'num\_procedures', 'number\_diagnoses',

'max\_glu\_serum', 'A1Cresult', 'metformin', 'repaglinide', 'nateglinide',

'chlorpropamide', 'glimepiride', 'acetohexamide', 'glipizide',

'glyburide', 'tolbutamide', 'pioglitazone', 'rosiglitazone', 'acarbose',

'miglitol', 'troglitazone', 'tolazamide', 'insulin',

'glyburide-metformin', 'glipizide-metformin',

'glimepiride-pioglitazone', 'metformin-rosiglitazone',

'metformin-pioglitazone', 'change', 'diabetesMed', 'readmitted'],

dtype='object')

int64=['time\_in\_hospital',

'num\_lab\_procedures', 'num\_procedures', 'number\_diagnoses']

admission\_type\_id', 'discharge\_disposition\_id', 'admission\_source\_id',

diabetic\_data[int64] = standardize(diabetic\_data[int64])

#Number of records from from the orginal dataset

Orginal\_Shape-diabetic\_data.shape[0]

4658

for i in diabetic\_data.columns: if diabetic\_data[i].dtypes == 'object': diabetic\_data[i]=diabetic\_data[i].astype("category").cat.codes

for i in diabetic\_data.columns:

if(len(diabetic\_data[i].unique()) <= 1):

print(print(i,diabetic\_data[i].unique()[:10],'length=',len(diabetic\_data[i].unique())))

print('===============================================================================')

metformin-rosiglitazone [0] length= 1

None

===============================================================================

#Dropping the unique value variable in the data

diabetic\_data = diabetic\_data.drop(['metformin-rosiglitazone','glimepiride-pioglitazone'], axis = 1)

#From the research paper we are dropping the duplicate records from the data other than there first visit

diabetic\_data = diabetic\_data.drop\_duplicates(subset= ['patient\_nbr'], keep = 'first')

Orginal\_Shape-diabetic\_data.shape[0]

33600

#Making the categories for assigned medical segments

diabetic\_data['diag\_1']=diabetic\_data.diag\_1.astype("category").cat.codes

diabetic\_data['diag\_2']=diabetic\_data.diag\_2.astype("category").cat.codes

diabetic\_data['diag\_3']=diabetic\_data.diag\_3.astype("category").cat.codes

diabetic\_data\_model = diabetic\_data.drop(['encounter\_id','patient\_nbr','race'], axis = 1)

diabetic\_data\_model.head()

|  | **gender** | **age** | **admission\_type\_id** | **discharge\_disposition\_id** | **admission\_source\_id** | **time\_in\_hospital** | **num\_lab\_procedures** | **num\_procedures** | **diag\_1** | **diag\_2** | **...** | **glyburide-metformin** | **glipizide-metformin** | **metformin-pioglitazone** | **change** | **diabetesMed** | **readmitted** | **log\_number\_emergency** | **log\_num\_medications** | **log\_number\_inpatient** | **log\_number\_outpatient** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0 | 5 | 5 | 25 | 1 | -1.138528 | -0.095884 | -0.785073 | 4 | 18 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0.0 | 0.693147 | 0.000000 | 0.000000 |
| **1** | 0 | 15 | 1 | 1 | 7 | -0.465107 | 0.821092 | -0.785073 | 4 | 4 | ... | 0 | 0 | 0 | 1 | 1 | 0 | 0.0 | 2.944439 | 0.000000 | 0.000000 |
| **2** | 0 | 25 | 1 | 1 | 7 | -0.801818 | -1.624178 | 2.154736 | 13 | 4 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0.0 | 2.639057 | 0.693147 | 1.098612 |
| **3** | 1 | 35 | 1 | 1 | 7 | -0.801818 | 0.056945 | -0.197111 | 7 | 4 | ... | 0 | 0 | 0 | 1 | 1 | 0 | 0.0 | 2.833213 | 0.000000 | 0.000000 |
| **4** | 1 | 45 | 1 | 1 | 7 | -1.138528 | 0.413547 | -0.785073 | 11 | 11 | ... | 0 | 0 | 0 | 1 | 1 | 0 | 0.0 | 2.197225 | 0.000000 | 0.000000 |

