Diabetes Prediction

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import copy
```

In [7]:
```
# Only Using diabetes_data.csv
raw_df = pd.read_csv("diabetes_data.csv")
raw_df.info()
```

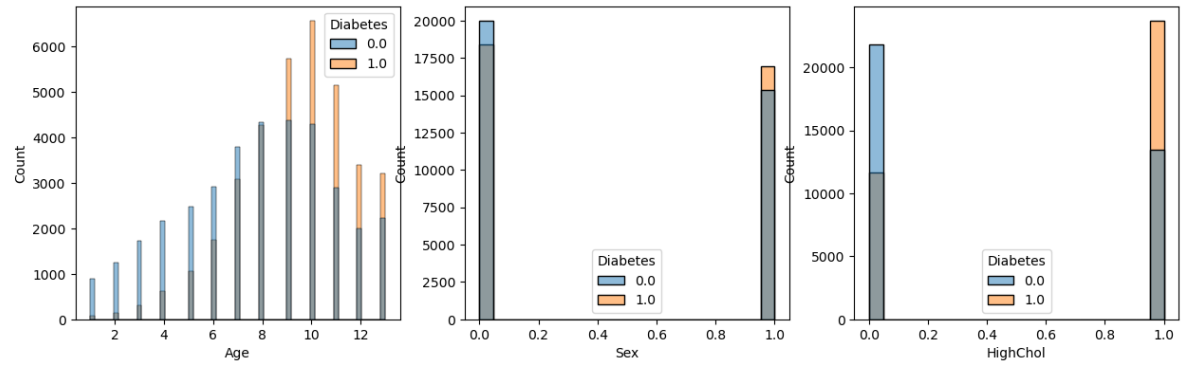
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70692 entries, 0 to 70691
Data columns (total 18 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Age                   70692 non-null  float64
 1   Sex                   70692 non-null  float64
 2   HighChol              70692 non-null  float64
 3   CholCheck             70692 non-null  float64
 4   BMI                   70692 non-null  float64
 5   Smoker                70692 non-null  float64
 6   HeartDiseaseorAttack  70692 non-null  float64
 7   PhysActivity          70692 non-null  float64
 8   Fruits                70692 non-null  float64
 9   Veggies               70692 non-null  float64
 10  HvyAlcoholConsump     70692 non-null  float64
 11  GenHlth               70692 non-null  float64
 12  MentHlth              70692 non-null  float64
 13  PhysHlth              70692 non-null  float64
 14  DiffWalk              70692 non-null  float64
 15  Stroke                70692 non-null  float64
 16  HighBP                70692 non-null  float64
 17  Diabetes              70692 non-null  float64
dtypes: float64(18)
memory usage: 9.7 MB
```

In [8]:
```
raw_df.describe().T
raw_df.isnull().sum() #No missing values
```

Out[8]:
```
Age                     0
Sex                     0
HighChol                0
CholCheck               0
BMI                     0
Smoker                  0
HeartDiseaseorAttack    0
PhysActivity            0
Fruits                  0
Veggies                 0
HvyAlcoholConsump       0
GenHlth                 0
MentHlth                0
PhysHlth                0
DiffWalk                0
Stroke                  0
HighBP                  0
Diabetes                0
dtype: int64
```

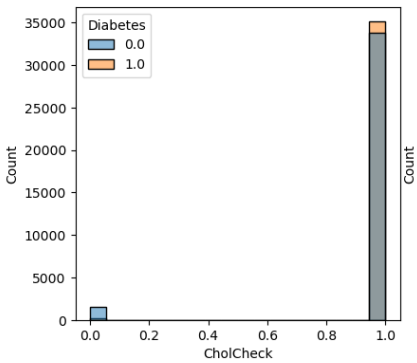
In [9]:
```
raw_df.hist(figsize=(20,20))
plt.show()
```

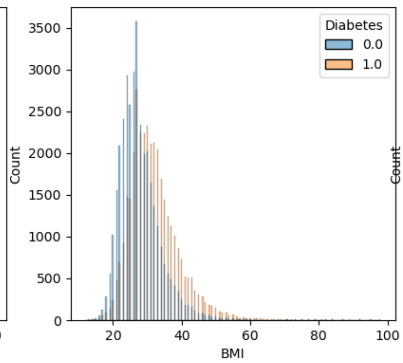
In [10]:
```
fig, ax = plt.subplots(6, 3, figsize=(15, 30))
i = 0
for col in raw_df.columns:
    if col=='MentHlth' or col=='PhysHlth':
        sns.kdeplot(data=raw_df, x=col, hue='Diabetes', ax=ax[i//3, i%3])
    else:
            sns.histplot(data=raw_df, x=col, hue='Diabetes', ax=ax[i//3, i%3])
    i+=1
plt.show()
```

