In [161]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [162]:

```
train = pd.read_csv("training_SyncPatient.csv").dropna()
test = pd.read_csv("test_SyncPatient.csv").dropna()
data = pd.concat([train,test])
```

EDA

In [163]:

```
# cleaning: DOB to Age
data["age"] = (2022 - data["YearOfBirth"]).astype(int)
data = data.drop("YearOfBirth",axis=1)
data = data.reset_index().drop("index",axis=1)
data
```

Out[163]:

	PatientGuid	DMIndicator	Gender	State	PracticeGuid	age
0	FB6EFC3D-1A20-4497-9CBD-00027CC5D220	0.0	M	SD	7BF4DAD8-5F67-4985-B911-20C9E89A3737	93
1	C6746626-6783-4650-A58F-00065649139A	0.0	F	TX	E7101967-2FF1-4B0F-8129-B0B429D1D15C	37
2	E05C6E8F-779F-4594-A388-000C635AE4D3	0.0	F	NJ	FC01A799-1CAF-464F-A86F-8A666AB86F32	38
3	EAEBD216-F847-4355-87B2-000D942E08F0	0.0	M	OH	EEBC95EF-79BE-4542-892E-98D3166BAB20	63
4	C7F10A80-4934-42D2-8540-000FBEBA75C8	0.0	F	FL	677BA32E-B4C4-48F2-86E4-08C42B135401	32
...	...	...	...	...	...	...
9943	96C0A4E6-1E3E-497E-9C4E-FFEC0E25AD3A	NaN	F	TX	E7101967-2FF1-4B0F-8129-B0B429D1D15C	44
9944	5845977A-3014-4301-92B3-FFF0A2EBBAD2	NaN	F	WA	EADEC07A-9901-411F-BBE3-04376029E1E8	36
9945	F948403A-ABE6-496D-B37D-FFF9A9D79767	NaN	F	CA	57B6F75F-CF0A-4225-BAD0-8222A7D4B489	67
9946	F764BC86-0CFA-4661-8D84-FFFA8E2B6080	NaN	F	CA	1A69F223-8409-4FDC-A26C-114677D2D4C3	62
9947	A411D8EA-81A2-4A7F-9EF9-FFFD8ECBE91C	NaN	F	NJ	6C808413-0201-4850-B906-5D2A8433A82D	88

9948 rows × 6 columns

In [164]:

```
# use training data for EDA
Patient = data.iloc[train.index,:]
```

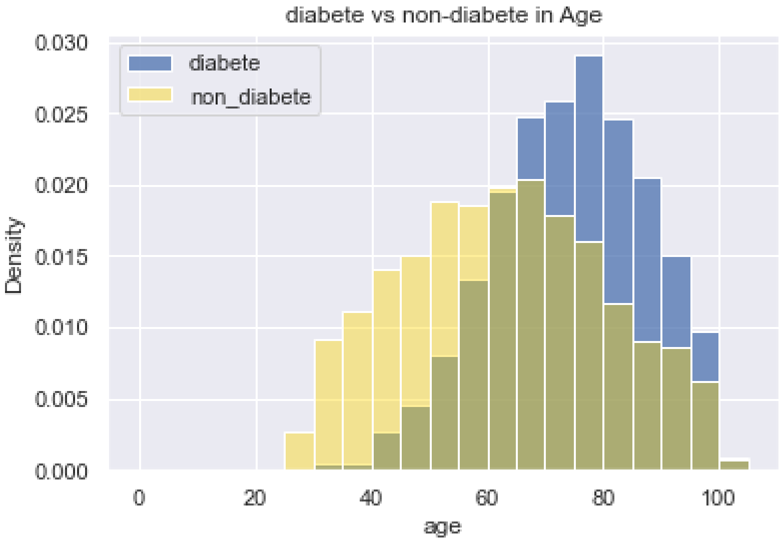
age

In [165]:

```python
diabete = Patient[Patient["DMIndicator"]==1]
non_diabete = Patient[Patient["DMIndicator"]==0]

Patient.loc[Patient["DMIndicator"]==1,"DMIndicator"] = "diabete"
Patient.loc[Patient["DMIndicator"]==0,"DMIndicator"] = "non-diabete"
```

In [191]:

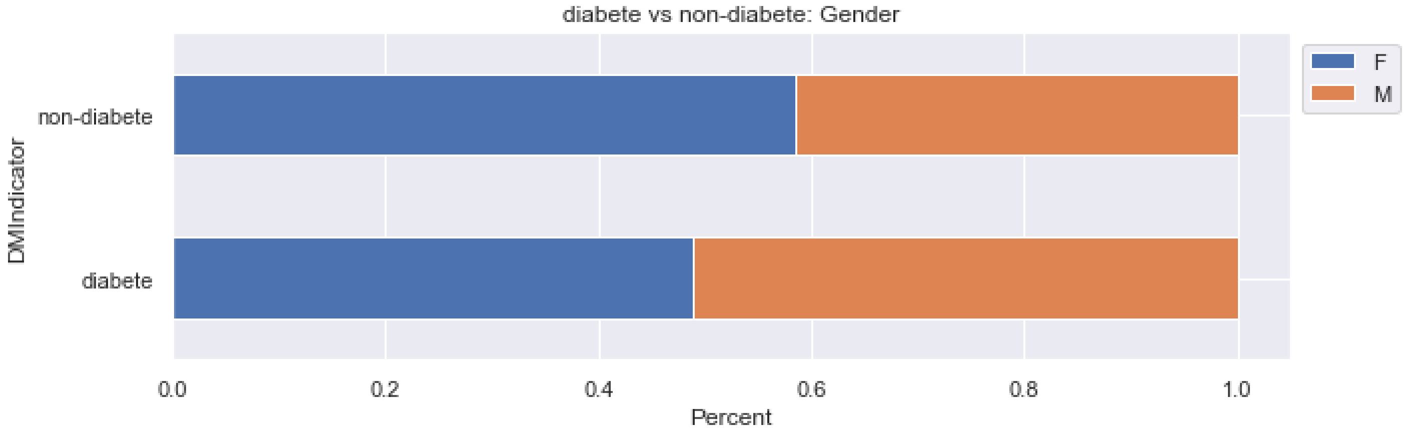
```python
sns.set(rc = {'figure.figsize':(6,4)})
bins = np.arange(0,110,5)
sns.histplot(diabete["age"], bins=bins,stat='density', label='diabete', ec='w');
sns.histplot(non_diabete["age"], bins=bins,stat='density', label='non_diabete', color='gold', alpha = 0.4, ec='w');
plt.title("diabete vs non-diabete in Age")
plt.legend();
```



gender

In [167]:

```
sns.set(rc = {'figure.figsize':(10,3)})
table = pd.crosstab(Patient.DMIndicator,Patient.Gender);
table.div(table.sum(1).astype(float), axis=0).plot(kind='barh',stacked=True);
plt.title('diabete vs non-diabete: Gender');
plt.legend(loc='best',bbox_to_anchor=(1, 1))
plt.xlabel('Percent');
plt.ylabel('DMIndicator');
```



smoke condition

In [168]:

```
SmokingStatus = pd.read_csv("training_SyncPatientSmokingStatus.csv")
SmokingStatus_desc = pd.read_csv("SyncSmokingStatus.csv")
```