5 rows × 40 columns

Exporting Data into csv for modeling

diabetic\_data\_model.to\_csv("final\_Modelused.csv",index=False )

**Data Modeling**

**Importing Requried Libraries for ML Modeling**

import numpy as np

import pandas as pd

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

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

from sklearn import tree

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.metrics import classification\_report

from imblearn.over\_sampling import SMOTE

from collections import Counter

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import AdaBoostClassifier

from sklearn.ensemble import AdaBoostClassifier

from sklearn.model\_selection import GridSearchCV

from sklearn.model\_selection import cross\_val\_score

diabetic\_data\_model= pd.read\_csv("final\_Modelused.csv")

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

X = diabetic\_data\_model.drop(['readmitted'], axis = 1)

Y = diabetic\_data\_model['readmitted']

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

random\_state = 7, stratify = Y)

#Checking the shape of the Data

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

shape of Xtrain,Xtest: (54532, 39) (13634, 39)

**Logistic Regression**

# create model logistic as logistic regression using Sklearn

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

logisticreg.fit(Xtrain, Ytrain)

Ylog = logisticreg.predict(Xtest)

C:\Users\satwi\Anaconda3\lib\site-packages\sklearn\linear\_model\logistic.py:433: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.

FutureWarning)

# 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: 0.9096376705295585

YScre = logisticreg.decision\_function(Xtest)

#Checking the confusion matrix

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

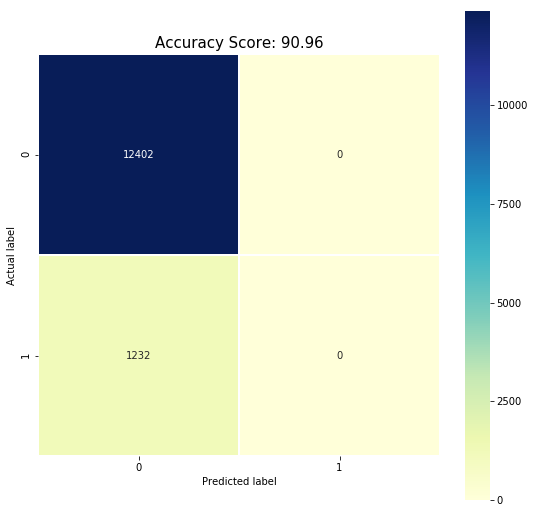
sns.heatmap(confusion\_matrix(Ytest, Ylog), annot=True, fmt="d", linewidths=.5, square = True, cmap = 'YlGnBu');

plt.ylabel('Actual label');

plt.xlabel('Predicted label');

all\_sample\_title = 'Accuracy Score: {0}'.format(round(logisticreg.score(Xtest, Ytest)\*100,2))

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



# Checking the summary of classification

print(classification\_report(Ytest, Ylog))

precision recall f1-score support

0 0.91 1.00 0.95 12402

1 0.00 0.00 0.00 1232

micro avg 0.91 0.91 0.91 13634

macro avg 0.45 0.50 0.48 13634

weighted avg 0.83 0.91 0.87 13634

C:\Users\satwi\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.

'precision', 'predicted', average, warn\_for)

C:\Users\satwi\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.

'precision', 'predicted', average, warn\_for)

C:\Users\satwi\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.

'precision', 'predicted', average, warn\_for)

**This translates to a no proportion of readmitted people predicted may be this is due to data inbalance**

**Checking the Lables**

diabetic\_data\_model.groupby('readmitted').size()

readmitted

0 62007

1 6159

dtype: int64

**Data was highly imbalanced with respect to readmissions (only 10% records for 30-day readmissions), leading to high accuracy. Moreover, the high accuracy could be attributed not to the generalizability of our model to diverse patient records but to the baseline accuracy of 90%: predicting that no patient would be readmitted.**

**Using SMOTE For balancing the data and spliting the data into 80 and 20 ratio**

print('Original dataset shape {}'.format(Counter(Y)))

sm = SMOTE(random\_state=20)

diabetes\_input\_new, diabetes\_output\_new = sm.fit\_sample(X, Y)

print('New dataset shape {}'.format(Counter(diabetes\_output\_new)))

diabetes\_input\_new = pd.DataFrame(diabetes\_input\_new, columns = list(X.columns))