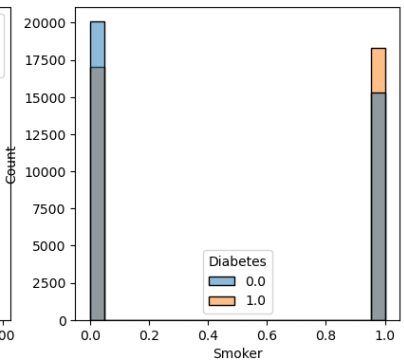

Diabetes
0.0
1.0
Count
Age

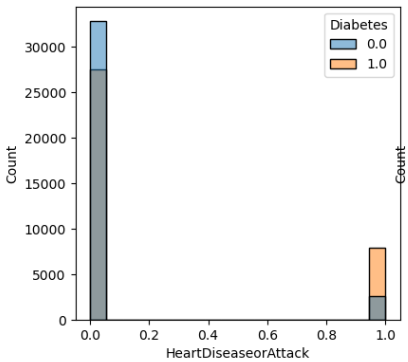
Diabetes
0.0
1.0
Count
Sex

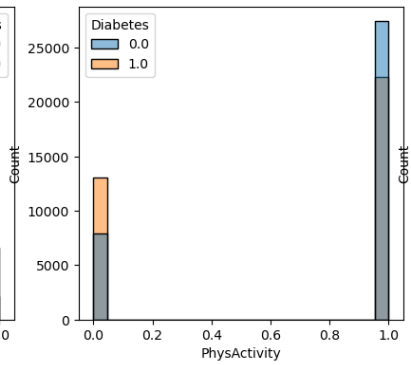
Diabetes
0.0
1.0
Count
HighChol

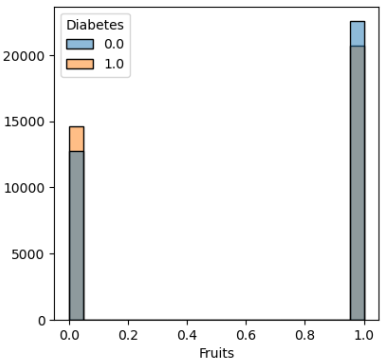

Diabetes
0.0
1.0
Count
CholCheck

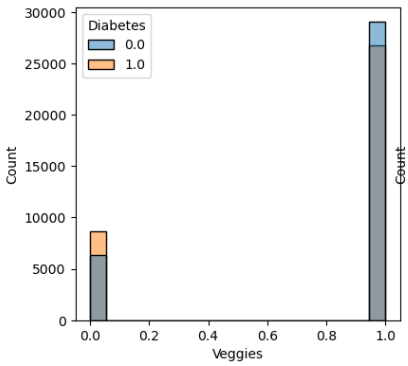

Diabetes
0.0
1.0
Count
BMI

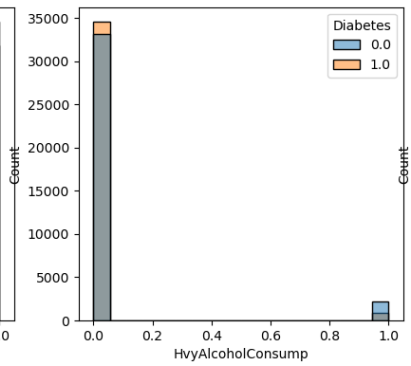

Diabetes
0.0
1.0
Count
Smoker

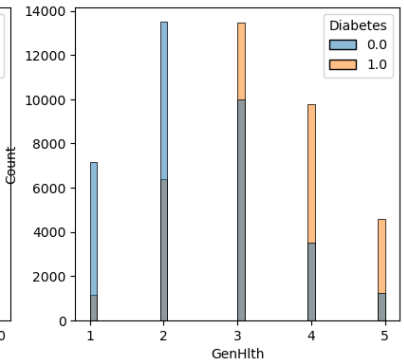

Diabetes
0.0
1.0
Count
HeartDiseaseorAttack

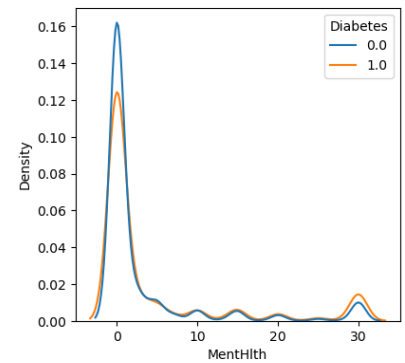

Diabetes
0.0
1.0
Count
PhysActivity

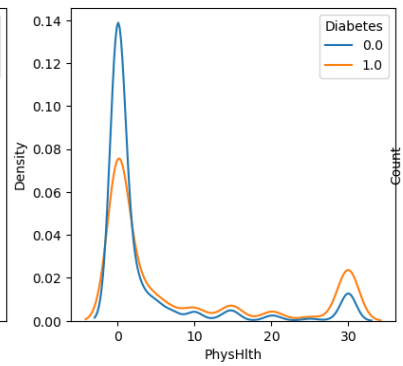

Diabetes
0.0
1.0
Count
Fruits

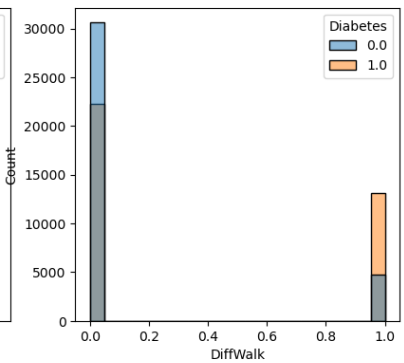

Diabetes
0.0
1.0
Count
Veggies


Diabetes
0.0
1.0
Count
HvyAlcoholConsump


Diabetes
0.0
1.0
Count
GenHlth

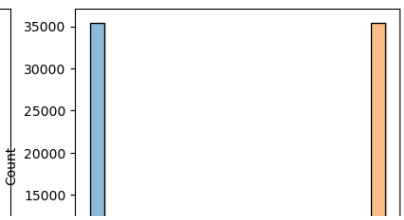

Diabetes
0.0
1.0
Density
MentHlth


Diabetes
0.0
1.0
Density
PhysHlth


Diabetes
0.0
1.0
Count
DiffWalk


Diabetes
0.0
1.0
Count


Count


Count

In [11]:
```
print('- Total Instatnces')
print(len(raw_df.Diabetes), end='\n\n')
print('- Values and Counts of the Class')
print(raw_df.Diabetes.value_counts(), end='\n\n')
print('- Zero R of the Dataset')
print(raw_df.Diabetes.value_counts().max()/len(raw_df.Diabetes))
```

```
- Total Instatnces
70692

- Values and Counts of the Class
0.0    35346
1.0    35346
Name: Diabetes, dtype: int64

- Zero R of the Dataset
0.5
```

In [12]:
```
X_df = raw_df.drop(columns=['Diabetes'])
y_df = raw_df['Diabetes']
```

