Describe problem

Since the living standard is getting higher and higher after entering into the 21st century, several problems occur along with unhealthy eating habits. According to the statistics, there will be overall 34.2 million people more having diabetes by 2020, 10.5% of which are the US population. As the number of people getting diabetes rises, the pressure of testing diabetes also increases. There turned out to be some negative facts when a patient takes diabetes tests in the hospital.

Analysis task

First, based on the list, patients are required to complete more than 10 items in order to get the test result, which is too time-consuming. Second, when the number of people who get diabete is increasing (National Report), the number of equipment becomes inefficient. Hospitals have to cost more on buying new equipment meanwhile maintaining the old machines. Third, over-listed testing requirements ask for a high medical expense, but some of them are actually not necessarily needed. Therefore, by doing this project, we wish to solve the problem of how hospitals can offer patients the most efficient way of diabetes testing.

To achieve this purpose, we will build a model that only tests the necessary features to get the correct medical result. If we could solve this problem, the patients' waiting and examining hours will be shortened; the equipment will be efficient, and hence reduce the costs of hospitals; moreover, by shrinking into the specific features, the testing expenses for the patients will also be reduced.

Data process

In [1]:
```
# This is a comment section, which will not be executed as part of the p
ython script
# matplotlib inline makes your matplotlib plot embedded in the jupyter n
otebook cell
%matplotlib inline

# Here "as np" is aliasing the library name to a short abbrev
import numpy as np

import pandas as pd
import seaborn as sns

# "from ... import ..." allows us to import specific submodule from the
 library
from matplotlib import pyplot as plt

### This is to set pandas to display all rows and columns
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

In [2]:
```
df = pd.read_csv('diabetic_data.csv')
df.head()
```

Out[2]:

	encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discharge_
0	2278392	8222157	Caucasian	Female	[0-10)	?	6	
1	149190	55629189	Caucasian	Female	[10-20)	?	1	
2	64410	86047875	AfricanAmerican	Female	[20-30)	?	1	
3	500364	82442376	Caucasian	Male	[30-40)	?	1	
4	16680	42519267	Caucasian	Male	[40-50)	?	1	

Here we check the top 5 results of the dataframe and find that some data has missing value as '?', so we want to replace the value as NaN.

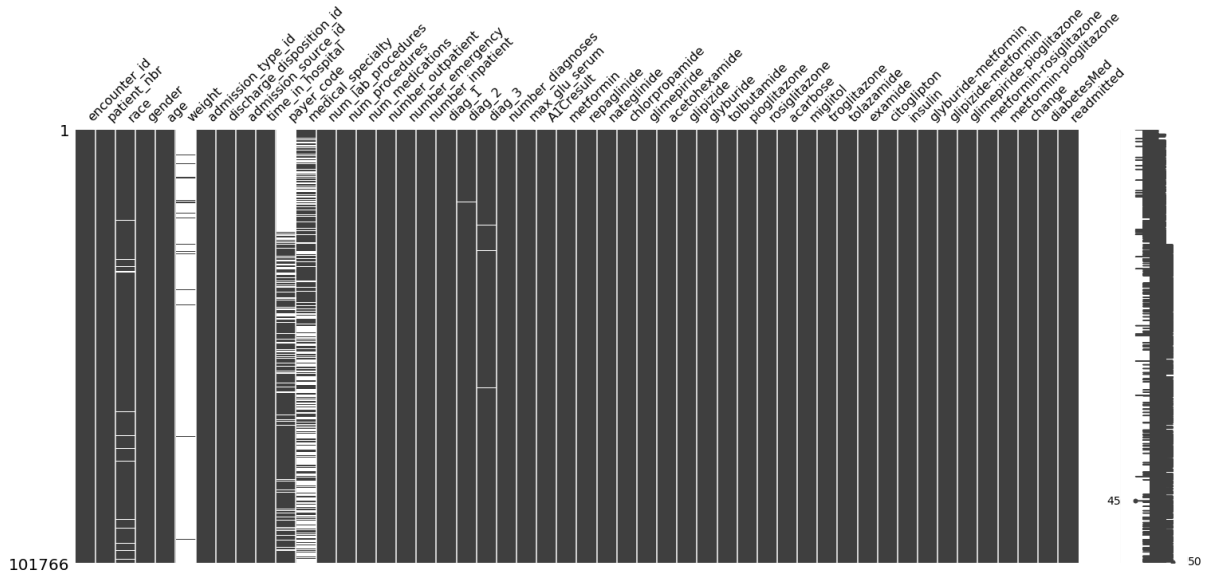
In [3]:
```
df = df.replace('?',np.nan)
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101766 entries, 0 to 101765
Data columns (total 50 columns):
 #   Column                    Non-Null Count   Dtype 
---  ------                    --------------   ----- 
 0   encounter_id              101766 non-null  int64 
 1   patient_nbr               101766 non-null  int64 
 2   race                      99493 non-null   object
 3   gender                    101766 non-null  object
 4   age                       101766 non-null  object
 5   weight                    3197 non-null    object
 6   admission_type_id         101766 non-null  int64 
 7   discharge_disposition_id  101766 non-null  int64 
 8   admission_source_id       101766 non-null  int64 
 9   time_in_hospital          101766 non-null  int64 
 10  payer_code                61510 non-null   object
 11  medical_specialty         51817 non-null   object
 12  num_lab_procedures        101766 non-null  int64 
 13  num_procedures            101766 non-null  int64 
 14  num_medications           101766 non-null  int64 
 15  number_outpatient         101766 non-null  int64 
 16  number_emergency          101766 non-null  int64 
 17  number_inpatient          101766 non-null  int64 
 18  diag_1                    101745 non-null  object
 19  diag_2                    101408 non-null  object
 20  diag_3                    100343 non-null  object
 21  number_diagnoses          101766 non-null  int64 
 22  max_glu_serum             101766 non-null  object
 23  A1Cresult                 101766 non-null  object
 24  metformin                 101766 non-null  object
 25  repaglinide               101766 non-null  object
 26  nateglinide               101766 non-null  object
 27  chlorpropamide            101766 non-null  object
 28  glimepiride               101766 non-null  object
 29  acetohexamide             101766 non-null  object
 30  glipizide                 101766 non-null  object
 31  glyburide                 101766 non-null  object
 32  tolbutamide               101766 non-null  object
 33  pioglitazone              101766 non-null  object
 34  rosiglitazone             101766 non-null  object
 35  acarbose                  101766 non-null  object
 36  miglitol                  101766 non-null  object
 37  troglitazone              101766 non-null  object
 38  tolazamide                101766 non-null  object
 39  examide                   101766 non-null  object
 40  citoglipton               101766 non-null  object
 41  insulin                   101766 non-null  object
 42  glyburide-metformin       101766 non-null  object
 43  glipizide-metformin       101766 non-null  object
 44  glimepiride-pioglitazone  101766 non-null  object
 45  metformin-rosiglitazone   101766 non-null  object
 46  metformin-pioglitazone    101766 non-null  object
 47  change                    101766 non-null  object
 48  diabetesMed               101766 non-null  object
 49  readmitted                101766 non-null  object
dtypes: int64(13), object(37)
memory usage: 38.8+ MB
```

In [5]:
```
# this python magics will allow plot to be embedded into the notebook
import matplotlib
import matplotlib.pyplot as plt
import warnings
warnings.simplefilter('ignore', DeprecationWarning)
%matplotlib inline

import missingno as mn

mn.matrix(df)
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7f48d25e14d0>



This image shows that the white block is the missing value, we try to remove the rows with empty data, resulting in the smaller number of data. So we decide to remove the three columns, such like weight, payer_code and medicanl_speciality.

In [6]:
```
all_cols = list(df.columns)
```

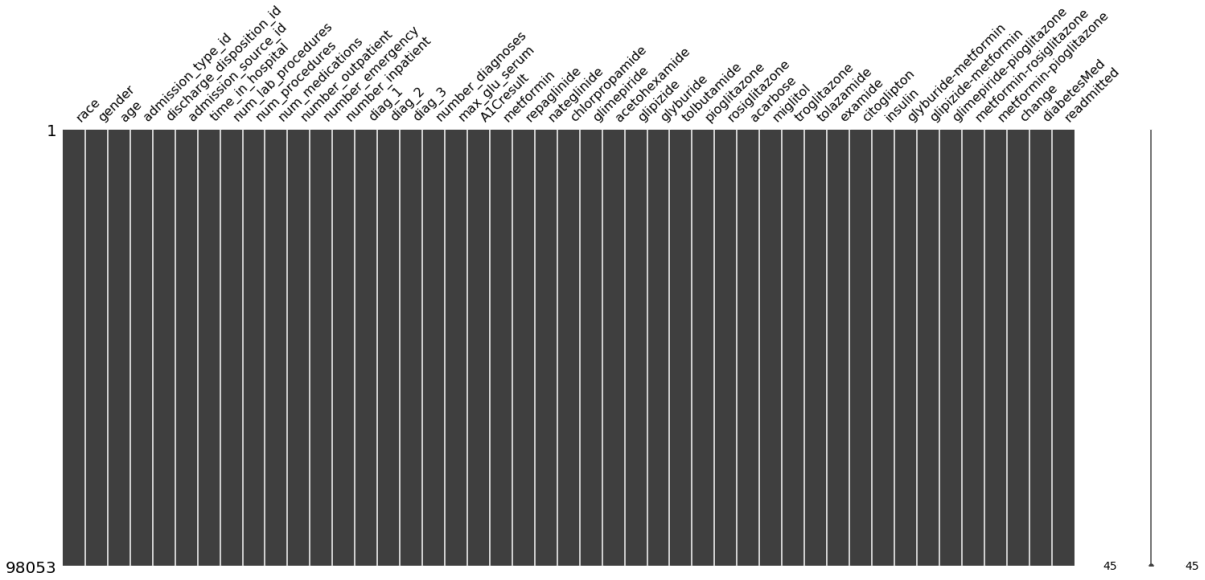
from sklearn.preprocessing import LabelEncoder #remove_col = [] le = LabelEncoder() for col in all_cols: y = le.fit_transform(df[col]) df[col] = y print(col,set(y))

In [7]:
```
df = df.drop(['encounter_id','patient_nbr',"weight","payer_code","medical_specialty"], axis = 1)
```