X\_train, X\_dev, Y\_train, Y\_dev = train\_test\_split(diabetes\_input\_new, diabetes\_output\_new, test\_size=0.20, random\_state=0)

Original dataset shape Counter({0: 62007, 1: 6159})

New dataset shape Counter({0: 62007, 1: 62007})

**Logistic Regression After SMOTE**

# create model logistic as logistic regression using Sklearn

from sklearn.linear\_model import LogisticRegression

logisticreg = LogisticRegression(tol=1e-7, penalty='l2', C=0.0005,solver='lbfgs',random\_state=0,max\_iter =300)

logisticreg.fit(X\_train, Y\_train)

Ylog\_SMOTE = logisticreg.predict(X\_dev)

print(" The accuracy of the Logistic regression model:" ,round(logisticreg.score(X\_train, Y\_train)\*100,2))

The accuracy of the Logistic regression model: 59.09

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

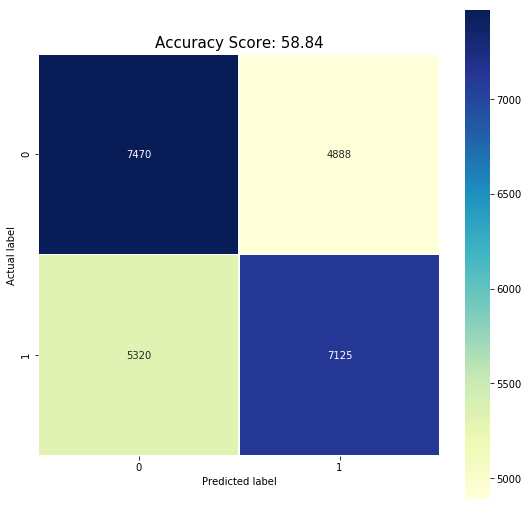
sns.heatmap(confusion\_matrix(Y\_dev, Ylog\_SMOTE), annot=True, fmt="d", linewidths=.5, square = True, cmap = 'YlGnBu');

plt.ylabel('Actual label');

plt.xlabel('Predicted label');

all\_sample\_title = 'Accuracy Score: {0}'.format(round(logisticreg.score(X\_dev, Y\_dev)\*100,2))

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



YScre\_aftersmote = logisticreg.decision\_function(X\_dev)

# Checking the summary of classification

print(classification\_report(Y\_dev, Ylog\_SMOTE))

precision recall f1-score support

0 0.58 0.60 0.59 12358

1 0.59 0.57 0.58 12445

micro avg 0.59 0.59 0.59 24803

macro avg 0.59 0.59 0.59 24803

weighted avg 0.59 0.59 0.59 24803

**Random Forest Classifier**

random\_forest = RandomForestClassifier(random\_state=30,n\_estimators=300)

random\_forest.fit(X\_train, Y\_train)

randforest = random\_forest.predict(X\_dev)

#Checking the accuracy

print(" Accuracy of Randomeforest: ", random\_forest.score(X\_dev, Y\_dev))

Accuracy of Randomeforest: 0.9505301778010724

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

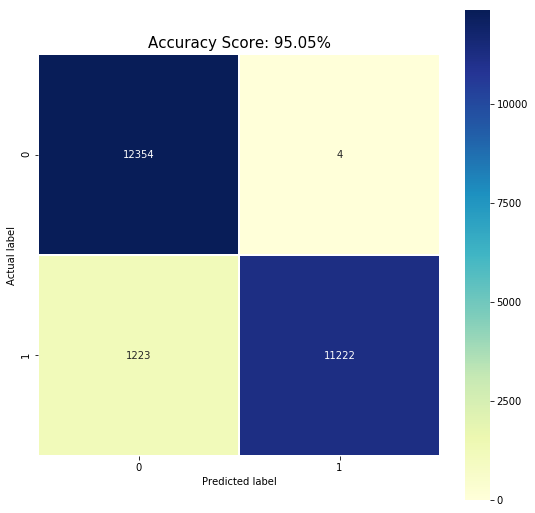
sns.heatmap(confusion\_matrix(Y\_dev, randforest), annot=True, fmt="d", linewidths=.5, square = True, cmap = 'YlGnBu');

plt.ylabel('Actual label');

plt.xlabel('Predicted label');

all\_sample\_title = 'Accuracy Score: {0}'.format(round(random\_forest.score(X\_dev, Y\_dev)\*100,2))+str('%')