In [13]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_df, y_df, random_state=42, s
print('                  X         y')
print('-'*34)
print('Train data |', X_train.shape, y_train.shape)
print('Test data  |', X_test.shape, y_test.shape)
```

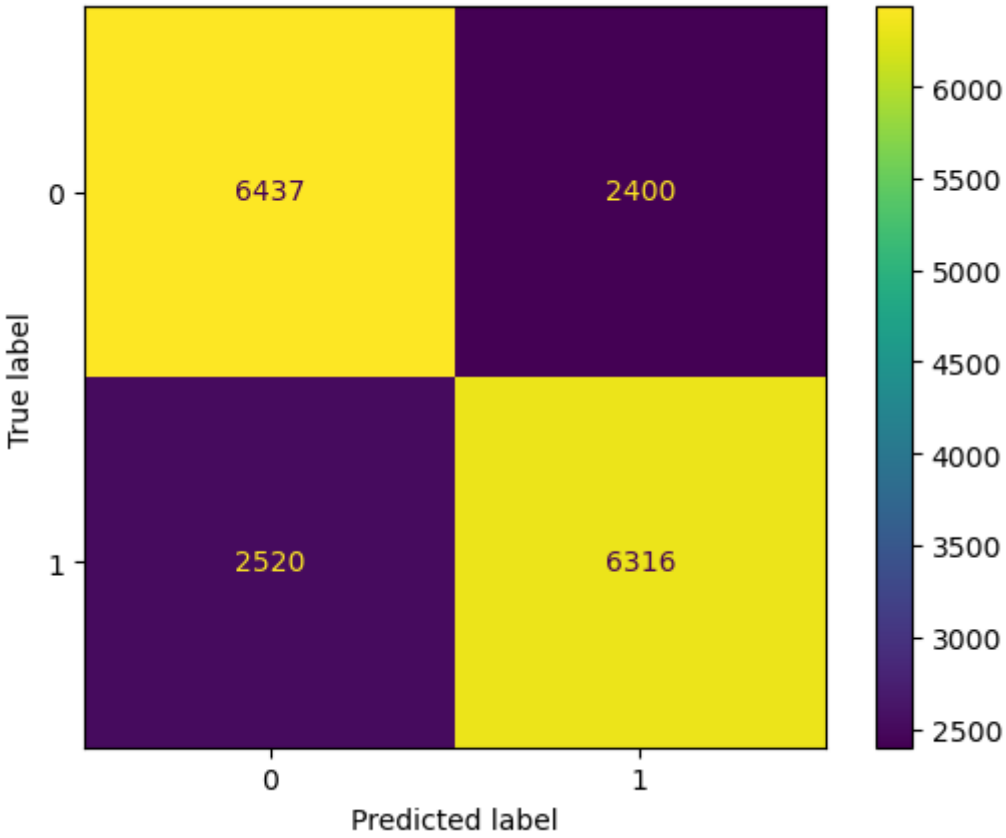
```
                  X         y
----------------------------------
Train data | (53019, 17) (53019,)
Test data  | (17673, 17) (17673,)
```

In [14]:
```
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.metrics import f1_score

naive_dict = {'Gaussian NB':GaussianNB(),
              'DecisionTreeClassifier':DecisionTreeClassifier(random_state=42),
              'LogisticRegression':LogisticRegression(max_iter=1000, random_state=
              'MLPClassifier':MLPClassifier(max_iter=1000, random_state=42),
              'RandomForestClassifier':RandomForestClassifier(random_state=42),
              'AdaBoostClassifier':AdaBoostClassifier(random_state=42)}
```

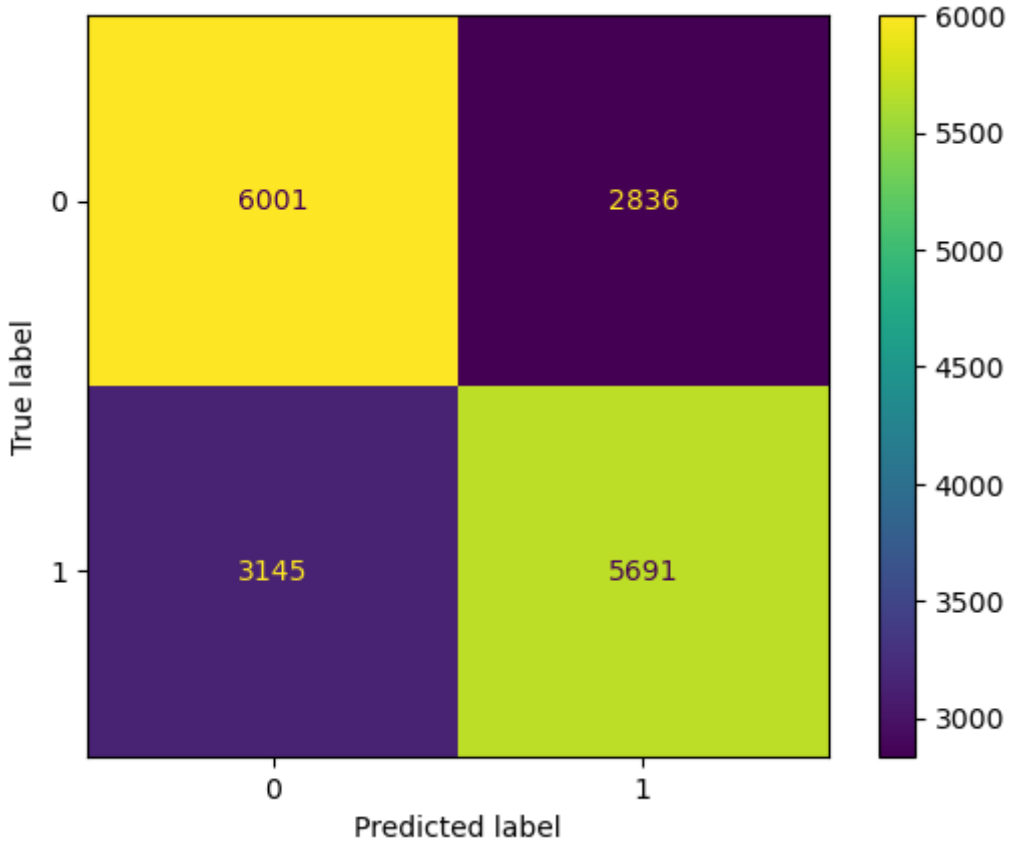
In [15]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, f1_score

for model in naive_dict:
    naive_dict[model].fit(X_train, y_train)
    score = naive_dict[model].score(X_test, y_test)
    pred = naive_dict[model].predict(X_test)
    print('-'*35)
    print('-', model)
    print('Accuracy:', score)
    print('f1 score:', f1_score(y_test, pred))
```

```
    ConfusionMatrixDisplay(confusion_matrix(y_test, pred)).plot()
    plt.show()
```

```
-----------------------------------
- Gaussian NB
Accuracy: 0.7216092344253947
f1 score: 0.7196900638103919
```