In [8]:
```
df.dropna(inplace = True)
```

In [9]:
```
# this python magics will allow plot to be embedded into the notebook
import matplotlib
import matplotlib.pyplot as plt
import warnings
warnings.simplefilter('ignore', DeprecationWarning)
%matplotlib inline

import missingno as mn

mn.matrix(df)
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7f48d213b350>



This graph shows that the result contains no missing value.

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 98053 entries, 1 to 101765
Data columns (total 45 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   race                      98053 non-null  object
 1   gender                    98053 non-null  object
 2   age                       98053 non-null  object
 3   admission_type_id         98053 non-null  int64 
 4   discharge_disposition_id  98053 non-null  int64 
 5   admission_source_id       98053 non-null  int64 
 6   time_in_hospital          98053 non-null  int64 
 7   num_lab_procedures        98053 non-null  int64 
 8   num_procedures            98053 non-null  int64 
 9   num_medications           98053 non-null  int64 
 10  number_outpatient         98053 non-null  int64 
 11  number_emergency          98053 non-null  int64 
 12  number_inpatient          98053 non-null  int64 
 13  diag_1                    98053 non-null  object
 14  diag_2                    98053 non-null  object
 15  diag_3                    98053 non-null  object
 16  number_diagnoses          98053 non-null  int64 
 17  max_glu_serum             98053 non-null  object
 18  A1Cresult                 98053 non-null  object
 19  metformin                 98053 non-null  object
 20  repaglinide               98053 non-null  object
 21  nateglinide               98053 non-null  object
 22  chlorpropamide            98053 non-null  object
 23  glimepiride               98053 non-null  object
 24  acetohexamide             98053 non-null  object
 25  glipizide                 98053 non-null  object
 26  glyburide                 98053 non-null  object
 27  tolbutamide               98053 non-null  object
 28  pioglitazone              98053 non-null  object
 29  rosiglitazone             98053 non-null  object
 30  acarbose                  98053 non-null  object
 31  miglitol                  98053 non-null  object
 32  troglitazone              98053 non-null  object
 33  tolazamide                98053 non-null  object
 34  examide                   98053 non-null  object
 35  citoglipton               98053 non-null  object
 36  insulin                   98053 non-null  object
 37  glyburide-metformin       98053 non-null  object
 38  glipizide-metformin       98053 non-null  object
 39  glimepiride-pioglitazone  98053 non-null  object
 40  metformin-rosiglitazone   98053 non-null  object
 41  metformin-pioglitazone    98053 non-null  object
 42  change                    98053 non-null  object
 43  diabetesMed               98053 non-null  object
 44  readmitted                98053 non-null  object
dtypes: int64(11), object(34)
memory usage: 34.4+ MB
```

Because some features are not int type, if we remove these features, the rest of data may not result into the good model. Hence, we decide to transform the string data into int data.

description of feature

race: Caucasian, Asian, African American, Hispanic, and other

Gender: male, female and unknown/invalid

Age: [0,10), [10,20), ... [90,100)

Admission type : 9 values

Discharge disposition: 29 values

Admission source: 21 values

time in hospital: remove

payer code: 23 values

medical specialty:

glucose serum test result: range of result

A1c test result: '>7' = greater than 7%

Label: change of medications

In [11]:
```
from sklearn.preprocessing import LabelEncoder
remove_col = []
all_cols = list(df.columns)
le = LabelEncoder()
for col in all_cols:
    if(df[col].dtypes != 'int64'):
        s = set(df[col])
        y = le.fit_transform(df[col])
        if len(set(y)) == 1:
            remove_col.append(col)
        df[col] = y
    print(col,set(y), s)
```

```
race {0, 1, 2, 3, 4} {'Asian', 'AfricanAmerican', 'Caucasian', 'Hispani
c', 'Other'}
gender {0, 1, 2} {'Female', 'Unknown/Invalid', 'Male'}
age {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} {'[30-40)', '[90-100)', '[20-30)',
'[60-70)', '[10-20)', '[40-50)', '[50-60)', '[70-80)', '[0-10)', '[80-9
0)'}
admission_type_id {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} {'[30-40)', '[90-10
0)', '[20-30)', '[60-70)', '[10-20)', '[40-50)', '[50-60)', '[70-80)',
'[0-10)', '[80-90)'}
discharge_disposition_id {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} {'[30-40)', '[9
0-100)', '[20-30)', '[60-70)', '[10-20)', '[40-50)', '[50-60)', '[70-8
0)', '[0-10)', '[80-90)'}
admission_source_id {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} {'[30-40)', '[90-10
0)', '[20-30)', '[60-70)', '[10-20)', '[40-50)', '[50-60)', '[70-80)',
'[0-10)', '[80-90)'}
time_in_hospital {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} {'[30-40)', '[90-100)',
'[20-30)', '[60-70)', '[10-20)', '[40-50)', '[50-60)', '[70-80)', '[0-1
0)', '[80-90)'}
num_lab_procedures {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} {'[30-40)', '[90-10
0)', '[20-30)', '[60-70)', '[10-20)', '[40-50)', '[50-60)', '[70-80)',
'[0-10)', '[80-90)'}
num_procedures {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} {'[30-40)', '[90-100)',
'[20-30)', '[60-70)', '[10-20)', '[40-50)', '[50-60)', '[70-80)', '[0-1
0)', '[80-90)'}
num_medications {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} {'[30-40)', '[90-100)',
'[20-30)', '[60-70)', '[10-20)', '[40-50)', '[50-60)', '[70-80)', '[0-1
0)', '[80-90)'}
number_outpatient {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} {'[30-40)', '[90-10
0)', '[20-30)', '[60-70)', '[10-20)', '[40-50)', '[50-60)', '[70-80)',
'[0-10)', '[80-90)'}
number_emergency {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} {'[30-40)', '[90-100)',
'[20-30)', '[60-70)', '[10-20)', '[40-50)', '[50-60)', '[70-80)', '[0-1
0)', '[80-90)'}
number_inpatient {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} {'[30-40)', '[90-100)',
'[20-30)', '[60-70)', '[10-20)', '[40-50)', '[50-60)', '[70-80)', '[0-1
0)', '[80-90)'}
diag_1 {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1
8, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71,
72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 1
06, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 12
0, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 13
4, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 14
8, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 16
2, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 17
6, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 19
0, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 20
4, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 21
8, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 23
2, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 24
6, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 26
0, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 27
4, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 28
8, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 30
2, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 31
```

```
6, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 33
0, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 34
4, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 35
8, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 37
2, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 38
6, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 40
0, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 41
4, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 42
8, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 44
2, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 45
6, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 47
0, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 48
4, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 49
8, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 51
2, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 52
6, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 54
0, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 55
4, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 56
8, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 58
2, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 59
6, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 61
0, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 62
4, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 63
8, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 65
2, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 66
6, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 68
0, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 69
4, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 70
8, 709, 710, 711, 712} {'34', '827', '404', '814', '883', '312', '895',
'973', '61', '171', '7', '443', '533', '893', '924', '333', '340', '89
7', '685', '891', '365', '664', '971', '174', '219', '39', '152', '29
6', '817', '318', '711', '535', '151', '654', '801', 'E909', '511', '39
1', '806', '425', '655', '942', '519', '501', '250.9', '682', '923', '2
50.12', '574', '47', '322', '474', '663', 'V53', '780', '785', '366',
'432', '865', '570', '217', '250.92', '250.3', '301', '293', '305', '24
2', '434', '374', '163', '199', '220', '153', '737', '866', '700', '60
5', '550', '131', '989', '619', '239', '916', '542', '719', '935', '55
8', '461', '854', '786', '939', '963', '523', '281', '782', '362', '23
0', '193', '440', '913', '873', '185', '974', '793', '135', '250.7', '8
16', '38', '578', '10', '516', '919', '824', 'V55', '250.51', '250.01',
'879', '475', '562', '331', '154', '892', '287', '208', '792', '567',
'280', '250.2', '834', '933', '692', '385', '569', '826', '566', '717',
'586', '237', '703', '825', '982', '361', '250.83', '698', 'V71', '79
7', 'V51', '904', '466', '571', '353', '959', '433', '694', '846', '84
7', 'V54', '683', '822', '843', '653', '521', '980', '860', '462', '80
7', '383', '914', '598', '477', '572', '308', '309', '661', '369', '43
1', '210', '524', '803', '464', '289', '332', '379', '187', '599', '24
6', '306', '323', '658', '815', '227', '162', '530', '955', '235', '58
2', '403', '581', '303', '575', '9', '348', '411', '515', '334', '709',
'402', '250.22', '941', '23', '310', '457', '161', '276', '188', '831',
'921', '986', '995', '486', '438', '999', '580', '236', '994', '311',
'250.81', '522', '623', '384', '640', '657', '716', '203', '680', '250.
41', '804', '632', '724', '695', '41', '94', '410', '192', '610', '88
1', '875', '560', '250.02', '471', '833', '861', '141', '596', '372',
'656', '252', '48', '422', '147', '429', '850', '250.91', '821', '455',
'756', '788', '229', '183', '745', '204', '992', '58', '864', '625', '8
36', '241', '781', '282', '732', '250.1', '157', '261', '441', '506',
```

```
'510', '288', '753', '600', '646', '634', '480', '148', '57', '534', '8
2', '299', '593', 'V07', '198', '216', '377', '250.6', '911', '250', '2
26', '357', '727', '922', '182', '205', '271', '421', '155', '114', '78
4', '945', '250.93', '347', '445', '715', '681', '738', '335', '591',
'996', '380', '250.53', '987', '934', '964', '565', 'V26', '79', '604',
'195', '607', '275', '851', '706', '344', '952', '529', '320', '540',
'170', '164', '465', '783', '350', '917', '253', '721', '250.42', '83
8', '52', '189', '417', '351', '3', '437', '736', '997', '194', '751',
'298', '906', '346', '250.43', '799', '880', '915', '977', '868', '75',
'146', '201', '207', '54', '358', '156', '590', '397', '184', '637', '1
75', '285', '747', '584', '250.8', '794', '473', '968', '787', '11', 'V
66', '791', '415', '490', '180', '723', '327', '454', '603', '307', '29
5', 'V45', '273', '427', '705', '292', '627', '592', '197', '735', '8',
'669', '970', '759', '649', '49', '179', '228', '324', '414', '614', '3
81', '965', 'V56', '944', '583', '446', '835', '250.33', '388', '690',
'133', '975', '920', '84', '844', '796', '98', '27', '789', '555', '51
8', '710', '870', '684', '972', '810', '579', '304', '720', '250.31',
'31', '297', '990', '376', '976', '728', '514', '823', '790', '648', 'V
57', '659', '853', '378', '78', '291', '398', '277', '444', '250.11',
'142', 'V25', '867', '966', '250.5', '191', '568', '485', '863', '373',
'746', '936', '962', '714', '622', '143', '172', 'V60', '837', '284',
'5', '693', 'V43', '149', '890', '112', '621', '341', '396', '611', '64
7', 'V67', '394', '882', '115', '405', '240', '426', '349', '601', '66
0', '251', '729', '250.03', '160', '435', '615', '430', '957', '840',
'363', '355', '448', '576', '730', '839', '665', '633', '354', '442',
'642', '969', '416', '117', '507', '279', '652', '886', '88', '725', '4
36', '481', '988', '813', '110', '928', '508', '300', '262', '541', '33
8', '643', '356', 'V63', '456', '196', '734', '451', 'V58', '551', '3
6', '726', '617', '173', '577', '413', '543', '238', '722', '871', '95
8', '337', '423', '250.4', '233', '585', '42', '325', '967', '852', '29
0', '250.82', '795', '536', '66', '272', '708', '842', '463', '266', '6
18', '214', '447', '696', '359', '424', '452', '812', '202', '453', '70
4', '707', '674', '845', '244', '556', '255', '336', '848', '626', '61
6', '983', '805', '491', '718', '645', '832', '250.21', '283', '53', '6
08', '382', '386', '644', '375', '412', '420', '487', '998', '294', '15
0', '537', '225', '250.23', '428', '286', '512', '527', '70', '263', '2
15', '370', '800', '352', '97', '250.32', '158', '820', '200', '494',
'686', '389', '808', '885', '602', '553', '500', '458', '552', '360',
'991', '470', '250.13', '731', '136', '314', '588', '478', '531', '21
8', '528', '274', '211', '526', '245', '278', '496', '564', '495', '39
5', '368', '594', '493', '691', '862', '573', '223', '212', '802', '59
5', '641', '492', '250.52', '532', '620', '345', '459', '342', '483',
'557', '513', '878', '733', '401', '482', '35', '145'}
diag_2 {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1
8, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71,
72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 1
06, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 12
0, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 13
4, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 14
8, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 16
2, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 17
6, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 19
0, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 20
4, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 21
```

```
8, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 23
2, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 24
6, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 26
0, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 27
4, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 28
8, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 30
2, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 31
6, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 33
0, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 34
4, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 35
8, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 37
2, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 38
6, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 40
0, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 41
4, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 42
8, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 44
2, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 45
6, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 47
0, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 48
4, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 49
8, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 51
2, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 52
6, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 54
0, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 55
4, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 56
8, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 58
2, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 59
6, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 61
0, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 62
4, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 63
8, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 65
2, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 66
6, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 68
0, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 69
4, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 70
8, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 72
2, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 73
6, 737, 738, 739} {'34', 'E918', '404', '814', '883', '312', '947', '46
0', '171', 'E819', '443', '533', '893', '7', '924', '333', '340', '68
5', '891', '365', '664', '174', '670', '152', '296', '318', '711', '53
5', '151', '654', '801', '511', '806', '425', '519', '250.9', '501', '9
42', '682', '923', '250.12', '574', '474', '322', '894', '663', '258',
'V53', '780', '517', '785', '366', '432', '865', '570', '217', 'E938',
'250.92', '250.3', '301', '242', '305', '293', '434', '374', '163', '19
9', '220', '153', 'V49', '737', 'E945', 'E924', '866', '550', '131', '9
89', 'E930', '619', '239', '542', '186', '916', 'E917', '719', '558',
'461', '786', '963', '702', '523', '281', '782', '362', '741', 'E915',
'V85', '99', '193', '440', '873', 'V08', '185', '974', 'E829', '913',
'793', '135', '250.7', 'E884', '816', '38', '578', '516', '919', '824',
'V55', '250.51', '250.01', '879', '811', '475', '302', '872', '562', '1
37', '331', '154', '892', '287', '208', '792', '567', 'E882', '280', '2
50.2', '933', '692', '569', '826', '566', '717', '586', 'E906', '713',
'E821', '703', '825', 'E900', '250.83', '698', '797', 'V64', '466', '57
1', '353', '701', '959', '319', '433', '694', '846', '847', '683', 'V5
4', '138', '822', 'E853', '843', '521', 'V17', 'E890', '980', '860', '4
62', '807', '383', '317', '598', '572', '477', '308', '309', '661', '36
9', '431', '524', '464', '289', '948', '910', 'E887', 'E939', '332', '3
```

```
79', '599', '246', '306', '46', 'E813', 'V09', '323', '658', 'V69', '81
5', '227', '162', '530', '955', '403', '581', '303', '575', '9', '348',
'411', '515', '709', '402', '250.22', '310', '457', 'E880', '276', 'V1
8', '188', '831', '921', '520', '995', '486', '438', '999', '580', 'E85
8', '994', 'V15', '311', '250.81', 'E816', '522', '623', '716', '203',
'680', '250.41', '724', '695', '41', '881', '410', '94', '610', '192',
'560', '250.02', 'E927', '748', '833', '861', 'V70', '141', '596', '37
2', '252', '656', '422', '850', '429', '250.91', '821', '455', '756',
'788', '183', '745', '204', '992', '864', '484', '40', '625', '781', '2
41', '282', '836', '250.1', '157', '261', '441', 'E850', '506', '753',
'288', '510', '600', '646', '634', '480', '534', '299', '260', 'E968',
'593', '198', '377', '250.6', 'V72', '911', '250', '472', '226', '357',
'727', '922', '182', '271', '205', '421', '155', '114', 'E885', '784',
'945', '250.93', 'E817', '347', '140', '715', '681', '738', 'E950', '33
5', '591', '996', '380', '250.53', '987', '934', 'E980', '565', '256',
'V16', '79', '604', '884', '195', '927', '607', '275', '851', '706', 'V
50', '344', '952', '529', '320', '259', '540', '164', '465', '783', '35
0', 'E937', 'E931', '917', '253', '232', '721', '250.42', '742', 'V44',
'52', '364', '189', '351', '712', '437', '736', '997', '755', 'E936',
'E879', '751', '298', 'E812', '953', 'E881', '906', '250.43', '346', '7
99', '915', '880', 'V42', '977', '868', '75', '201', '54', '343', '35
8', 'E934', '156', '590', '397', '912', '285', 'E854', '747', '584', '2
50.8', 'V14', '794', '473', '968', '787', 'V66', '11', 'E929', '791',
'415', '490', '180', '723', '327', '908', '454', 'E868', '603', '307',
'E878', '295', 'V45', '273', 'E916', '427', '705', '292', '627', '592',
'E947', '197', '8', '759', '649', '179', '228', '324', 'E932', '414',
'614', '381', '965', '944', '583', '446', '250.33', '388', '975', 'V2
3', '920', '844', '796', 'V12', 'V61', '27', 'V02', '789', '555', '51
8', '710', '870', '684', '972', '909', 'E905', '810', '304', '579', '3
1', '250.31', 'E944', 'V65', '297', '990', '376', 'E935', '728', '823',
'790', '514', '648', 'V57', '659', '853', '378', '78', '96', '291', '26
8', '277', '398', '444', '250.11', 'V25', '905', '867', 'E826', 'E942',
'250.5', '191', '568', '485', '758', '373', 'E888', '863', '746', '27
0', '962', '714', '622', '172', '837', 'V60', '284', 'V43', '5', '693',
'112', '396', '621', '341', '316', '611', '394', '647', '882', '115',
'405', '240', '426', '349', '601', 'V86', '251', '729', '250.03', '43
5', '615', '430', '269', '840', '355', '448', '576', '730', '665', '35
4', 'E941', '442', '642', '117', '416', '969', '507', '279', '652', 'E9
33', '88', '725', '436', '481', '813', '110', 'E818', '750', '300', '50
8', '262', '338', '869', '356', 'V63', '456', '196', '734', '451', 'V5
8', 'E849', 'E919', '726', '617', '173', '577', '413', '543', '238', '1
23', 'V62', '722', '871', '958', '337', '423', '250.4', '233', '585',
'42', '907', '325', '967', '852', 'V46', '290', '250.82', '795', '536',
'V10', '66', '272', '842', '463', '266', '618', '214', '447', '696', '4
24', '359', '452', '812', 'E928', '202', '453', '707', '244', '556', '8
45', 'E883', '255', '336', '626', '616', '805', '491', '718', '832', '6
45', '250.21', '382', '53', '283', '608', '386', '644', '412', '420',
'487', '998', '294', '150', '537', '250.23', '225', 'V11', '918', '42
8', '286', '512', '527', '70', '263', '215', '800', '352', '250.32', '8
20', '200', '494', '389', '686', '130', '808', '602', '553', '500', 'E8
14', '458', '552', '360', '991', '470', '250.13', '731', '136', '314',
'588', '478', 'E870', '531', '218', '528', '274', '211', '245', '278',
'496', '564', '754', '495', '395', '368', '594', '493', '691', '862',
'573', '223', '212', '802', '595', 'V13', '641', '492', '250.52', '53
2', '620', '345', '342', '459', '483', '513', '557', '733', '401', '48
2', '35', '145'}
diag_3 {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1
```

```
8, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71,
72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 1
06, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 12
0, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 13
4, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 14
8, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 16
2, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 17
6, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 19
0, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 20
4, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 21
8, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 23
2, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 24
6, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 26
0, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 27
4, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 28
8, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 30
2, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 31
6, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 33
0, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 34
4, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 35
8, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 37
2, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 38
6, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 40
0, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 41
4, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 42
8, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 44
2, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 45
6, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 47
0, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 48
4, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 49
8, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 51
2, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 52
6, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 54
0, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 55
4, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 56
8, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 58
2, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 59
6, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 61
0, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 62
4, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 63
8, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 65
2, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 66
6, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 68
0, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 69
4, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 70
8, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 72
2, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 73
6, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 75
0, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 76
4, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 77
8, 779, 780, 781, 782, 783, 784, 785} {'34', '404', '814', '883', '17',
'312', '460', '171', 'E819', '443', '533', '893', '7', '924', '333', '3
40', '685', '891', '365', '365.44', '664', '971', '174', '670', '152',
'296', '318', '711', '535', 'E955', '654', '151', '801', '511', '391',
```

```
'425', 'V06', '655', '519', '250.9', '501', '942', '682', '923', '250.1
2', '574', '47', '663', '258', 'V53', '387', '780', '517', '785', '36
6', '432', 'E938', '570', '865', '217', '250.92', '250.3', '301', '29
3', '305', '242', '434', '374', '163', '199', 'V49', '153', '220', '73
7', 'E945', '866', 'E924', '605', 'E865', '131', '550', '989', 'E930',
'111', '619', '239', '916', '542', '186', 'E917', '719', '935', '558',
'461', '854', '786', '702', '523', '281', '782', '362', '741', 'V85',
'E915', '230', '193', '440', '873', 'V08', '185', '243', '913', '793',
'135', '250.7', 'E884', '816', '38', '578', '516', '919', '824', 'V55',
'250.51', '250.01', '879', '811', '475', '872', '562', '331', '154', '8
92', '287', '315', '208', '792', '567', 'E882', '280', '250.2', '834',
'933', '692', '385', '569', '826', '566', '717', '586', 'E906', '713',
'703', '825', '757', '361', 'E900', '250.83', '698', '797', 'V64', '46
6', '571', '353', '701', '959', '433', '319', '694', 'E852', '847', 'V5
4', '138', 'E853', '653', '822', '521', 'V17', '980', '860', '462', '80
7', '383', '598', '477', '572', '317', '308', '309', '661', '369', '43
1', '524', '464', '289', '948', '910', 'E887', '332', 'E939', '379', '5
99', '246', '306', 'E813', 'V09', '323', '658', '815', '227', '162', '5
30', '955', '235', '582', '403', '581', '303', '575', '9', '348', '41
1', '515', '334', '709', '402', '250.22', '310', '457', 'E880', '161',
'276', 'V18', '188', '831', '921', '995', '486', '438', '999', '580',
'943', '236', 'E858', 'V15', '311', '250.81', 'E816', '522', '623', '38
4', '657', '716', '203', '680', '250.41', '724', '94', '41', '881', '41
0', '695', '610', '192', '875', '560', '250.02', 'E927', '861', '132',
'V70', 'E904', '141', '122', '596', '372', '252', '656', '429', '850',
'250.91', 'E886', '821', '455', '756', '788', '183', '745', '204', '99
2', '864', 'E828', '484', '625', '781', '836', '282', '732', '241', '25
0.1', '157', '261', '441', 'E850', '506', '753', '288', '510', '600',
'646', '877', '57', '480', '148', '534', '299', '260', '593', 'V07', '1
98', '216', '377', '250.6', 'V72', '911', '250', '472', '226', '357',
'727', '922', '182', '205', '271', '421', '155', 'E876', 'E885', '784',
'E949', '945', '250.93', 'E817', '347', '445', 'E894', '715', '681', 'E
950', '738', '335', '591', '996', '380', '250.53', '987', '934', '930',
'E965', 'E980', '565', '256', 'V16', '79', '604', '884', '195', '275',
'607', '851', '706', '344', '952', '529', 'E822', '259', '540', '170',
'164', 'E943', '465', '783', '350', 'E937', 'E931', '917', '253', '72
1', '250.42', '742', 'V44', '838', '671', '189', '417', '351', '712',
'3', '437', '736', '997', '755', 'E936', 'E879', '751', 'E956', '298',
'E825', 'E812', '953', 'E881', '906', '250.43', '346', '799', '915', '8
80', 'V42', '75', '868', '146', '201', '54', '343', '358', 'E855', 'E93
4', '156', '397', '590', '912', '265', '175', '285', 'E854', '747', '58
4', '250.8', '697', '956', 'V14', 'V01', '794', '473', '11', '787', 'V6
6', 'E946', 'E929', '791', '415', '490', '180', '723', '327', '908', '4
54', '603', '307', 'E878', '525', 'V45', '295', '273', 'E916', '427',
'705', '292', '627', '592', 'E947', '197', '735', '8', '970', '759', '6
69', '649', '179', '49', '228', 'E932', '414', '614', 'E864', '381', '9
65', '944', '583', '446', '388', '690', '920', 'V23', '744', '844', '79
6', 'V12', 'V61', '27', '14', 'V02', '789', '555', '518', '710', '870',
'684', '972', '909', 'E905', '810', '304', '579', '720', '250.31', 'E94
4', 'V65', '297', 'E920', '876', '376', 'E935', '728', '823', '790', '5
14', '648', 'V57', '659', '853', '378', '78', '291', '268', '398', '27
7', '444', '250.11', 'V25', '905', '867', 'E826', 'E942', '250.5', '96
6', '270', '568', '191', '758', '373', 'E888', '485', '746', '863', '96
2', '714', '622', '172', 'V60', '837', '284', '313', 'V43', '693', '5',
'890', '112', '396', '621', '341', '611', '394', '647', '882', '115',
'405', '240', '426', '349', '660', '601', '251', 'V22', '729', 'V86',
'250.03', '435', '430', 'E901', '840', '355', '448', '576', '538', '73
```

```
0', '665', '354', 'E941', '442', '642', 'E987', '951', '416', '117', '9
69', '841', '507', '279', '652', 'E933', '88', '725', '436', '481', '81
3', '110', 'E818', '750', '928', '300', '508', '262', '338', '643', '35
6', 'E861', '456', '196', '734', '451', 'V58', 'E849', 'V63', 'E919',
'726', '617', '173', '577', '413', 'V27', '543', '238', '123', 'V62',
'722', '871', '958', '337', '423', '250.4', '233', '585', '42', '907',
'967', '852', 'V46', '290', '250.82', '795', '536', 'V10', '66', '272',
'708', '842', '266', '618', '214', '447', '696', '424', '359', '452',
'812', 'E928', '202', '453', '139', '704', '707', '674', '244', '845',
'556', 'E883', 'E912', '255', '336', '848', '626', '616', 'V03', '805',
'491', '718', '250.21', '608', '382', '53', '283', '386', '644', '412',
'420', '487', '998', '294', '150', '537', '250.23', 'V11', '225', '91
8', '428', '752', '286', '512', '527', '70', '263', '215', '800', '37
0', '158', '820', '200', '494', '389', '686', '808', '597', '602', '55
3', '500', '458', '552', 'E892', '360', '991', '470', '250.13', '731',
'136', '314', '588', '478', 'E870', '531', '218', '528', '274', '211',
'245', '278', '496', '564', '495', '395', '754', '368', '594', '493',
'862', '573', '223', '802', '624', '595', 'V13', '641', '250.52', '49
2', '532', '620', '345', '459', '342', '483', '557', 'E815', '733', '40
1', '482', '35'}
number_diagnoses {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69,
70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87,
88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 10
4, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 11
8, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 13
2, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 14
6, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 16
0, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 17
4, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 18
8, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 20
2, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 21
6, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 23
0, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 24
4, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 25
8, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 27
2, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 28
6, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 30
0, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 31
4, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 32
8, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 34
2, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 35
6, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 37
0, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 38
4, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 39
8, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 41
2, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 42
6, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 44
0, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 45
4, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 46
8, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 48
2, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 49
6, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 51
0, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 52
4, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 53
```