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



print(classification\_report(Y\_dev, randforest))

precision recall f1-score support

0 0.91 1.00 0.95 12358

1 1.00 0.90 0.95 12445

micro avg 0.95 0.95 0.95 24803

macro avg 0.95 0.95 0.95 24803

weighted avg 0.95 0.95 0.95 24803

#Determining which features are most important from random forest

feature\_names = X\_train.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))

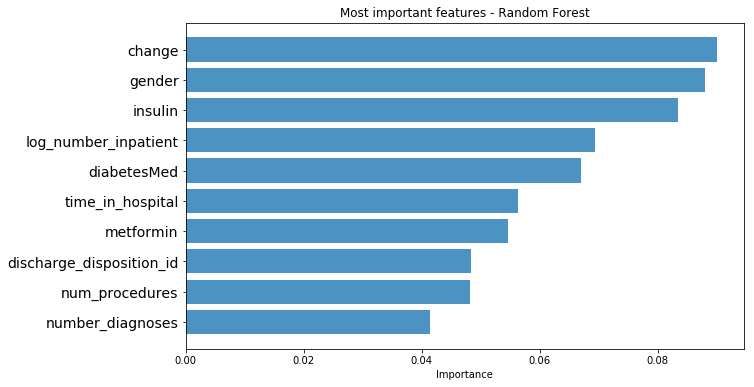
plt.barh(range(len(most\_imp\_features)), most\_imp\_features.Importance, align='center', alpha=0.8)

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**

#Creating a AdaBoosted Classification model,

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

adaClassification.fit(X\_train, Y\_train)

adaClassificationPred = adaClassification.predict(X\_dev)

print("The accurary of AdaBoosted Classification model: ", adaClassification.score(X\_dev, Y\_dev))

The accurary of AdaBoosted Classification model: 0.8360278998508245

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

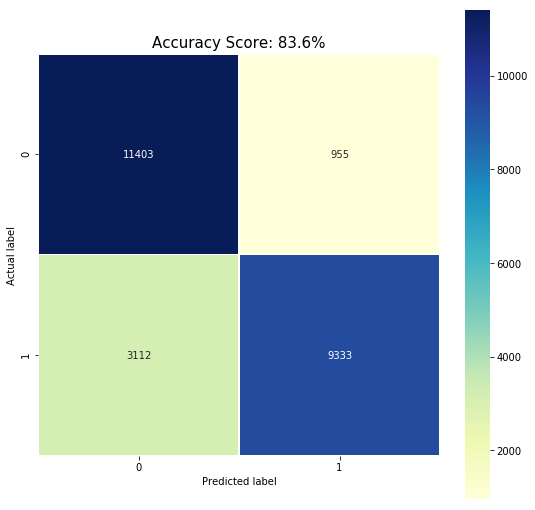
sns.heatmap(confusion\_matrix(Y\_dev, adaClassificationPred), annot=True, fmt="d", linewidths=.5, square = True, cmap = 'YlGnBu');

plt.ylabel('Actual label');

plt.xlabel('Predicted label');

all\_sample\_title = 'Accuracy Score: {0}'.format(round(adaClassification.score(X\_dev, Y\_dev)\*100,2))+str('%')

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



#Checking the precision and recall of ada boost ClassificationPred

print(classification\_report(Y\_dev, adaClassificationPred))

precision recall f1-score support

0 0.79 0.92 0.85 12358

1 0.91 0.75 0.82 12445

micro avg 0.84 0.84 0.84 24803

macro avg 0.85 0.84 0.83 24803

weighted avg 0.85 0.84 0.83 24803

# calculate the FP and TP from the roc\_curve metric

from sklearn.metrics import roc\_curve

yadaclas = adaClassification.decision\_function(X\_dev)

fpr\_adaclass, tpr\_adaclass, thresholds = roc\_curve(Y\_dev, adaClassificationPred)