In [169]:

```
# merge
Patient = Patient.merge(SmokingStatus,how="left",left_on="PatientGuid",right_on="PatientGuid")
Patient = Patient.merge(SmokingStatus_desc,how="left",left_on="SmokingStatusGuid",right_on="SmokingStatusGuid")

# check duplicates
print("duplicates before:",sum(Patient["PatientGuid"].duplicated()))

# keeping the most recent effective record
Patient = Patient.sort_values(by="EffectiveYear",ascending=False)
Patient = Patient.drop_duplicates("PatientGuid")
Patient = Patient.sort_index().reset_index().drop("index",axis=1)

print("duplicates after:",sum(Patient["PatientGuid"].duplicated()))
```

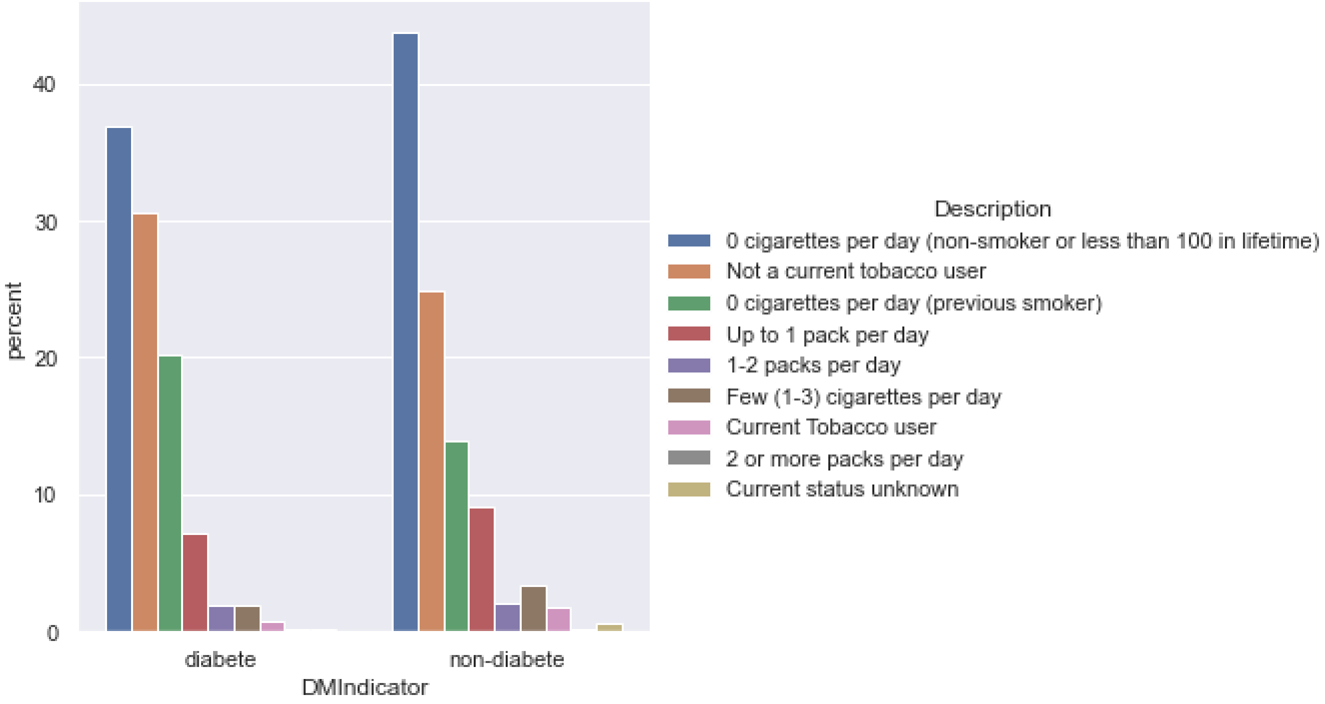
```
duplicates before: 325
duplicates after: 0
```

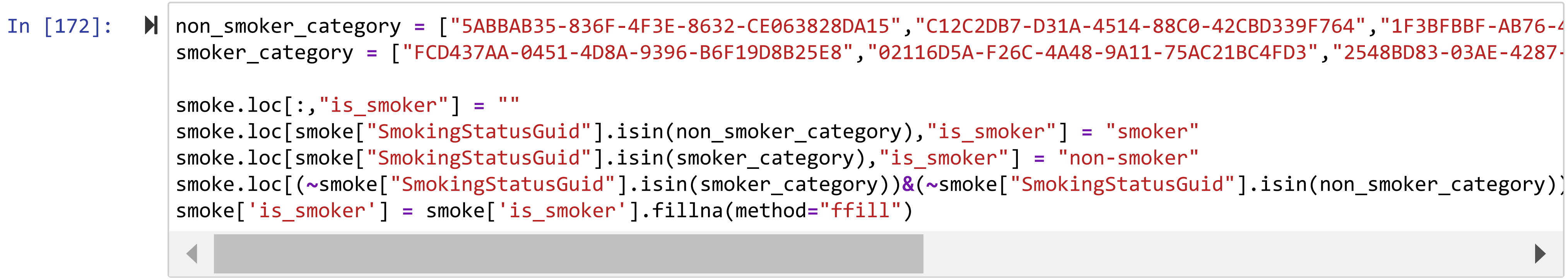
In [170]:

```
smoke = Patient[~Patient["SmokingStatusGuid"].isnull()]
```

In [171]:

```
sns.set(rc = {'figure.figsize':(10,5)})
p = smoke.groupby('DMIndicator')['Description'].value_counts(normalize=True).mul(100).rename('percent').reset_index()
.pipe((sns.catplot,'data'), x='DMIndicator',y='percent',hue='Description',kind='bar');

# overall, about 85% of patients in the train dataset are non-smoker, no matter if they have diabete

# the category definition is not very clear, such as no big diff between 0 cig per day(prev smoker) and not a current
# so let's just break into two category: present smoker and non-smoker
```