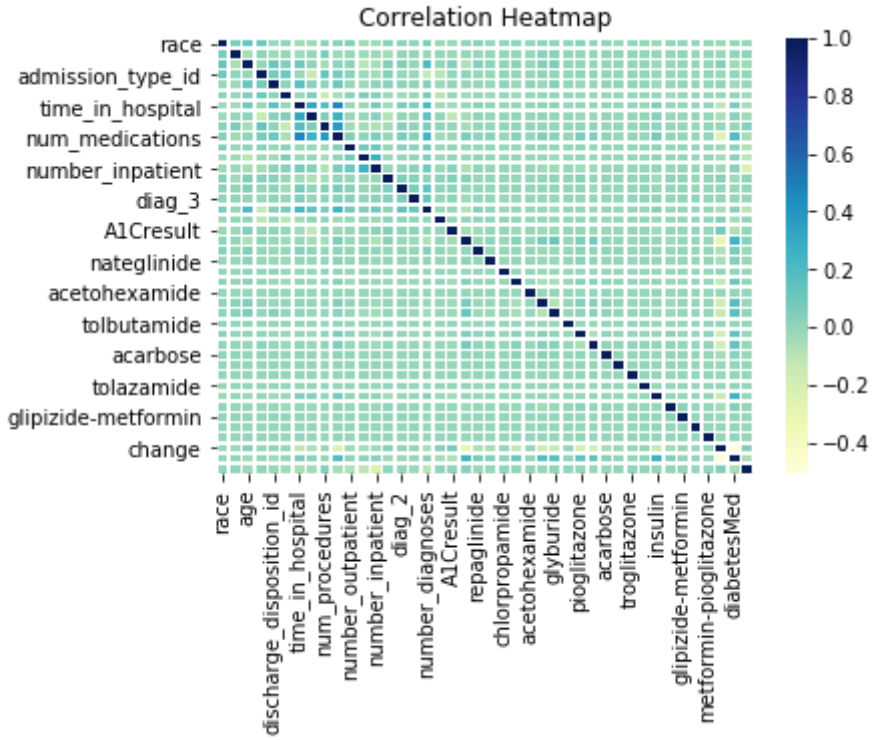
```
8, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 55
2, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 56
6, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 58
0, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 59
4, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 60
8, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 62
2, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 63
6, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 65
0, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 66
4, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 67
8, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 69
2, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 70
6, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 72
0, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 73
4, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 74
8, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 76
2, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 77
6, 777, 778, 779, 780, 781, 782, 783, 784, 785} {'34', '404', '814', '8
83', '17', '312', '460', '171', 'E819', '443', '533', '893', '7', '92
4', '333', '340', '685', '891', '365', '365.44', '664', '971', '174',
'670', '152', '296', '318', '711', '535', 'E955', '654', '151', '801',
'511', '391', '425', 'V06', '655', '519', '250.9', '501', '942', '682',
'923', '250.12', '574', '47', '663', '258', 'V53', '387', '780', '517',
'785', '366', '432', 'E938', '570', '865', '217', '250.92', '250.3', '3
01', '293', '305', '242', '434', '374', '163', '199', 'V49', '153', '22
0', '737', 'E945', '866', 'E924', '605', 'E865', '131', '550', '989',
'E930', '111', '619', '239', '916', '542', '186', 'E917', '719', '935',
'558', '461', '854', '786', '702', '523', '281', '782', '362', '741',
'V85', 'E915', '230', '193', '440', '873', 'V08', '185', '243', '913',
'793', '135', '250.7', 'E884', '816', '38', '578', '516', '919', '824',
'V55', '250.51', '250.01', '879', '811', '475', '872', '562', '331', '1
54', '892', '287', '315', '208', '792', '567', 'E882', '280', '250.2',
'834', '933', '692', '385', '569', '826', '566', '717', '586', 'E906',
'713', '703', '825', '757', '361', 'E900', '250.83', '698', '797', 'V6
4', '466', '571', '353', '701', '959', '433', '319', '694', 'E852', '84
7', 'V54', '138', 'E853', '653', '822', '521', 'V17', '980', '860', '46
2', '807', '383', '598', '477', '572', '317', '308', '309', '661', '36
9', '431', '524', '464', '289', '948', '910', 'E887', '332', 'E939', '3
79', '599', '246', '306', 'E813', 'V09', '323', '658', '815', '227', '1
62', '530', '955', '235', '582', '403', '581', '303', '575', '9', '34
8', '411', '515', '334', '709', '402', '250.22', '310', '457', 'E880',
'161', '276', 'V18', '188', '831', '921', '995', '486', '438', '999',
'580', '943', '236', 'E858', 'V15', '311', '250.81', 'E816', '522', '62
3', '384', '657', '716', '203', '680', '250.41', '724', '94', '41', '88
1', '410', '695', '610', '192', '875', '560', '250.02', 'E927', '861',
'132', 'V70', 'E904', '141', '122', '596', '372', '252', '656', '429',
'850', '250.91', 'E886', '821', '455', '756', '788', '183', '745', '20
4', '992', '864', 'E828', '484', '625', '781', '836', '282', '732', '24
1', '250.1', '157', '261', '441', 'E850', '506', '753', '288', '510',
'600', '646', '877', '57', '480', '148', '534', '299', '260', '593', 'V
07', '198', '216', '377', '250.6', 'V72', '911', '250', '472', '226',
'357', '727', '922', '182', '205', '271', '421', '155', 'E876', 'E885',
'784', 'E949', '945', '250.93', 'E817', '347', '445', 'E894', '715', '6
81', 'E950', '738', '335', '591', '996', '380', '250.53', '987', '934',
'930', 'E965', 'E980', '565', '256', 'V16', '79', '604', '884', '195',
'275', '607', '851', '706', '344', '952', '529', 'E822', '259', '540',
'170', '164', 'E943', '465', '783', '350', 'E937', 'E931', '917', '25
```

```
3', '721', '250.42', '742', 'V44', '838', '671', '189', '417', '351',
'712', '3', '437', '736', '997', '755', 'E936', 'E879', '751', 'E956',
'298', 'E825', 'E812', '953', 'E881', '906', '250.43', '346', '799', '9
15', '880', 'V42', '75', '868', '146', '201', '54', '343', '358', 'E85
5', 'E934', '156', '397', '590', '912', '265', '175', '285', 'E854', '7
47', '584', '250.8', '697', '956', 'V14', 'V01', '794', '473', '11', '7
87', 'V66', 'E946', 'E929', '791', '415', '490', '180', '723', '327',
'908', '454', '603', '307', 'E878', '525', 'V45', '295', '273', 'E916',
'427', '705', '292', '627', '592', 'E947', '197', '735', '8', '970', '7
59', '669', '649', '179', '49', '228', 'E932', '414', '614', 'E864', '3
81', '965', '944', '583', '446', '388', '690', '920', 'V23', '744', '84
4', '796', 'V12', 'V61', '27', '14', 'V02', '789', '555', '518', '710',
'870', '684', '972', '909', 'E905', '810', '304', '579', '720', '250.3
1', 'E944', 'V65', '297', 'E920', '876', '376', 'E935', '728', '823',
'790', '514', '648', 'V57', '659', '853', '378', '78', '291', '268', '3
98', '277', '444', '250.11', 'V25', '905', '867', 'E826', 'E942', '250.
5', '966', '270', '568', '191', '758', '373', 'E888', '485', '746', '86
3', '962', '714', '622', '172', 'V60', '837', '284', '313', 'V43', '69
3', '5', '890', '112', '396', '621', '341', '611', '394', '647', '882',
'115', '405', '240', '426', '349', '660', '601', '251', 'V22', '729',
'V86', '250.03', '435', '430', 'E901', '840', '355', '448', '576', '53
8', '730', '665', '354', 'E941', '442', '642', 'E987', '951', '416', '1
17', '969', '841', '507', '279', '652', 'E933', '88', '725', '436', '48
1', '813', '110', 'E818', '750', '928', '300', '508', '262', '338', '64
3', '356', 'E861', '456', '196', '734', '451', 'V58', 'E849', 'V63', 'E
919', '726', '617', '173', '577', '413', 'V27', '543', '238', '123', 'V
62', '722', '871', '958', '337', '423', '250.4', '233', '585', '42', '9
07', '967', '852', 'V46', '290', '250.82', '795', '536', 'V10', '66',
'272', '708', '842', '266', '618', '214', '447', '696', '424', '359',
'452', '812', 'E928', '202', '453', '139', '704', '707', '674', '244',
'845', '556', 'E883', 'E912', '255', '336', '848', '626', '616', 'V03',
'805', '491', '718', '250.21', '608', '382', '53', '283', '386', '644',
'412', '420', '487', '998', '294', '150', '537', '250.23', 'V11', '22
5', '918', '428', '752', '286', '512', '527', '70', '263', '215', '80
0', '370', '158', '820', '200', '494', '389', '686', '808', '597', '60
2', '553', '500', '458', '552', 'E892', '360', '991', '470', '250.13',
'731', '136', '314', '588', '478', 'E870', '531', '218', '528', '274',
'211', '245', '278', '496', '564', '495', '395', '754', '368', '594',
'493', '862', '573', '223', '802', '624', '595', 'V13', '641', '250.5
2', '492', '532', '620', '345', '459', '342', '483', '557', 'E815', '73
3', '401', '482', '35'}
max_glu_serum {0, 1, 2, 3} {'Norm', 'None', '>200', '>300'}
A1Cresult {0, 1, 2, 3} {'Norm', '>7', 'None', '>8'}
metformin {0, 1, 2, 3} {'No', 'Steady', 'Up', 'Down'}
repaglinide {0, 1, 2, 3} {'No', 'Steady', 'Up', 'Down'}
nateglinide {0, 1, 2, 3} {'No', 'Steady', 'Up', 'Down'}
chlorpropamide {0, 1, 2, 3} {'No', 'Steady', 'Up', 'Down'}
glimepiride {0, 1, 2, 3} {'No', 'Steady', 'Up', 'Down'}
acetohexamide {0, 1} {'No', 'Steady'}
glipizide {0, 1, 2, 3} {'No', 'Steady', 'Up', 'Down'}
glyburide {0, 1, 2, 3} {'No', 'Steady', 'Up', 'Down'}
tolbutamide {0, 1} {'No', 'Steady'}
pioglitazone {0, 1, 2, 3} {'No', 'Steady', 'Up', 'Down'}
rosiglitazone {0, 1, 2, 3} {'No', 'Steady', 'Up', 'Down'}
acarbose {0, 1, 2, 3} {'No', 'Steady', 'Up', 'Down'}
miglitol {0, 1, 2, 3} {'No', 'Steady', 'Up', 'Down'}
troglitazone {0, 1} {'No', 'Steady'}
```

```
tolazamide {0, 1, 2} {'No', 'Steady', 'Up'}
examide {0} {'No'}
citoglipton {0} {'No'}
insulin {0, 1, 2, 3} {'Up', 'Steady', 'No', 'Down'}
glyburide-metformin {0, 1, 2, 3} {'No', 'Steady', 'Up', 'Down'}
glipizide-metformin {0, 1} {'No', 'Steady'}
glimepiride-pioglitazone {0, 1} {'No', 'Steady'}
metformin-rosiglitazone {0} {'No'}
metformin-pioglitazone {0, 1} {'No', 'Steady'}
change {0, 1} {'No', 'Ch'}
diabetesMed {0, 1} {'No', 'Yes'}
readmitted {0, 1, 2} {'NO', '<30', '>30'}
```