**Hyperparameters Tunning for AdaBoosted**

#Trying to tune the model

ada\_boost = AdaBoostClassifier(n\_estimators = 10, learning\_rate = 0.2, random\_state = 30)

boostprameters ={

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

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

},

adaboost = GridSearchCV(ada\_boost, cv=2, n\_jobs=2, param\_grid=boostprameters)

%%time

adaboost.fit(X\_train,Y\_train)

Wall time: 10min 57s

GridSearchCV(cv=2, error\_score='raise-deprecating',

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

learning\_rate=0.2, n\_estimators=10, random\_state=30),

fit\_params=None, iid='warn', n\_jobs=2,

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)

adapred = adaboost.predict(X\_dev)

adaboost.score(X\_dev, Y\_dev)

0.9464984074507116

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

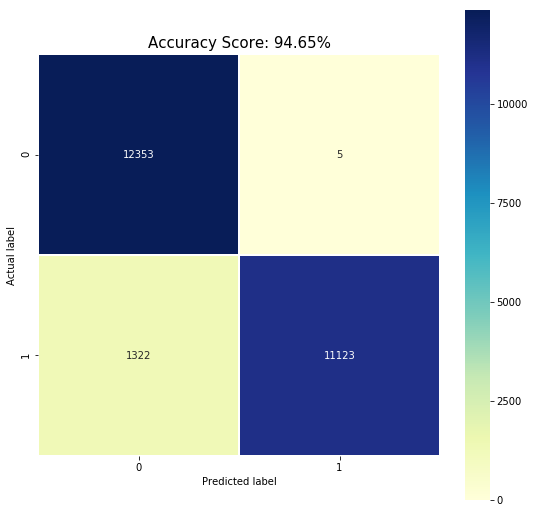
sns.heatmap(confusion\_matrix(Y\_dev, adapred), annot=True, fmt="d", linewidths=.5, square = True, cmap = 'YlGnBu');

all\_sample\_title = 'Accuracy Score: {0}'.format(round(adaboost.score(X\_dev, Y\_dev)\*100,2))+str('%')

plt.ylabel('Actual label');

plt.xlabel('Predicted label');

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



adaboostpred = adaboost.predict(X\_dev)

print(classification\_report(Y\_dev, adaboostpred))

precision recall f1-score support

0 0.90 1.00 0.95 12358

1 1.00 0.89 0.94 12445

micro avg 0.95 0.95 0.95 24803

macro avg 0.95 0.95 0.95 24803

weighted avg 0.95 0.95 0.95 24803

ygridadascore = adaboost.decision\_function(X\_dev)

**Decision Tree**

DecisionTree = DecisionTreeClassifier(max\_depth=28, criterion = "entropy", min\_samples\_split=10)

print("Cross Validation score: {:.2%}".format(np.mean(cross\_val\_score(DecisionTree, X\_train, Y\_train, cv=10))))

DecisionTree.fit(X\_train, Y\_train)

print("Dev Set score: {:.2%}".format(DecisionTree.score(X\_dev, Y\_dev)))

Cross Validation score: 90.80%

Dev Set score: 91.04%

ypreddt = DecisionTree.predict(X\_dev)

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

Accuracy is 91.04140628149821

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

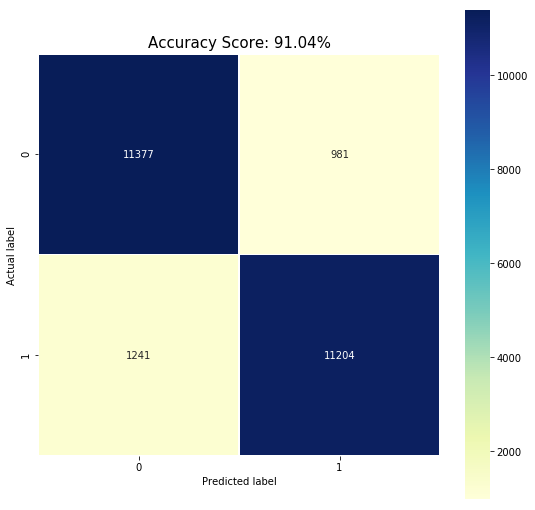
sns.heatmap(confusion\_matrix(Y\_dev, ypreddt), annot=True, fmt="d", linewidths=.5, square = True, cmap = 'YlGnBu');