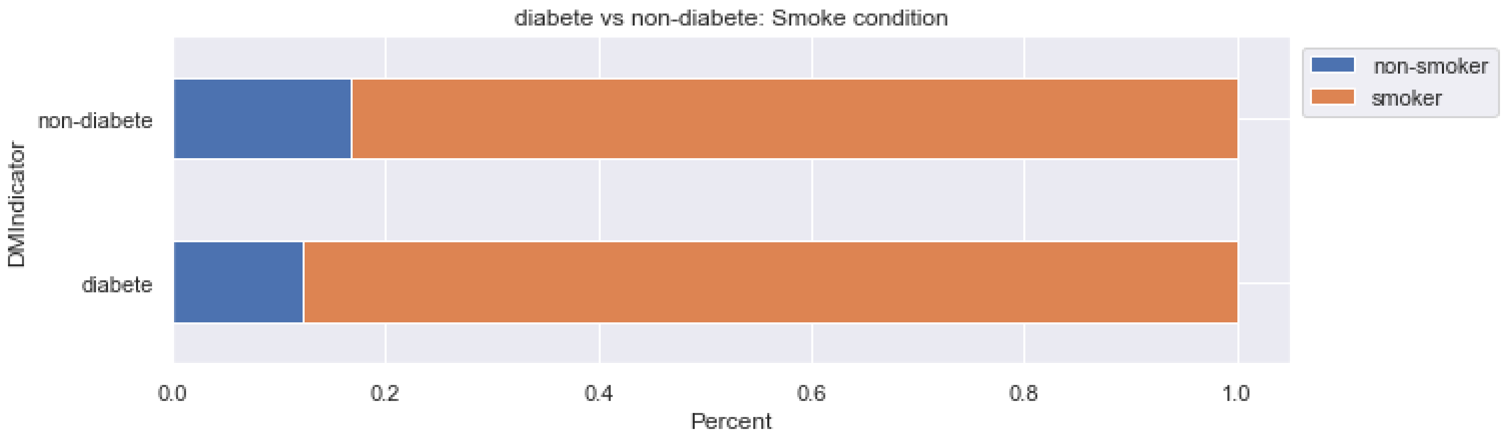


In [172]:

```
non_smoker_category = ["5ABBAB35-836F-4F3E-8632-CE063828DA15","C12C2DB7-D31A-4514-88C0-42CBD339F764","1F3BFBBF-AB76-4
smoker_category = ["FCD437AA-0451-4D8A-9396-B6F19D8B25E8","02116D5A-F26C-4A48-9A11-75AC21BC4FD3","2548BD83-03AE-4287-

smoke.loc[:,"is_smoker"] = ""
smoke.loc[smoke["SmokingStatusGuid"].isin(non_smoker_category),"is_smoker"] = "smoker"
smoke.loc[smoke["SmokingStatusGuid"].isin(smoker_category),"is_smoker"] = "non-smoker"
smoke.loc[(~smoke["SmokingStatusGuid"].isin(smoker_category))&(~smoke["SmokingStatusGuid"].isin(non_smoker_category))
smoke['is_smoker'] = smoke['is_smoker'].fillna(method="ffill")
```

In [198]:

```
sns.set(rc = {'figure.figsize':(10,3)})
table = pd.crosstab(smoke.DMIndicator,smoke.is_smoker);
table.div(table.sum(1).astype(float), axis=0).plot(kind='barh',stacked=True);
plt.title('diabete vs non-diabete: Smoke condition');
plt.legend(loc='best',bbox_to_anchor=(1, 1))
plt.xlabel('Percent');
plt.ylabel('DMIndicator');
# by breaking to binary category:smoker or non-smoker, there's little difference between diabete and non-diabete pati
```



transcript data

In [174]:

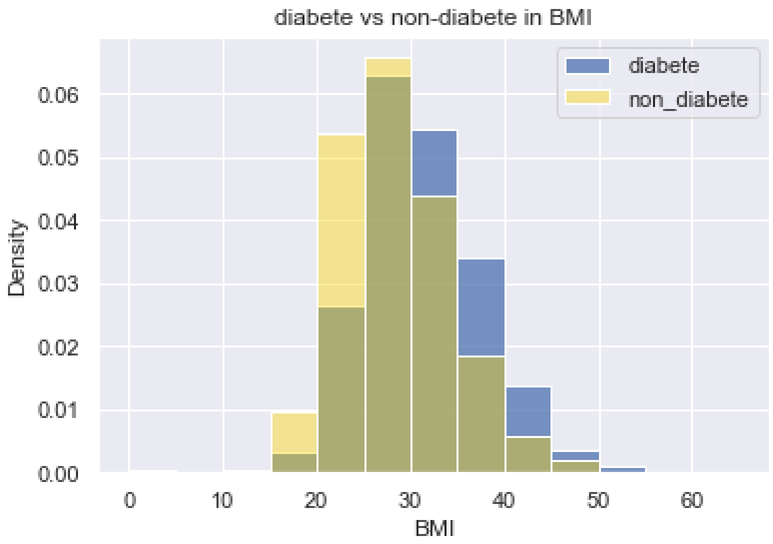
```
transcript = pd.read_csv("training_SyncTranscript.csv")
# drop null or 0 values
transcript = transcript[transcript["BMI"] != 0].drop(["HeartRate","PhysicianSpecialty"],axis=1)

# drop duplicates: keeping the most recent effective record
transcript = transcript.sort_values(by="VisitYear",ascending=False)
transcript = transcript.drop_duplicates("PatientGuid")
transcript = transcript.sort_index().reset_index().drop("index",axis=1)

# impute null with mean
transcript = transcript.fillna(transcript.mean())
```

In [175]:

```
data = pd.merge(data,transcript,how="left",on="PatientGuid")
```

In [196]:

```
diabete = data[data["DMIndicator"]==1]
non_diabete = data[data["DMIndicator"]==0]

sns.set(rc = {'figure.figsize':(6,4)})
bins = np.arange(0,70,5)
sns.histplot(diabete["BMI"], bins=bins,stat='density', label='diabete', ec='w');
sns.histplot(non_diabete["BMI"], bins=bins,stat='density', label='non_diabete', color='gold', alpha  = 0.4,ec='w');
plt.title("diabete vs non-diabete in BMI")
plt.legend();

## we can see that BMI of diabete patients is higher
```



In [195]:

```
non_diabete
```

Out[195]:

	PatientGuid	DMIndicator	Gender	State	PracticeGuid	age	TranscriptGuid	VisitYear	Height	Weight	BMI	SystolicBP	Diastoli
0	FB6EFC3D-1A20-4497-9CBD-00027CC5D220	0.0	M	SD	7BF4DAD8-5F67-4985-B911-20C9E89A3737	93	E5C73D9C-8FC2-4BCD-A705-6A204B40EB8E	2009	69.0	131.0	19.343	142.000000	66.00
1	C6746626-6783-4650-A58F-00065649139A	0.0	F	TX	E7101967-2FF1-4B0F-8129-B0B429D1D15C	37	721D128B-F44C-4C10-8037-925674F932EE	2012	66.5	162.0	25.753	112.000000	60.00
2	E05C6E8F-779F-4594-A388-000C635AE4D3	0.0	F	NJ	FC01A799-1CAF-464F-A86F-8A666AB86F32	38	C22E3B6D-9C8B-4CBE-9BCF-EAE31C8E635E	2010	64.0	198.0	33.983	124.000000	86.00
3	EAEBD216-F847-4355-87B2-000D942E08F0	0.0	M	OH	EEBC95EF-79BE-4542-892E-98D3166BAB20	63	4DEF0402-9D22-4789-AA49-7E981C61C111	2012	72.0	244.0	33.089	125.000000	70.00
4	C7F10A80-4934-42D2-8540-000FBEBA75C8	0.0	F	FL	677BA32E-B4C4-48F2-86E4-08C42B135401	32	4E0B4C2C-96F9-4028-ADB9-1346347D03F6	2009	62.0	186.8	34.162	126.554941	76.87
...	...	...	...	...	...	...	...	...	...	...	...	...	
6595	83492E15-745E-4A3E-A05E-A737E5088CEB	0.0	M	CA	DD0D8C67-1F38-4BFE-A54E-191AB0C66FCA	97	344AFD72-AF40-47BC-869C-55CB195BC488	2012	72.0	170.0	23.054	137.000000	82.00
6596	9CAE08C1-F6E4-4B9A-A1D4-A7392CF5159B	0.0	M	CA	7AFFC5D8-05B5-405E-9A9F-8D18190A5FEF	53	329A965B-27B4-4951-A9BC-3F1BE0AF655B	2011	68.5	156.0	23.372	114.000000	84.00
6597	FEF04377-E07A-4389-9493-A749165D8D78	0.0	F	SD	7BF4DAD8-5F67-4985-B911-20C9E89A3737	64	5188F879-98CF-4076-8369-D439EDB28F61	2010	67.0	209.0	32.730	126.000000	86.00

	PatientGuid	DMIndicator	Gender	State	PracticeGuid	age	TranscriptGuid	VisitYear	Height	Weight	BMI	SystolicBP	Diastoli
6598	052D137A-DFB2-4806-9434-A74A54A46E14	0.0	M	FL	6373C626-559A-40B5-9936-AEEE8B4CAB5E	66	035A0F44-1F11-4A9A-B001-4A729710FF76	2012	69.0	234.0	34.552	144.000000	90.00
6599	269A2938-916A-495C-B68A-A74B723F83E4	0.0	M	FL	BD7ECDCC-4EBE-4042-A51F-A9E85DBAA7DD	55	74E93414-9C7B-4D56-B0E5-BCB3DF634B96	2010	72.0	119.0	16.138	124.000000	78.00

5334 rows × 16 columns

diagnosis data (icd9)

In [83]:

```python
def categorize_icd9code(code):
    icd9code = {
        '272': 'Disorders of lipoid metabolism',
        '401': 'Essential hypertension',
        '585': 'Chronic renal failure',
        '715': 'Osteoarthrosis and allied disorders',
        '414': 'Other forms of chronic ischemic heart disease',
        '782': 'Symptoms involving skin and other integumentary tissue',
        '443': 'Other peripheral vascular disease',
        '428': 'Heart failure',
        '285': 'Other and unspecified anemias',
        '781': 'Symptoms involving nervous and musculoskeletal systems',
        '276': 'Disorders of fluid, electrolyte, and acid-base balance',
        '791': 'Nonspecific findings on examination of urine',
        'v03+v04': 'prophylactic vaccination and inoculation',
        '600': 'Hyperplasia of prostate',
        '715': 'certain conditions originating in the perinatal period',
        '716': 'Other and unspecified arthropathies',
        '496': 'Chronic airway obstruction, not elsewhere classified',
        '438': 'Late effects of cerebrovascular disease',
        '461': 'Acute sinusitis',
        '706': 'Diseases of sebaceous glands',
        '314': 'Hyperkinetic syndrome of childhood',
        '300':'Neurotic disorders'
    }
    code = code.split('.')[0]
    if ('V03' in code.upper()) or ('V04' in code.upper()): return 'prophylactic vaccination and inoculation'
    elif ('E' in code.upper()) or ('V' in code.upper()): return 'Other Supplementary'
    elif int(code) == 272: return 'Disorders of lipoid metabolism'
    elif int(code) == 401: return 'Essential hypertension'
    elif int(code) == 585: return 'Chronic renal failure'
    elif int(code) == 715: return 'Osteoarthrosis and allied disorders'
    elif int(code) == 414: return 'Other forms of chronic ischemic heart disease'
    elif int(code) == 782: return 'Symptoms involving skin and other integumentary tissue'
    elif int(code) == 443: return 'Other peripheral vascular disease'
    elif int(code) == 428: return 'Heart failure'
    elif int(code) == 285: return 'Other and unspecified anemias'
    elif int(code) == 781: return 'Symptoms involving nervous and musculoskeletal systems'
    elif int(code) == 276: return 'Disorders of fluid, electrolyte, and acid-base balance'
    elif int(code) == 791: return 'Nonspecific findings on examination of urine'
    elif int(code) == 600: return 'Hyperplasia of prostate'
    elif int(code) == 715: return 'certain conditions originating in the perinatal period'
    elif int(code) == 716: return 'Other and unspecified arthropathies'
    elif int(code) == 496: return 'Chronic airway obstruction, not elsewhere classified'
    elif int(code) == 438: return 'Late effects of cerebrovascular disease'
    elif int(code) == 461: return 'Acute sinusitis'
    elif int(code) == 706: return 'Diseases of sebaceous glands'
    elif int(code) == 314: return 'Hyperkinetic syndrome of childhood'
    elif int(code) == 300: return 'Neurotic disorders'
    else: return 'Other Comorbidity categories'
```

In [84]:

```
Diagnosis = pd.read_csv("training_SyncDiagnosis.csv")
Diagnosis['ICD9CodeCategory'] = Diagnosis.ICD9Code.apply(lambda x:categorize_icd9code(x))
```

In [85]:

```
# aggregate and get dummies of ICD9CodeCategory
diagnosis_agg = Diagnosis[['ICD9CodeCategory']]
diagnosis_agg.index = Diagnosis.PatientGuid
diagnosis_agg = pd.get_dummies(diagnosis_agg,prefix='',prefix_sep='').reset_index().groupby('PatientGuid').sum()
data = data.set_index("PatientGuid").join(diagnosis_agg).reset_index()
```

```
In [86]: data.columns
```

```
Out[86]: Index(['PatientGuid', 'DMIndicator', 'Gender', 'State', 'PracticeGuid', 'age',
                'TranscriptGuid', 'VisitYear', 'Height', 'Weight', 'BMI', 'SystolicBP',
                'DiastolicBP', 'RespiratoryRate', 'Temperature', 'UserGuid',
                'Acute sinusitis',
                'Chronic airway obstruction, not elsewhere classified',
                'Chronic renal failure', 'Diseases of sebaceous glands',
                'Disorders of fluid, electrolyte, and acid-base balance',
                'Disorders of lipoid metabolism', 'Essential hypertension',
                'Heart failure', 'Hyperkinetic syndrome of childhood',
                'Hyperplasia of prostate', 'Late effects of cerebrovascular disease',
                'Neurotic disorders', 'Nonspecific findings on examination of urine',
                'Osteoarthrosis and allied disorders', 'Other Comorbidity categories',
                'Other Supplementary', 'Other and unspecified anemias',
                'Other and unspecified arthropathies',
                'Other forms of chronic ischemic heart disease',
                'Other peripheral vascular disease',
                'Symptoms involving nervous and musculoskeletal systems',
                'Symptoms involving skin and other integumentary tissue',
                'prophylactic vaccination and inoculation'],
               dtype='object')
```

```
In [ ]:
```

```
In [ ]:
```

In [49]:

```
'''
# I think we need to take state population into consideration
# but not sure if should do some adjustment for onehot encoding of state

pop = pd.read_csv("pop_2012.csv")
pat_by_state = Patient.groupby("State").agg(len).reset_index()
pat_pop_merged = pat_by_state.merge(pop,how="inner",left_on="State",right_on="Code")[["State_x","PatientGuid","POP_20
pat_pop_merged.loc[:,"patient density"] = pat_pop_merged["PatientGuid"]/pat_pop_merged["POP_2012"]
pat_pop_merged = pat_pop_merged.sort_values("patient density",ascending=False)

for i in range(len(onehot)):
    for j in range(len(pat_pop_merged)):
        if onehot.loc[i,"State_"+str(pat_pop_merged.loc[j,"State_x"])] == 1:
            onehot.loc[i,"State_"+str(pat_pop_merged.loc[j,"State_x"])] = pat_pop_merged.loc[j,"patient density"]

'''
```

Out[49]:

```
'\n# I think we need to take state population into consideration\n# but not sure if should do some adjustment for o
nehot encoding of state \n\npop = pd.read_csv("pop_2012.csv")\npat_by_state = Patient.groupby("State").agg(len).res
et_index()\npat_pop_merged = pat_by_state.merge(pop,how="inner",left_on="State",right_on="Code")[["State_x","Patien
tGuid","POP_2012"]]\npat_pop_merged.loc[:,"patient density"] = pat_pop_merged["PatientGuid"]/pat_pop_merged["POP_20
12"]\npat_pop_merged = pat_pop_merged.sort_values("patient density",ascending=False)\n\nfor i in range(len(oneho
t)):\n    for j in range(len(pat_pop_merged)):\n        if onehot.loc[i,"State_"+str(pat_pop_merged.loc[j,"State_
x"])] == 1:\n            onehot.loc[i,"State_"+str(pat_pop_merged.loc[j,"State_x"])] = pat_pop_merged.loc[j,"patien
t density"]\n            \n'
```

Feature Engineering

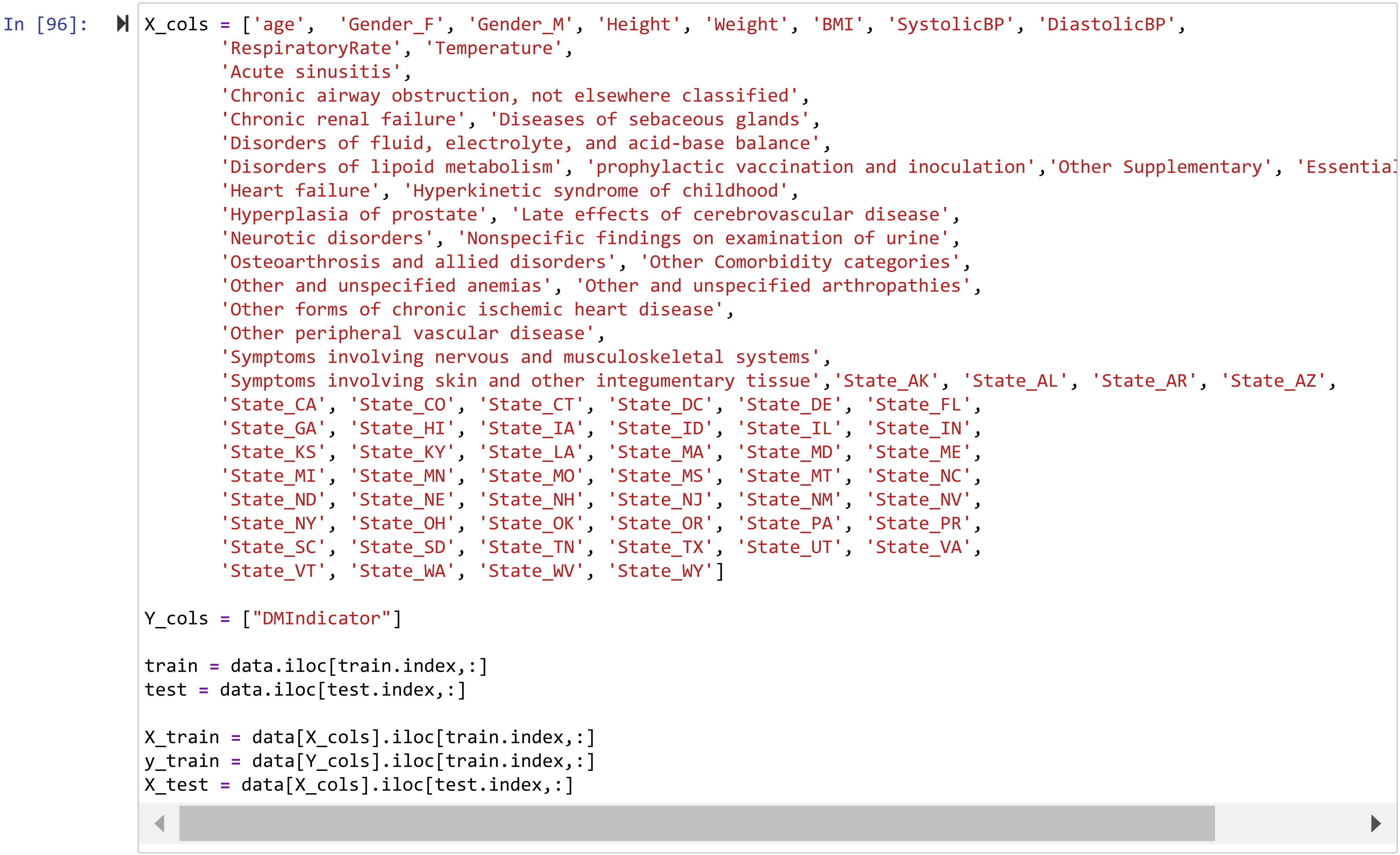
In [87]:

```
# covert DMIndicator back to boolean values
train.loc[train["DMIndicator"]=="diabete","DMIndicator"] = 1
train.loc[train["DMIndicator"]=="non-diabete","DMIndicator"] = 0

train.loc[train["Gender"]=="M","Gender"] = 1
train.loc[train["Gender"]=="F","Gender"] = 0

train["DMIndicator"] = train["DMIndicator"].astype(float)
```

One hot encoding

In [88]:

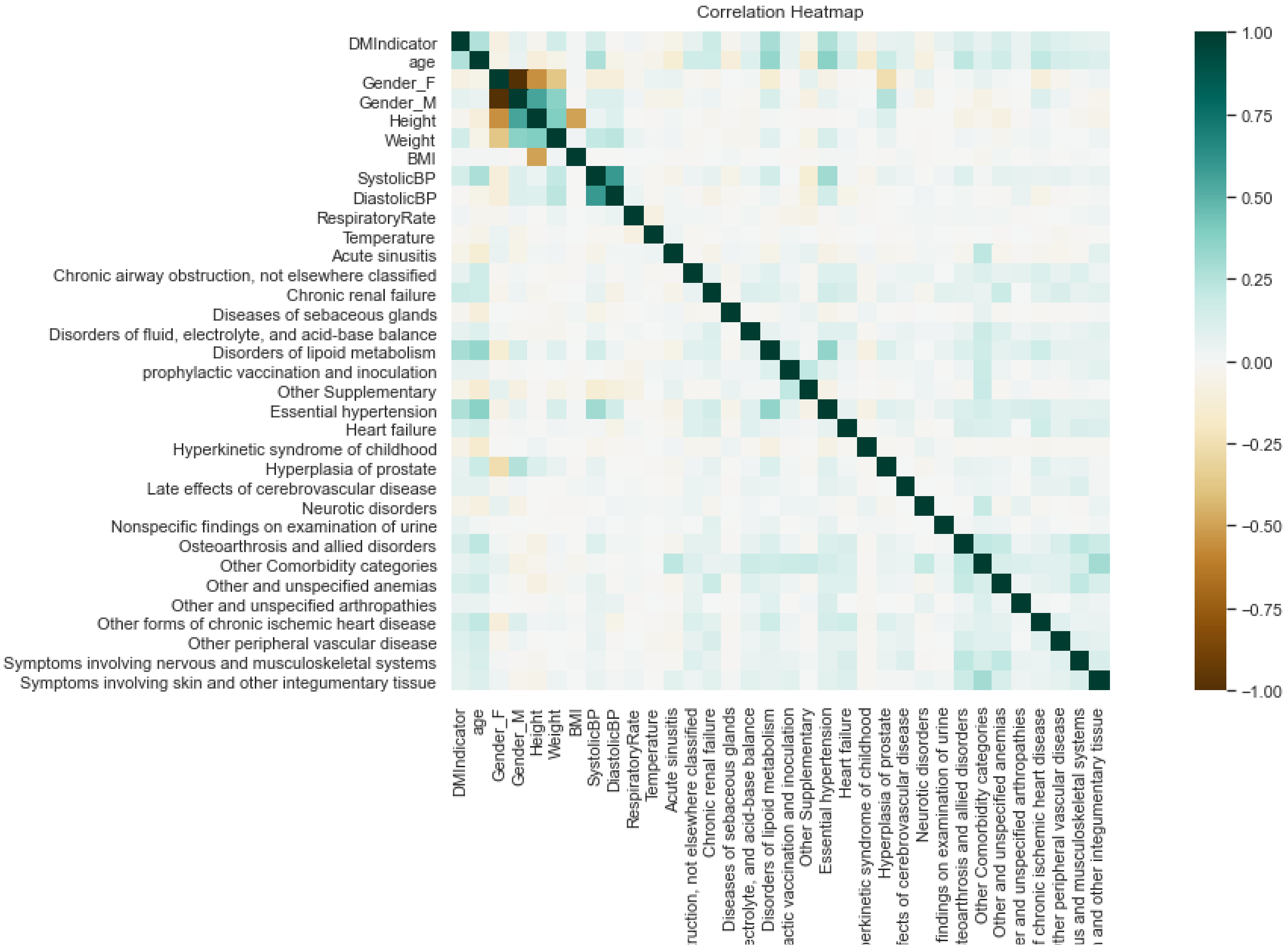
```
onehot = pd.get_dummies(data[["State","Gender"]],columns=["State","Gender"])


data = pd.concat([data,onehot],axis=1)
data = data.drop(["State","Gender"],axis=1)
```

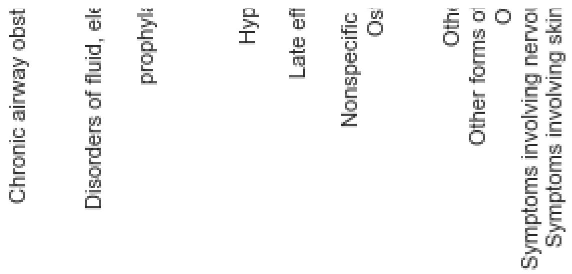


In [96]:

```
X_cols = ['age',  'Gender_F', 'Gender_M', 'Height', 'Weight', 'BMI', 'SystolicBP', 'DiastolicBP',
       'RespiratoryRate', 'Temperature',
       'Acute sinusitis',
       'Chronic airway obstruction, not elsewhere classified',
       'Chronic renal failure', 'Diseases of sebaceous glands',
       'Disorders of fluid, electrolyte, and acid-base balance',
       'Disorders of lipoid metabolism', 'prophylactic vaccination and inoculation','Other Supplementary', 'Essential
       'Heart failure', 'Hyperkinetic syndrome of childhood',
       'Hyperplasia of prostate', 'Late effects of cerebrovascular disease',
       'Neurotic disorders', 'Nonspecific findings on examination of urine',
       'Osteoarthrosis and allied disorders', 'Other Comorbidity categories',
       'Other and unspecified anemias', 'Other and unspecified arthropathies',
       'Other forms of chronic ischemic heart disease',
       'Other peripheral vascular disease',
       'Symptoms involving nervous and musculoskeletal systems',
       'Symptoms involving skin and other integumentary tissue','State_AK', 'State_AL', 'State_AR', 'State_AZ',
       'State_CA', 'State_CO', 'State_CT', 'State_DC', 'State_DE', 'State_FL',
       'State_GA', 'State_HI', 'State_IA', 'State_ID', 'State_IL', 'State_IN',
       'State_KS', 'State_KY', 'State_LA', 'State_MA', 'State_MD', 'State_ME',
       'State_MI', 'State_MN', 'State_MO', 'State_MS', 'State_MT', 'State_NC',
       'State_ND', 'State_NE', 'State_NH', 'State_NJ', 'State_NM', 'State_NV',
       'State_NY', 'State_OH', 'State_OK', 'State_OR', 'State_PA', 'State_PR',
       'State_SC', 'State_SD', 'State_TN', 'State_TX', 'State_UT', 'State_VA',
       'State_VT', 'State_WA', 'State_WV', 'State_WY']

Y_cols = ["DMIndicator"]

train = data.iloc[train.index,:]
test = data.iloc[test.index,:]

X_train = data[X_cols].iloc[train.index,:]
y_train = data[Y_cols].iloc[train.index,:]
X_test = data[X_cols].iloc[test.index,:]
```

```
In [97]:   plt.figure(figsize=(20,8));
           corr_heatmap = sns.heatmap(data[Y_cols+X_cols[:33]].corr(),vmin=-1, vmax=1,cmap='BrBG',square=True);
           corr_heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':12}, pad=10);
```





Oversampling

In [98]:

```
cnt_non_diabete = train[train['DMIndicator'] == 0]["DMIndicator"].count()
train_class_diabete = train[train['DMIndicator'] == 1]
train_class_non_diabete = train[train['DMIndicator'] == 0]

#OverSampling
train_class_diabete_oversample = train_class_diabete.sample(cnt_non_diabete, replace=True)
train_oversampled = pd.concat([train_class_non_diabete, train_class_diabete_oversample], axis=0)

print('Random over-sampling:')
print(train_oversampled['DMIndicator'].value_counts())

X_train_oversampled = train_oversampled[X_cols]
y_train_oversampled = train_oversampled[Y_cols]
```

```
Random over-sampling:
0.0    5334
1.0    5334
Name: DMIndicator, dtype: int64
```

Standardization

In [99]:

```
 from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
X_train_oversampled_std = scalar.fit_transform(X_train_oversampled)
X_test_std = scalar.fit_transform(X_test)
```

Modeling

logistic regression

In [100]:

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(penalty='l2',random_state=42)
logreg.fit(X_train_oversampled_std, y_train_oversampled)
y_train_pred = logreg.predict(X_train_oversampled_std)
y_test_pred = logreg.predict(X_test_std)
```

Cross validation

In [102]:

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(logreg, X_train_oversampled_std, y_train_oversampled, cv=5)
print('Cross-Validation Accuracy Scores', scores)
```

```
Cross-Validation Accuracy Scores [0.74695408 0.75679475 0.75726336 0.73230192 0.74777309]
```

In [103]:

```
from collections import Counter
Counter(y_train_pred)
```

Out[103]:

```
Counter({0.0: 5092, 1.0: 5576})
```

Classification report

In [104]:

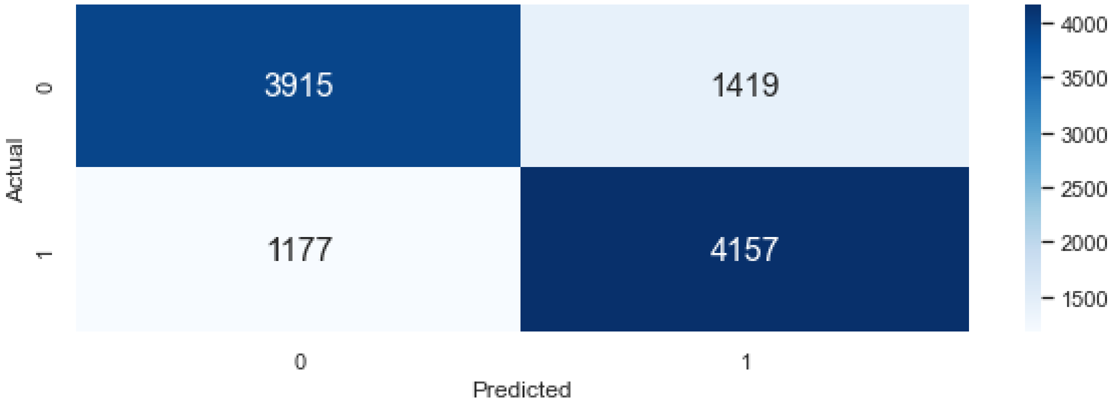
```
from sklearn.metrics import accuracy_score, classification_report
print(classification_report(y_train_oversampled, y_train_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.77      0.73      0.75      5334
         1.0       0.75      0.78      0.76      5334

    accuracy                           0.76     10668
   macro avg       0.76      0.76      0.76     10668
weighted avg       0.76      0.76      0.76     10668
```

Confusion matrix

In [105]:

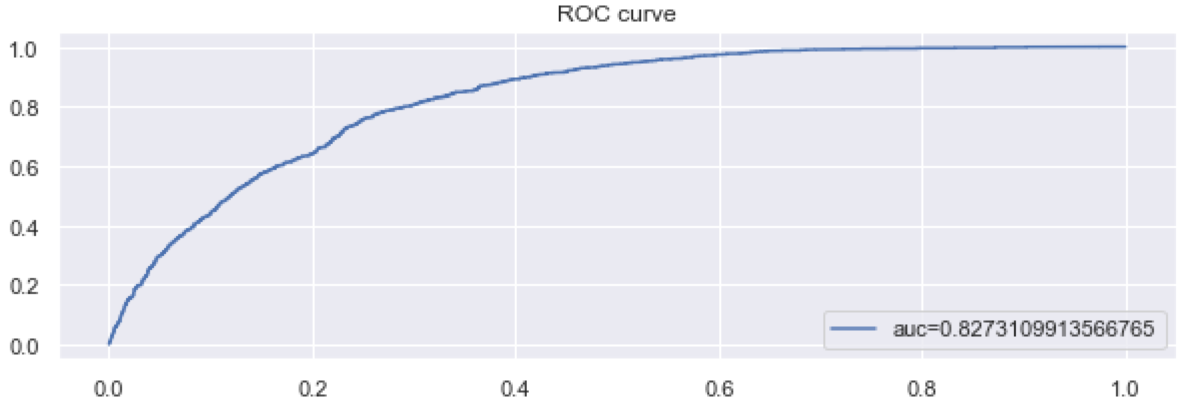
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_train_oversampled, y_train_pred)
sns.heatmap(cm, annot=True, fmt = 'd', cmap = 'Blues', annot_kws = {'size': 16})
plt.xlabel('Predicted')
plt.ylabel('Actual');
```



ROC curve

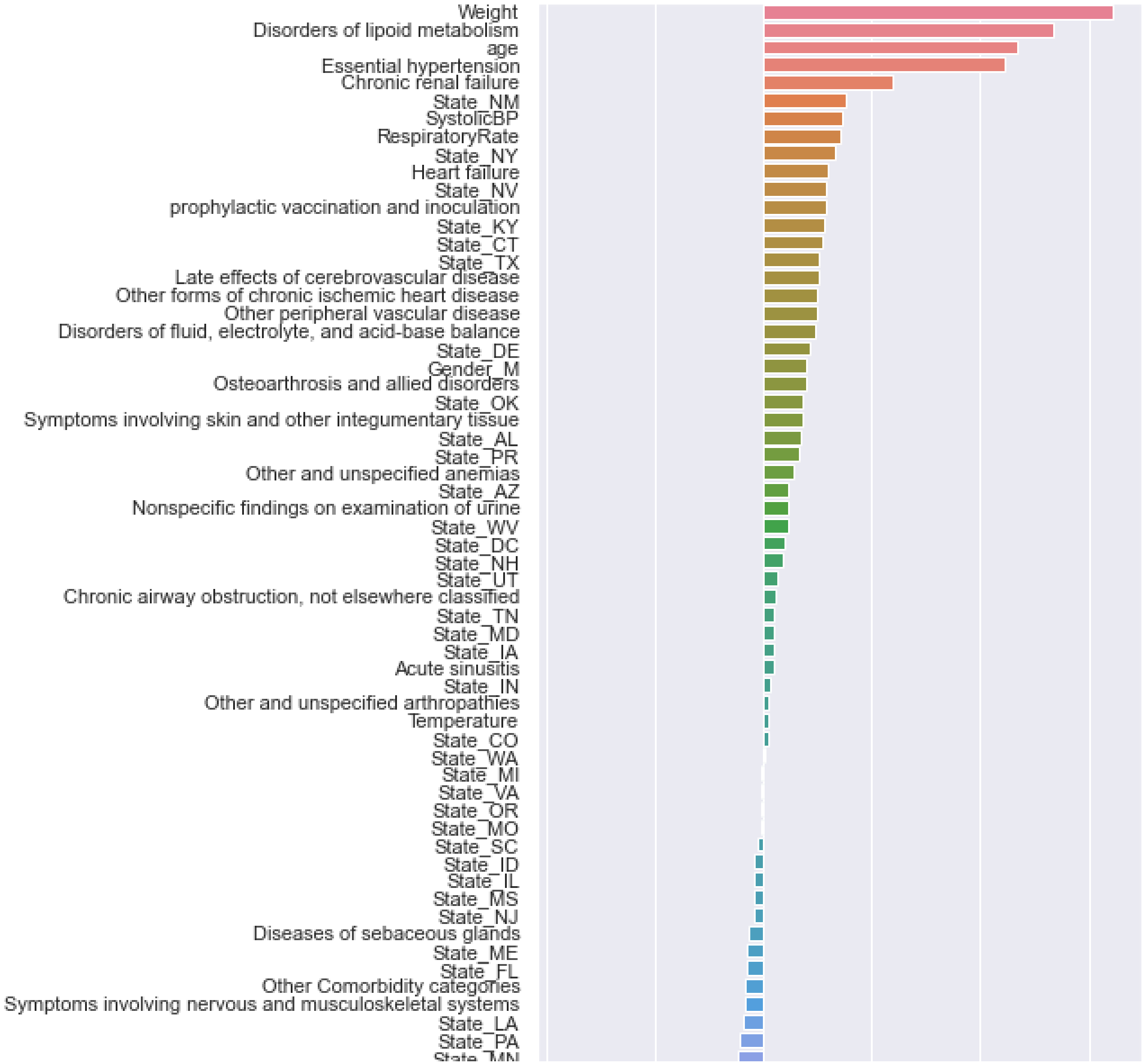
In [106]:

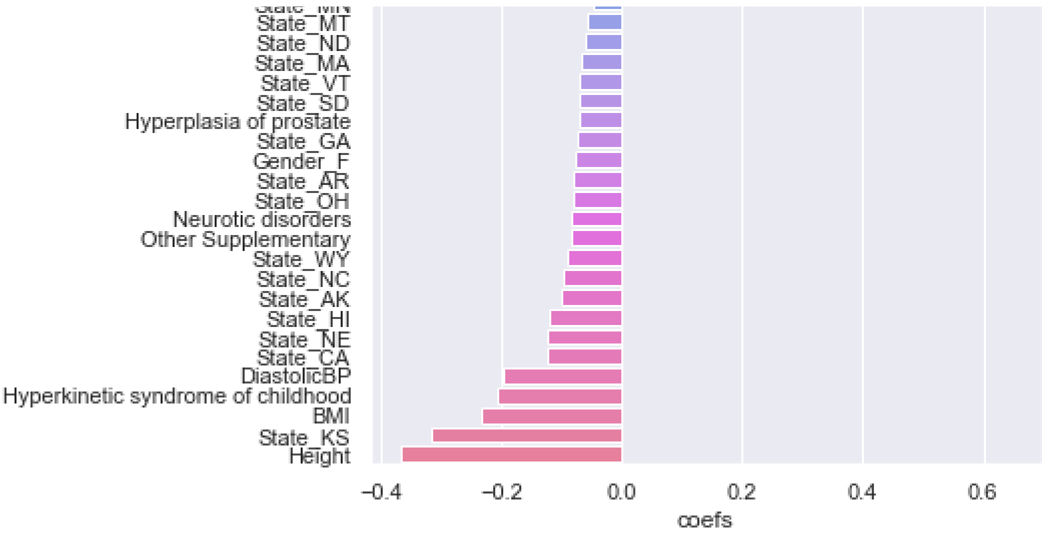
```
# create a ROC curve
from sklearn import metrics

y_pred_proba = logreg.predict_proba(X_train_oversampled_std)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_train_oversampled,y_pred_proba)

auc = metrics.roc_auc_score(y_train_oversampled,y_pred_proba)
plt.plot(fpr,tpr,label = "auc="+str(auc))
plt.legend(loc=4)
plt.title("ROC curve")
plt.show()
```



Feature coefs

```
In [109]:  data_feature = pd.DataFrame({"feature":X_train_oversampled.columns,"coefs":logreg.coef_[0]}).sort_values("coefs",asce
           sns.set(rc = {'figure.figsize':(6,15)})
           sns.barplot(x="coefs",y="feature",data=data_feature,palette="husl");
           plt.ylabel("");
```





Brier score

In [159]:

```
from sklearn.metrics import brier_score_loss
brier_score_loss(y_train_oversampled, y_pred_proba)
```

Out[159]:

```
0.16893541566986592
```

Output forecast DMIndicatorForecast

In [212]:

```
test_SyncPatientForecast = pd.DataFrame({"PracticeGuid":test["PracticeGuid"],"DMIndicatorForecast":y_test_pred})
test_SyncPatientForecast.to_csv("test_SyncPatientForecast.csv")
```

In [213]:

```
test_SyncPatientForecast
```

Out[213]:

	PracticeGuid	DMIndicatorForecast
0	4D27688B-C925-4513-9CF9-8D281ACC6712	1.0
1	44C560D5-82B4-436A-9C72-C090F5377FD0	0.0
2	9891CFAA-9B40-4120-AE20-3A1D86064898	0.0
3	64F84808-F87B-41CF-8E4B-5E0F456359B4	1.0
4	BD209FBC-E92C-4392-A085-1DDA42AF37BA	0.0
...	...	...
3343	E7101967-2FF1-4B0F-8129-B0B429D1D15C	0.0
3344	EADEC07A-9901-411F-BBE3-04376029E1E8	0.0
3345	57B6F75F-CF0A-4225-BAD0-8222A7D4B489	1.0
3346	1A69F223-8409-4FDC-A26C-114677D2D4C3	1.0
3347	6C808413-0201-4850-B906-5D2A8433A82D	0.0

3348 rows × 2 columns