In [12]:
```
cols = df.columns
```

In [13]:
```
vars_to_use = [item for item in cols if item not in remove_col] # pick v
ars
plt.title('Correlation Heatmap')
sns.heatmap(df[vars_to_use].corr(),annot=False, fmt="f", cmap="YlGnBu",
linewidths=0.5)
plt.show()
```



Here, we want to find the main dimensions which help us to compress the data into small size.

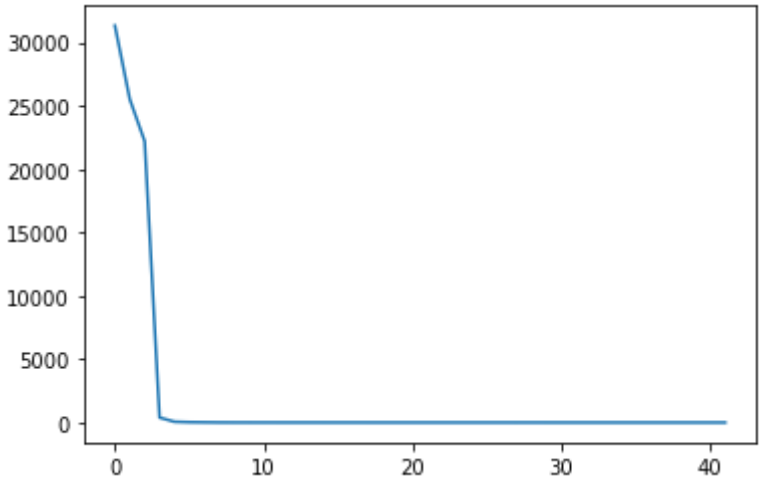
We use PCA to calculate the cov of dataframe and compute the eigenvalues which help us to find the weight of each feature.

finally, we find the age, race and gender is top 3 weight feature, after we build the decision tree, we will implement the pca data.

In [14]:
```
cov = df[vars_to_use].cov()
```

In [15]:
```
import numpy as np
eig_val, eig_vec = np.linalg.eig(cov)
```

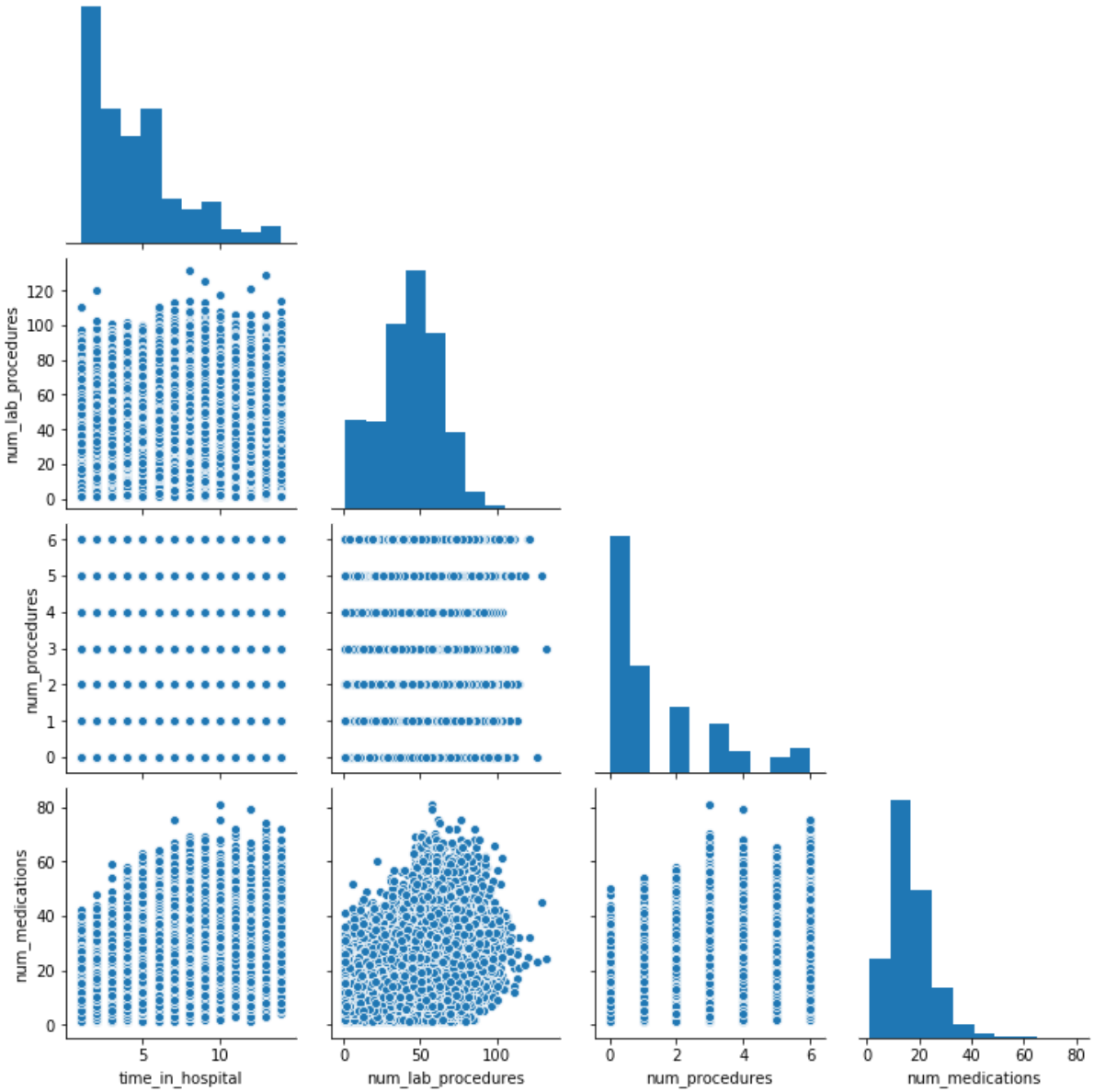
In [16]:
```
plt.plot(eig_val)
plt.show()
```