plt.ylabel('Actual label');

plt.xlabel('Predicted label');

all\_sample\_title = 'Accuracy Score: {0}'.format(round(accuracy\_score(Y\_dev,ypreddt)\*100,2))+str('%')

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



print(classification\_report(Y\_dev, ypreddt))

precision recall f1-score support

0 0.90 0.92 0.91 12358

1 0.92 0.90 0.91 12445

micro avg 0.91 0.91 0.91 24803

macro avg 0.91 0.91 0.91 24803

weighted avg 0.91 0.91 0.91 24803

**Comparing the models**

**Comparing the models using roc curve**

#Saving the probabilites

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

fpr\_log, tpr\_log, thresholds = roc\_curve(Y\_dev, YScre\_aftersmote)#logistic after smote regression

fpr\_rf, tpr\_rf, thresholds = roc\_curve(Y\_dev, randforest)#random forest classifier

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

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

fpr\_dt, tpr\_dt, thresholds = roc\_curve(Y\_dev,ypreddt )

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

plt.plot(fpr\_log\_beforesmote, tpr\_log\_beforesmote, label='Logistic regression before smote')

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 hyper parameters')

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')

plt.title('ROC')

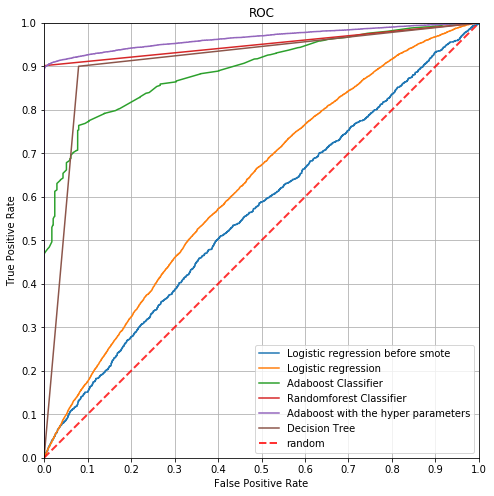
plt.legend(loc="lower right")

plt.show()

C:\Users\satwi\Anaconda3\lib\site-packages\matplotlib\figure.py:98: 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.

"Adding an axes using the same arguments as a previous axes "



**Our objective is to predict readmission with in 30 days (TRUE POSITIVE),**

**Adaclassifier with hyperparameters & randomforest give us good sensitivity (POSITIVE)**

**Comparing Model Accuracy**

print("Accuracy of Logistic regression model before smote:" ,round((logisticreg.score(Xtest, Ytest)\*100),2))

print('-------After SMOTE-----')

print("Accuracy of Logistic regression model:" ,round((logisticreg.score(X\_dev, Y\_dev)\*100),2))

print("Accuracy of Random forest classification: ", round((random\_forest.score(X\_dev, Y\_dev)\*100),2))

print("Accuracy of AdaBoosted Classification model: ", round((adaClassification.score(X\_dev, Y\_dev)\*100),2))

print("Accuracy of Hyperparameter Tuning AdaBoosted Classification model: ", round((adaboost.score(X\_dev, Y\_dev)\*100),2))

print( "Accuracy of Decision Tree ", round((accuracy\_score(Y\_dev,ypreddt)\*100),2))

Accuracy of Logistic regression model before smote: 59.18

-------After SMOTE-----

Accuracy of Logistic regression model: 58.55

Accuracy of Random forest classification: 95.05

Accuracy of AdaBoosted Classification model: 83.6

Accuracy of Hyperparameter Tuning AdaBoosted Classification model: 94.65

Accuracy of Decision Tree 91.04