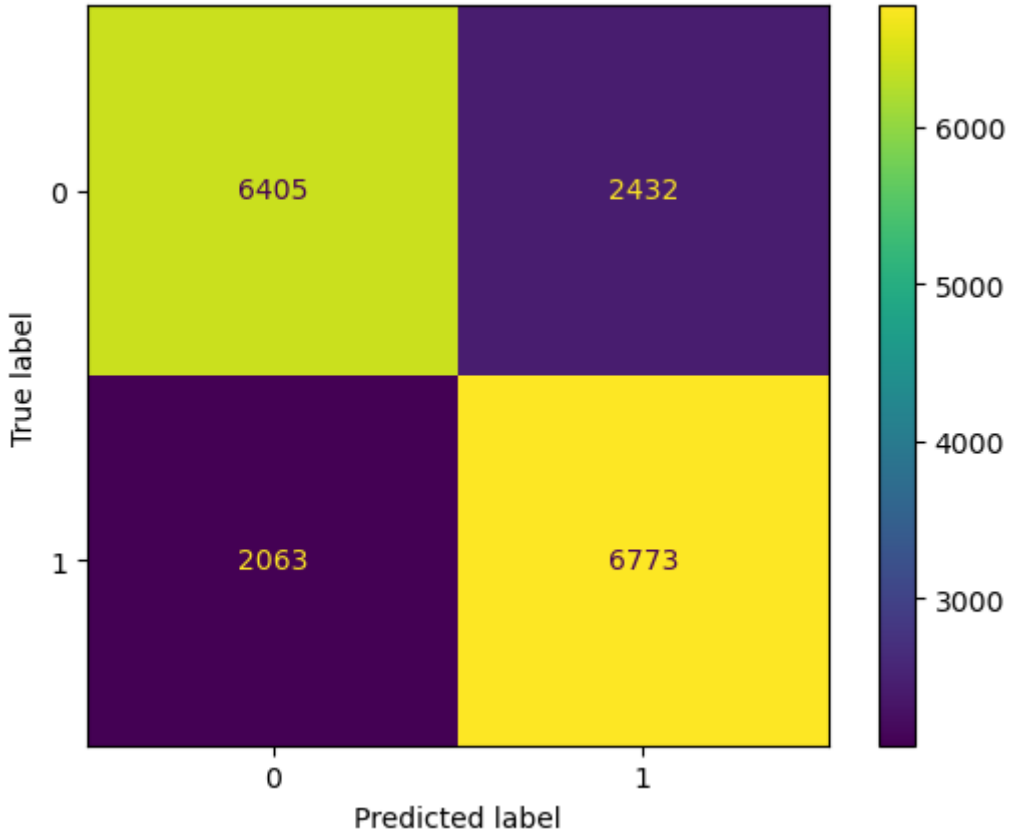

```
-----------------------------------
- DecisionTreeClassifier
Accuracy: 0.6615741526622532
f1 score: 0.6555318781316593
```