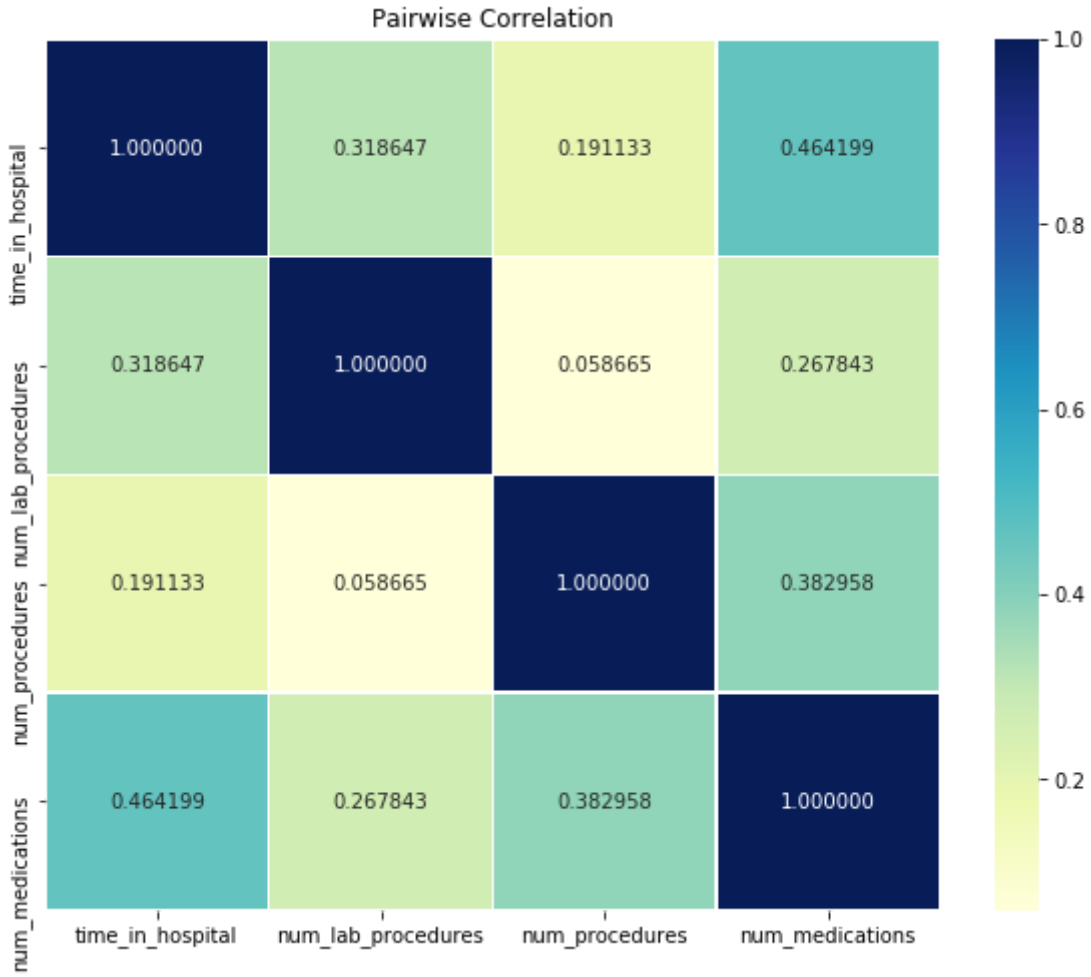
In [17]:
```
total = sum(eig_val)
for i in range(len(vars_to_use)):
    print(vars_to_use[i], eig_val[i]/total)
```

```
race 0.39371963370161045
gender 0.32071661180698147
age 0.27898098889598616
admission_type_id 0.004940628669071916
discharge_disposition_id 0.0007863448647460391
admission_source_id 0.0003488994564904768
time_in_hospital 0.00020742521587846076
num_lab_procedures 8.056661791769524e-05
num_procedures 4.278747638718541e-05
num_medications 3.1962816785196095e-05
number_outpatient 2.815692321463492e-05
number_emergency 2.3828341800340387e-05
number_inpatient 2.037861369839961e-05
diag_1 1.8301208164267663e-05
diag_2 9.560643801103227e-06
diag_3 9.059654317634512e-06
number_diagnoses 8.811036289405853e-06
max_glu_serum 5.437036485727402e-06
A1Cresult 4.26095095337505e-06
metformin 3.2156528882597043e-06
repaglinide 3.0535587865937676e-06
nateglinide 1.9030822730948097e-06
chlorpropamide 1.6863149100688978e-06
glimepiride 1.3458166365260478e-06
acetohexamide 1.1950344923795745e-06
glipizide 1.158105289754156e-06
glyburide 9.303173442006989e-07
tolbutamide 6.753521246440119e-07
pioglitazone 7.099124868457109e-07
rosiglitazone 2.2943448742906505e-07
acarbose 9.623910700510974e-08
miglitol 8.655586839291668e-08
troglitazone 4.190327638178571e-08
tolazamide 1.2911284514719714e-08
insulin 5.506481605160644e-09
glyburide-metformin 5.121281885739713e-09
glipizide-metformin 2.817716662402021e-09
glimepiride-pioglitazone 1.6637609771029708e-09
metformin-pioglitazone 3.844473614645289e-10
change 1.2817254696140358e-10
diabetesMed 1.2814110177242792e-10
readmitted 1.2816199263514072e-10
```

In [18]:
```
sns.pairplot(df[["time_in_hospital", "num_lab_procedures", "num_procedures","num_medications"]],corner=True)
plt.show()
```



In [19]:
```
plt.figure(figsize=(10,8))
sns.heatmap(df[["time_in_hospital", "num_lab_procedures", "num_procedures","num_medications"]].corr(), annot=True, fmt="f", cmap="YlGnBu", linewidths=0.5)
plt.title('Pairwise Correlation')
plt.show()
```



Approach

During the analyzing process, we used two approaches to figure out whether a person has diabetes or not. The first approach is the decision tree model, which is a supervised machine learning technique for inducing a decision tree from training data. A decision tree is a predictive model which is a mapping from observations about an item to conclusions about its target value. Here, take the top three features for example. Change, Insulin and Metformin are at the top of the tree, which means they are the most important features in this training model. According to entropy theory, Information entropy is usually used to describe the average amount of information that results from the entire random distribution and is more statistically specific.

From the equation, it can be seen that the information entropy H(X) is the cumulative value of each self-information, and since each is an integer positive, the more times the random variable takes the value, the more times it accumulates, the greater the entropy of the information, the greater the degree of confusion, and the lower the purity. The broader the distribution, the greater the entropy, and within the same defined domain, due to the impulse distribution in the broadness of the distribution < Gaussian distribution < uniform distribution Therefore, the relationship between entropy is pulse distribution of information entropy < Gaussian distribution of information entropy < uniform distribution of information entropy. It can be proved mathematically that entropy is greatest when the distribution of random variables is uniform, i.e. when the number of states is highest. Applied to our model, we use the information gain (Information entropy - conditional entropy). Information gain represents the degree to which

information complexity (uncertainty) is reduced under one condition. In the decision tree algorithm, our aim is to select one feature at a time. If there are multiple features, the information gain can be used as a measure. If a feature is selected with the greatest information gain (the greatest reduction in information uncertainty), then we choose that feature. In our model, Change, Insulin and Metformin are the greatest information gains. The other approach we used is Principal Component Analysis (PCA). PCA is a process of identifying the high level of information retention of the data, and then replacing the raw data with the most significant features of the data. It is one of the most important methods of dimensionality reduction. In simple terms, it is the transformation of data from the original space into a new feature space, e.g., the original space is three-dimensional (x,y ,z), x, y, and z are the three bases of the original space, and we can somehow use the new coordinate system (a,b, c) to represent the original data, then a, b, and c are the new bases, and they form the new feature space. In the new feature space, it may be that all data projections on c are close to zero, i.e. negligible, then we can just Use (a,b) to represent the data so that the data are reduced from three-dimensional (x,y,z) to two-dimensional (a,b). We zero-average the raw data, then derive the covariance matrix, then derive the eigenvectors and eigenvalues from the covariance matrix, and these eigenvectors form the new feature space. Here, we use PCA in order to keep data information as much as possible.

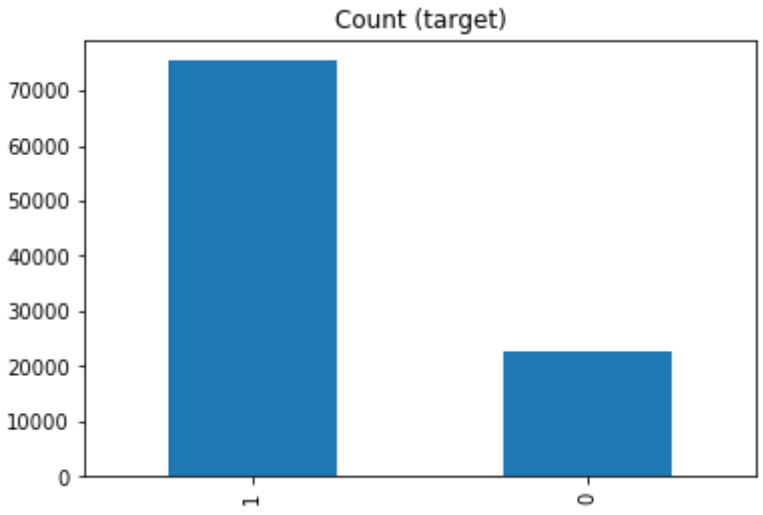
In [20]:
```
X = df.drop(['diabetesMed'],axis = 1)
Y = df['diabetesMed']
```

In [21]:
```
X.columns
```

Out[21]:
```
Index(['race', 'gender', 'age', 'admission_type_id',
       'discharge_disposition_id', 'admission_source_id', 'time_in_hosp
ital',
       'num_lab_procedures', 'num_procedures', 'num_medications',
       'number_outpatient', 'number_emergency', 'number_inpatient', 'di
ag_1',
       'diag_2', 'diag_3', 'number_diagnoses', 'max_glu_serum', 'A1Cres
ult',
       'metformin', 'repaglinide', 'nateglinide', 'chlorpropamide',
       'glimepiride', 'acetohexamide', 'glipizide', 'glyburide', 'tolbu
tamide',
       'pioglitazone', 'rosiglitazone', 'acarbose', 'miglitol', 'trogli
tazone',
       'tolazamide', 'examide', 'citoglipton', 'insulin',
       'glyburide-metformin', 'glipizide-metformin',
       'glimepiride-pioglitazone', 'metformin-rosiglitazone',
       'metformin-pioglitazone', 'change', 'readmitted'],
      dtype='object')
```

In [22]:
```
target_count = Y.value_counts() #count numbers of each class
#show the amount in each class
target_count.plot(kind='bar', title='Count (target)');
target_count = list(target_count)
print('Class 0:', target_count[0])
print('Class 1:', target_count[1])
```

```
Class 0: 75351
Class 1: 22702
```



In [23]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size=0.2, r
andom_state=50)

print("Num training samples:", len(X_train), "Num testing samples:", len
(X_test))
```

```
Num training samples: 78442 Num testing samples: 19611
```

In [24]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

'''
Write your code below
'''
clf = tree.DecisionTreeClassifier(max_depth = 10)
clf.fit(X_train, y_train)
```

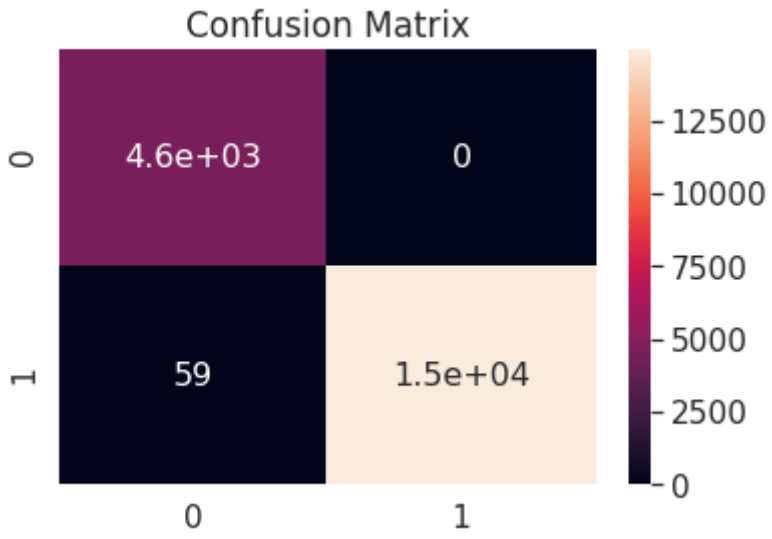
Out[24]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gin
i',
                       max_depth=10, max_features=None, max_leaf_nodes=
None,
                       min_impurity_decrease=0.0, min_impurity_split=No
ne,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecate
d',
                       random_state=None, splitter='best')
```

In [25]:
```
from sklearn.metrics import accuracy_score

pred = clf.predict(X_test)
accuracy_score(y_test, pred)
```

Out[25]:
```
0.9969914843710163
```

In [26]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sn

conf_mat = confusion_matrix(y_test, pred)

df_cm = pd.DataFrame(conf_mat, range(2),range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm, annot=True,annot_kws={"size": 16})
plt.title('Confusion Matrix')
```

Out[26]: Text(0.5, 1, 'Confusion Matrix')



In [27]:
```
import numpy as np
eig_val, eig_vec = np.linalg.eig(X.cov())
```

In [28]:
```
total = sum(eig_val)
for i in range(len(X.columns)):
    print(list(X.columns)[i], eig_val[i]/total)
```

```
race 0.39372051387646606
gender 0.32071732752690674
age 0.27898161282038775
admission_type_id 0.004940636139079
discharge_disposition_id 0.0007862742874496 03
admission_source_id 0.00034889080972700444
time_in_hospital 0.00020742457807398936
num_lab_procedures 8.05649535956088e-05
num_procedures 4.2787424724582593e-05
num_medications 3.1939194988707784e-05
number_outpatient 2.815635988418637e-05
number_emergency 2.3828113056375835e-05
number_inpatient 2.037760692226999e-05
diag_1 1.83011235729596e-05
diag_2 9.545460244535023e-06
diag_3 8.956350091558296e-06
number_diagnoses 8.708928404775846e-06
max_glu_serum 5.419141337098398e-06
A1Cresult 3.6851105297643363e-06
metformin 3.048173041108532e-06
repaglinide 3.1430626430777565e-06
nateglinide 1.8670075303499174e-06
chlorpropamide 1.6818247809929268e-06
glimepiride 1.3023704896382129e-06
acetohexamide 1.1940954864915756e-06
glipizide 9.318161477772862e-07
glyburide 6.857481623469074e-07
tolbutamide 7.122797109482077e-07
pioglitazone 2.3006284110119314e-07
rosiglitazone 9.629272879472573e-08
acarbose 8.675530575906845e-08
miglitol 4.1903950449149815e-08
troglitazone 1.2921740673430335e-08
tolazamide 5.50654747654115e-09
examide 5.122449820049057e-09
citoglipton 2.8182984502665465e-09
insulin 1.6637754123141602e-09
glyburide-metformin 3.844496597908314e-10
glipizide-metformin 1.281728520728904e-10
glimepiride-pioglitazone 1.2814160562773894e-10
metformin-rosiglitazone 1.281623043207319e-10
metformin-pioglitazone 0.0
change 0.0
readmitted 0.0
```

Improve, implement PCA

In [29]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=6)
pca.fit(X)
pca.explained_variance_ratio_
```

Out[29]:
```
array([3.93720514e-01, 3.20717328e-01, 2.78981613e-01, 4.94063614e-03,
       7.86274287e-04, 3.48890810e-04])
```

In [30]:
```
pca_X = pca.transform(X)
```

In [31]:
```
#pd.DataFrame(pca.components_,columns=X.columns,index = ['PC-1','PC-2','PC-3'])
```

In [32]:
```
from sklearn.model_selection import train_test_split

pca_X_train, pca_X_test, y_train, y_test = train_test_split(pca_X,Y,test_size=0.2, random_state=50)

print("Num training samples:", len(pca_X_train), "Num testing samples:", len(pca_X_test))
```

```
Num training samples: 78442 Num testing samples: 19611
```

In [33]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

'''
Write your code below
'''
clf = tree.DecisionTreeClassifier(max_depth = 10)
clf.fit(pca_X_train, y_train)
```

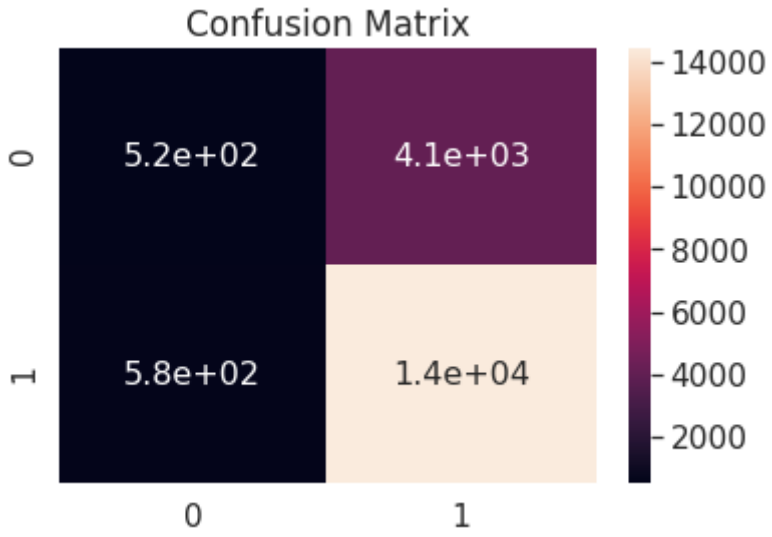
Out[33]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=10, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

In [34]:
```
from sklearn.metrics import accuracy_score

pred = clf.predict(pca_X_test)
accuracy_score(y_test, pred)
```

Out[34]:
```
0.7623782571006068
```

In [35]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sn

conf_mat = confusion_matrix(y_test, pred)

df_cm = pd.DataFrame(conf_mat, range(2),range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm, annot=True,annot_kws={"size": 16})
plt.title('Confusion Matrix')
```

Out[35]: Text(0.5, 1, 'Confusion Matrix')



drop personal information

In [36]:
```
X = df.drop(['diabetesMed'],axis = 1)
X = X.iloc[:,17:]
Y = df['diabetesMed']
```

In [37]:
```
X.columns
```

Out[37]:
```
Index(['max_glu_serum', 'A1Cresult', 'metformin', 'repaglinide', 'nateg
linide',
       'chlorpropamide', 'glimepiride', 'acetohexamide', 'glipizide',
       'glyburide', 'tolbutamide', 'pioglitazone', 'rosiglitazone', 'ac
arbose',
       'miglitol', 'troglitazone', 'tolazamide', 'examide', 'citoglipto
n',
       'insulin', 'glyburide-metformin', 'glipizide-metformin',
       'glimepiride-pioglitazone', 'metformin-rosiglitazone',
       'metformin-pioglitazone', 'change', 'readmitted'],
      dtype='object')
```

In [38]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size=0.2, r
andom_state=50)

print("Num training samples:", len(X_train), "Num testing samples:", len
(X_test))
```

```
Num training samples: 78442 Num testing samples: 19611
```

In [39]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

'''
Write your code below
'''
clf = tree.DecisionTreeClassifier(max_depth = 10)
clf.fit(X_train, y_train)
```

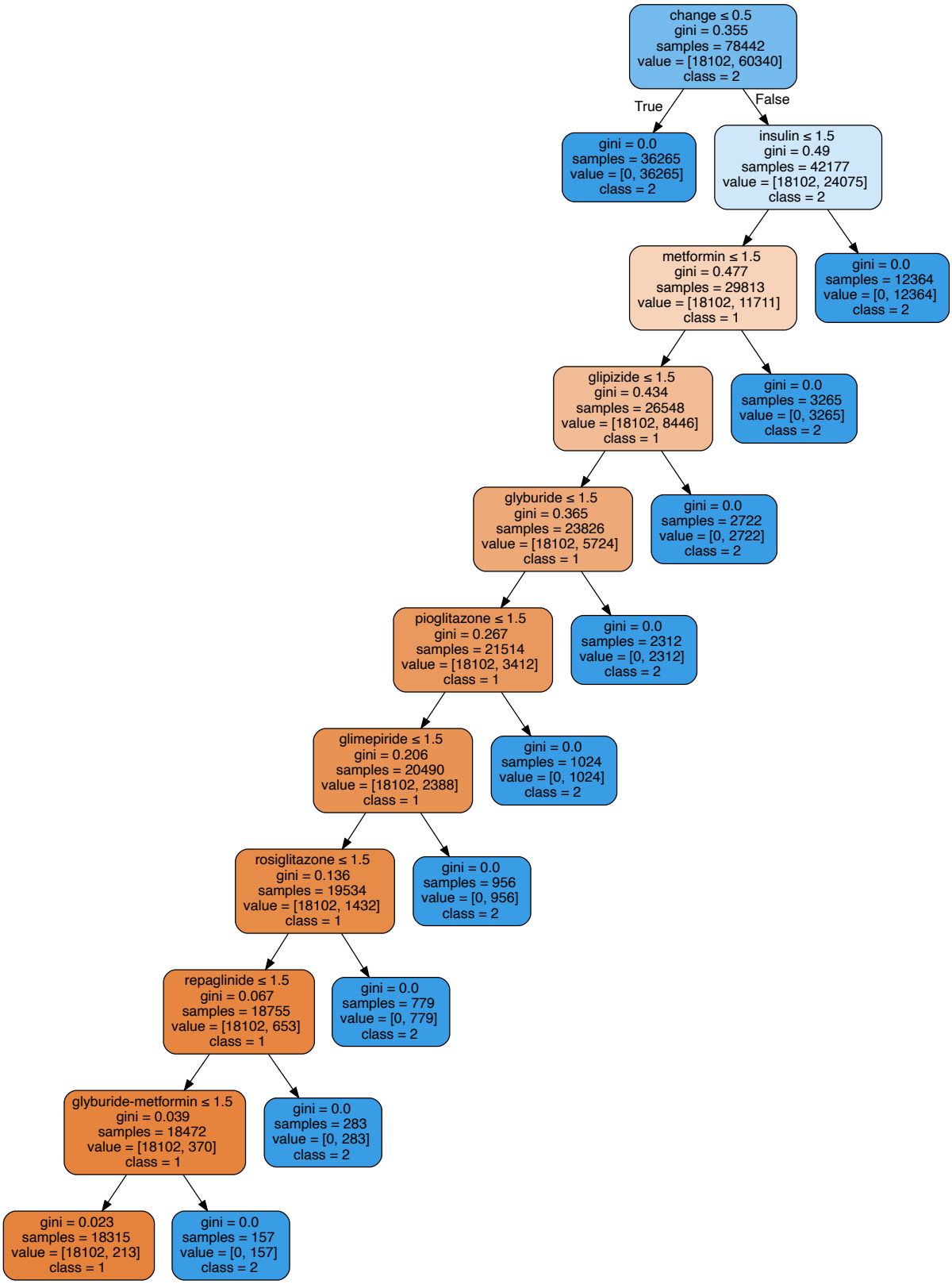
Out[39]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gin
i',
                       max_depth=10, max_features=None, max_leaf_nodes=
None,
                       min_impurity_decrease=0.0, min_impurity_split=No
ne,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecate
d',
                       random_state=None, splitter='best')
```