- LogisticRegression

Accuracy: 0.745657217224014

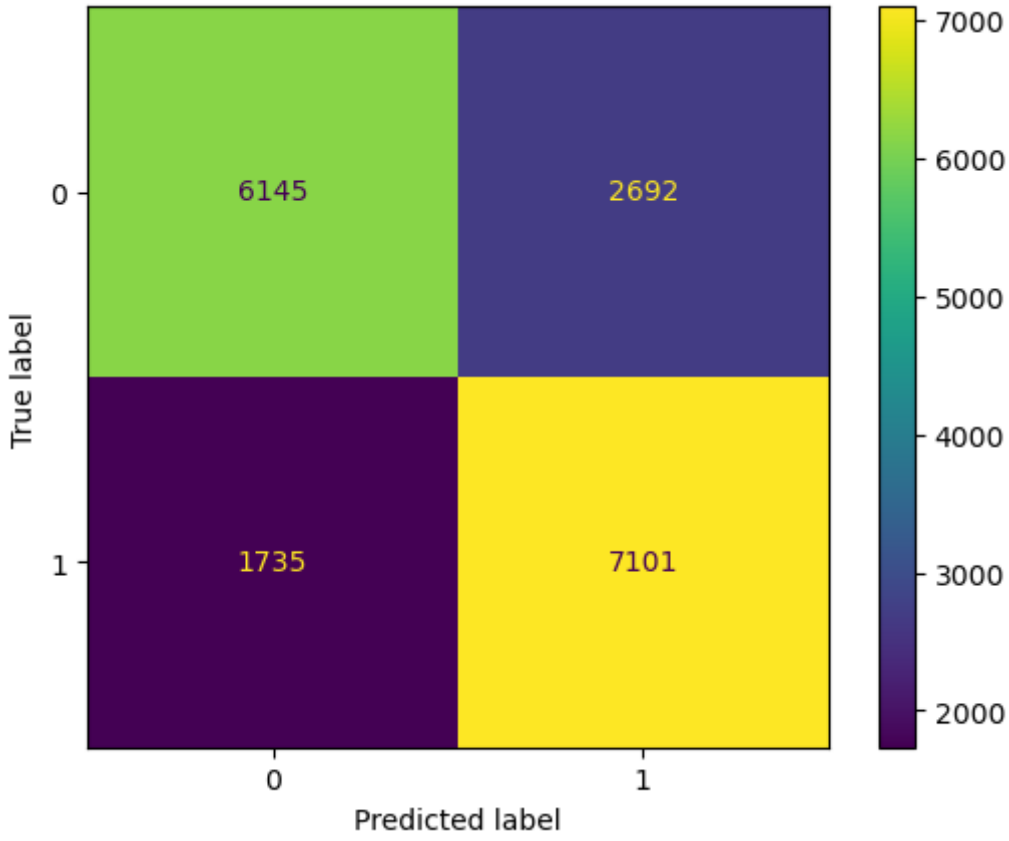
f1 score: 0.750845296823901



- MLPClassifier

Accuracy: 0.7495048944717931

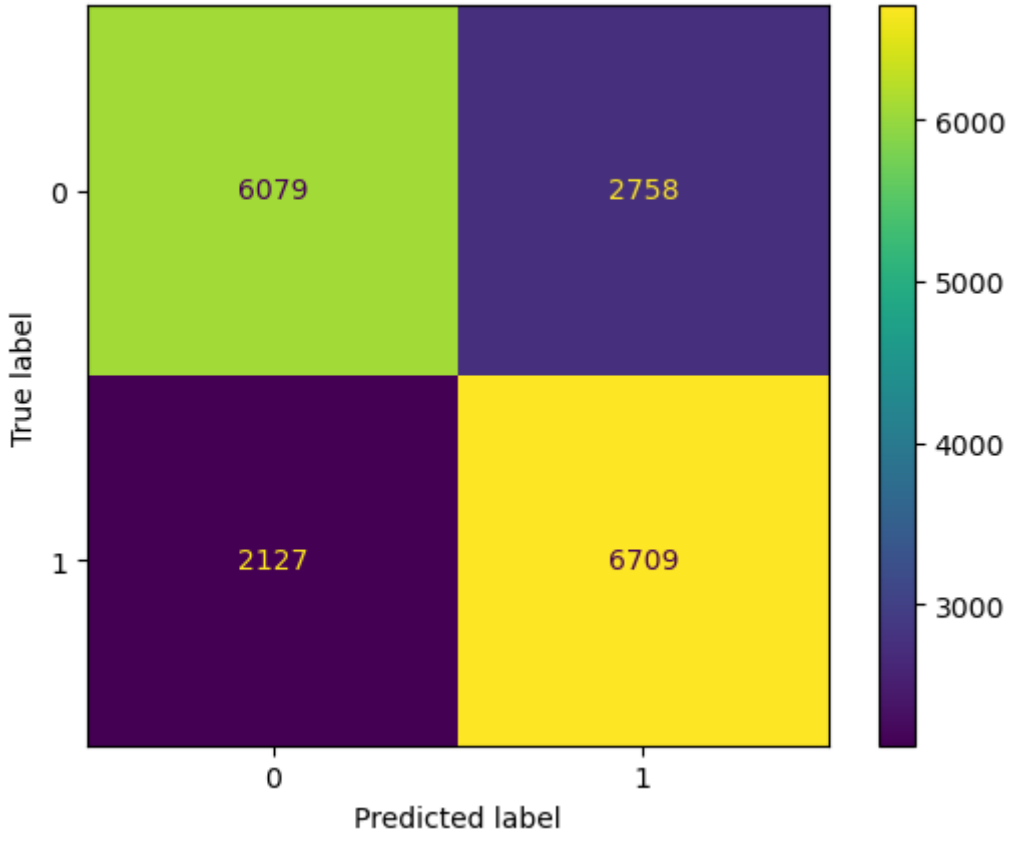
f1 score: 0.762359761661925



- RandomForestClassifier

Accuracy: 0.7235896565382222

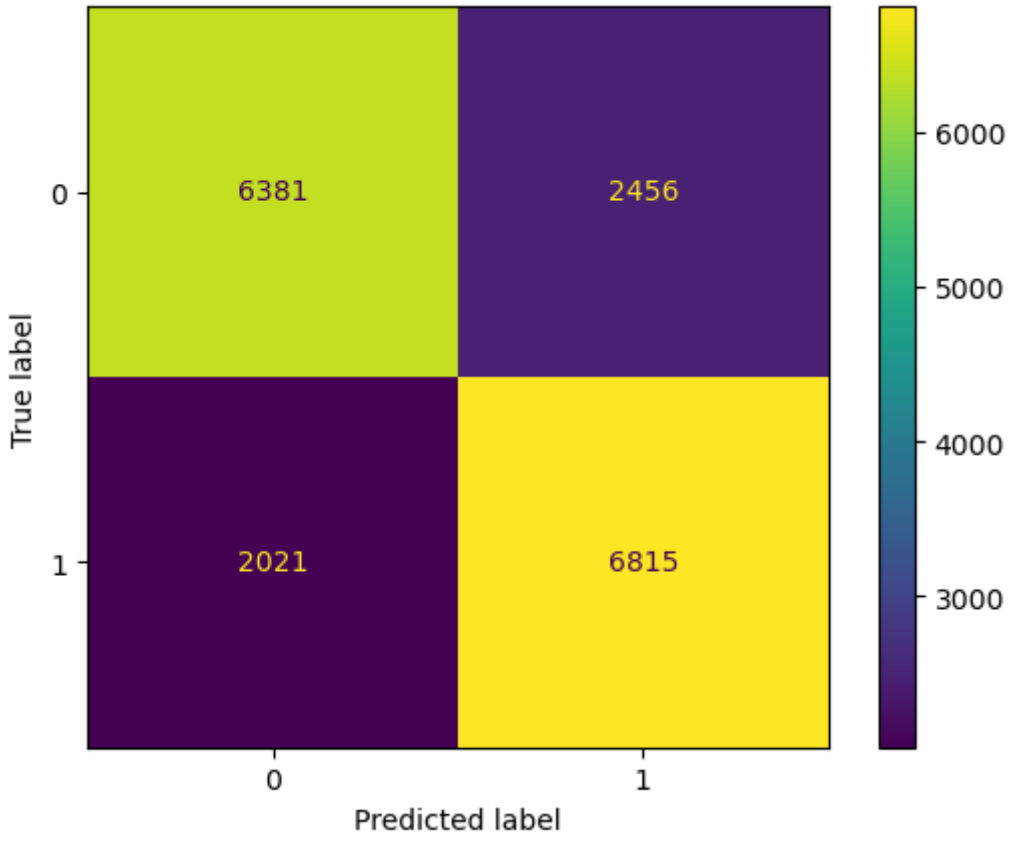
f1 score: 0.7331038627547398



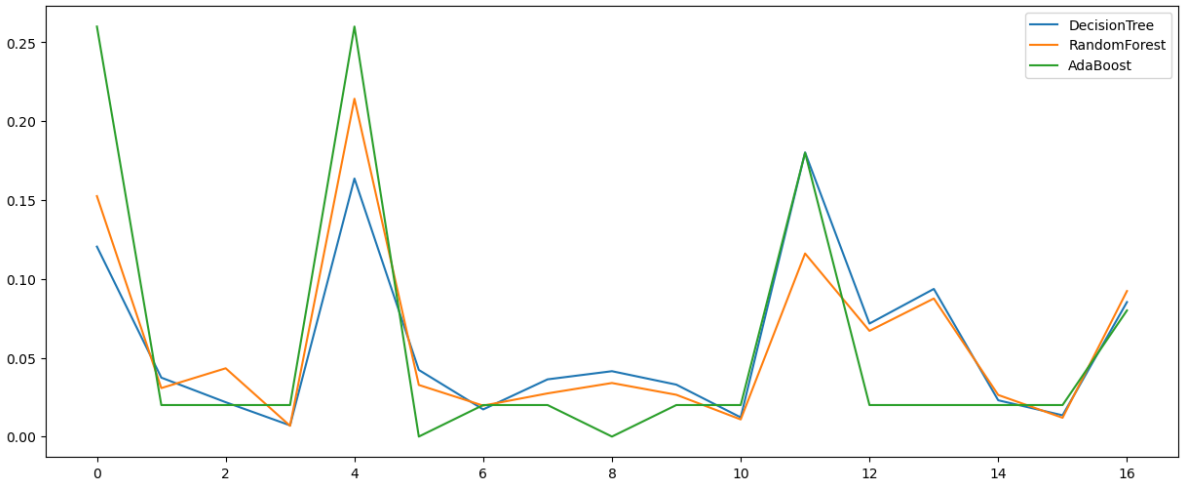
- AdaBoostClassifier

Accuracy: 0.7466757200248967

f1 score: 0.7527475561937372



In [16]:
```
plt.figure(figsize=(15,6))
plt.plot(naive_dict['DecisionTreeClassifier'].feature_importances_, label='Decisio
plt.plot(naive_dict['RandomForestClassifier'].feature_importances_, label='RandomF
plt.plot(naive_dict['AdaBoostClassifier'].feature_importances_, label='AdaBoost')
plt.legend()
plt.show()
```



In [17]:
```
from sklearn.feature_selection import RFE

dt = naive_dict['DecisionTreeClassifier']
rf = naive_dict['RandomForestClassifier']
ab = naive_dict['AdaBoostClassifier']
RFE_dt = RFE(dt, n_features_to_select=7)
RFE_rf = RFE(rf, n_features_to_select=7)
RFE_ab = RFE(ab, n_features_to_select=7)
RFE_dt.fit(X_train, y_train)
RFE_rf.fit(X_train, y_train)
RFE_ab.fit(X_train, y_train)
```

Out[17]:
```
RFE(estimator=AdaBoostClassifier(random_state=42), n_features_to_select=7)
```

In [18]:
```
dt_rank = RFE_dt.ranking_
rf_rank = RFE_rf.ranking_
ab_rank = RFE_ab.ranking_
rank = pd.DataFrame({'DecisionTree':dt_rank,
                     'RandomForest':rf_rank,
                     'AdaBoostClassifier':ab_rank}).set_index(X_df.columns)
rank
```

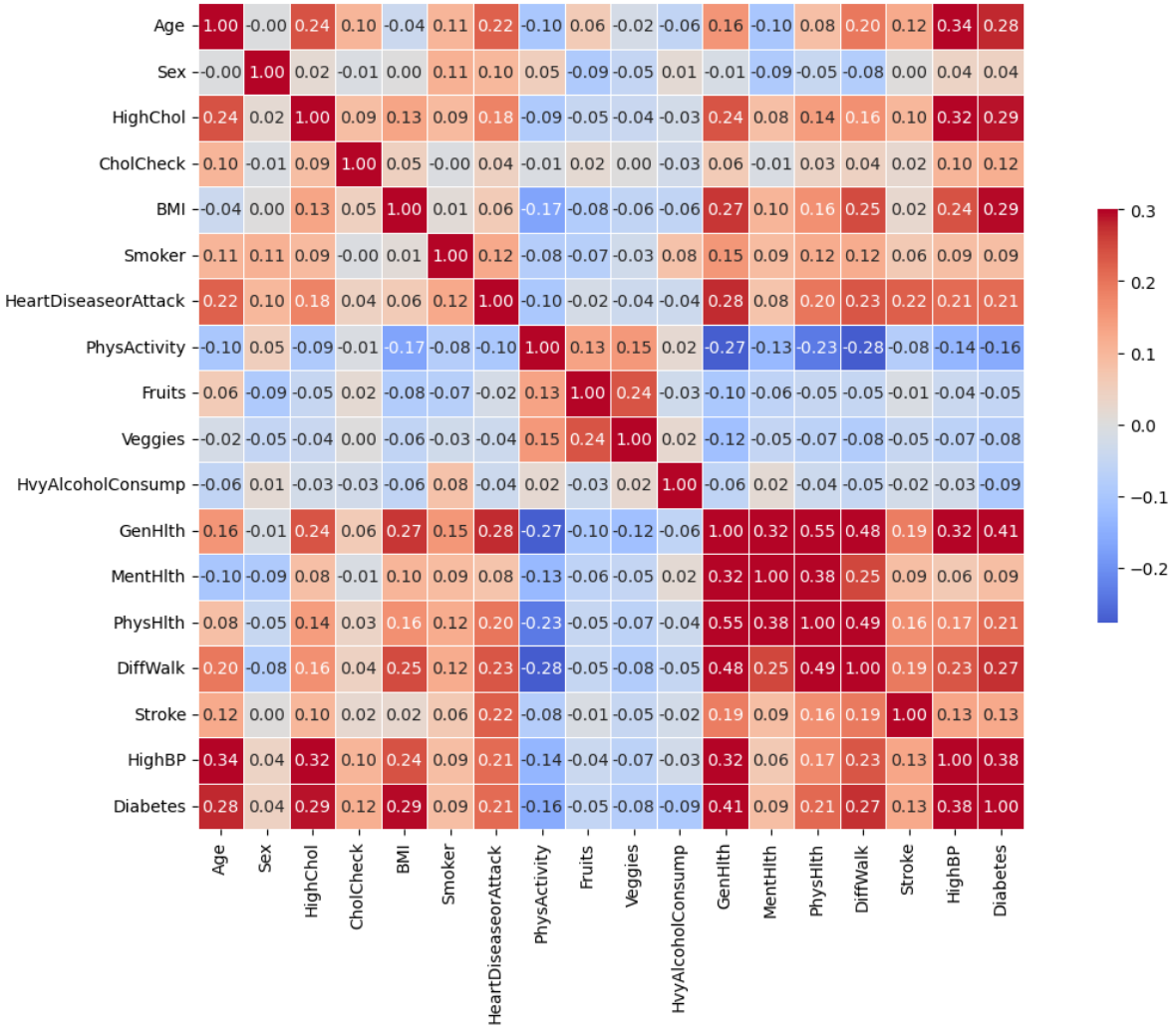
Out[18]:

	DecisionTree	RandomForest	AdaBoostClassifier
Age	1	1	1
Sex	4	4	9
HighChol	7	1	8
CholCheck	11	11	7
BMI	1	1	1
Smoker	3	3	10
HeartDiseaseorAttack	8	8	1
PhysActivity	2	7	5
Fruits	1	2	11
Veggies	5	6	4
HvyAlcoholConsump	10	10	3
GenHlth	1	1	1
MentHlth	1	1	1
PhysHlth	1	1	6
DiffWalk	6	5	2
Stroke	9	9	1
HighBP	1	1	1

In [19]:
```
must = []
maybe = []
for i in rank.index:
    if rank.loc[i, :].sum()==3:
        must.append(i)
    elif 1 in rank.loc[i, :].values:
        maybe.append(i)
```

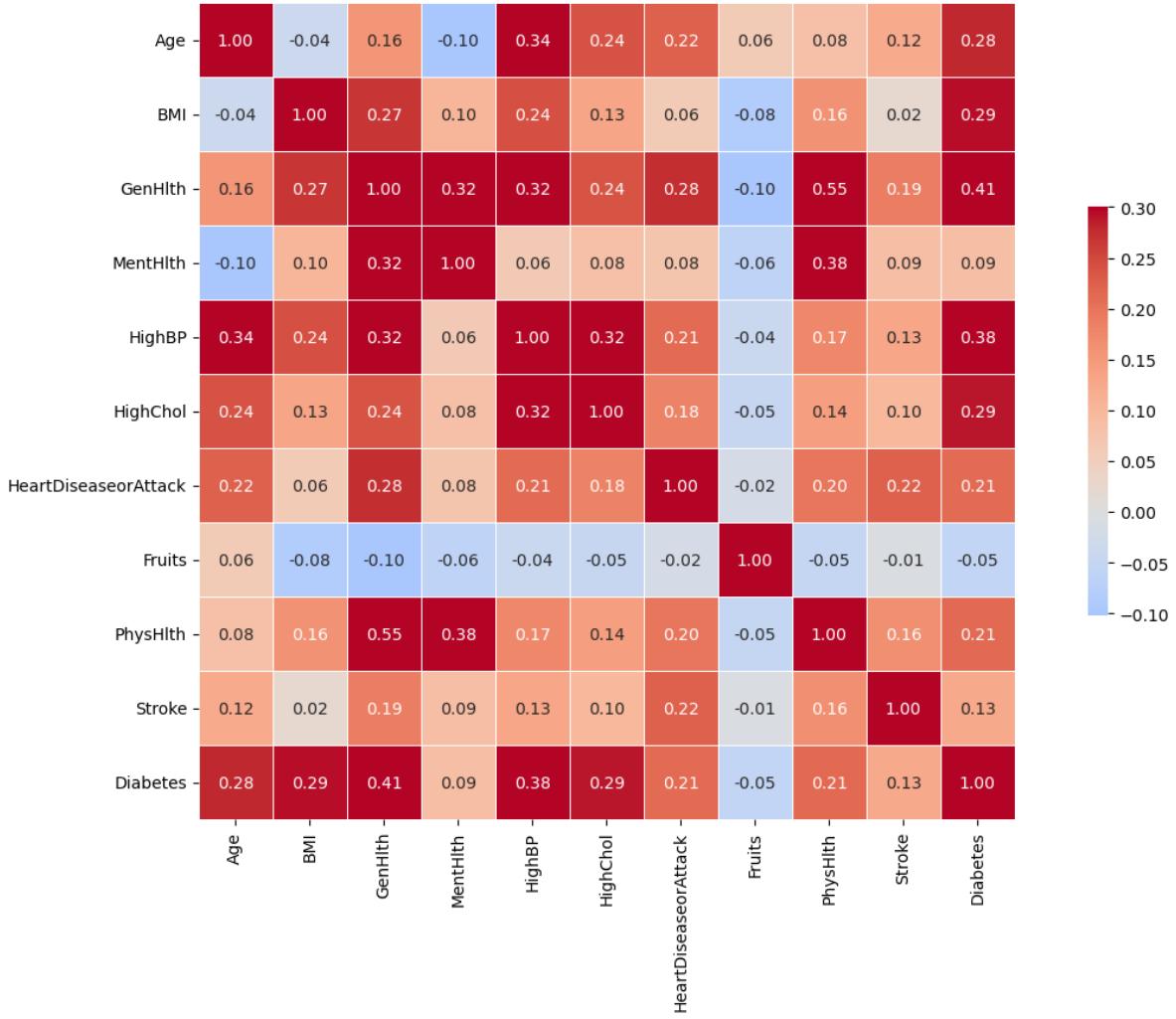
In [20]:
```
g = sns.heatmap(raw_df.corr(), vmax=.3, center=0,
                square=True, linewidths=.5,
                cbar_kws={"shrink": .5}, annot=True,
                fmt='.2f', cmap='coolwarm')
print('must : ', must)
print('maybe :', maybe)
g.figure.set_size_inches(16,9)
plt.show()
```

```
must :  ['Age', 'BMI', 'GenHlth', 'MentHlth', 'HighBP']
maybe : ['HighChol', 'HeartDiseaseorAttack', 'Fruits', 'PhysHlth', 'Stroke']
```



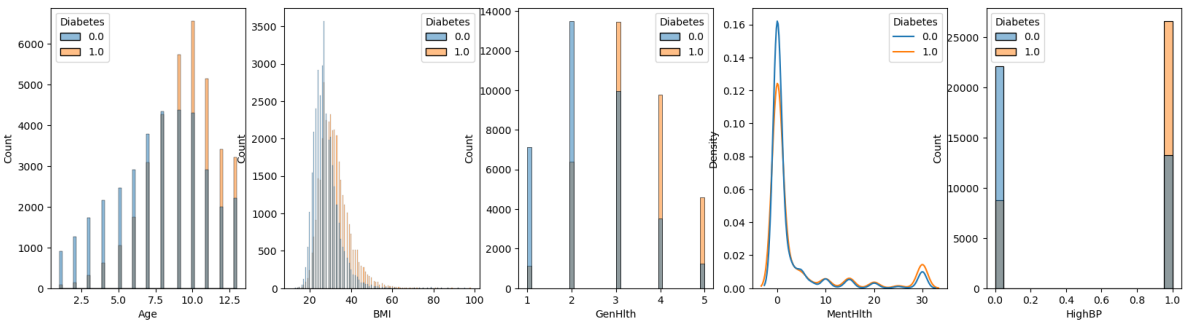
In [21]:
```
reduced_df = raw_df[must+maybe+['Diabetes']]
g = sns.heatmap(reduced_df.corr(), vmax=.3, center=0,
                square=True, linewidths=.5,
                cbar_kws={"shrink": .5}, annot=True,
                fmt='.2f', cmap='coolwarm')
print('must : ', must)
print('maybe :', maybe)
g.figure.set_size_inches(16,9)
plt.show()
```

```
must :  ['Age', 'BMI', 'GenHlth', 'MentHlth', 'HighBP']
maybe : ['HighChol', 'HeartDiseaseorAttack', 'Fruits', 'PhysHlth', 'Stroke']
```



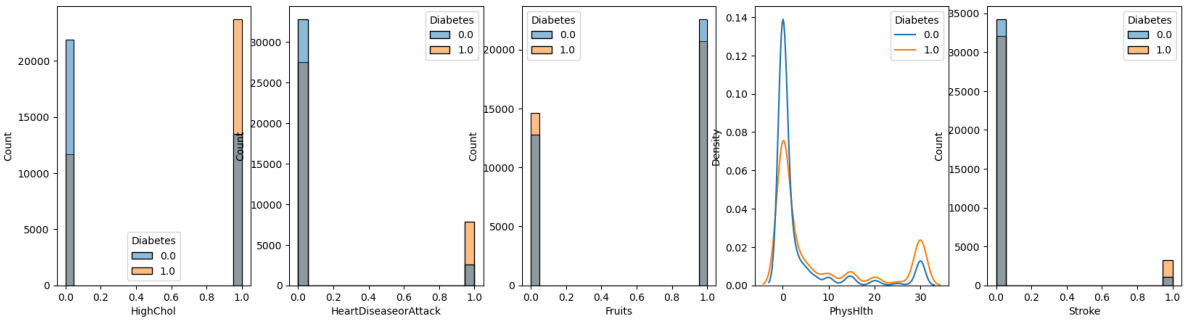
In [22]:

```
fig, ax = plt.subplots(ncols=5, figsize=(20, 5))
i = 0
for col in must:
    if col=='MentHlth' or col=='PhysHlth':
        sns.kdeplot(data=raw_df, x=col, hue='Diabetes', ax=ax[i])
    else:
            sns.histplot(data=raw_df, x=col, hue='Diabetes', ax=ax[i])
    i+=1
plt.show()
```



In [23]:

```
fig, ax = plt.subplots(ncols=5, figsize=(20, 5))
i = 0
for col in maybe:
    if col=='MentHlth' or col=='PhysHlth':
        sns.kdeplot(data=raw_df, x=col, hue='Diabetes', ax=ax[i])
    else:
            sns.histplot(data=raw_df, x=col, hue='Diabetes', ax=ax[i])
    i+=1
plt.show()
```


Count
HighChol
Diabetes
0.0
1.0
HeartDiseaseorAttack
Fruits
Density
PhysHlth
Stroke