In [40]:
```
from sklearn.metrics import accuracy_score

pred = clf.predict(X_test)
accuracy_score(y_test, pred)
```

Out[40]:
```
0.9969914843710163
```

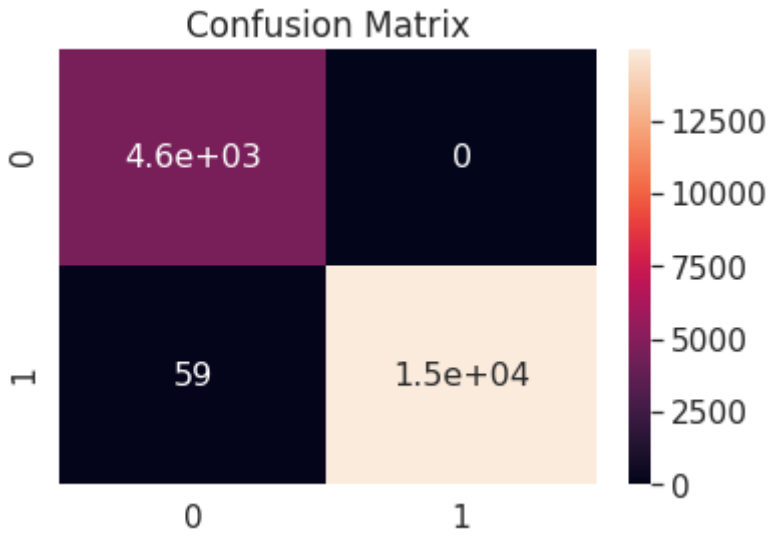
In [41]:
```
import graphviz
import os
feature_names = list(X_test.columns)
class_names = ['1', '2', '3']

'''
Write your code below
'''
dot_data = tree.export_graphviz(clf, out_file=None,feature_names=feature
_names,\
                                class_names=class_names, filled=True,\
                                rounded=True,special_characters=True)
graph = graphviz.Source(dot_data)
graph
```

Out[41]:


change ≤ 0.5
gini = 0.355
samples = 78442
value = [18102, 60340]
class = 2
True
False
gini = 0.0
samples = 36265
value = [0, 36265]
class = 2
insulin ≤ 1.5
gini = 0.49
samples = 42177
value = [18102, 24075]
class = 2
metformin ≤ 1.5
gini = 0.477
samples = 29813
value = [18102, 11711]
class = 1
gini = 0.0
samples = 12364
value = [0, 12364]
class = 2
glipizide ≤ 1.5
gini = 0.434
samples = 26548
value = [18102, 8446]
class = 1
gini = 0.0
samples = 3265
value = [0, 3265]
class = 2
glyburide ≤ 1.5
gini = 0.365
samples = 23826
value = [18102, 5724]
class = 1
gini = 0.0
samples = 2722
value = [0, 2722]
class = 2
pioglitazone ≤ 1.5
gini = 0.267
samples = 21514
value = [18102, 3412]
class = 1
gini = 0.0
samples = 2312
value = [0, 2312]
class = 2
glimepiride ≤ 1.5
gini = 0.206
samples = 20490
value = [18102, 2388]
class = 1
gini = 0.0
samples = 1024
value = [0, 1024]
class = 2
rosiglitazone ≤ 1.5
gini = 0.136
samples = 19534
value = [18102, 1432]
class = 1
gini = 0.0
samples = 956
value = [0, 956]
class = 2
repaglinide ≤ 1.5
gini = 0.067
samples = 18755
value = [18102, 653]
class = 1
gini = 0.0
samples = 779
value = [0, 779]
class = 2
glyburide-metformin ≤ 1.5
gini = 0.039
samples = 18472
value = [18102, 370]
class = 1
gini = 0.0
samples = 283
value = [0, 283]
class = 2
gini = 0.023
samples = 18315
value = [18102, 213]
class = 1
gini = 0.0
samples = 157
value = [0, 157]
class = 2

In [42]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sn

conf_mat = confusion_matrix(y_test, pred)

df_cm = pd.DataFrame(conf_mat, range(2),range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm, annot=True,annot_kws={"size": 16})
plt.title('Confusion Matrix')
```

Out[42]: Text(0.5, 1, 'Confusion Matrix')



PCA

In [43]:
```
eig_val, eig_vec = np.linalg.eig(X.cov())
total = sum(eig_val)
for i in range(len(X.columns)):
    print(list(X.columns)[i], eig_val[i]/total)
```

```
max_glu_serum 0.2892371734366437
A1Cresult 0.19098547308153727
metformin 0.1257690625183079
repaglinide 0.10450697760082313
nateglinide 0.06152198638137348
chlorpropamide 0.0546181298110954
glimepiride 0.04279445553045121
acetohexamide 0.039309389524198594
glipizide 0.03014765437933832
glyburide 0.023168489673578523
tolbutamide 0.022274490539067434
pioglitazone 0.007460722394389842
rosiglitazone 0.0031143323077734465
acarbose 0.0028069447184867493
miglitol 0.0013540144668306628
troglitazone 0.0004176890886220288
tolazamide 0.00017786732431120157
examide 0.00016550693351677416
citoglipton 9.104884067032685e-05
insulin 5.375436250722393e-05
glyburide-metformin 1.2418152582818944e-05
glipizide-metformin 4.1390036304904155e-06
glimepiride-pioglitazone 4.1398015600616465e-06
metformin-rosiglitazone 4.140128703670846e-06
metformin-pioglitazone 0.0
change 0.0
readmitted 0.0
```

In [44]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=6)
pca.fit(X)
pca.explained_variance_ratio_
```

Out[44]:
```
array([0.28923717, 0.19098547, 0.12576906, 0.10450698, 0.06152199,
       0.05461813])
```

In [45]:
```
pca_X = pca.transform(X)
```

In [46]:
```
from sklearn.model_selection import train_test_split

pca_X_train, pca_X_test, y_train, y_test = train_test_split(pca_X,Y,test
_size=0.2, random_state=50)

print("Num training samples:", len(pca_X_train), "Num testing samples:",
len(pca_X_test))
```

```
Num training samples: 78442 Num testing samples: 19611
```

In [47]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

'''
Write your code below
'''
clf = tree.DecisionTreeClassifier(max_depth = 10)
clf.fit(pca_X_train, y_train)
```

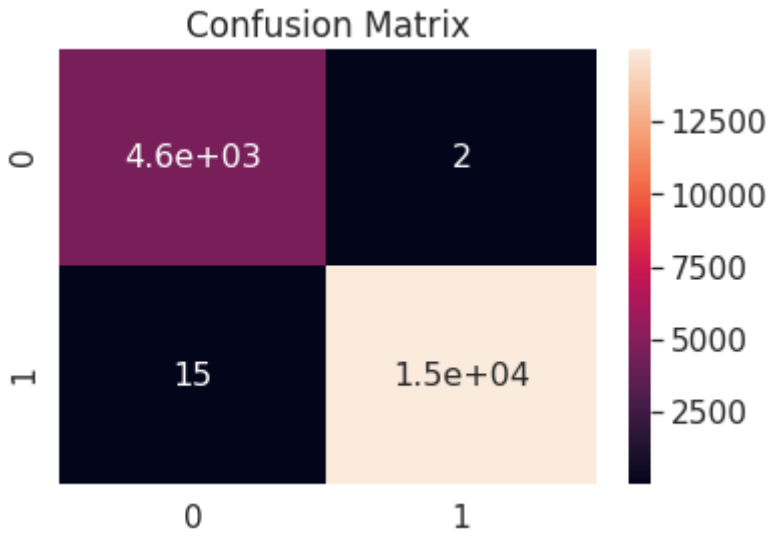
Out[47]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gin
i',
                       max_depth=10, max_features=None, max_leaf_nodes=
None,
                       min_impurity_decrease=0.0, min_impurity_split=No
ne,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecate
d',
                       random_state=None, splitter='best')
```

In [48]:
```
from sklearn.metrics import accuracy_score

pred = clf.predict(pca_X_test)
accuracy_score(y_test, pred)
```

Out[48]:
```
0.9991331395645301
```

In [49]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sn

conf_mat = confusion_matrix(y_test, pred)

df_cm = pd.DataFrame(conf_mat, range(2),range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm, annot=True,annot_kws={"size": 16})
plt.title('Confusion Matrix')
```

Out[49]: Text(0.5, 1, 'Confusion Matrix')



Analysis Results

For this project, we aim to improve model accuracy by reducing feature quantities. To better evaluate the results, we calculated the accuracy and got a confusion matrix before we implemented the analysis of the Principal Component Analysis (PCA). So that we can compare the accuracy and the confusion matrix before and after we alter the data. These two measurements are the most direct way to see the efficiency of drop less weighted features through PCA. To start with, after data preparation and reclassification, we first calculate the accuracy of the original model, which is 99.7%. Then we implemented PCA, the accuracy suddenly dropped to 76%. After getting the weight of each feature, according to the decision tree's estimation on features, the top ones are the main features of medication-related features such as insulin. Furthermore, we found that the bottom features on the decision tree are less weighted and have small significance in affecting the data. These features are all related to personal identities in a total of 16. So we decide to remove these 16 features of personal information such as age, gender, etc. Then we tested for accuracy on the original model (without 16 features) firstly, the accuracy remained 99.7%. However, after PCA implementation, the accuracy increased to 99.9%. (Exhibit 3) This is a very good result because it's pretty close to 100%. Indicating the model is well established. That by removing less weighted features, we can reduce distraction caused by these features, so we could reach a better accuracy of the model.

Insights gained

Through the analysis on the decision tree and the PCA, we reduced the number of required testing features and locked on the accuracy of remaining features. It is found that after removing the features with low weight, the accuracy of our model increased from 76% to 99.9%. This accuracy means the remaining feature data with high weight are the proper features we are looking for to answer our research question. These proper features include the related hormone and the medication that can well control the diabetes.

Future works

Currently, our work has answered the original question. And we believe there can be more updates on our future work. For instance, how can we further increase the accuracy? What are the correlations of the remaining features to diabetes? And how can we further upgrade our current model. These are the future improvements we can consider deeper.

In [ ]: