HW2 - Type 1 Diabetes

Theory questions

Question 1

To evaluate how well our model performs at T1D classification, we need to have evaluation metrics that measures of its performances/accuracy. Which evaluation metric is more important to us: model accuracy or model performance? Give a simple example that illustrates your claim.

Answer 1

Model accuracy is given by the equation : (TP+TN)/(TP+TN+FP+FN). However, model performances is given by the variables : TP, TN, FP, FN, sensityvity (proportion of people with a condition who are correctly identified by a test as indeed having that condition (TP/(TP+FN)), specificity (proportion of people without a condition who are correctly identified by a test as indeed not having the condition TN/(TN+FP)), PPV (TP/(TP+FP)), NPV (TN/(TN+FN)) and F1 score (average between sensitivyty and PPV). (Lecture 08)

In the case of the detection of Type 1 Diabetes, model performances is more important to us than model accuracy. Indeed, model accuracy does not takes in count all of the parameters, and cannot predict in a good way the errors of the model, so it may mislead us. However, using the model performances provides better insights. For example, if we get accuracy 0.95 (95%) (like in lecture 08), which is high, we could get the 5% rest that may all be false negatives. So we would have failed in not detecting a lot of people that have T1D and are diagnosed as negatives. And our goal is to detect a maximum of positive people that are indeed positive, and to not miss a lot of positives. (In other words, we want high TP and low FN, and high TN and low FP in the same way). However, with the model performances, we can avoid this error by calculating sensitivity which will tell us the amount of people that have T1D and were diagnosed as well vs people who do are positive too but aren't diagnosed positives.

Question 2

T1D is often associated with other comorbidities such as a heart attack. You are asked to design a ML algorithm to predict which patients are going to suffer a heart attack. Relevant patient features for the algorithm may include blood pressure (BP), body-mass index (BMI), age (A), level of physical activity (P), and income (I). You should choose between two classifiers: the first uses only BP and BMI features and the other one uses all of the features available to you. Explain the pros and cons of each choice.

Answer 2

- *Pros and cons of classifier that uses only some of the features:*

  Pros:

- Simplification of the model, easier to interpret, and to extrapolate to other situations.
- There could be useless or redundant features. In this specific case: the age (A) feature could be directly linked to blood pressure (BP), and the physical activity (P) could be correlated with BMI for example.
- Less risk of overfitting.
- Low complexity, lower training time.

Cons:

- Using only some of the features could create bias (not all patients with high blood pressure are old and vice versa) and therefore underfitting.
- Loss of information from the non-used features.
- Less precise results and/or erroneous.

- *Pros and cons of classifier that uses all available features:*

Pros:

- The more information we use, the more accurate the results will be and the less bias we will get.
- No loss of information, we may figure unexpected correlations that will help with building the model (classification for example).

Cons:

- Results are harder to interpret.
- High complexity, higher training time.

Question 3

A histologist wants to use machine learning to tell the difference between pancreas biopsies that show signs of T1D and those that do not. She has already come up with dozens of measurements to take, such as color, size, uniformity and cell-count, but she isn't sure which model to use. The biopsies are really similar, and it is difficult to distinguish them from the human eye, or by just looking at the features. Which of the following is better: logistic regression, linear SVM or nonlinear SVM? Explain your answer.

Answer 3

Logistic regression and linear SMV are two algorithms with relatively similar performance that will assume that our data is linearly separable. A linear SVM will probably give the worst result as it assumes that the data can be linearly separated and only looks for the margins (distance between the line and the support vectors) between them, maybe with very low margins the results will improve. LR won't give good results either as it assumes too tgat our data is linearly separable.

We will advise the histologist to use nonlinear SMV as the difference between the pancreas isn't linear and are difficult to distinguish.

It is important to precise that not one algorithm will solve all types of problems, so generally, it is usually advisable to first try to use LR to see how the model does, if it fails then you can try using nonlinear SVM.

Question 4

What are the differences between LR and linear SVM and what is the difference in the effect/concept of their hyper-parameters tuning?

Answer 4

- Differences between LR and linear SVM: LR and linear SVM are supervised machine learning algorithms. They are both used to solve classification problems (sorting data into categories). SVM tries to maximize the margin between the closest support vectors, while LR tries to maximize the posterior class probability. Moreover, SVM works well with unstructured and semi-structured data like text and images while LR works with already identified independent variables. An other difference is the risk of overfitting : the risk of overfitting is less in SVM, while LR is vulnerable to it.

- Differences in the effect/concept of their hyper-parameters tuning: Most of the machine learning and deep learning algorithms have some parameters that can be adjusted which are called hyperparameters. We need to set hyperparameters before we train the models.

In LR, the hyperparameters are the learning rate and regularization parameter. The learning rate controls how quickly the model is adapted to the problem, and the regularization parameter reduces overfitting. In SMV, the hyperparameters are also the learning rate, and tradeoff parameter (C). The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example.

(For nonlinear kernel SVM we alo use Gamma γ, when we use the Gaussian RBF kernel. γ decides that how much curvature we want in a decision boundary. When it is high, it means more curvature.)

Credits :

Lecture 09, Lecture 10

https://medium.com/axum-labs/logistic-regression-vs-support-vector-machines-svm-c335610a3d16

https://towardsdatascience.com/hyperparameter-tuning-for-support-vector-machines-c-and-gamma-parameters-6a5097416167

http://pavelbazin.com/post/linear-regression-hyperparameters/

https://medium.com/@myselfaman12345/c-and-gamma-in-svm-e6cee48626be

Coding assignment

In [20]:
```
import numpy as np
import itertools
from sklearn.datasets import make_blobs
```

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from scipy.stats import norm
import pickle
import sys
import pandas as pd
import matplotlib as mpl
import seaborn as sns
import matplotlib.pyplot as plt
mpl.style.use(['ggplot'])
%matplotlib inline
plt.rcParams['axes.labelsize'] = 14
plt.rcParams['xtick.labelsize'] = 12
plt.rcParams['ytick.labelsize'] = 12
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.mixture import GaussianMixture
from sklearn.pipeline import Pipeline
from IPython.display import display, clear_output
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.metrics import plot_confusion_matrix, roc_auc_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
```

In [2]:
```
# Importing Data
df = pd.read_csv('HW2_data.csv')
```

Preprocessing data

In [3]:
```
# replace missing answers with Nans
df_clean = df.replace(r'^\s*$', np.nan, regex=True)
```

In [4]:
```
# remove Nans and replace them using random sampling of each series values
def nan2num_samp(df):

    for col in list(df.columns):
        try:
            a = np.array(df[col].value_counts().index)
            size = df[col].value_counts(dropna=False).loc[np.nan]
            p = np.array(df[col].value_counts(normalize=True))

            sample = np.random.choice(a, size=size, p=p)
```

```
            index_nan = list(df[df[col].isna()].index)
            for i in range(len(index_nan)):
                df.loc[index_nan[i], col] = sample[i]
        except Exception:
            pass
    return pd.DataFrame(df)
```

In [5]:
```
df_clean = nan2num_samp (df_clean)
```

In [6]:
```
# remove improbable age if there is
df_clean = df_clean.drop(df_clean[df_clean.Age > 120].index).drop(df_clean[df_clean.Age < 0].index)
```

In [7]:
```
# removing output column
Y = df_clean[['Diagnosis']]
T1D_clean = df_clean.drop(columns=['Diagnosis'])
```

In [8]:
```
# move to binary
Y = pd.get_dummies(data=Y, drop_first=True)
df_clean = pd.get_dummies(data=df_clean, drop_first=True) # keeping copy of whole dataset cleaned
T1D_clean = pd.get_dummies(data=T1D_clean, drop_first=True)
```

In [9]:
```
T1D_clean.describe() # overview on the cleaned data
```

Out[9]:

	Age	Family History	Gender_Male	Increased Urination_Yes	Increased Thirst_Yes	Sudden Weight Loss_Yes	Weakness_Yes	Increased Hunger_Yes	Genital Thrush_Yes	Visual Blurring_Yes
count	565.000000	565.000000	565.000000	565.000000	565.000000	565.000000	565.000000	565.000000	565.000000	565.000000
mean	48.169912	0.506195	0.637168	0.483186	0.449558	0.414159	0.571681	0.447788	0.217699	0.449558
std	12.295828	0.500405	0.481243	0.500160	0.497890	0.493013	0.495274	0.497707	0.413047	0.497890
min	16.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	39.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	48.000000	1.000000	1.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
75%	57.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000	1.000000
max	90.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

In [10]:
```
Y.describe() # stats of Diagnosis
```

Out[10]:

	Diagnosis_Positive

	Diagnosis_Positive
count	565.000000
mean	0.607080
std	0.488832
min	0.000000
25%	0.000000
50%	1.000000
75%	1.000000
max	1.000000

Test-train split

In [11]:
```
# spliting the data

X_train, X_test, y_train, y_test = train_test_split(T1D_clean, Y, test_size=0.2, random_state=0, stratify=Y)
```

Data visualization and exploration

Analysis to show that the distribution of the features is similar between test and train

In [12]:
```
# construction of comparison table for binary features

table_cols = ['Positive Feature', 'Train %', 'Test %', 'Delta Train-Test %']
comp_dis = pd.DataFrame(columns=table_cols) # comp_dis = comparison of distribution
x_train_comp = X_train.drop(['Age'], axis=1).to_numpy()
x_test_comp = X_test.drop(['Age'], axis=1).to_numpy()

#first column
comp_dis['Positive Feature'] = T1D_clean.columns.drop('Age')

#second column
for i, col in enumerate(x_train_comp.transpose()):
    comp_dis.loc[i, 'Train %'] = round((np.sum(col) / col.size) * 100, 1)

#third column
for i, col in enumerate(x_test_comp.transpose()):
    comp_dis.loc[i, 'Test %'] = round((np.sum(col) / col.size) * 100, 1)

#fourth column
comp_dis['Delta Train-Test %'] = comp_dis['Train %'] - comp_dis['Test %']
```

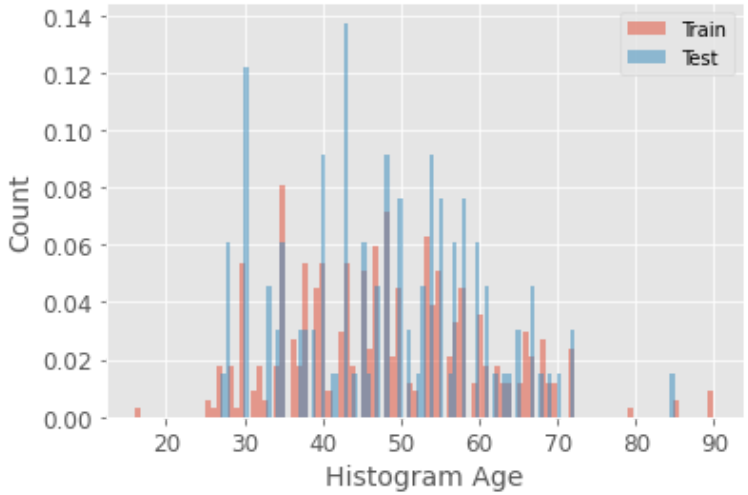
```
comp_dis
```

Out[12]:

	Positive Feature	Train %	Test %	Delta Train-Test %
0	Family History	50.2	52.2	-2
1	Gender_Male	63.9	62.8	1.1
2	Increased Urination_Yes	48.2	48.7	-0.5
3	Increased Thirst_Yes	46	40.7	5.3
4	Sudden Weight Loss_Yes	40.3	46	-5.7
5	Weakness_Yes	56.4	60.2	-3.8
6	Increased Hunger_Yes	45.1	43.4	1.7
7	Genital Thrush_Yes	20.6	26.5	-5.9
8	Visual Blurring_Yes	45.8	41.6	4.2
9	Itching_Yes	49.3	44.2	5.1
10	Irritability_Yes	23	25.7	-2.7
11	Delayed Healing_Yes	46.9	43.4	3.5
12	Partial Paresis_Yes	42.7	42.5	0.2
13	Muscle Stiffness_Yes	36.7	37.2	-0.5
14	Hair Loss_Yes	34.3	41.6	-7.3
15	Obesity_Yes	17.5	14.2	3.3

In [13]:

```
# comparison of distribution for non-binary feature 'Age'

x_train_age = pd.DataFrame(X_train['Age'])
x_test_age = pd.DataFrame(X_test['Age'])

bins = 100
plt.hist(x_train_age, bins, density=True, alpha=0.5, label='Train')
plt.hist(x_test_age, bins, density=True, alpha=0.5, label='Test')
plt.xlabel('Histogram Age')
plt.ylabel('Count')
plt.legend(loc='upper right')
plt.show()
```



In [14]:
```
mu_train, std_train = norm.fit(x_train_age)
mu_test, std_test = norm.fit(x_test_age)

print("Mu of train is: " + str("{0:.2f}".format(mu_train)) + ", Std of train is:" + str("{0:.2f}".format(std_train)))
print("Mu of test is: " + str("{0:.2f}".format(mu_test)) + ", Std of test is:" + str("{0:.2f}".format(std_test)))
```

```
Mu of train is: 48.18, Std of train is:12.34
Mu of test is: 48.12, Std of test is:12.08
```

Question 3) a

i) What issues could an imbalance of features between train and test cause?

Answer

An imbalance of features between train and test could cause a false generalization and a false classification, and may not be representative of the reality.

ii) How can you solve the issue ?

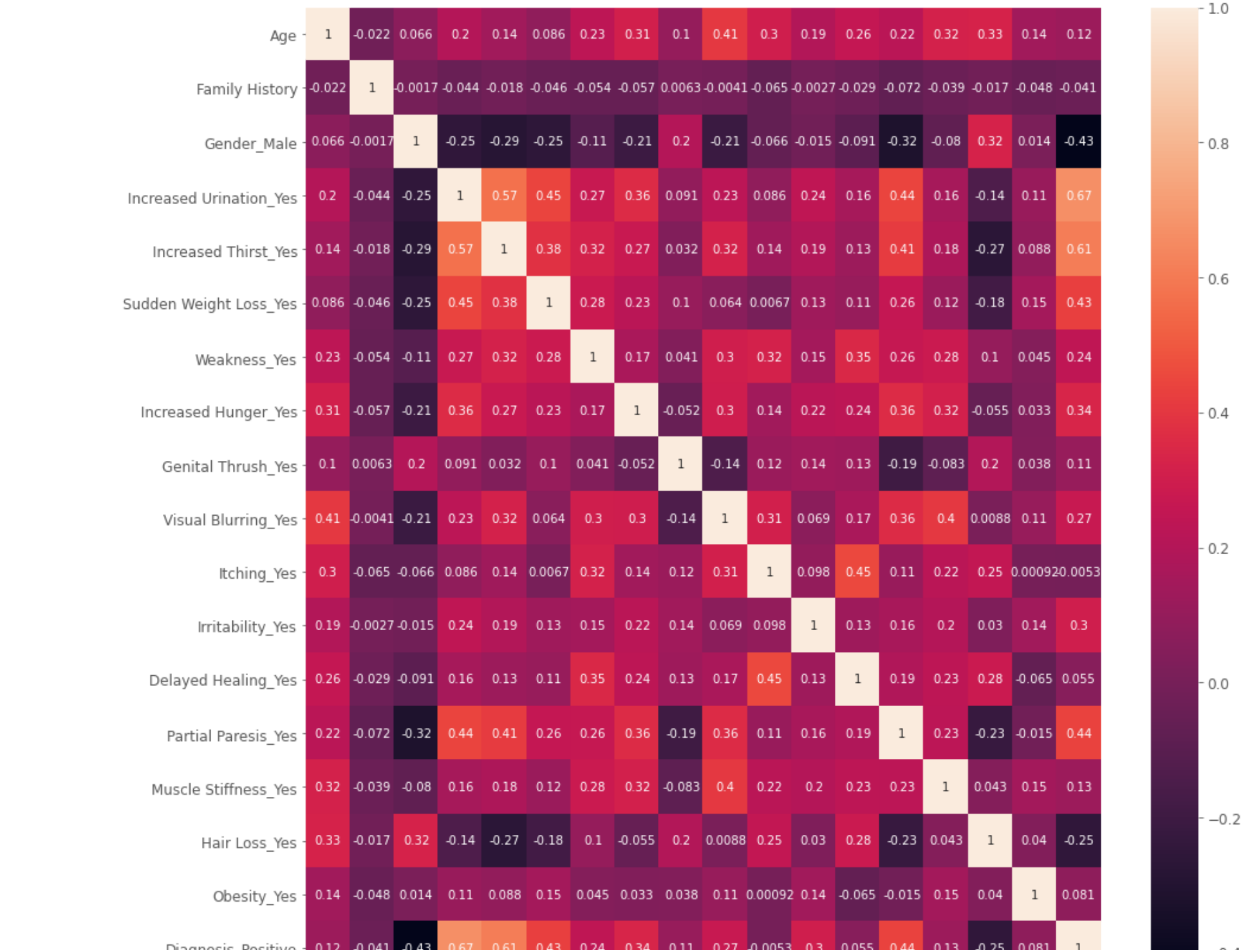
Answer

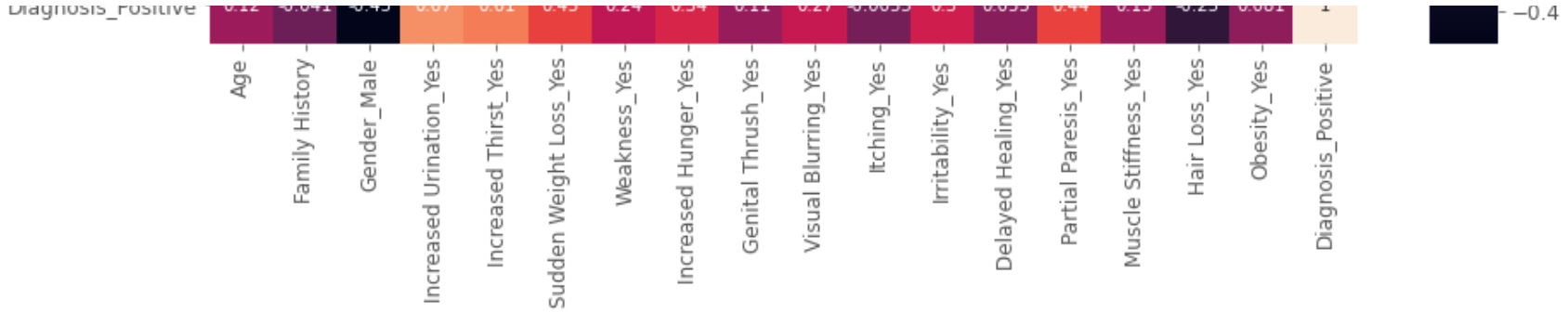
We can solve the issue by adding more data in order to balance the classes, oversample minority classes, undersample majority classes.

Plots showing the relationship between feature and label

In [15]:
```
# Heatmap - checking correlation of Diagnosis with other features
plt.figure(figsize = (15,15))
sns.heatmap(df_clean.corr(), annot = True)
```

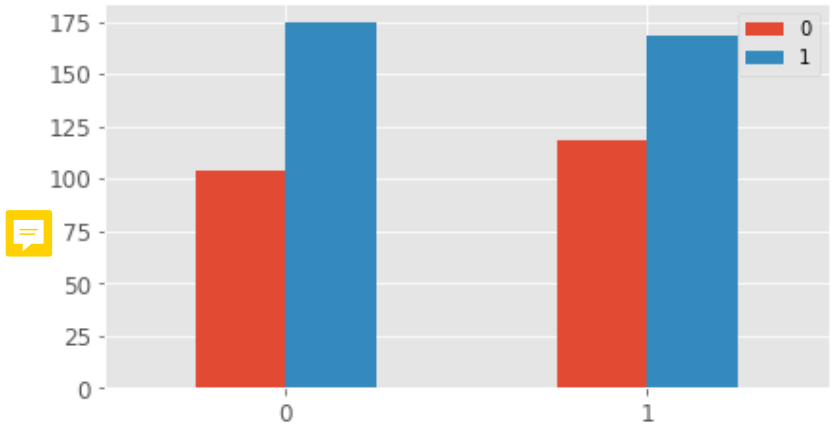
Out[15]: <AxesSubplot:>

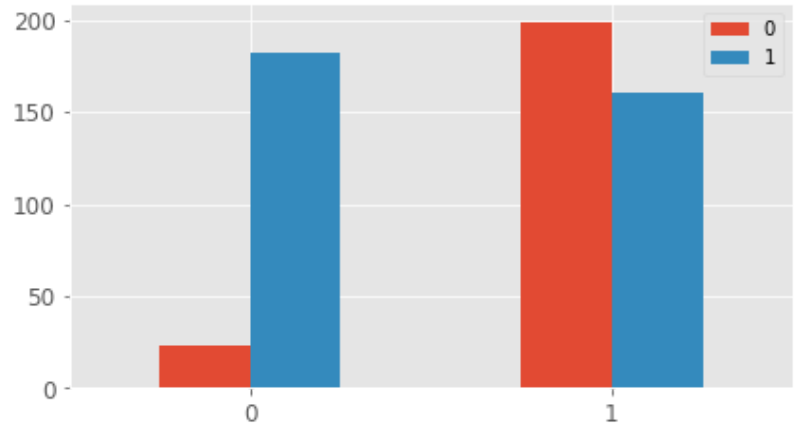

Age 1 -0.022 0.066 0.2 0.14 0.086 0.23 0.31 0.1 0.41 0.3 0.19 0.26 0.22 0.32 0.33 0.14 0.12
Family History -0.022 1 -0.0017 -0.044 -0.018 -0.046 -0.054 -0.057 0.0063 -0.0041 -0.065 -0.0027 -0.029 -0.072 -0.039 -0.017 -0.048 -0.041
Gender_Male 0.066 -0.0017 1 -0.25 -0.29 -0.25 -0.11 -0.21 0.2 -0.21 -0.066 -0.015 -0.091 -0.32 -0.08 0.32 0.014 -0.43
Increased Urination_Yes 0.2 -0.044 -0.25 1 0.57 0.45 0.27 0.36 0.091 0.23 0.086 0.24 0.16 0.44 0.16 -0.14 0.11 0.67
Increased Thirst_Yes 0.14 -0.018 -0.29 0.57 1 0.38 0.32 0.27 0.032 0.32 0.14 0.19 0.13 0.41 0.18 -0.27 0.088 0.61
Sudden Weight Loss_Yes 0.086 -0.046 -0.25 0.45 0.38 1 0.28 0.23 0.1 0.064 0.0067 0.13 0.11 0.26 0.12 -0.18 0.15 0.43
Weakness_Yes 0.23 -0.054 -0.11 0.27 0.32 0.28 1 0.17 0.041 0.3 0.32 0.15 0.35 0.26 0.28 0.1 0.045 0.24
Increased Hunger_Yes 0.31 -0.057 -0.21 0.36 0.27 0.23 0.17 1 -0.052 0.3 0.14 0.22 0.24 0.36 0.32 -0.055 0.033 0.34
Genital Thrush_Yes 0.1 0.0063 0.2 0.091 0.032 0.1 0.041 -0.052 1 -0.14 0.12 0.14 0.13 -0.19 -0.083 0.2 0.038 0.11
Visual Blurring_Yes 0.41 -0.0041 -0.21 0.23 0.32 0.064 0.3 0.3 -0.14 1 0.31 0.069 0.17 0.36 0.4 0.0088 0.11 0.27
Itching_Yes 0.3 -0.065 -0.066 0.086 0.14 0.0067 0.32 0.14 0.12 0.31 1 0.098 0.45 0.11 0.22 0.25 0.00092 -0.0053
Irritability_Yes 0.19 -0.0027 -0.015 0.24 0.19 0.13 0.15 0.22 0.14 0.069 0.098 1 0.13 0.16 0.2 0.03 0.14 0.3
Delayed Healing_Yes 0.26 -0.029 -0.091 0.16 0.13 0.11 0.35 0.24 0.13 0.17 0.45 0.13 1 0.19 0.23 0.28 -0.065 0.055
Partial Paresis_Yes 0.22 -0.072 -0.32 0.44 0.41 0.26 0.26 0.36 -0.19 0.36 0.11 0.16 0.19 1 0.23 -0.23 -0.015 0.44
Muscle Stiffness_Yes 0.32 -0.039 -0.08 0.16 0.18 0.12 0.28 0.32 -0.083 0.4 0.22 0.2 0.23 0.23 1 0.043 0.15 0.13
Hair Loss_Yes 0.33 -0.017 0.32 -0.14 -0.27 -0.18 0.1 -0.055 0.2 0.0088 0.25 0.03 0.28 -0.23 0.043 1 0.04 -0.25
Obesity_Yes 0.14 -0.048 0.014 0.11 0.088 0.15 0.045 0.033 0.038 0.11 0.00092 0.14 -0.065 -0.015 0.15 0.04 1 0.081
Diagnosis_Positive 0.12 -0.041 -0.43 0.67 0.61 0.43 0.24 0.34 0.11 0.27 -0.0053 0.3 0.055 0.44 0.13 -0.25 0.081 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

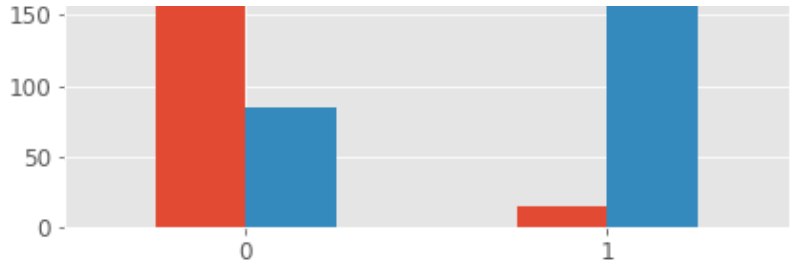


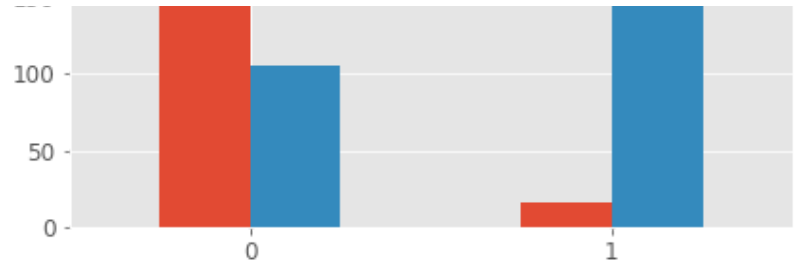
The heatmap shows the highest correlations of positive diagnosis to increased urination and increased thirst.

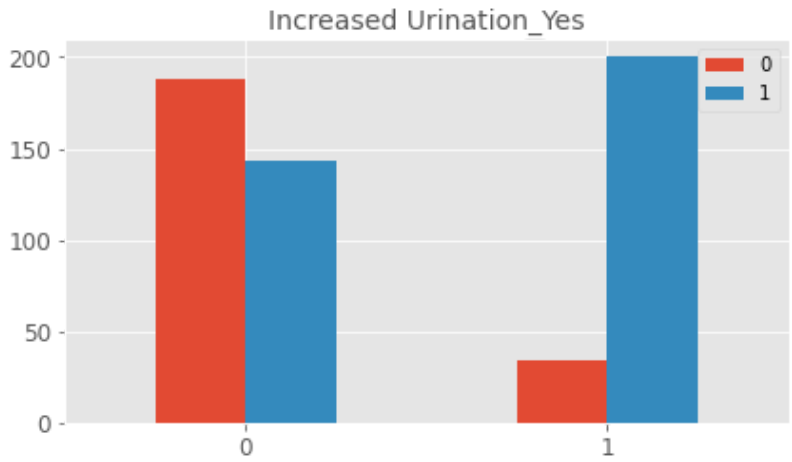
In [41]:
```
# Plots showing the features' frequency according to Diagnosis

idx=1
plt.figure(figsize=(15, 35))
for col in df_clean.drop(columns=['Age','Diagnosis_Positive']).columns:
    ax = plt.subplot(8, 2, idx)
    df_clean[[col,'Diagnosis_Positive']].groupby(col)['Diagnosis_Positive'].value_counts().unstack().plot.bar(ax=ax,
    plt.ylabel='Count'
    plt.legend(loc='upper right')
    idx+=1

# Red bars '0' = Diagnosis Negative
# Blue bars '1' = Diagnosis Positive
```

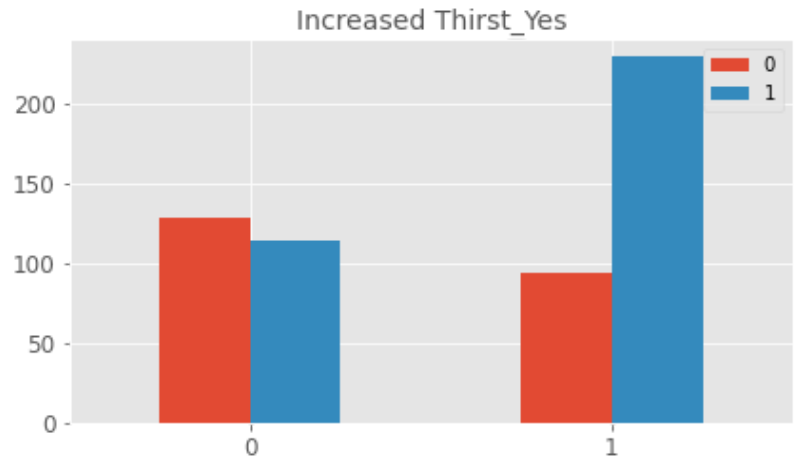


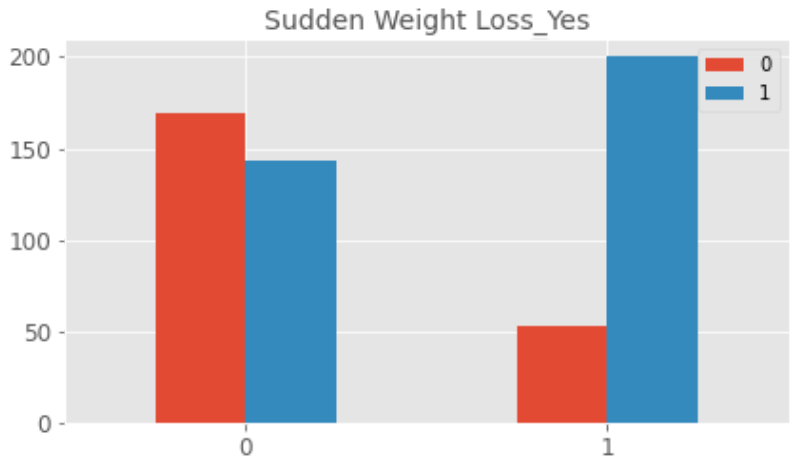


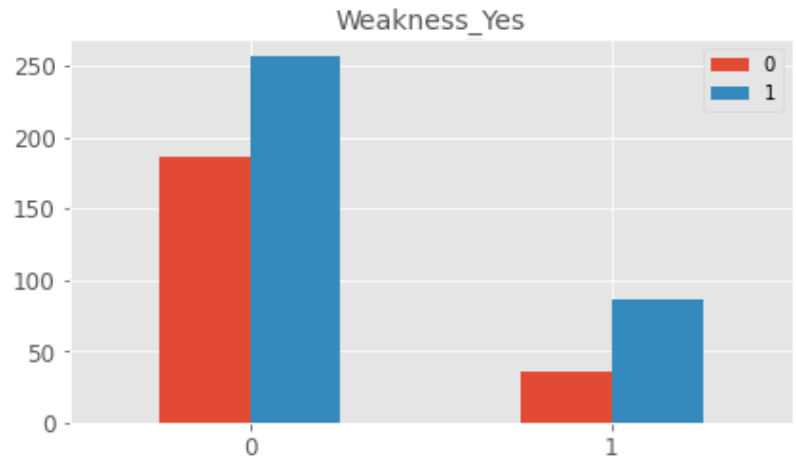

150
100
50
0
0
1

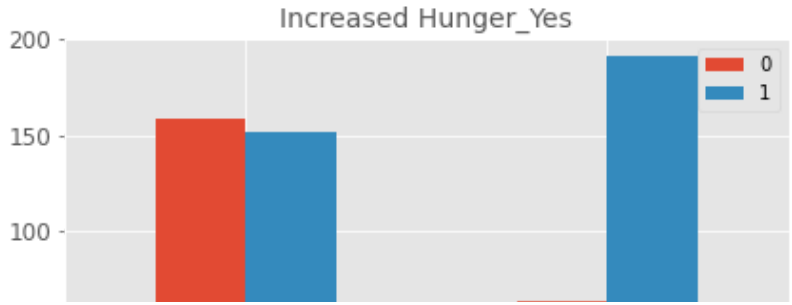

100
50
0
0
1

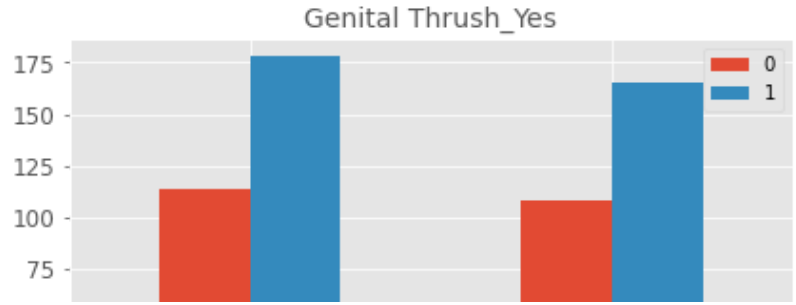

Increased Urination_Yes
200
150
100
50
0
0
1
0
1

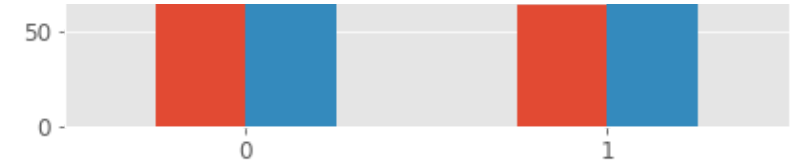

Increased Thirst_Yes
200
150
100
50
0
0
1
0
1

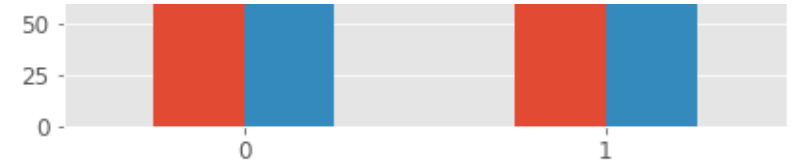

Sudden Weight Loss_Yes
200
150
100
50
0
0
1
0
1

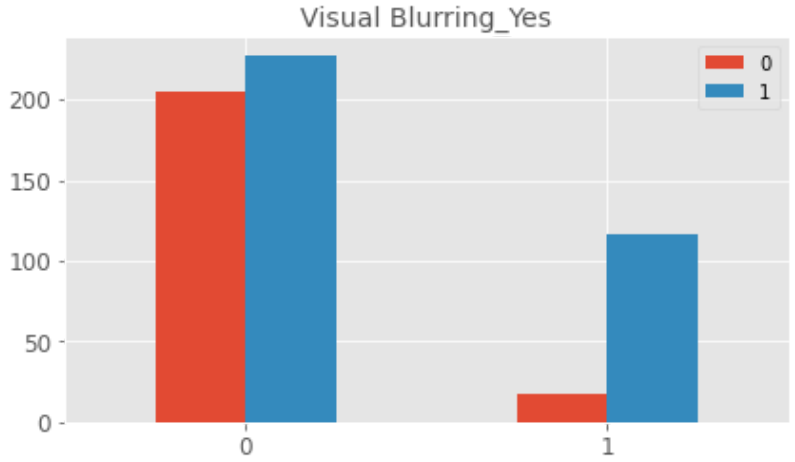

Weakness_Yes
250
200
150
100
50
0
0
1
0
1

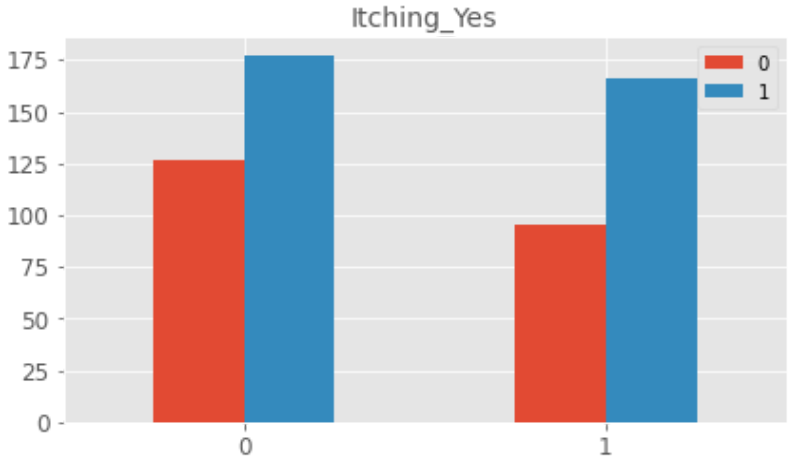

Increased Hunger_Yes
200
150
100
0
1

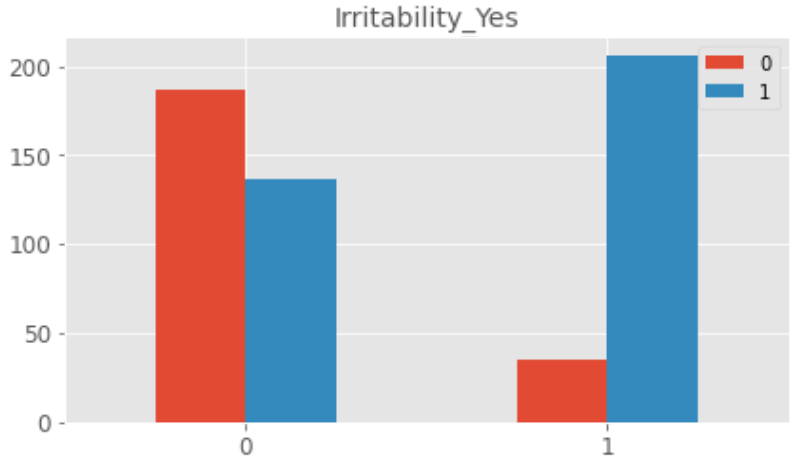

Genital Thrush_Yes
175
150
125
100
75
0
1

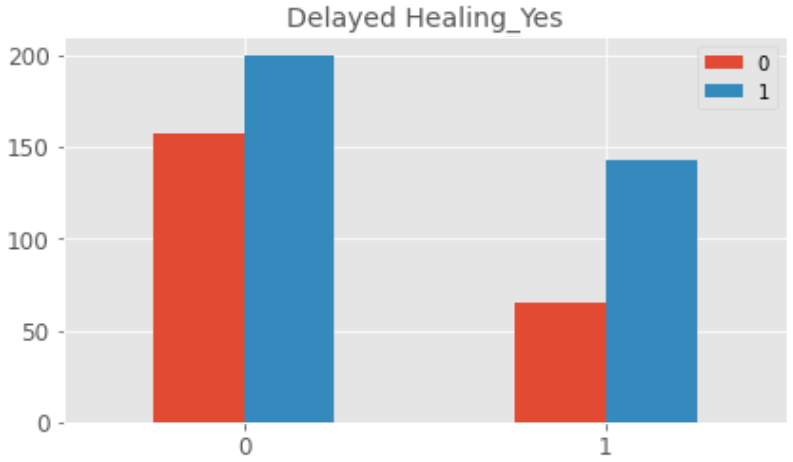

50
0
0
1

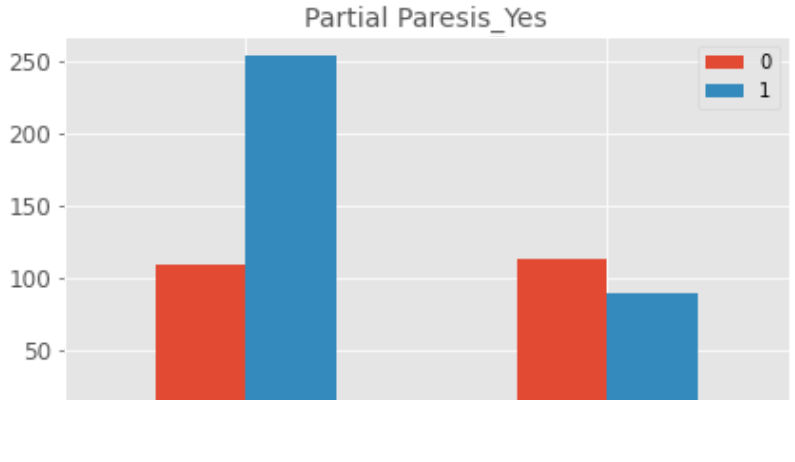

50
25
0
0
1

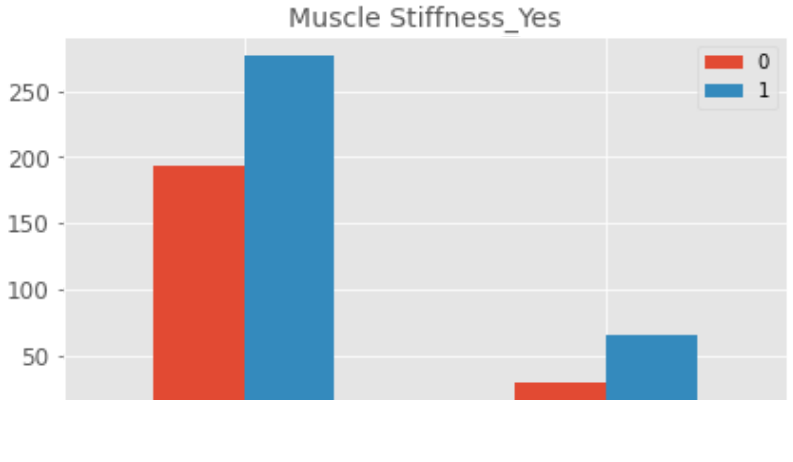

Visual Blurring_Yes
200
150
100
50
0
0
1
0
1

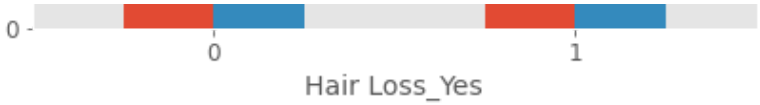

Itching_Yes
175
150
125
100
75
50
25
0
0
1
0
1

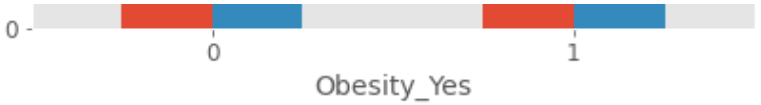

Irritability_Yes
200
150
100
50
0
0
1
0
1


Delayed Healing_Yes
200
150
100
50
0
0
1
0
1

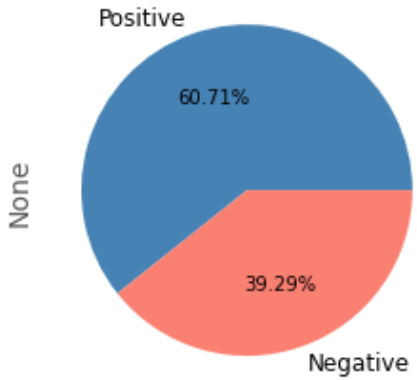

Partial Paresis_Yes
250
200
150
100
50
0
1


Muscle Stiffness_Yes
250
200
150
100
50
0
1





In [17]:
```
# Pie Diagram for visualization of Diagnosis Proportions

Y.value_counts().plot(kind="pie", labels=['Positive','Negative'], colors = ['steelblue', 'salmon'], autopct='%1.2f%%'
plt.show()
```



Question 3) d

i) Was there anything unexpected?

Answer

Surprisingly, the fact that someone has family history in T1D doesn't affect so much the diagnosis. In addition, we see that the gender influes much on the diagnosis : the amount of negative women is way lower than negative men, when we expected this feature to be balanced. We can also remark that obesity isn't a factor to T1D : number of positive non-obese people are way higher than positive obese people.

ii) Are there any features that you feel will be particularly important to your model? Explain why.

Answer

The features that seem particularly important to our model are increased thirst, increased urination and gender (we explained why the gender seems important to look on 3)d.i.). Indeed, we can see from the graphs that most of people with increased thirst or increased urination are diagnosed positive.

Encoding the data as one hot vector

In [18]:
```
enc = OneHotEncoder(handle_unknown='ignore')
enc_Diagnosis = enc.fit_transform(Y)
```

```
enc_T1D = enc.fit_transform (T1D_clean)
```

Chosing, building and optimizing Machin learning models

Linear models

In [21]:
```
# Logistic regression - LogReg

lmbda = np.array([0.001, 0.01, 1, 10, 100, 1000])
pen = ['l1', 'l2']
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
log_reg = LogisticRegression(random_state=5, penalty=pen, C = 1/lmbda, solver='saga')
pipe = Pipeline(steps=[('scale', StandardScaler()), ('logistic', log_reg)])
LogReg = GridSearchCV(estimator=pipe, param_grid={'logistic__C': 1/lmbda, 'logistic__penalty': pen},
                      scoring=['accuracy','f1','roc_auc'], cv=skf,
                      refit='roc_auc', verbose=3, return_train_score=True)
LogReg.fit(X_train, np.ravel(y_train))
chosen_LogReg = LogReg.best_estimator_

y_pred_test = chosen_LogReg.predict(X_test)
y_pred_proba_test = chosen_LogReg.predict_proba(X_test)

y_pred_train = chosen_LogReg.predict(X_train)
y_pred_proba_train = chosen_LogReg.predict_proba(X_train)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
[CV 1/5] END logistic__C=1000.0, logistic__penalty=l1; accuracy: (train=0.947, test=0.901) f1: (train=0.957, test=0.91
9) roc_auc: (train=0.986, test=0.952) total time=   0.1s
[CV 2/5] END logistic__C=1000.0, logistic__penalty=l1; accuracy: (train=0.945, test=0.934) f1: (train=0.954, test=0.94
4) roc_auc: (train=0.982, test=0.971) total time=   0.0s
[CV 3/5] END logistic__C=1000.0, logistic__penalty=l1; accuracy: (train=0.945, test=0.878) f1: (train=0.955, test=0.89
7) roc_auc: (train=0.983, test=0.961) total time=   0.0s
[CV 4/5] END logistic__C=1000.0, logistic__penalty=l1; accuracy: (train=0.925, test=0.911) f1: (train=0.939, test=0.92
9) roc_auc: (train=0.981, test=0.974) total time=   0.0s
[CV 5/5] END logistic__C=1000.0, logistic__penalty=l1; accuracy: (train=0.936, test=0.933) f1: (train=0.947, test=0.94
7) roc_auc: (train=0.980, test=0.973) total time=   0.0s
[CV 1/5] END logistic__C=1000.0, logistic__penalty=l2; accuracy: (train=0.947, test=0.901) f1: (train=0.957, test=0.91
9) roc_auc: (train=0.986, test=0.952) total time=   0.0s
[CV 2/5] END logistic__C=1000.0, logistic__penalty=l2; accuracy: (train=0.945, test=0.934) f1: (train=0.954, test=0.94
4) roc_auc: (train=0.982, test=0.971) total time=   0.0s
[CV 3/5] END logistic__C=1000.0, logistic__penalty=l2; accuracy: (train=0.945, test=0.878) f1: (train=0.955, test=0.89
7) roc_auc: (train=0.983, test=0.961) total time=   0.0s
[CV 4/5] END logistic__C=1000.0, logistic__penalty=l2; accuracy: (train=0.925, test=0.911) f1: (train=0.939, test=0.92
9) roc_auc: (train=0.981, test=0.974) total time=   0.0s
[CV 5/5] END logistic__C=1000.0, logistic__penalty=l2; accuracy: (train=0.936, test=0.933) f1: (train=0.947, test=0.94
7) roc_auc: (train=0.980, test=0.973) total time=   0.0s
[CV 1/5] END logistic__C=100.0, logistic__penalty=l1; accuracy: (train=0.947, test=0.901) f1: (train=0.957, test=0.91
9) roc_auc: (train=0.986, test=0.952) total time=   0.0s
```

```
[CV 2/5] END logistic__C=100.0, logistic__penalty=l1; accuracy: (train=0.945, test=0.934) f1: (train=0.954, test=0.94
4) roc_auc: (train=0.982, test=0.971) total time=   0.0s
[CV 3/5] END logistic__C=100.0, logistic__penalty=l1; accuracy: (train=0.945, test=0.878) f1: (train=0.955, test=0.89
7) roc_auc: (train=0.983, test=0.960) total time=   0.0s
[CV 4/5] END logistic__C=100.0, logistic__penalty=l1; accuracy: (train=0.925, test=0.911) f1: (train=0.939, test=0.92
9) roc_auc: (train=0.981, test=0.974) total time=   0.0s
[CV 5/5] END logistic__C=100.0, logistic__penalty=l1; accuracy: (train=0.936, test=0.933) f1: (train=0.947, test=0.94
7) roc_auc: (train=0.980, test=0.973) total time=   0.0s
[CV 1/5] END logistic__C=100.0, logistic__penalty=l2; accuracy: (train=0.947, test=0.901) f1: (train=0.957, test=0.91
9) roc_auc: (train=0.986, test=0.952) total time=   0.0s
[CV 2/5] END logistic__C=100.0, logistic__penalty=l2; accuracy: (train=0.945, test=0.934) f1: (train=0.954, test=0.94
4) roc_auc: (train=0.982, test=0.971) total time=   0.0s
[CV 3/5] END logistic__C=100.0, logistic__penalty=l2; accuracy: (train=0.945, test=0.878) f1: (train=0.955, test=0.89
7) roc_auc: (train=0.983, test=0.960) total time=   0.0s
[CV 4/5] END logistic__C=100.0, logistic__penalty=l2; accuracy: (train=0.925, test=0.911) f1: (train=0.939, test=0.92
9) roc_auc: (train=0.981, test=0.974) total time=   0.0s
[CV 5/5] END logistic__C=100.0, logistic__penalty=l2; accuracy: (train=0.936, test=0.933) f1: (train=0.947, test=0.94
7) roc_auc: (train=0.980, test=0.973) total time=   0.0s
[CV 1/5] END logistic__C=1.0, logistic__penalty=l1; accuracy: (train=0.950, test=0.890) f1: (train=0.959, test=0.909)
roc_auc: (train=0.986, test=0.957) total time=   0.0s
[CV 2/5] END logistic__C=1.0, logistic__penalty=l1; accuracy: (train=0.947, test=0.923) f1: (train=0.957, test=0.935)
roc_auc: (train=0.982, test=0.974) total time=   0.0s
[CV 3/5] END logistic__C=1.0, logistic__penalty=l1; accuracy: (train=0.936, test=0.878) f1: (train=0.948, test=0.895)
roc_auc: (train=0.982, test=0.958) total time=   0.0s
[CV 4/5] END logistic__C=1.0, logistic__penalty=l1; accuracy: (train=0.923, test=0.911) f1: (train=0.936, test=0.929)
roc_auc: (train=0.981, test=0.976) total time=   0.0s
[CV 5/5] END logistic__C=1.0, logistic__penalty=l1; accuracy: (train=0.934, test=0.922) f1: (train=0.945, test=0.939)
roc_auc: (train=0.980, test=0.972) total time=   0.0s
[CV 1/5] END logistic__C=1.0, logistic__penalty=l2; accuracy: (train=0.945, test=0.890) f1: (train=0.954, test=0.909)
roc_auc: (train=0.986, test=0.955) total time=   0.0s
[CV 2/5] END logistic__C=1.0, logistic__penalty=l2; accuracy: (train=0.945, test=0.934) f1: (train=0.954, test=0.944)
roc_auc: (train=0.982, test=0.974) total time=   0.0s
[CV 3/5] END logistic__C=1.0, logistic__penalty=l2; accuracy: (train=0.953, test=0.878) f1: (train=0.961, test=0.895)
roc_auc: (train=0.982, test=0.957) total time=   0.0s
[CV 4/5] END logistic__C=1.0, logistic__penalty=l2; accuracy: (train=0.923, test=0.911) f1: (train=0.936, test=0.929)
roc_auc: (train=0.980, test=0.973) total time=   0.0s
[CV 5/5] END logistic__C=1.0, logistic__penalty=l2; accuracy: (train=0.928, test=0.933) f1: (train=0.941, test=0.947)
roc_auc: (train=0.980, test=0.975) total time=   0.0s
[CV 1/5] END logistic__C=0.1, logistic__penalty=l1; accuracy: (train=0.922, test=0.890) f1: (train=0.937, test=0.911)
roc_auc: (train=0.978, test=0.948) total time=   0.0s
[CV 2/5] END logistic__C=0.1, logistic__penalty=l1; accuracy: (train=0.925, test=0.912) f1: (train=0.939, test=0.926)
roc_auc: (train=0.973, test=0.978) total time=   0.0s
[CV 3/5] END logistic__C=0.1, logistic__penalty=l1; accuracy: (train=0.909, test=0.900) f1: (train=0.926, test=0.914)
roc_auc: (train=0.973, test=0.960) total time=   0.0s
[CV 4/5] END logistic__C=0.1, logistic__penalty=l1; accuracy: (train=0.909, test=0.956) f1: (train=0.926, test=0.964)
roc_auc: (train=0.971, test=0.978) total time=   0.0s
[CV 5/5] END logistic__C=0.1, logistic__penalty=l1; accuracy: (train=0.923, test=0.900) f1: (train=0.936, test=0.922)
roc_auc: (train=0.974, test=0.948) total time=   0.0s
[CV 1/5] END logistic__C=0.1, logistic__penalty=l2; accuracy: (train=0.928, test=0.890) f1: (train=0.940, test=0.909)
roc_auc: (train=0.984, test=0.960) total time=   0.0s
[CV 2/5] END logistic__C=0.1, logistic__penalty=l2; accuracy: (train=0.936, test=0.934) f1: (train=0.948, test=0.944)
roc_auc: (train=0.981, test=0.980) total time=   0.0s
```

```
[CV 3/5] END logistic__C=0.1, logistic__penalty=l2; accuracy: (train=0.934, test=0.889) f1: (train=0.944, test=0.904)
roc_auc: (train=0.980, test=0.956) total time=   0.0s
[CV 4/5] END logistic__C=0.1, logistic__penalty=l2; accuracy: (train=0.923, test=0.900) f1: (train=0.936, test=0.919)
roc_auc: (train=0.979, test=0.972) total time=   0.0s
[CV 5/5] END logistic__C=0.1, logistic__penalty=l2; accuracy: (train=0.920, test=0.889) f1: (train=0.933, test=0.911)
roc_auc: (train=0.978, test=0.980) total time=   0.0s
[CV 1/5] END logistic__C=0.01, logistic__penalty=l1; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.753)
roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END logistic__C=0.01, logistic__penalty=l1; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.753)
roc_auc: (train=0.846, test=0.850) total time=   0.0s
[CV 3/5] END logistic__C=0.01, logistic__penalty=l1; accuracy: (train=0.608, test=0.600) f1: (train=0.756, test=0.750)
roc_auc: (train=0.911, test=0.864) total time=   0.0s
[CV 4/5] END logistic__C=0.01, logistic__penalty=l1; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.759)
roc_auc: (train=0.837, test=0.886) total time=   0.0s
[CV 5/5] END logistic__C=0.01, logistic__penalty=l1; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.759)
roc_auc: (train=0.857, test=0.805) total time=   0.0s
[CV 1/5] END logistic__C=0.01, logistic__penalty=l2; accuracy: (train=0.911, test=0.890) f1: (train=0.927, test=0.909)
roc_auc: (train=0.971, test=0.944) total time=   0.0s
[CV 2/5] END logistic__C=0.01, logistic__penalty=l2; accuracy: (train=0.911, test=0.923) f1: (train=0.927, test=0.936)
roc_auc: (train=0.970, test=0.981) total time=   0.0s
[CV 3/5] END logistic__C=0.01, logistic__penalty=l2; accuracy: (train=0.912, test=0.867) f1: (train=0.928, test=0.887)
roc_auc: (train=0.971, test=0.945) total time=   0.0s
[CV 4/5] END logistic__C=0.01, logistic__penalty=l2; accuracy: (train=0.895, test=0.911) f1: (train=0.914, test=0.929)
roc_auc: (train=0.974, test=0.971) total time=   0.0s
[CV 5/5] END logistic__C=0.01, logistic__penalty=l2; accuracy: (train=0.917, test=0.878) f1: (train=0.931, test=0.903)
roc_auc: (train=0.968, test=0.971) total time=   0.0s
[CV 1/5] END logistic__C=0.001, logistic__penalty=l1; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.75
3) roc_auc: (train=0.500, test=0.500) total time=   0.0s
[CV 2/5] END logistic__C=0.001, logistic__penalty=l1; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.75
3) roc_auc: (train=0.500, test=0.500) total time=   0.0s
[CV 3/5] END logistic__C=0.001, logistic__penalty=l1; accuracy: (train=0.608, test=0.600) f1: (train=0.756, test=0.75
0) roc_auc: (train=0.500, test=0.500) total time=   0.0s
[CV 4/5] END logistic__C=0.001, logistic__penalty=l1; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.75
9) roc_auc: (train=0.500, test=0.500) total time=   0.0s
[CV 5/5] END logistic__C=0.001, logistic__penalty=l1; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.75
9) roc_auc: (train=0.500, test=0.500) total time=   0.0s
[CV 1/5] END logistic__C=0.001, logistic__penalty=l2; accuracy: (train=0.748, test=0.703) f1: (train=0.827, test=0.80
0) roc_auc: (train=0.953, test=0.915) total time=   0.0s
[CV 2/5] END logistic__C=0.001, logistic__penalty=l2; accuracy: (train=0.729, test=0.736) f1: (train=0.816, test=0.82
1) roc_auc: (train=0.951, test=0.976) total time=   0.0s
[CV 3/5] END logistic__C=0.001, logistic__penalty=l2; accuracy: (train=0.740, test=0.722) f1: (train=0.823, test=0.81
2) roc_auc: (train=0.956, test=0.916) total time=   0.0s
[CV 4/5] END logistic__C=0.001, logistic__penalty=l2; accuracy: (train=0.732, test=0.733) f1: (train=0.818, test=0.81
8) roc_auc: (train=0.952, test=0.963) total time=   0.0s
[CV 5/5] END logistic__C=0.001, logistic__penalty=l2; accuracy: (train=0.724, test=0.778) f1: (train=0.813, test=0.84
6) roc_auc: (train=0.952, test=0.954) total time=   0.0s
```

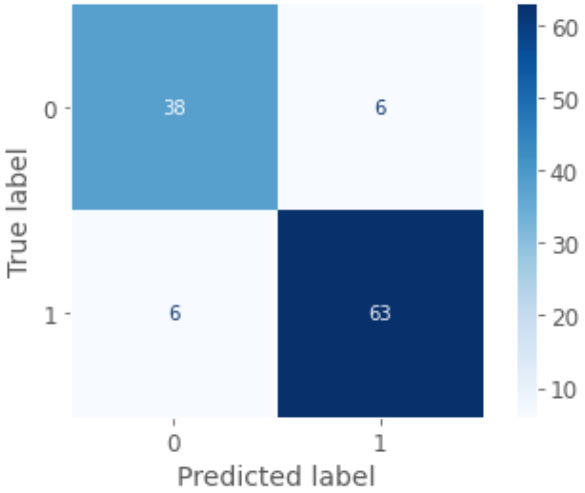
In [22]:

```
def plot_and_calc (x_test, y_test, y_pred_test, y_pred_proba_test):

    plot_confusion_matrix(chosen_LogReg,x_test,y_test, cmap=plt.cm.Blues)
    plt.grid(False)
```

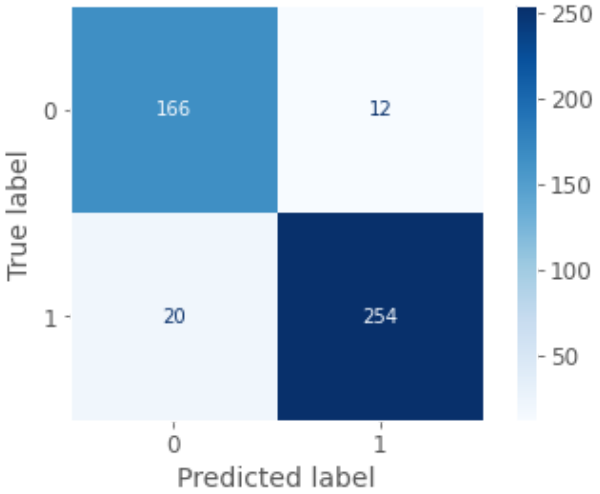
```
    plt.show()

    calc_TN = lambda y_true, y_pred: confusion_matrix(y_true, y_pred)[0, 0]
    calc_FP = lambda y_true, y_pred: confusion_matrix(y_true, y_pred)[0, 1]
    calc_FN = lambda y_true, y_pred: confusion_matrix(y_true, y_pred)[1, 0]
    calc_TP = lambda y_true, y_pred: confusion_matrix(y_true, y_pred)[1, 1]

    TN = calc_TN(y_test,y_pred_test)
    FP = calc_FP(y_test,y_pred_test)
    FN = calc_FN(y_test,y_pred_test)
    TP = calc_TP(y_test,y_pred_test)
    Se = TP/(TP+FN)
    Sp = TN/(TN+FP)
    PPV = TP/(TP+FP)
    NPV = TN/(TN+FN)
    Acc = (TP+TN)/(TP+TN+FP+FN)
    F1 = (2*Se*PPV)/(Se+PPV)
    print('Accuracy is {:.2f}. \nF1 is {:.2f}.'.format(Acc,F1))
    print('AUROC is {:.2f}'.format(roc_auc_score(y_test, y_pred_proba_test[:,1])))
```

In [23]:
```
print('Test set:')
plot_and_calc (X_test, y_test, y_pred_test, y_pred_proba_test)
print('Train set:')
plot_and_calc (X_train, y_train, y_pred_train, y_pred_proba_train)
```

```
Test set:
```




```
Accuracy is 0.89.
F1 is 0.91.
AUROC is 0.96
Train set:
```




```
Accuracy is 0.93.
F1 is 0.94.
AUROC is 0.98
```

In [24]:

```python
# Linear SVM
from sklearn.svm import SVC

C = np.array([0.001, 0.01, 1, 10, 100, 1000])
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
svc = SVC(probability=True)
pipe = Pipeline(steps=[('scale', StandardScaler()), ('svm', svc)])
svm_lin = GridSearchCV(estimator=pipe,
             param_grid={'svm__kernel':['linear'], 'svm__C':C},
             scoring=['accuracy','f1','roc_auc'],
             cv=skf, refit='roc_auc', verbose=3, return_train_score=True)
svm_lin.fit(X_train, np.ravel(y_train))
chosen_svm_lin = svm_lin.best_estimator_

y_pred_test = chosen_svm_lin.predict(X_test)
y_pred_proba_test = chosen_svm_lin.predict_proba(X_test)

y_pred_train = chosen_svm_lin.predict(X_train)
y_pred_proba_train = chosen_svm_lin.predict_proba(X_train)
```

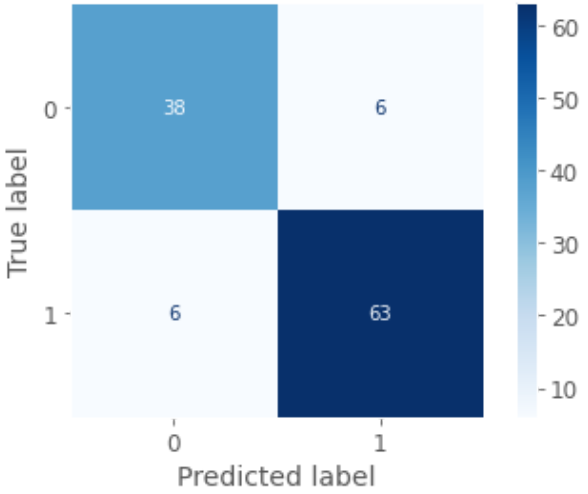
```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
[CV 1/5] END svm__C=0.001, svm__kernel=linear; accuracy: (train=0.881, test=0.824) f1: (train=0.906, test=0.864) roc_a
uc: (train=0.955, test=0.917) total time=   0.1s
[CV 2/5] END svm__C=0.001, svm__kernel=linear; accuracy: (train=0.820, test=0.868) f1: (train=0.866, test=0.900) roc_a
uc: (train=0.955, test=0.979) total time=   0.0s
[CV 3/5] END svm__C=0.001, svm__kernel=linear; accuracy: (train=0.870, test=0.800) f1: (train=0.899, test=0.845) roc_a
uc: (train=0.957, test=0.916) total time=   0.0s
```

```
[CV 4/5] END svm__C=0.001, svm__kernel=linear; accuracy: (train=0.818, test=0.822) f1: (train=0.866, test=0.869) roc_a
uc: (train=0.954, test=0.964) total time=   0.0s
[CV 5/5] END svm__C=0.001, svm__kernel=linear; accuracy: (train=0.809, test=0.856) f1: (train=0.859, test=0.891) roc_a
uc: (train=0.954, test=0.952) total time=   0.0s
[CV 1/5] END svm__C=0.01, svm__kernel=linear; accuracy: (train=0.892, test=0.890) f1: (train=0.908, test=0.907) roc_au
c: (train=0.977, test=0.959) total time=   0.0s
[CV 2/5] END svm__C=0.01, svm__kernel=linear; accuracy: (train=0.909, test=0.890) f1: (train=0.923, test=0.904) roc_au
c: (train=0.974, test=0.981) total time=   0.0s
[CV 3/5] END svm__C=0.01, svm__kernel=linear; accuracy: (train=0.914, test=0.867) f1: (train=0.927, test=0.882) roc_au
c: (train=0.975, test=0.942) total time=   0.0s
[CV 4/5] END svm__C=0.01, svm__kernel=linear; accuracy: (train=0.906, test=0.900) f1: (train=0.921, test=0.919) roc_au
c: (train=0.974, test=0.970) total time=   0.0s
[CV 5/5] END svm__C=0.01, svm__kernel=linear; accuracy: (train=0.903, test=0.878) f1: (train=0.917, test=0.901) roc_au
c: (train=0.973, test=0.971) total time=   0.0s
[CV 1/5] END svm__C=1.0, svm__kernel=linear; accuracy: (train=0.942, test=0.912) f1: (train=0.952, test=0.927) roc_au
c: (train=0.982, test=0.955) total time=   0.0s
[CV 2/5] END svm__C=1.0, svm__kernel=linear; accuracy: (train=0.942, test=0.945) f1: (train=0.952, test=0.954) roc_au
c: (train=0.979, test=0.975) total time=   0.0s
[CV 3/5] END svm__C=1.0, svm__kernel=linear; accuracy: (train=0.950, test=0.889) f1: (train=0.959, test=0.907) roc_au
c: (train=0.977, test=0.952) total time=   0.0s
[CV 4/5] END svm__C=1.0, svm__kernel=linear; accuracy: (train=0.939, test=0.933) f1: (train=0.950, test=0.945) roc_au
c: (train=0.979, test=0.974) total time=   0.1s
[CV 5/5] END svm__C=1.0, svm__kernel=linear; accuracy: (train=0.936, test=0.933) f1: (train=0.947, test=0.946) roc_au
c: (train=0.977, test=0.972) total time=   0.0s
[CV 1/5] END svm__C=10.0, svm__kernel=linear; accuracy: (train=0.945, test=0.901) f1: (train=0.954, test=0.917) roc_au
c: (train=0.982, test=0.955) total time=   0.1s
[CV 2/5] END svm__C=10.0, svm__kernel=linear; accuracy: (train=0.942, test=0.945) f1: (train=0.952, test=0.954) roc_au
c: (train=0.978, test=0.976) total time=   0.2s
[CV 3/5] END svm__C=10.0, svm__kernel=linear; accuracy: (train=0.948, test=0.878) f1: (train=0.956, test=0.897) roc_au
c: (train=0.976, test=0.950) total time=   0.1s
[CV 4/5] END svm__C=10.0, svm__kernel=linear; accuracy: (train=0.939, test=0.933) f1: (train=0.950, test=0.945) roc_au
c: (train=0.979, test=0.973) total time=   0.1s
[CV 5/5] END svm__C=10.0, svm__kernel=linear; accuracy: (train=0.939, test=0.933) f1: (train=0.950, test=0.946) roc_au
c: (train=0.975, test=0.971) total time=   0.1s
[CV 1/5] END svm__C=100.0, svm__kernel=linear; accuracy: (train=0.945, test=0.901) f1: (train=0.954, test=0.917) roc_a
uc: (train=0.982, test=0.955) total time=   0.9s
[CV 2/5] END svm__C=100.0, svm__kernel=linear; accuracy: (train=0.939, test=0.945) f1: (train=0.950, test=0.954) roc_a
uc: (train=0.978, test=0.976) total time=   1.0s
[CV 3/5] END svm__C=100.0, svm__kernel=linear; accuracy: (train=0.948, test=0.878) f1: (train=0.956, test=0.897) roc_a
uc: (train=0.976, test=0.950) total time=   1.1s
[CV 4/5] END svm__C=100.0, svm__kernel=linear; accuracy: (train=0.939, test=0.933) f1: (train=0.950, test=0.945) roc_a
uc: (train=0.979, test=0.973) total time=   0.8s
[CV 5/5] END svm__C=100.0, svm__kernel=linear; accuracy: (train=0.945, test=0.933) f1: (train=0.955, test=0.946) roc_a
uc: (train=0.975, test=0.972) total time=   0.6s
[CV 1/5] END svm__C=1000.0, svm__kernel=linear; accuracy: (train=0.945, test=0.901) f1: (train=0.954, test=0.917) roc_
auc: (train=0.982, test=0.955) total time=  12.9s
[CV 2/5] END svm__C=1000.0, svm__kernel=linear; accuracy: (train=0.939, test=0.945) f1: (train=0.950, test=0.954) roc_
auc: (train=0.978, test=0.976) total time=  10.6s
[CV 3/5] END svm__C=1000.0, svm__kernel=linear; accuracy: (train=0.948, test=0.878) f1: (train=0.956, test=0.897) roc_
auc: (train=0.976, test=0.950) total time=  11.7s
[CV 4/5] END svm__C=1000.0, svm__kernel=linear; accuracy: (train=0.939, test=0.933) f1: (train=0.950, test=0.945) roc_
auc: (train=0.978, test=0.973) total time=  10.8s
```

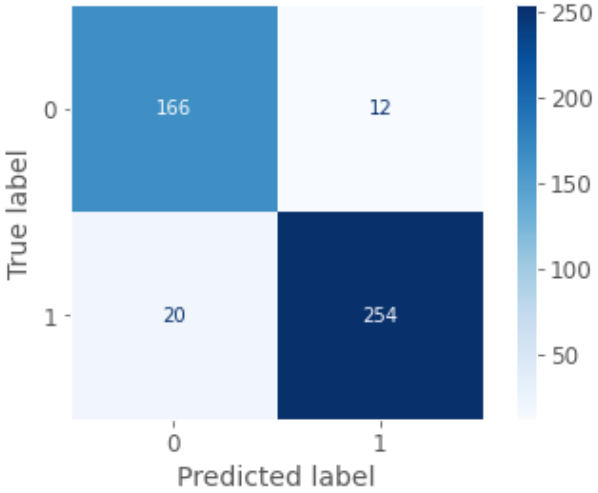
```
[CV 5/5] END svm__C=1000.0, svm__kernel=linear; accuracy: (train=0.945, test=0.933) f1: (train=0.955, test=0.946) roc_
auc: (train=0.975, test=0.972) total time=   6.0s
```

In [25]:
```
print('Test set:')
plot_and_calc (X_test, y_test, y_pred_test, y_pred_proba_test)
print('Train set:')
plot_and_calc (X_train, y_train, y_pred_train, y_pred_proba_train)
```

```
Test set:
```




```
Accuracy is 0.91.
F1 is 0.93.
AUROC is 0.95
Train set:
```




```
Accuracy is 0.94.
F1 is 0.95.
```

```
AUROC is 0.98
```

Non-Linear models

In [26]:
```
# Non Linear SVM - Kernels: rbf and poly

C = np.array([0.001, 0.01, 1, 10, 100, 1000])
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
svc = SVC(probability=True)
pipe = Pipeline(steps=[('scale', StandardScaler()), ('svm', svc)])
svm_nonlin = GridSearchCV(estimator=pipe,
                  param_grid={'svm__kernel':['rbf','poly'], 'svm__C':C, 'svm__degree':[3], 'svm__gamma':['auto','scale
                  scoring=['accuracy','f1','roc_auc'],
                  cv=skf, refit='roc_auc', verbose=3, return_train_score=True)
svm_nonlin.fit(X_train, np.ravel(y_train))
chosen_svm_nonlin = svm_nonlin.best_estimator_
print(svm_nonlin.best_params_)

y_pred_test = chosen_svm_nonlin.predict(X_test)
y_pred_proba_test = chosen_svm_nonlin.predict_proba(X_test)

y_pred_train = chosen_svm_nonlin.predict(X_train)
y_pred_proba_train = chosen_svm_nonlin.predict_proba(X_train)
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
[CV 1/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1: (t
rain=0.755, test=0.753) roc_auc: (train=0.970, test=0.944) total time=   0.1s
[CV 2/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1: (t
rain=0.755, test=0.753) roc_auc: (train=0.971, test=0.986) total time=   0.1s
[CV 3/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.608, test=0.600) f1: (t
rain=0.756, test=0.750) roc_auc: (train=0.974, test=0.942) total time=   0.1s
[CV 4/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1: (t
rain=0.754, test=0.759) roc_auc: (train=0.973, test=0.970) total time=   0.1s
[CV 5/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1: (t
rain=0.754, test=0.759) roc_auc: (train=0.971, test=0.969) total time=   0.1s
[CV 1/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.987, test=0.975) total time=   0.0s
[CV 2/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.990, test=0.991) total time=   0.1s
[CV 3/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.608, test=0.600) f1:
(train=0.756, test=0.750) roc_auc: (train=0.988, test=0.960) total time=   0.0s
[CV 4/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.605, test=0.611) f1:
(train=0.754, test=0.759) roc_auc: (train=0.989, test=0.984) total time=   0.0s
[CV 5/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.605, test=0.611) f1:
(train=0.754, test=0.759) roc_auc: (train=0.989, test=0.980) total time=   0.0s
[CV 1/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.970, test=0.944) total time=   0.1s
[CV 2/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.971, test=0.986) total time=   0.1s
```

```
[CV 3/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.608, test=0.600) f1:
(train=0.756, test=0.750) roc_auc: (train=0.974, test=0.942) total time=   0.1s
[CV 4/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1:
(train=0.754, test=0.759) roc_auc: (train=0.973, test=0.970) total time=   0.1s
[CV 5/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1:
(train=0.754, test=0.759) roc_auc: (train=0.971, test=0.969) total time=   0.1s
[CV 1/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.987, test=0.975) total time=   0.0s
[CV 2/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.990, test=0.991) total time=   0.0s
[CV 3/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.608, test=0.600) f1:
(train=0.756, test=0.750) roc_auc: (train=0.988, test=0.960) total time=   0.0s
[CV 4/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.605, test=0.611) f1:
(train=0.754, test=0.759) roc_auc: (train=0.989, test=0.984) total time=   0.0s
[CV 5/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.605, test=0.611) f1:
(train=0.754, test=0.759) roc_auc: (train=0.989, test=0.980) total time=   0.0s
[CV 1/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1: (tr
ain=0.755, test=0.753) roc_auc: (train=0.970, test=0.944) total time=   0.1s
[CV 2/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1: (tr
ain=0.755, test=0.753) roc_auc: (train=0.971, test=0.986) total time=   0.1s
[CV 3/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.608, test=0.600) f1: (tr
ain=0.756, test=0.750) roc_auc: (train=0.974, test=0.942) total time=   0.1s
[CV 4/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1: (tr
ain=0.754, test=0.759) roc_auc: (train=0.973, test=0.970) total time=   0.1s
[CV 5/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1: (tr
ain=0.754, test=0.759) roc_auc: (train=0.971, test=0.969) total time=   0.1s
[CV 1/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.607, test=0.604) f1: (t
rain=0.755, test=0.753) roc_auc: (train=0.987, test=0.975) total time=   0.0s
[CV 2/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.607, test=0.604) f1: (t
rain=0.755, test=0.753) roc_auc: (train=0.990, test=0.991) total time=   0.0s
[CV 3/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.608, test=0.600) f1: (t
rain=0.756, test=0.750) roc_auc: (train=0.988, test=0.959) total time=   0.0s
[CV 4/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.605, test=0.611) f1: (t
rain=0.754, test=0.759) roc_auc: (train=0.989, test=0.984) total time=   0.0s
[CV 5/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.605, test=0.611) f1: (t
rain=0.754, test=0.759) roc_auc: (train=0.989, test=0.980) total time=   0.0s
[CV 1/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1: (t
rain=0.755, test=0.753) roc_auc: (train=0.970, test=0.944) total time=   0.1s
[CV 2/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1: (t
rain=0.755, test=0.753) roc_auc: (train=0.971, test=0.986) total time=   0.1s
[CV 3/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.608, test=0.600) f1: (t
rain=0.756, test=0.750) roc_auc: (train=0.974, test=0.942) total time=   0.1s
[CV 4/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1: (t
rain=0.754, test=0.759) roc_auc: (train=0.973, test=0.970) total time=   0.1s
[CV 5/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1: (t
rain=0.754, test=0.759) roc_auc: (train=0.971, test=0.969) total time=   0.1s
[CV 1/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.987, test=0.975) total time=   0.0s
[CV 2/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.990, test=0.991) total time=   0.0s
[CV 3/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.608, test=0.600) f1:
(train=0.756, test=0.750) roc_auc: (train=0.988, test=0.959) total time=   0.0s
```

```
[CV 4/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.605, test=0.611) f1:
(train=0.754, test=0.759) roc_auc: (train=0.989, test=0.984) total time=   0.0s
[CV 5/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.605, test=0.611) f1:
(train=0.754, test=0.759) roc_auc: (train=0.989, test=0.980) total time=   0.0s
[CV 1/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.989, test=0.967) f1: (tra
in=0.991, test=0.972) roc_auc: (train=0.999, test=0.994) total time=   0.0s
[CV 2/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.986, test=0.956) f1: (tra
in=0.989, test=0.964) roc_auc: (train=0.999, test=0.995) total time=   0.0s
[CV 3/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.986, test=0.944) f1: (tra
in=0.989, test=0.953) roc_auc: (train=0.999, test=0.993) total time=   0.0s
[CV 4/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.986, test=0.989) f1: (tra
in=0.989, test=0.991) roc_auc: (train=0.999, test=0.997) total time=   0.0s
[CV 5/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.986, test=0.989) f1: (tra
in=0.989, test=0.991) roc_auc: (train=1.000, test=0.998) total time=   0.0s
[CV 1/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.989, test=0.978) f1: (tr
ain=0.991, test=0.982) roc_auc: (train=1.000, test=0.995) total time=   0.0s
[CV 2/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.989, test=0.967) f1: (tr
ain=0.991, test=0.973) roc_auc: (train=1.000, test=0.993) total time=   0.0s
[CV 3/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.992, test=0.956) f1: (tr
ain=0.993, test=0.963) roc_auc: (train=1.000, test=0.987) total time=   0.0s
[CV 4/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.989, test=0.967) f1: (tr
ain=0.991, test=0.972) roc_auc: (train=1.000, test=0.997) total time=   0.0s
[CV 5/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.986, test=0.956) f1: (tr
ain=0.988, test=0.963) roc_auc: (train=1.000, test=0.995) total time=   0.0s
[CV 1/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.989, test=0.967) f1: (tr
ain=0.991, test=0.972) roc_auc: (train=0.999, test=0.994) total time=   0.0s
[CV 2/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.986, test=0.956) f1: (tr
ain=0.989, test=0.964) roc_auc: (train=0.999, test=0.995) total time=   0.0s
[CV 3/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.986, test=0.944) f1: (tr
ain=0.989, test=0.953) roc_auc: (train=0.999, test=0.993) total time=   0.0s
[CV 4/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.986, test=0.989) f1: (tr
ain=0.989, test=0.991) roc_auc: (train=0.999, test=0.997) total time=   0.0s
[CV 5/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.986, test=0.989) f1: (tr
ain=0.989, test=0.991) roc_auc: (train=1.000, test=0.998) total time=   0.0s
[CV 1/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.989, test=0.978) f1: (t
rain=0.991, test=0.982) roc_auc: (train=1.000, test=0.995) total time=   0.0s
[CV 2/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.989, test=0.967) f1: (t
rain=0.991, test=0.973) roc_auc: (train=1.000, test=0.993) total time=   0.0s
[CV 3/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.992, test=0.956) f1: (t
rain=0.993, test=0.963) roc_auc: (train=1.000, test=0.987) total time=   0.0s
[CV 4/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.989, test=0.967) f1: (t
rain=0.991, test=0.972) roc_auc: (train=1.000, test=0.997) total time=   0.0s
[CV 5/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.986, test=0.956) f1: (t
rain=0.988, test=0.963) roc_auc: (train=1.000, test=0.995) total time=   0.0s
[CV 1/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.997, test=0.967) f1: (tr
ain=0.998, test=0.972) roc_auc: (train=1.000, test=0.991) total time=   0.0s
[CV 2/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.994, test=0.956) f1: (tr
ain=0.995, test=0.964) roc_auc: (train=1.000, test=0.995) total time=   0.0s
[CV 3/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.997, test=0.967) f1: (tr
ain=0.998, test=0.972) roc_auc: (train=1.000, test=0.993) total time=   0.0s
[CV 4/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.994, test=0.989) f1: (tr
ain=0.995, test=0.991) roc_auc: (train=1.000, test=1.000) total time=   0.0s
```

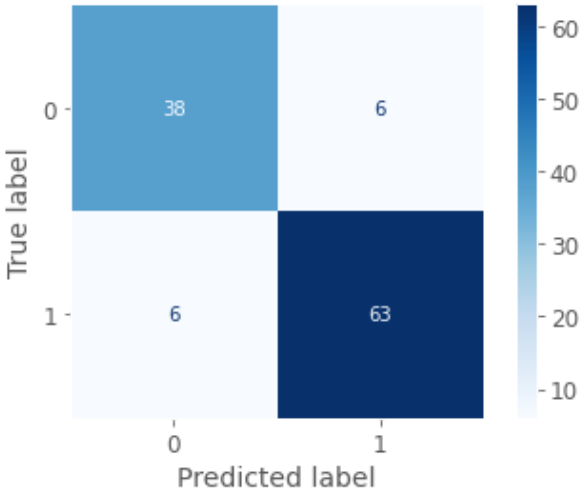
```
[CV 5/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.994, test=0.989) f1: (tr
ain=0.995, test=0.991) roc_auc: (train=1.000, test=0.998) total time=   0.0s
[CV 1/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.997, test=0.956) f1: (t
rain=0.998, test=0.963) roc_auc: (train=1.000, test=0.991) total time=   0.0s
[CV 2/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.994, test=0.967) f1: (t
rain=0.995, test=0.973) roc_auc: (train=1.000, test=0.993) total time=   0.0s
[CV 3/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.997, test=0.967) f1: (t
rain=0.998, test=0.972) roc_auc: (train=1.000, test=0.989) total time=   0.0s
[CV 4/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=1.000, test=0.978) f1: (t
rain=1.000, test=0.982) roc_auc: (train=1.000, test=0.999) total time=   0.0s
[CV 5/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=1.000, test=0.967) f1: (t
rain=1.000, test=0.972) roc_auc: (train=1.000, test=0.997) total time=   0.0s
[CV 1/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.997, test=0.967) f1: (t
rain=0.998, test=0.972) roc_auc: (train=1.000, test=0.991) total time=   0.0s
[CV 2/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.994, test=0.956) f1: (t
rain=0.995, test=0.964) roc_auc: (train=1.000, test=0.995) total time=   0.0s
[CV 3/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.997, test=0.967) f1: (t
rain=0.998, test=0.972) roc_auc: (train=1.000, test=0.993) total time=   0.0s
[CV 4/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.994, test=0.989) f1: (t
rain=0.995, test=0.991) roc_auc: (train=1.000, test=1.000) total time=   0.0s
[CV 5/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.994, test=0.989) f1: (t
rain=0.995, test=0.991) roc_auc: (train=1.000, test=0.998) total time=   0.0s
[CV 1/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.997, test=0.956) f1:
(train=0.998, test=0.963) roc_auc: (train=1.000, test=0.991) total time=   0.0s
[CV 2/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.994, test=0.967) f1:
(train=0.995, test=0.973) roc_auc: (train=1.000, test=0.993) total time=   0.0s
[CV 3/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.997, test=0.967) f1:
(train=0.998, test=0.972) roc_auc: (train=1.000, test=0.989) total time=   0.0s
[CV 4/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=1.000, test=0.978) f1:
(train=1.000, test=0.982) roc_auc: (train=1.000, test=0.999) total time=   0.0s
[CV 5/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=1.000, test=0.967) f1:
(train=1.000, test=0.972) roc_auc: (train=1.000, test=0.997) total time=   0.0s
[CV 1/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.997, test=0.978) f1: (t
rain=0.998, test=0.982) roc_auc: (train=1.000, test=0.992) total time=   0.0s
[CV 2/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.994, test=0.945) f1: (t
rain=0.995, test=0.954) roc_auc: (train=1.000, test=0.992) total time=   0.0s
[CV 3/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.997, test=0.956) f1: (t
rain=0.998, test=0.962) roc_auc: (train=1.000, test=0.995) total time=   0.0s
[CV 4/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.994, test=0.978) f1: (t
rain=0.995, test=0.981) roc_auc: (train=1.000, test=1.000) total time=   0.1s
[CV 5/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.994, test=0.978) f1: (t
rain=0.995, test=0.981) roc_auc: (train=1.000, test=0.998) total time=   0.0s
[CV 1/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.997, test=0.967) f1:
(train=0.998, test=0.972) roc_auc: (train=1.000, test=0.994) total time=   0.0s
[CV 2/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.994, test=0.967) f1:
(train=0.995, test=0.972) roc_auc: (train=1.000, test=0.996) total time=   0.0s
[CV 3/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.997, test=0.956) f1:
(train=0.998, test=0.962) roc_auc: (train=1.000, test=0.989) total time=   0.0s
[CV 4/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.994, test=0.944) f1:
(train=0.995, test=0.953) roc_auc: (train=1.000, test=0.993) total time=   0.0s
[CV 5/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.994, test=0.933) f1:
(train=0.995, test=0.943) roc_auc: (train=1.000, test=0.994) total time=   0.0s
```

```
[CV 1/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.997, test=0.978) f1:
(train=0.998, test=0.982) roc_auc: (train=1.000, test=0.992) total time=   0.1s
[CV 2/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.994, test=0.945) f1:
(train=0.995, test=0.954) roc_auc: (train=1.000, test=0.992) total time=   0.0s
[CV 3/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.997, test=0.956) f1:
(train=0.998, test=0.962) roc_auc: (train=1.000, test=0.995) total time=   0.0s
[CV 4/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.994, test=0.978) f1:
(train=0.995, test=0.981) roc_auc: (train=1.000, test=1.000) total time=   0.0s
[CV 5/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.994, test=0.978) f1:
(train=0.995, test=0.981) roc_auc: (train=1.000, test=0.998) total time=   0.0s
[CV 1/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.997, test=0.967) f1:
(train=0.998, test=0.972) roc_auc: (train=1.000, test=0.994) total time=   0.0s
[CV 2/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.994, test=0.967) f1:
(train=0.995, test=0.972) roc_auc: (train=1.000, test=0.996) total time=   0.0s
[CV 3/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.997, test=0.956) f1:
(train=0.998, test=0.962) roc_auc: (train=1.000, test=0.989) total time=   0.0s
[CV 4/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.994, test=0.944) f1:
(train=0.995, test=0.953) roc_auc: (train=1.000, test=0.993) total time=   0.0s
[CV 5/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.994, test=0.933) f1:
(train=0.995, test=0.943) roc_auc: (train=1.000, test=0.994) total time=   0.0s
[CV 1/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=1.000, test=0.934) f1:
(train=1.000, test=0.943) roc_auc: (train=1.000, test=0.971) total time=   0.0s
[CV 2/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=1.000, test=0.956) f1:
(train=1.000, test=0.963) roc_auc: (train=1.000, test=0.961) total time=   0.0s
[CV 3/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=1.000, test=0.967) f1:
(train=1.000, test=0.971) roc_auc: (train=1.000, test=0.970) total time=   0.0s
[CV 4/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=1.000, test=0.967) f1:
(train=1.000, test=0.972) roc_auc: (train=1.000, test=0.991) total time=   0.0s
[CV 5/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=1.000, test=0.956) f1:
(train=1.000, test=0.963) roc_auc: (train=1.000, test=0.985) total time=   0.0s
[CV 1/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=1.000, test=0.934) f1:
(train=1.000, test=0.945) roc_auc: (train=1.000, test=0.983) total time=   0.0s
[CV 2/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=1.000, test=0.967) f1:
(train=1.000, test=0.972) roc_auc: (train=1.000, test=0.987) total time=   0.0s
[CV 3/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=1.000, test=0.911) f1:
(train=1.000, test=0.925) roc_auc: (train=1.000, test=0.944) total time=   0.0s
[CV 4/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=1.000, test=0.922) f1:
(train=1.000, test=0.936) roc_auc: (train=1.000, test=0.965) total time=   0.0s
[CV 5/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=1.000, test=0.911) f1:
(train=1.000, test=0.925) roc_auc: (train=1.000, test=0.966) total time=   0.0s
[CV 1/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=1.000, test=0.934) f1:
(train=1.000, test=0.943) roc_auc: (train=1.000, test=0.971) total time=   0.0s
[CV 2/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=1.000, test=0.956) f1:
(train=1.000, test=0.963) roc_auc: (train=1.000, test=0.961) total time=   0.0s
[CV 3/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=1.000, test=0.967) f1:
(train=1.000, test=0.971) roc_auc: (train=1.000, test=0.970) total time=   0.0s
[CV 4/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=1.000, test=0.967) f1:
(train=1.000, test=0.972) roc_auc: (train=1.000, test=0.991) total time=   0.0s
[CV 5/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=1.000, test=0.956) f1:
(train=1.000, test=0.963) roc_auc: (train=1.000, test=0.985) total time=   0.0s
[CV 1/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=1.000, test=0.934) f1:
(train=1.000, test=0.945) roc_auc: (train=1.000, test=0.983) total time=   0.0s
```

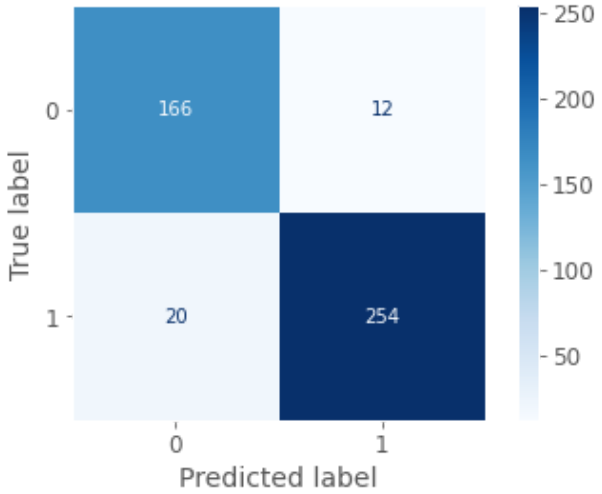
```
[CV 2/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=1.000, test=0.967) f1:
(train=1.000, test=0.972) roc_auc: (train=1.000, test=0.987) total time=   0.0s
[CV 3/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=1.000, test=0.911) f1:
(train=1.000, test=0.925) roc_auc: (train=1.000, test=0.944) total time=   0.0s
[CV 4/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=1.000, test=0.922) f1:
(train=1.000, test=0.936) roc_auc: (train=1.000, test=0.965) total time=   0.0s
[CV 5/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=1.000, test=0.911) f1:
(train=1.000, test=0.925) roc_auc: (train=1.000, test=0.966) total time=   0.0s
{'svm__C': 1.0, 'svm__degree': 3, 'svm__gamma': 'auto', 'svm__kernel': 'rbf'}
```

In [27]:
```
print('Test set:')
plot_and_calc (X_test, y_test, y_pred_test, y_pred_proba_test)
print('Train set:')
plot_and_calc (X_train, y_train, y_pred_train, y_pred_proba_train)
```

```
Test set:
```




```
Accuracy is 0.92.
F1 is 0.93.
AUROC is 0.98
Train set:
```




```
Accuracy is 0.99.
F1 is 0.99.
AUROC is 1.00
```

Question 5) c.

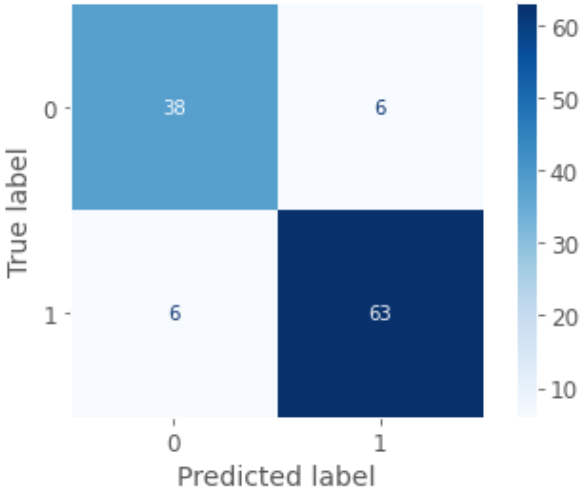
What performs best on this dataset? Linear or non-linear models?

Answer

We can see that the highest values of Accuracy, F1 and AUROC appear all in the non-linear SVM models. According to these results, we can assume that non-linear models perform best on this dataset.

Feature selection

In [28]:
```
# Random Forest Classifier

from sklearn.metrics import plot_confusion_matrix, roc_auc_score,plot_roc_curve
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(X_train, np.ravel(y_train))
y_pred_test = rfc.predict(X_test)
y_pred_proba_test = rfc.predict_proba(X_test)

plot_and_calc (X_test, y_test, y_pred_test, y_pred_proba_test)
```

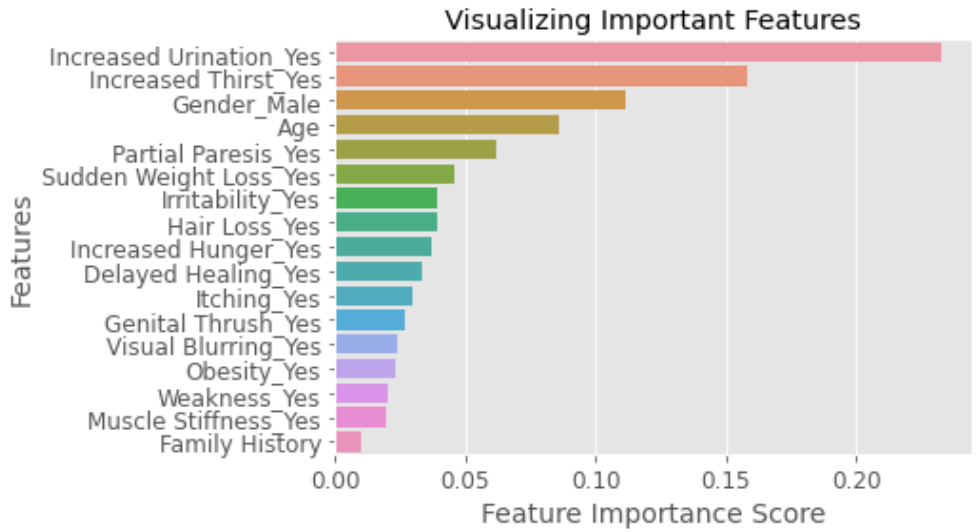



```
Accuracy is 0.96.
F1 is 0.97.
AUROC is 0.99
```

In [29]:
```
feature_imp = pd.Series(rfc.feature_importances_, index = X_train.columns).sort_values(ascending=False)
feature_imp
```

Out[29]:
```
Increased Urination_Yes    0.232665
Increased Thirst_Yes       0.158055
Gender_Male                0.111457
Age                        0.086018
Partial Paresis_Yes        0.061830
Sudden Weight Loss_Yes     0.045881
Irritability_Yes           0.039705
Hair Loss_Yes              0.039455
Increased Hunger_Yes       0.037140
Delayed Healing_Yes        0.033316
Itching_Yes                0.029517
Genital Thrush_Yes         0.026961
Visual Blurring_Yes        0.024305
Obesity_Yes                0.023561
Weakness_Yes               0.020507
Muscle Stiffness_Yes       0.019546
Family History             0.010082
dtype: float64
```

In [30]:
```
sns.barplot(x = feature_imp, y = feature_imp.index)
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
plt.show()
```



Question 6) a

i) What are the 2 most important features according to the random forest?

Answer

The two most important features according to the random forest are:

- Increased Urination
- Increased Thirst

ii) Does this match up exactly with the feature exploration you did?

Answer

Yes, this exactly matches up with the feature exploration we did.

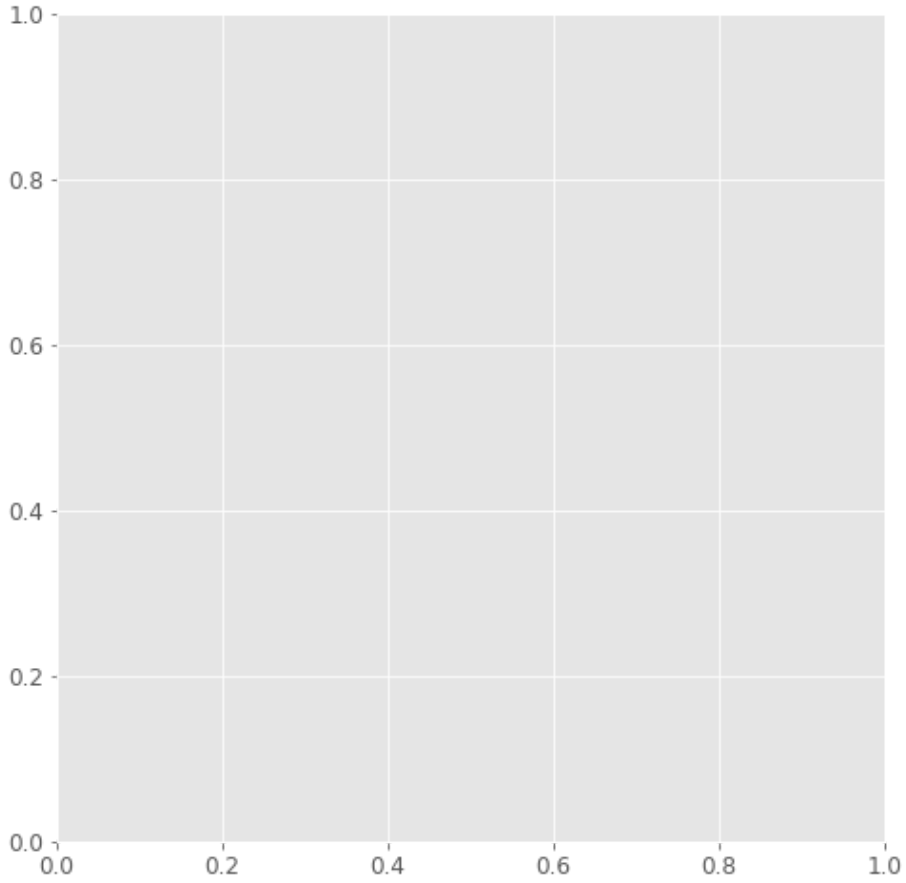
Data Separability Visualization

Dimensionality reduction

In [31]:
```
n_components = 2
pca = PCA(n_components=n_components, whiten=True)

X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)

def plt_2d_pca(X_pca,y):
    fig = plt.figure(figsize=(8, 8))
```

```
    ax = fig.add_subplot(111, aspect='equal')
    ax.scatter(X_pca[y==0, 0], X_pca[y==0, 1], color='b')
    ax.scatter(X_pca[y==1, 0], X_pca[y==1, 1], color='r')
    ax.legend(('Positive Diagnosis','Negative Diagnosis'))
    ax.plot([0], [0], "ko")
    ax.arrow(0, 0, 0, 1, head_width=0.05, length_includes_head=True, head_length=0.1, fc='k', ec='k')
    ax.arrow(0, 0, 1, 0, head_width=0.05, length_includes_head=True, head_length=0.1, fc='k', ec='k')
    ax.set_xlabel('$U_1$')
    ax.set_ylabel('$U_2$')
    ax.set_title('2D PCA')

plt_2d_pca(X_test_pca,y_test)
plt.show()
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-31-b02be8f8921d> in <module>
     18     ax.set_title('2D PCA')
     19
---> 20 plt_2d_pca(X_test_pca,y_test)
     21 plt.show()

<ipython-input-31-b02be8f8921d> in plt_2d_pca(X_pca, y)
      8     fig = plt.figure(figsize=(8, 8))
      9     ax = fig.add_subplot(111, aspect='equal')
---> 10     ax.scatter(X_pca[y==0, 0], X_pca[y==0, 1], color='b')
     11     ax.scatter(X_pca[y==1, 0], X_pca[y==1, 1], color='r')
     12     ax.legend(('Positive Diagnosis','Negative Diagnosis'))

IndexError: too many indices for array: array is 2-dimensional, but 3 were indexed
```



Question 7) b

How separable is your data when reduced to just two features?

Answer

In [32]:
```
def calc (x_test, y_test, y_pred_test, y_pred_proba_test):

    calc_TN = lambda y_true, y_pred: confusion_matrix(y_true, y_pred)[0, 0]
    calc_FP = lambda y_true, y_pred: confusion_matrix(y_true, y_pred)[0, 1]
    calc_FN = lambda y_true, y_pred: confusion_matrix(y_true, y_pred)[1, 0]
    calc_TP = lambda y_true, y_pred: confusion_matrix(y_true, y_pred)[1, 1]

    TN = calc_TN(y_test,y_pred_test)
    FP = calc_FP(y_test,y_pred_test)
    FN = calc_FN(y_test,y_pred_test)
    TP = calc_TP(y_test,y_pred_test)
    Se = TP/(TP+FN)
```

```
    Sp = TN/(TN+FP)
    PPV = TP/(TP+FP)
    NPV = TN/(TN+FN)
    Acc = (TP+TN)/(TP+TN+FP+FN)
    F1 = (2*Se*PPV)/(Se+PPV)
    print('Accuracy is {:.2f}. \nF1 is {:.2f}.'.format(Acc,F1))
    print('AUROC is {:.2f}'.format(roc_auc_score(y_test, y_pred_proba_test[:,1])))
```

Training the same models above on the dimensionality-reduced training set

In [33]:
```
#Logistic regression
lmbda = np.array([0.001, 0.01, 1, 10, 100, 1000])
pen = ['l1', 'l2']
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
log_reg_2d = LogisticRegression(random_state=5, penalty=pen, C = 1/lmbda, solver='saga')
pipe = Pipeline(steps=[('scale', StandardScaler()), ('logistic', log_reg_2d)])
LogReg2D = GridSearchCV(estimator=pipe, param_grid={'logistic__C': 1/lmbda, 'logistic__penalty': pen},
                  scoring=['accuracy','f1','roc_auc'], cv=skf,
                  refit='roc_auc', verbose=3, return_train_score=True)
LogReg2D.fit(X_train_pca, np.ravel(y_train))
chosen_LogReg2D = LogReg2D.best_estimator_

y_pred_test_2d = chosen_LogReg2D.predict(X_test_pca)
y_pred_proba_test_2d = chosen_LogReg2D.predict_proba(X_test_pca)

y_pred_train_2d = chosen_LogReg2D.predict(X_train_pca)
y_pred_proba_train_2d = chosen_LogReg2D.predict_proba(X_train_pca)

print('Test set:')
calc (X_test_pca, y_test, y_pred_test_2d, y_pred_proba_test_2d)
print('Train set:')
calc (X_train_pca, y_train, y_pred_train_2d, y_pred_proba_train_2d)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
[CV 1/5] END logistic__C=1000.0, logistic__penalty=l1; accuracy: (train=0.845, test=0.758) f1: (train=0.872, test=0.80
0) roc_auc: (train=0.931, test=0.868) total time=   0.0s
[CV 2/5] END logistic__C=1000.0, logistic__penalty=l1; accuracy: (train=0.834, test=0.879) f1: (train=0.860, test=0.90
1) roc_auc: (train=0.912, test=0.955) total time=   0.0s
[CV 3/5] END logistic__C=1000.0, logistic__penalty=l1; accuracy: (train=0.837, test=0.822) f1: (train=0.865, test=0.84
9) roc_auc: (train=0.928, test=0.889) total time=   0.0s
[CV 4/5] END logistic__C=1000.0, logistic__penalty=l1; accuracy: (train=0.845, test=0.856) f1: (train=0.872, test=0.87
6) roc_auc: (train=0.917, test=0.932) total time=   0.0s
[CV 5/5] END logistic__C=1000.0, logistic__penalty=l1; accuracy: (train=0.823, test=0.856) f1: (train=0.853, test=0.88
7) roc_auc: (train=0.916, test=0.936) total time=   0.0s
[CV 1/5] END logistic__C=1000.0, logistic__penalty=l2; accuracy: (train=0.845, test=0.758) f1: (train=0.872, test=0.80
0) roc_auc: (train=0.931, test=0.868) total time=   0.0s
[CV 2/5] END logistic__C=1000.0, logistic__penalty=l2; accuracy: (train=0.834, test=0.879) f1: (train=0.860, test=0.90
1) roc_auc: (train=0.912, test=0.955) total time=   0.0s
```

```
[CV 3/5] END logistic__C=1000.0, logistic__penalty=l2; accuracy: (train=0.837, test=0.822) f1: (train=0.865, test=0.84
9) roc_auc: (train=0.928, test=0.889) total time=   0.0s
[CV 4/5] END logistic__C=1000.0, logistic__penalty=l2; accuracy: (train=0.845, test=0.856) f1: (train=0.872, test=0.87
6) roc_auc: (train=0.917, test=0.932) total time=   0.0s
[CV 5/5] END logistic__C=1000.0, logistic__penalty=l2; accuracy: (train=0.823, test=0.856) f1: (train=0.853, test=0.88
7) roc_auc: (train=0.916, test=0.936) total time=   0.0s
[CV 1/5] END logistic__C=100.0, logistic__penalty=l1; accuracy: (train=0.845, test=0.758) f1: (train=0.872, test=0.80
0) roc_auc: (train=0.931, test=0.868) total time=   0.0s
[CV 2/5] END logistic__C=100.0, logistic__penalty=l1; accuracy: (train=0.834, test=0.879) f1: (train=0.860, test=0.90
1) roc_auc: (train=0.912, test=0.955) total time=   0.0s
[CV 3/5] END logistic__C=100.0, logistic__penalty=l1; accuracy: (train=0.837, test=0.822) f1: (train=0.865, test=0.84
9) roc_auc: (train=0.928, test=0.889) total time=   0.0s
[CV 4/5] END logistic__C=100.0, logistic__penalty=l1; accuracy: (train=0.845, test=0.856) f1: (train=0.872, test=0.87
6) roc_auc: (train=0.917, test=0.933) total time=   0.0s
[CV 5/5] END logistic__C=100.0, logistic__penalty=l1; accuracy: (train=0.823, test=0.856) f1: (train=0.853, test=0.88
7) roc_auc: (train=0.916, test=0.936) total time=   0.0s
[CV 1/5] END logistic__C=100.0, logistic__penalty=l2; accuracy: (train=0.845, test=0.758) f1: (train=0.872, test=0.80
0) roc_auc: (train=0.931, test=0.868) total time=   0.0s
[CV 2/5] END logistic__C=100.0, logistic__penalty=l2; accuracy: (train=0.834, test=0.879) f1: (train=0.860, test=0.90
1) roc_auc: (train=0.912, test=0.955) total time=   0.0s
[CV 3/5] END logistic__C=100.0, logistic__penalty=l2; accuracy: (train=0.837, test=0.822) f1: (train=0.865, test=0.84
9) roc_auc: (train=0.928, test=0.889) total time=   0.0s
[CV 4/5] END logistic__C=100.0, logistic__penalty=l2; accuracy: (train=0.840, test=0.856) f1: (train=0.867, test=0.87
6) roc_auc: (train=0.917, test=0.932) total time=   0.0s
[CV 5/5] END logistic__C=100.0, logistic__penalty=l2; accuracy: (train=0.823, test=0.856) f1: (train=0.853, test=0.88
7) roc_auc: (train=0.916, test=0.936) total time=   0.0s
[CV 1/5] END logistic__C=1.0, logistic__penalty=l1; accuracy: (train=0.845, test=0.769) f1: (train=0.872, test=0.807)
roc_auc: (train=0.931, test=0.868) total time=   0.0s
[CV 2/5] END logistic__C=1.0, logistic__penalty=l1; accuracy: (train=0.839, test=0.879) f1: (train=0.866, test=0.901)
roc_auc: (train=0.912, test=0.955) total time=   0.0s
[CV 3/5] END logistic__C=1.0, logistic__penalty=l1; accuracy: (train=0.837, test=0.822) f1: (train=0.865, test=0.849)
roc_auc: (train=0.928, test=0.889) total time=   0.0s
[CV 4/5] END logistic__C=1.0, logistic__penalty=l1; accuracy: (train=0.845, test=0.856) f1: (train=0.872, test=0.876)
roc_auc: (train=0.917, test=0.933) total time=   0.0s
[CV 5/5] END logistic__C=1.0, logistic__penalty=l1; accuracy: (train=0.823, test=0.856) f1: (train=0.853, test=0.887)
roc_auc: (train=0.916, test=0.937) total time=   0.0s
[CV 1/5] END logistic__C=1.0, logistic__penalty=l2; accuracy: (train=0.845, test=0.758) f1: (train=0.872, test=0.800)
roc_auc: (train=0.931, test=0.870) total time=   0.0s
[CV 2/5] END logistic__C=1.0, logistic__penalty=l2; accuracy: (train=0.831, test=0.879) f1: (train=0.858, test=0.901)
roc_auc: (train=0.913, test=0.956) total time=   0.0s
[CV 3/5] END logistic__C=1.0, logistic__penalty=l2; accuracy: (train=0.834, test=0.822) f1: (train=0.863, test=0.849)
roc_auc: (train=0.928, test=0.888) total time=   0.0s
[CV 4/5] END logistic__C=1.0, logistic__penalty=l2; accuracy: (train=0.840, test=0.856) f1: (train=0.867, test=0.876)
roc_auc: (train=0.916, test=0.935) total time=   0.0s
[CV 5/5] END logistic__C=1.0, logistic__penalty=l2; accuracy: (train=0.823, test=0.856) f1: (train=0.853, test=0.887)
roc_auc: (train=0.916, test=0.935) total time=   0.0s
[CV 1/5] END logistic__C=0.1, logistic__penalty=l1; accuracy: (train=0.861, test=0.791) f1: (train=0.884, test=0.822)
roc_auc: (train=0.931, test=0.869) total time=   0.0s
[CV 2/5] END logistic__C=0.1, logistic__penalty=l1; accuracy: (train=0.834, test=0.879) f1: (train=0.862, test=0.901)
roc_auc: (train=0.912, test=0.947) total time=   0.0s
[CV 3/5] END logistic__C=0.1, logistic__penalty=l1; accuracy: (train=0.851, test=0.833) f1: (train=0.876, test=0.857)
roc_auc: (train=0.928, test=0.890) total time=   0.0s
```

```
[CV 4/5] END logistic__C=0.1, logistic__penalty=l1; accuracy: (train=0.843, test=0.856) f1: (train=0.870, test=0.876)
roc_auc: (train=0.917, test=0.929) total time=   0.0s
[CV 5/5] END logistic__C=0.1, logistic__penalty=l1; accuracy: (train=0.837, test=0.844) f1: (train=0.864, test=0.877)
roc_auc: (train=0.916, test=0.941) total time=   0.0s
[CV 1/5] END logistic__C=0.1, logistic__penalty=l2; accuracy: (train=0.845, test=0.758) f1: (train=0.873, test=0.800)
roc_auc: (train=0.929, test=0.865) total time=   0.0s
[CV 2/5] END logistic__C=0.1, logistic__penalty=l2; accuracy: (train=0.820, test=0.868) f1: (train=0.852, test=0.893)
roc_auc: (train=0.911, test=0.957) total time=   0.0s
[CV 3/5] END logistic__C=0.1, logistic__penalty=l2; accuracy: (train=0.831, test=0.822) f1: (train=0.862, test=0.849)
roc_auc: (train=0.924, test=0.886) total time=   0.0s
[CV 4/5] END logistic__C=0.1, logistic__penalty=l2; accuracy: (train=0.823, test=0.844) f1: (train=0.855, test=0.870)
roc_auc: (train=0.916, test=0.938) total time=   0.0s
[CV 5/5] END logistic__C=0.1, logistic__penalty=l2; accuracy: (train=0.820, test=0.856) f1: (train=0.851, test=0.887)
roc_auc: (train=0.914, test=0.931) total time=   0.0s
[CV 1/5] END logistic__C=0.01, logistic__penalty=l1; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.753)
roc_auc: (train=0.929, test=0.871) total time=   0.0s
[CV 2/5] END logistic__C=0.01, logistic__penalty=l1; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.753)
roc_auc: (train=0.912, test=0.947) total time=   0.0s
[CV 3/5] END logistic__C=0.01, logistic__penalty=l1; accuracy: (train=0.608, test=0.600) f1: (train=0.756, test=0.750)
roc_auc: (train=0.925, test=0.894) total time=   0.0s
[CV 4/5] END logistic__C=0.01, logistic__penalty=l1; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.759)
roc_auc: (train=0.917, test=0.929) total time=   0.0s
[CV 5/5] END logistic__C=0.01, logistic__penalty=l1; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.759)
roc_auc: (train=0.912, test=0.945) total time=   0.0s
[CV 1/5] END logistic__C=0.01, logistic__penalty=l2; accuracy: (train=0.850, test=0.769) f1: (train=0.883, test=0.826)
roc_auc: (train=0.926, test=0.853) total time=   0.0s
[CV 2/5] END logistic__C=0.01, logistic__penalty=l2; accuracy: (train=0.820, test=0.868) f1: (train=0.861, test=0.897)
roc_auc: (train=0.908, test=0.955) total time=   0.0s
[CV 3/5] END logistic__C=0.01, logistic__penalty=l2; accuracy: (train=0.837, test=0.822) f1: (train=0.874, test=0.857)
roc_auc: (train=0.919, test=0.884) total time=   0.0s
[CV 4/5] END logistic__C=0.01, logistic__penalty=l2; accuracy: (train=0.815, test=0.878) f1: (train=0.857, test=0.903)
roc_auc: (train=0.911, test=0.942) total time=   0.0s
[CV 5/5] END logistic__C=0.01, logistic__penalty=l2; accuracy: (train=0.829, test=0.811) f1: (train=0.866, test=0.860)
roc_auc: (train=0.911, test=0.924) total time=   0.0s
[CV 1/5] END logistic__C=0.001, logistic__penalty=l1; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.75
3) roc_auc: (train=0.500, test=0.500) total time=   0.0s
[CV 2/5] END logistic__C=0.001, logistic__penalty=l1; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.75
3) roc_auc: (train=0.500, test=0.500) total time=   0.0s
[CV 3/5] END logistic__C=0.001, logistic__penalty=l1; accuracy: (train=0.608, test=0.600) f1: (train=0.756, test=0.75
0) roc_auc: (train=0.500, test=0.500) total time=   0.0s
[CV 4/5] END logistic__C=0.001, logistic__penalty=l1; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.75
9) roc_auc: (train=0.500, test=0.500) total time=   0.0s
[CV 5/5] END logistic__C=0.001, logistic__penalty=l1; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.75
9) roc_auc: (train=0.500, test=0.500) total time=   0.0s
[CV 1/5] END logistic__C=0.001, logistic__penalty=l2; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.75
3) roc_auc: (train=0.924, test=0.851) total time=   0.0s
[CV 2/5] END logistic__C=0.001, logistic__penalty=l2; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.75
3) roc_auc: (train=0.907, test=0.954) total time=   0.0s
[CV 3/5] END logistic__C=0.001, logistic__penalty=l2; accuracy: (train=0.608, test=0.600) f1: (train=0.756, test=0.75
0) roc_auc: (train=0.919, test=0.880) total time=   0.0s
[CV 4/5] END logistic__C=0.001, logistic__penalty=l2; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.75
9) roc_auc: (train=0.910, test=0.942) total time=   0.0s
```

```
[CV 5/5] END logistic__C=0.001, logistic__penalty=l2; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.75
9) roc_auc: (train=0.911, test=0.924) total time=   0.0s
Test set:
Accuracy is 0.79.
F1 is 0.85.
AUROC is 0.91
Train set:
Accuracy is 0.77.
F1 is 0.83.
AUROC is 0.92
```

In [34]:
```
# Linear SVM

C = np.array([0.001, 0.01, 1, 10, 100, 1000])
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
svc = SVC(probability=True)
pipe = Pipeline(steps=[('scale', StandardScaler()), ('svm', svc)])
svm_lin_2d = GridSearchCV(estimator=pipe,
             param_grid={'svm__kernel':['linear'], 'svm__C':C},
             scoring=['accuracy','f1','roc_auc'],
             cv=skf, refit='roc_auc', verbose=3, return_train_score=True)
svm_lin_2d.fit(X_train_pca, np.ravel(y_train))
chosen_svm_lin_2d = svm_lin_2d.best_estimator_

y_pred_test_2d = chosen_svm_lin_2d.predict(X_test_pca)
y_pred_proba_test_2d = chosen_svm_lin_2d.predict_proba(X_test_pca)

y_pred_train_2d = chosen_svm_lin_2d.predict(X_train_pca)
y_pred_proba_train_2d = chosen_svm_lin_2d.predict_proba(X_train_pca)

print('Test set:')
calc (X_test_pca, y_test, y_pred_test_2d, y_pred_proba_test_2d)
print('Train set:')
calc (X_train_pca, y_train, y_pred_train_2d, y_pred_proba_train_2d)
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
[CV 1/5] END svm__C=0.001, svm__kernel=linear; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.753) roc_a
uc: (train=0.916, test=0.831) total time=   0.0s
[CV 2/5] END svm__C=0.001, svm__kernel=linear; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.753) roc_a
uc: (train=0.896, test=0.947) total time=   0.1s
[CV 3/5] END svm__C=0.001, svm__kernel=linear; accuracy: (train=0.608, test=0.600) f1: (train=0.756, test=0.750) roc_a
uc: (train=0.909, test=0.871) total time=   0.0s
[CV 4/5] END svm__C=0.001, svm__kernel=linear; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.759) roc_a
uc: (train=0.899, test=0.938) total time=   0.0s
[CV 5/5] END svm__C=0.001, svm__kernel=linear; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.759) roc_a
uc: (train=0.902, test=0.917) total time=   0.0s
[CV 1/5] END svm__C=0.01, svm__kernel=linear; accuracy: (train=0.848, test=0.725) f1: (train=0.872, test=0.762) roc_au
c: (train=0.920, test=0.841) total time=   0.0s
[CV 2/5] END svm__C=0.01, svm__kernel=linear; accuracy: (train=0.812, test=0.879) f1: (train=0.840, test=0.899) roc_au
```

```
c: (train=0.898, test=0.949) total time=   0.0s
[CV 3/5] END svm__C=0.01, svm__kernel=linear; accuracy: (train=0.829, test=0.789) f1: (train=0.856, test=0.816) roc_au
c: (train=0.913, test=0.875) total time=   0.0s
[CV 4/5] END svm__C=0.01, svm__kernel=linear; accuracy: (train=0.815, test=0.844) f1: (train=0.845, test=0.870) roc_au
c: (train=0.903, test=0.942) total time=   0.0s
[CV 5/5] END svm__C=0.01, svm__kernel=linear; accuracy: (train=0.815, test=0.867) f1: (train=0.842, test=0.893) roc_au
c: (train=0.904, test=0.921) total time=   0.0s
[CV 1/5] END svm__C=1.0, svm__kernel=linear; accuracy: (train=0.861, test=0.791) f1: (train=0.884, test=0.822) roc_au
c: (train=0.931, test=0.868) total time=   0.0s
[CV 2/5] END svm__C=1.0, svm__kernel=linear; accuracy: (train=0.837, test=0.901) f1: (train=0.863, test=0.917) roc_au
c: (train=0.912, test=0.946) total time=   0.0s
[CV 3/5] END svm__C=1.0, svm__kernel=linear; accuracy: (train=0.851, test=0.833) f1: (train=0.876, test=0.857) roc_au
c: (train=0.927, test=0.892) total time=   0.0s
[CV 4/5] END svm__C=1.0, svm__kernel=linear; accuracy: (train=0.848, test=0.844) f1: (train=0.874, test=0.865) roc_au
c: (train=0.916, test=0.929) total time=   0.0s
[CV 5/5] END svm__C=1.0, svm__kernel=linear; accuracy: (train=0.840, test=0.856) f1: (train=0.864, test=0.885) roc_au
c: (train=0.915, test=0.941) total time=   0.0s
[CV 1/5] END svm__C=10.0, svm__kernel=linear; accuracy: (train=0.861, test=0.791) f1: (train=0.884, test=0.822) roc_au
c: (train=0.931, test=0.867) total time=   0.0s
[CV 2/5] END svm__C=10.0, svm__kernel=linear; accuracy: (train=0.837, test=0.901) f1: (train=0.863, test=0.917) roc_au
c: (train=0.911, test=0.946) total time=   0.0s
[CV 3/5] END svm__C=10.0, svm__kernel=linear; accuracy: (train=0.851, test=0.833) f1: (train=0.876, test=0.857) roc_au
c: (train=0.927, test=0.892) total time=   0.0s
[CV 4/5] END svm__C=10.0, svm__kernel=linear; accuracy: (train=0.848, test=0.844) f1: (train=0.874, test=0.865) roc_au
c: (train=0.916, test=0.929) total time=   0.0s
[CV 5/5] END svm__C=10.0, svm__kernel=linear; accuracy: (train=0.840, test=0.856) f1: (train=0.864, test=0.885) roc_au
c: (train=0.915, test=0.941) total time=   0.0s
[CV 1/5] END svm__C=100.0, svm__kernel=linear; accuracy: (train=0.861, test=0.791) f1: (train=0.884, test=0.822) roc_a
uc: (train=0.931, test=0.867) total time=   0.1s
[CV 2/5] END svm__C=100.0, svm__kernel=linear; accuracy: (train=0.837, test=0.901) f1: (train=0.863, test=0.917) roc_a
uc: (train=0.911, test=0.946) total time=   0.1s
[CV 3/5] END svm__C=100.0, svm__kernel=linear; accuracy: (train=0.851, test=0.833) f1: (train=0.876, test=0.857) roc_a
uc: (train=0.927, test=0.892) total time=   0.1s
[CV 4/5] END svm__C=100.0, svm__kernel=linear; accuracy: (train=0.848, test=0.844) f1: (train=0.874, test=0.865) roc_a
uc: (train=0.916, test=0.929) total time=   0.1s
[CV 5/5] END svm__C=100.0, svm__kernel=linear; accuracy: (train=0.840, test=0.856) f1: (train=0.864, test=0.885) roc_a
uc: (train=0.915, test=0.941) total time=   0.1s
[CV 1/5] END svm__C=1000.0, svm__kernel=linear; accuracy: (train=0.861, test=0.791) f1: (train=0.884, test=0.822) roc_
auc: (train=0.931, test=0.867) total time=   0.5s
[CV 2/5] END svm__C=1000.0, svm__kernel=linear; accuracy: (train=0.837, test=0.901) f1: (train=0.863, test=0.917) roc_
auc: (train=0.911, test=0.946) total time=   0.8s
[CV 3/5] END svm__C=1000.0, svm__kernel=linear; accuracy: (train=0.851, test=0.833) f1: (train=0.876, test=0.857) roc_
auc: (train=0.927, test=0.892) total time=   0.7s
[CV 4/5] END svm__C=1000.0, svm__kernel=linear; accuracy: (train=0.848, test=0.844) f1: (train=0.874, test=0.865) roc_
auc: (train=0.916, test=0.929) total time=   1.2s
[CV 5/5] END svm__C=1000.0, svm__kernel=linear; accuracy: (train=0.840, test=0.856) f1: (train=0.864, test=0.885) roc_
auc: (train=0.915, test=0.941) total time=   1.0s
Test set:
Accuracy is 0.82.
F1 is 0.86.
AUROC is 0.91
Train set:
```

```
Accuracy is 0.85.
F1 is 0.88.
AUROC is 0.92
```

In [35]:
```
# Non Linear SVM - Kernels: rbf and poly

C = np.array([0.001, 0.01, 1, 10, 100, 1000])
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
svc = SVC(probability=True)
pipe = Pipeline(steps=[('scale', StandardScaler()), ('svm', svc)])
svm_nonlin_2d = GridSearchCV(estimator=pipe,
                 param_grid={'svm__kernel':['rbf','poly'], 'svm__C':C, 'svm__degree':[3], 'svm__gamma':['auto','scale
                 scoring=['accuracy','f1','roc_auc'],
                 cv=skf, refit='roc_auc', verbose=3, return_train_score=True)
svm_nonlin_2d.fit(X_train_pca, np.ravel(y_train))
chosen_svm_nonlin_2d = svm_nonlin_2d.best_estimator_
print(svm_nonlin_2d.best_params_)

y_pred_test_2d = chosen_svm_nonlin_2d.predict(X_test_pca)
y_pred_proba_test_2d = chosen_svm_nonlin_2d.predict_proba(X_test_pca)

y_pred_train_2d = chosen_svm_nonlin_2d.predict(X_train_pca)
y_pred_proba_train_2d = chosen_svm_nonlin_2d.predict_proba(X_train_pca)

print('Test set:')
calc (X_test_pca, y_test, y_pred_test_2d, y_pred_proba_test_2d)
print('Train set:')
calc (X_train_pca, y_train, y_pred_train_2d, y_pred_proba_train_2d)
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
[CV 1/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1: (t
rain=0.755, test=0.753) roc_auc: (train=0.921, test=0.846) total time=   0.1s
[CV 2/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1: (t
rain=0.755, test=0.753) roc_auc: (train=0.908, test=0.954) total time=   0.1s
[CV 3/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.608, test=0.600) f1: (t
rain=0.756, test=0.750) roc_auc: (train=0.916, test=0.885) total time=   0.1s
[CV 4/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1: (t
rain=0.754, test=0.759) roc_auc: (train=0.913, test=0.925) total time=   0.1s
[CV 5/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1: (t
rain=0.754, test=0.759) roc_auc: (train=0.913, test=0.918) total time=   0.0s
[CV 1/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.895, test=0.818) total time=   0.0s
[CV 2/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.871, test=0.923) total time=   0.0s
[CV 3/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.608, test=0.600) f1:
(train=0.756, test=0.750) roc_auc: (train=0.893, test=0.824) total time=   0.0s
[CV 4/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.605, test=0.611) f1:
(train=0.754, test=0.759) roc_auc: (train=0.874, test=0.923) total time=   0.0s
[CV 5/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.605, test=0.611) f1:
```

```
(train=0.754, test=0.759) roc_auc: (train=0.881, test=0.891) total time=   0.0s
[CV 1/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.921, test=0.846) total time=   0.1s
[CV 2/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.908, test=0.954) total time=   0.0s
[CV 3/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.608, test=0.600) f1:
(train=0.756, test=0.750) roc_auc: (train=0.916, test=0.885) total time=   0.0s
[CV 4/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1:
(train=0.754, test=0.759) roc_auc: (train=0.913, test=0.925) total time=   0.0s
[CV 5/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1:
(train=0.754, test=0.759) roc_auc: (train=0.913, test=0.918) total time=   0.0s
[CV 1/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.895, test=0.818) total time=   0.0s
[CV 2/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.871, test=0.923) total time=   0.0s
[CV 3/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.608, test=0.600) f1:
(train=0.756, test=0.750) roc_auc: (train=0.893, test=0.824) total time=   0.0s
[CV 4/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.605, test=0.611) f1:
(train=0.754, test=0.759) roc_auc: (train=0.874, test=0.923) total time=   0.0s
[CV 5/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.605, test=0.611) f1:
(train=0.754, test=0.759) roc_auc: (train=0.881, test=0.891) total time=   0.0s
[CV 1/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1: (tr
ain=0.755, test=0.753) roc_auc: (train=0.921, test=0.846) total time=   0.0s
[CV 2/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1: (tr
ain=0.755, test=0.753) roc_auc: (train=0.908, test=0.953) total time=   0.0s
[CV 3/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.608, test=0.600) f1: (tr
ain=0.756, test=0.750) roc_auc: (train=0.916, test=0.885) total time=   0.1s
[CV 4/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1: (tr
ain=0.754, test=0.759) roc_auc: (train=0.913, test=0.925) total time=   0.0s
[CV 5/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1: (tr
ain=0.754, test=0.759) roc_auc: (train=0.913, test=0.918) total time=   0.0s
[CV 1/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.726, test=0.681) f1: (t
rain=0.810, test=0.788) roc_auc: (train=0.898, test=0.821) total time=   0.0s
[CV 2/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.709, test=0.736) f1: (t
rain=0.804, test=0.821) roc_auc: (train=0.874, test=0.926) total time=   0.0s
[CV 3/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.729, test=0.667) f1: (t
rain=0.814, test=0.769) roc_auc: (train=0.895, test=0.827) total time=   0.0s
[CV 4/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.707, test=0.756) f1: (t
rain=0.802, test=0.833) roc_auc: (train=0.875, test=0.922) total time=   0.0s
[CV 5/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.704, test=0.733) f1: (t
rain=0.800, test=0.821) roc_auc: (train=0.881, test=0.892) total time=   0.0s
[CV 1/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1: (t
rain=0.755, test=0.753) roc_auc: (train=0.921, test=0.846) total time=   0.0s
[CV 2/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1: (t
rain=0.755, test=0.753) roc_auc: (train=0.908, test=0.953) total time=   0.0s
[CV 3/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.608, test=0.600) f1: (t
rain=0.756, test=0.750) roc_auc: (train=0.916, test=0.885) total time=   0.0s
[CV 4/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1: (t
rain=0.754, test=0.759) roc_auc: (train=0.913, test=0.925) total time=   0.1s
[CV 5/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1: (t
rain=0.754, test=0.759) roc_auc: (train=0.913, test=0.918) total time=   0.1s
[CV 1/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.726, test=0.681) f1:
```

```
(train=0.810, test=0.788) roc_auc: (train=0.898, test=0.821) total time=   0.0s
[CV 2/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.709, test=0.736) f1:
(train=0.804, test=0.821) roc_auc: (train=0.874, test=0.926) total time=   0.0s
[CV 3/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.729, test=0.667) f1:
(train=0.814, test=0.769) roc_auc: (train=0.895, test=0.827) total time=   0.0s
[CV 4/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.707, test=0.756) f1:
(train=0.802, test=0.833) roc_auc: (train=0.875, test=0.922) total time=   0.0s
[CV 5/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.704, test=0.733) f1:
(train=0.800, test=0.821) roc_auc: (train=0.881, test=0.892) total time=   0.0s
[CV 1/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.878, test=0.802) f1: (tra
in=0.897, test=0.833) roc_auc: (train=0.941, test=0.879) total time=   0.0s
[CV 2/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.848, test=0.879) f1: (tra
in=0.871, test=0.899) roc_auc: (train=0.928, test=0.963) total time=   0.0s
[CV 3/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.865, test=0.856) f1: (tra
in=0.888, test=0.871) roc_auc: (train=0.938, test=0.918) total time=   0.0s
[CV 4/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.851, test=0.867) f1: (tra
in=0.876, test=0.887) roc_auc: (train=0.933, test=0.930) total time=   0.0s
[CV 5/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.859, test=0.822) f1: (tra
in=0.881, test=0.857) roc_auc: (train=0.935, test=0.935) total time=   0.0s
[CV 1/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.817, test=0.758) f1: (tr
ain=0.862, test=0.823) roc_auc: (train=0.934, test=0.857) total time=   0.0s
[CV 2/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.798, test=0.857) f1: (tr
ain=0.849, test=0.893) roc_auc: (train=0.902, test=0.954) total time=   0.0s
[CV 3/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.815, test=0.789) f1: (tr
ain=0.860, test=0.838) roc_auc: (train=0.920, test=0.881) total time=   0.0s
[CV 4/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.798, test=0.867) f1: (tr
ain=0.850, test=0.898) roc_auc: (train=0.917, test=0.946) total time=   0.0s
[CV 5/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.812, test=0.811) f1: (tr
ain=0.859, test=0.862) roc_auc: (train=0.918, test=0.938) total time=   0.0s
[CV 1/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.878, test=0.802) f1: (tr
ain=0.897, test=0.833) roc_auc: (train=0.941, test=0.879) total time=   0.0s
[CV 2/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.848, test=0.879) f1: (tr
ain=0.871, test=0.899) roc_auc: (train=0.928, test=0.963) total time=   0.0s
[CV 3/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.865, test=0.856) f1: (tr
ain=0.888, test=0.871) roc_auc: (train=0.938, test=0.918) total time=   0.0s
[CV 4/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.851, test=0.867) f1: (tr
ain=0.876, test=0.887) roc_auc: (train=0.933, test=0.930) total time=   0.0s
[CV 5/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.859, test=0.822) f1: (tr
ain=0.881, test=0.857) roc_auc: (train=0.935, test=0.935) total time=   0.0s
[CV 1/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.817, test=0.758) f1: (t
rain=0.862, test=0.823) roc_auc: (train=0.934, test=0.857) total time=   0.0s
[CV 2/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.798, test=0.857) f1: (t
rain=0.849, test=0.893) roc_auc: (train=0.902, test=0.954) total time=   0.0s
[CV 3/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.815, test=0.789) f1: (t
rain=0.860, test=0.838) roc_auc: (train=0.920, test=0.881) total time=   0.0s
[CV 4/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.798, test=0.867) f1: (t
rain=0.850, test=0.898) roc_auc: (train=0.917, test=0.946) total time=   0.0s
[CV 5/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.812, test=0.811) f1: (t
rain=0.859, test=0.862) roc_auc: (train=0.918, test=0.938) total time=   0.0s
[CV 1/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.886, test=0.813) f1: (tr
ain=0.904, test=0.841) roc_auc: (train=0.954, test=0.899) total time=   0.0s
[CV 2/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.870, test=0.879) f1: (tr
```

```
ain=0.888, test=0.899) roc_auc: (train=0.948, test=0.959) total time=   0.1s
[CV 3/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.865, test=0.856) f1: (tr
ain=0.886, test=0.871) roc_auc: (train=0.948, test=0.941) total time=   0.0s
[CV 4/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.867, test=0.856) f1: (tr
ain=0.887, test=0.879) roc_auc: (train=0.949, test=0.936) total time=   0.0s
[CV 5/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.870, test=0.867) f1: (tr
ain=0.888, test=0.889) roc_auc: (train=0.947, test=0.945) total time=   0.0s
[CV 1/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.825, test=0.758) f1: (t
rain=0.868, test=0.823) roc_auc: (train=0.936, test=0.857) total time=   0.1s
[CV 2/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.814, test=0.879) f1: (t
rain=0.858, test=0.906) roc_auc: (train=0.901, test=0.951) total time=   0.1s
[CV 3/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.826, test=0.800) f1: (t
rain=0.866, test=0.839) roc_auc: (train=0.917, test=0.877) total time=   0.0s
[CV 4/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.820, test=0.856) f1: (t
rain=0.862, test=0.889) roc_auc: (train=0.903, test=0.942) total time=   0.1s
[CV 5/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.829, test=0.811) f1: (t
rain=0.867, test=0.860) roc_auc: (train=0.912, test=0.923) total time=   0.1s
[CV 1/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.886, test=0.813) f1: (t
rain=0.904, test=0.841) roc_auc: (train=0.954, test=0.899) total time=   0.0s
[CV 2/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.870, test=0.879) f1: (t
rain=0.888, test=0.899) roc_auc: (train=0.948, test=0.959) total time=   0.1s
[CV 3/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.865, test=0.856) f1: (t
rain=0.886, test=0.871) roc_auc: (train=0.948, test=0.941) total time=   0.0s
[CV 4/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.867, test=0.856) f1: (t
rain=0.887, test=0.879) roc_auc: (train=0.949, test=0.936) total time=   0.0s
[CV 5/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.870, test=0.867) f1: (t
rain=0.888, test=0.889) roc_auc: (train=0.947, test=0.945) total time=   0.0s
[CV 1/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.825, test=0.758) f1:
(train=0.868, test=0.823) roc_auc: (train=0.936, test=0.857) total time=   0.1s
[CV 2/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.814, test=0.879) f1:
(train=0.858, test=0.906) roc_auc: (train=0.901, test=0.951) total time=   0.1s
[CV 3/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.826, test=0.800) f1:
(train=0.866, test=0.839) roc_auc: (train=0.917, test=0.877) total time=   0.0s
[CV 4/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.820, test=0.856) f1:
(train=0.862, test=0.889) roc_auc: (train=0.903, test=0.942) total time=   0.1s
[CV 5/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.829, test=0.811) f1:
(train=0.867, test=0.860) roc_auc: (train=0.912, test=0.923) total time=   0.1s
[CV 1/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.884, test=0.835) f1: (t
rain=0.902, test=0.860) roc_auc: (train=0.960, test=0.922) total time=   0.0s
[CV 2/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.875, test=0.868) f1: (t
rain=0.894, test=0.891) roc_auc: (train=0.953, test=0.951) total time=   0.0s
[CV 3/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.870, test=0.856) f1: (t
rain=0.890, test=0.871) roc_auc: (train=0.955, test=0.941) total time=   0.0s
[CV 4/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.878, test=0.867) f1: (t
rain=0.897, test=0.891) roc_auc: (train=0.953, test=0.936) total time=   0.0s
[CV 5/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.878, test=0.856) f1: (t
rain=0.897, test=0.879) roc_auc: (train=0.952, test=0.957) total time=   0.0s
[CV 1/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.837, test=0.769) f1:
(train=0.874, test=0.826) roc_auc: (train=0.931, test=0.858) total time=   0.9s
[CV 2/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.806, test=0.879) f1:
(train=0.850, test=0.906) roc_auc: (train=0.901, test=0.951) total time=   0.6s
[CV 3/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.826, test=0.800) f1:
```

```
(train=0.866, test=0.839) roc_auc: (train=0.917, test=0.877) total time=   0.3s
[CV 4/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.820, test=0.856) f1:
(train=0.862, test=0.889) roc_auc: (train=0.904, test=0.943) total time=   0.7s
[CV 5/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.829, test=0.811) f1:
(train=0.867, test=0.860) roc_auc: (train=0.912, test=0.923) total time=   0.5s
[CV 1/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.884, test=0.835) f1:
(train=0.902, test=0.860) roc_auc: (train=0.960, test=0.922) total time=   0.0s
[CV 2/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.875, test=0.868) f1:
(train=0.894, test=0.891) roc_auc: (train=0.953, test=0.951) total time=   0.0s
[CV 3/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.870, test=0.856) f1:
(train=0.890, test=0.871) roc_auc: (train=0.955, test=0.941) total time=   0.0s
[CV 4/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.878, test=0.867) f1:
(train=0.897, test=0.891) roc_auc: (train=0.953, test=0.936) total time=   0.0s
[CV 5/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.878, test=0.856) f1:
(train=0.897, test=0.879) roc_auc: (train=0.952, test=0.957) total time=   0.0s
[CV 1/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.837, test=0.769) f1:
(train=0.874, test=0.826) roc_auc: (train=0.931, test=0.858) total time=   0.6s
[CV 2/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.806, test=0.879) f1:
(train=0.850, test=0.906) roc_auc: (train=0.901, test=0.951) total time=   0.6s
[CV 3/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.826, test=0.800) f1:
(train=0.866, test=0.839) roc_auc: (train=0.917, test=0.877) total time=   0.2s
[CV 4/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.820, test=0.856) f1:
(train=0.862, test=0.889) roc_auc: (train=0.904, test=0.943) total time=   1.2s
[CV 5/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.829, test=0.811) f1:
(train=0.867, test=0.860) roc_auc: (train=0.912, test=0.923) total time=   0.6s
[CV 1/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.895, test=0.857) f1:
(train=0.913, test=0.879) roc_auc: (train=0.962, test=0.915) total time=   0.1s
[CV 2/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.867, test=0.901) f1:
(train=0.888, test=0.919) roc_auc: (train=0.955, test=0.945) total time=   0.1s
[CV 3/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.881, test=0.833) f1:
(train=0.902, test=0.851) roc_auc: (train=0.957, test=0.923) total time=   0.1s
[CV 4/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.878, test=0.856) f1:
(train=0.897, test=0.883) roc_auc: (train=0.959, test=0.941) total time=   0.1s
[CV 5/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.873, test=0.833) f1:
(train=0.893, test=0.860) roc_auc: (train=0.949, test=0.944) total time=   0.1s
[CV 1/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.837, test=0.769) f1:
(train=0.874, test=0.826) roc_auc: (train=0.931, test=0.859) total time=   5.4s
[CV 2/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.806, test=0.879) f1:
(train=0.850, test=0.906) roc_auc: (train=0.901, test=0.951) total time=   4.1s
[CV 3/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.826, test=0.800) f1:
(train=0.866, test=0.839) roc_auc: (train=0.917, test=0.877) total time=   2.9s
[CV 4/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.820, test=0.856) f1:
(train=0.862, test=0.889) roc_auc: (train=0.904, test=0.943) total time=   8.5s
[CV 5/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.829, test=0.811) f1:
(train=0.867, test=0.860) roc_auc: (train=0.912, test=0.923) total time=   4.1s
[CV 1/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.895, test=0.857) f1:
(train=0.913, test=0.879) roc_auc: (train=0.962, test=0.915) total time=   0.1s
[CV 2/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.867, test=0.901) f1:
(train=0.888, test=0.919) roc_auc: (train=0.955, test=0.945) total time=   0.2s
[CV 3/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.881, test=0.833) f1:
(train=0.902, test=0.851) roc_auc: (train=0.957, test=0.923) total time=   0.1s
[CV 4/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.878, test=0.856) f1:
```

```
(train=0.897, test=0.883) roc_auc: (train=0.959, test=0.941) total time=   0.1s
[CV 5/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.873, test=0.833) f1:
(train=0.893, test=0.860) roc_auc: (train=0.949, test=0.944) total time=   0.1s
[CV 1/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.837, test=0.769) f1:
(train=0.874, test=0.826) roc_auc: (train=0.931, test=0.859) total time=   9.8s
[CV 2/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.806, test=0.879) f1:
(train=0.850, test=0.906) roc_auc: (train=0.901, test=0.951) total time=   4.9s
[CV 3/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.826, test=0.800) f1:
(train=0.866, test=0.839) roc_auc: (train=0.917, test=0.877) total time=   3.1s
[CV 4/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.820, test=0.856) f1:
(train=0.862, test=0.889) roc_auc: (train=0.904, test=0.943) total time=   6.3s
[CV 5/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.829, test=0.811) f1:
(train=0.867, test=0.860) roc_auc: (train=0.912, test=0.923) total time=   9.8s
{'svm__C': 100.0, 'svm__degree': 3, 'svm__gamma': 'auto', 'svm__kernel': 'rbf'}
Test set:
Accuracy is 0.81.
F1 is 0.84.
AUROC is 0.93
Train set:
Accuracy is 0.87.
F1 is 0.89.
AUROC is 0.95
```

Training the same models on the best two features from section 6

In [36]:
```
# Best two features form section 6: Increased Urination, Increased Thirst

X_train_2d = X_train[['Increased Thirst_Yes','Increased Urination_Yes']]
X_test_2d = X_test[['Increased Thirst_Yes','Increased Urination_Yes']]
```

In [37]:
```
#Logistic regression
lmbda = np.array([0.001, 0.01, 1, 10, 100, 1000])
pen = ['l1', 'l2']
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
log_reg_2d = LogisticRegression(random_state=5, penalty=pen, C = 1/lmbda, solver='saga')
pipe = Pipeline(steps=[('scale', StandardScaler()), ('logistic', log_reg_2d)])
LogReg2D = GridSearchCV(estimator=pipe, param_grid={'logistic__C': 1/lmbda, 'logistic__penalty': pen},
                  scoring=['accuracy','f1','roc_auc'], cv=skf,
                  refit='roc_auc', verbose=3, return_train_score=True)
LogReg2D.fit(X_train_2d, np.ravel(y_train))
chosen_LogReg2D = LogReg2D.best_estimator_

y_pred_test_2d = chosen_LogReg2D.predict(X_test_2d)
y_pred_proba_test_2d = chosen_LogReg2D.predict_proba(X_test_2d)

y_pred_train_2d = chosen_LogReg2D.predict(X_train_2d)
y_pred_proba_train_2d = chosen_LogReg2D.predict_proba(X_train_2d)
```

```
print('Test set:')
calc (X_test_2d, y_test, y_pred_test_2d, y_pred_proba_test_2d)
print('Train set:')
calc (X_train_2d, y_train, y_pred_train_2d, y_pred_proba_train_2d)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
[CV 1/5] END logistic__C=1000.0, logistic__penalty=l1; accuracy: (train=0.867, test=0.846) f1: (train=0.887, test=0.86
8) roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END logistic__C=1000.0, logistic__penalty=l1; accuracy: (train=0.856, test=0.890) f1: (train=0.879, test=0.90
2) roc_auc: (train=0.904, test=0.912) total time=   0.0s
[CV 3/5] END logistic__C=1000.0, logistic__penalty=l1; accuracy: (train=0.870, test=0.833) f1: (train=0.890, test=0.85
4) roc_auc: (train=0.911, test=0.864) total time=   0.0s
[CV 4/5] END logistic__C=1000.0, logistic__penalty=l1; accuracy: (train=0.851, test=0.911) f1: (train=0.873, test=0.92
6) roc_auc: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END logistic__C=1000.0, logistic__penalty=l1; accuracy: (train=0.870, test=0.833) f1: (train=0.888, test=0.86
7) roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END logistic__C=1000.0, logistic__penalty=l2; accuracy: (train=0.867, test=0.846) f1: (train=0.887, test=0.86
8) roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END logistic__C=1000.0, logistic__penalty=l2; accuracy: (train=0.856, test=0.890) f1: (train=0.879, test=0.90
2) roc_auc: (train=0.904, test=0.912) total time=   0.0s
[CV 3/5] END logistic__C=1000.0, logistic__penalty=l2; accuracy: (train=0.870, test=0.833) f1: (train=0.890, test=0.85
4) roc_auc: (train=0.911, test=0.864) total time=   0.0s
[CV 4/5] END logistic__C=1000.0, logistic__penalty=l2; accuracy: (train=0.851, test=0.911) f1: (train=0.873, test=0.92
6) roc_auc: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END logistic__C=1000.0, logistic__penalty=l2; accuracy: (train=0.870, test=0.833) f1: (train=0.888, test=0.86
7) roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END logistic__C=100.0, logistic__penalty=l1; accuracy: (train=0.867, test=0.846) f1: (train=0.887, test=0.86
8) roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END logistic__C=100.0, logistic__penalty=l1; accuracy: (train=0.856, test=0.890) f1: (train=0.879, test=0.90
2) roc_auc: (train=0.904, test=0.912) total time=   0.0s
[CV 3/5] END logistic__C=100.0, logistic__penalty=l1; accuracy: (train=0.870, test=0.833) f1: (train=0.890, test=0.85
4) roc_auc: (train=0.911, test=0.864) total time=   0.0s
[CV 4/5] END logistic__C=100.0, logistic__penalty=l1; accuracy: (train=0.851, test=0.911) f1: (train=0.873, test=0.92
6) roc_auc: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END logistic__C=100.0, logistic__penalty=l1; accuracy: (train=0.870, test=0.833) f1: (train=0.888, test=0.86
7) roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END logistic__C=100.0, logistic__penalty=l2; accuracy: (train=0.867, test=0.846) f1: (train=0.887, test=0.86
8) roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END logistic__C=100.0, logistic__penalty=l2; accuracy: (train=0.856, test=0.890) f1: (train=0.879, test=0.90
2) roc_auc: (train=0.904, test=0.912) total time=   0.0s
[CV 3/5] END logistic__C=100.0, logistic__penalty=l2; accuracy: (train=0.870, test=0.833) f1: (train=0.890, test=0.85
4) roc_auc: (train=0.911, test=0.864) total time=   0.0s
[CV 4/5] END logistic__C=100.0, logistic__penalty=l2; accuracy: (train=0.851, test=0.911) f1: (train=0.873, test=0.92
6) roc_auc: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END logistic__C=100.0, logistic__penalty=l2; accuracy: (train=0.870, test=0.833) f1: (train=0.888, test=0.86
7) roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END logistic__C=1.0, logistic__penalty=l1; accuracy: (train=0.867, test=0.846) f1: (train=0.887, test=0.868)
roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END logistic__C=1.0, logistic__penalty=l1; accuracy: (train=0.856, test=0.890) f1: (train=0.879, test=0.902)
roc_auc: (train=0.904, test=0.912) total time=   0.0s
[CV 3/5] END logistic__C=1.0, logistic__penalty=l1; accuracy: (train=0.870, test=0.833) f1: (train=0.890, test=0.854)
roc_auc: (train=0.911, test=0.864) total time=   0.0s
```

```
[CV 4/5] END logistic__C=1.0, logistic__penalty=l1; accuracy: (train=0.851, test=0.911) f1: (train=0.873, test=0.926)
roc_auc: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END logistic__C=1.0, logistic__penalty=l1; accuracy: (train=0.870, test=0.833) f1: (train=0.888, test=0.867)
roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END logistic__C=1.0, logistic__penalty=l2; accuracy: (train=0.867, test=0.846) f1: (train=0.887, test=0.868)
roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END logistic__C=1.0, logistic__penalty=l2; accuracy: (train=0.856, test=0.890) f1: (train=0.879, test=0.902)
roc_auc: (train=0.904, test=0.912) total time=   0.0s
[CV 3/5] END logistic__C=1.0, logistic__penalty=l2; accuracy: (train=0.870, test=0.833) f1: (train=0.890, test=0.854)
roc_auc: (train=0.911, test=0.864) total time=   0.0s
[CV 4/5] END logistic__C=1.0, logistic__penalty=l2; accuracy: (train=0.851, test=0.911) f1: (train=0.873, test=0.926)
roc_auc: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END logistic__C=1.0, logistic__penalty=l2; accuracy: (train=0.870, test=0.833) f1: (train=0.888, test=0.867)
roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END logistic__C=0.1, logistic__penalty=l1; accuracy: (train=0.867, test=0.846) f1: (train=0.887, test=0.868)
roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END logistic__C=0.1, logistic__penalty=l1; accuracy: (train=0.856, test=0.890) f1: (train=0.879, test=0.902)
roc_auc: (train=0.904, test=0.912) total time=   0.0s
[CV 3/5] END logistic__C=0.1, logistic__penalty=l1; accuracy: (train=0.870, test=0.833) f1: (train=0.890, test=0.854)
roc_auc: (train=0.911, test=0.864) total time=   0.0s
[CV 4/5] END logistic__C=0.1, logistic__penalty=l1; accuracy: (train=0.851, test=0.911) f1: (train=0.873, test=0.926)
roc_auc: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END logistic__C=0.1, logistic__penalty=l1; accuracy: (train=0.870, test=0.833) f1: (train=0.888, test=0.867)
roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END logistic__C=0.1, logistic__penalty=l2; accuracy: (train=0.867, test=0.846) f1: (train=0.887, test=0.868)
roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END logistic__C=0.1, logistic__penalty=l2; accuracy: (train=0.856, test=0.890) f1: (train=0.879, test=0.902)
roc_auc: (train=0.904, test=0.912) total time=   0.0s
[CV 3/5] END logistic__C=0.1, logistic__penalty=l2; accuracy: (train=0.870, test=0.833) f1: (train=0.890, test=0.854)
roc_auc: (train=0.911, test=0.864) total time=   0.0s
[CV 4/5] END logistic__C=0.1, logistic__penalty=l2; accuracy: (train=0.851, test=0.911) f1: (train=0.873, test=0.926)
roc_auc: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END logistic__C=0.1, logistic__penalty=l2; accuracy: (train=0.870, test=0.833) f1: (train=0.888, test=0.867)
roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END logistic__C=0.01, logistic__penalty=l1; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.753)
roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END logistic__C=0.01, logistic__penalty=l1; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.753)
roc_auc: (train=0.846, test=0.850) total time=   0.0s
[CV 3/5] END logistic__C=0.01, logistic__penalty=l1; accuracy: (train=0.608, test=0.600) f1: (train=0.756, test=0.750)
roc_auc: (train=0.911, test=0.864) total time=   0.0s
[CV 4/5] END logistic__C=0.01, logistic__penalty=l1; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.759)
roc_auc: (train=0.837, test=0.886) total time=   0.0s
[CV 5/5] END logistic__C=0.01, logistic__penalty=l1; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.759)
roc_auc: (train=0.857, test=0.805) total time=   0.0s
[CV 1/5] END logistic__C=0.01, logistic__penalty=l2; accuracy: (train=0.867, test=0.846) f1: (train=0.887, test=0.868)
roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END logistic__C=0.01, logistic__penalty=l2; accuracy: (train=0.856, test=0.890) f1: (train=0.879, test=0.902)
roc_auc: (train=0.904, test=0.912) total time=   0.0s
[CV 3/5] END logistic__C=0.01, logistic__penalty=l2; accuracy: (train=0.870, test=0.833) f1: (train=0.890, test=0.854)
roc_auc: (train=0.911, test=0.864) total time=   0.0s
[CV 4/5] END logistic__C=0.01, logistic__penalty=l2; accuracy: (train=0.851, test=0.911) f1: (train=0.873, test=0.926)
roc_auc: (train=0.896, test=0.943) total time=   0.0s
```

```
[CV 5/5] END logistic__C=0.01, logistic__penalty=l2; accuracy: (train=0.870, test=0.833) f1: (train=0.888, test=0.867)
roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END logistic__C=0.001, logistic__penalty=l1; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.75
3) roc_auc: (train=0.500, test=0.500) total time=   0.0s
[CV 2/5] END logistic__C=0.001, logistic__penalty=l1; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.75
3) roc_auc: (train=0.500, test=0.500) total time=   0.0s
[CV 3/5] END logistic__C=0.001, logistic__penalty=l1; accuracy: (train=0.608, test=0.600) f1: (train=0.756, test=0.75
0) roc_auc: (train=0.500, test=0.500) total time=   0.0s
[CV 4/5] END logistic__C=0.001, logistic__penalty=l1; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.75
9) roc_auc: (train=0.500, test=0.500) total time=   0.0s
[CV 5/5] END logistic__C=0.001, logistic__penalty=l1; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.75
9) roc_auc: (train=0.500, test=0.500) total time=   0.0s
[CV 1/5] END logistic__C=0.001, logistic__penalty=l2; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.75
3) roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END logistic__C=0.001, logistic__penalty=l2; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.75
3) roc_auc: (train=0.904, test=0.912) total time=   0.0s
[CV 3/5] END logistic__C=0.001, logistic__penalty=l2; accuracy: (train=0.608, test=0.600) f1: (train=0.756, test=0.75
0) roc_auc: (train=0.911, test=0.864) total time=   0.0s
[CV 4/5] END logistic__C=0.001, logistic__penalty=l2; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.75
9) roc_auc: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END logistic__C=0.001, logistic__penalty=l2; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.75
9) roc_auc: (train=0.908, test=0.897) total time=   0.0s
Test set:
Accuracy is 0.83.
F1 is 0.86.
AUROC is 0.88
Train set:
Accuracy is 0.86.
F1 is 0.88.
AUROC is 0.91
```

In [38]:

```
# Linear SVM

C = np.array([0.001, 0.01, 1, 10, 100, 1000])
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
svc = SVC(probability=True)
pipe = Pipeline(steps=[('scale', StandardScaler()), ('svm', svc)])
svm_lin_2d = GridSearchCV(estimator=pipe,
             param_grid={'svm__kernel':['linear'], 'svm__C':C},
             scoring=['accuracy','f1','roc_auc'],
             cv=skf, refit='roc_auc', verbose=3, return_train_score=True)
svm_lin_2d.fit(X_train_2d, np.ravel(y_train))
chosen_svm_lin_2d = svm_lin_2d.best_estimator_

y_pred_test_2d = chosen_svm_lin_2d.predict(X_test_2d)
y_pred_proba_test_2d = chosen_svm_lin_2d.predict_proba(X_test_2d)

y_pred_train_2d = chosen_svm_lin_2d.predict(X_train_2d)
y_pred_proba_train_2d = chosen_svm_lin_2d.predict_proba(X_train_2d)
```

```
print('Test set:')
calc (X_test_2d, y_test, y_pred_test_2d, y_pred_proba_test_2d)
print('Train set:')
calc (X_train_2d, y_train, y_pred_train_2d, y_pred_proba_train_2d)
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
[CV 1/5] END svm__C=0.001, svm__kernel=linear; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.753) roc_a
uc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END svm__C=0.001, svm__kernel=linear; accuracy: (train=0.607, test=0.604) f1: (train=0.755, test=0.753) roc_a
uc: (train=0.904, test=0.912) total time=   0.1s
[CV 3/5] END svm__C=0.001, svm__kernel=linear; accuracy: (train=0.608, test=0.600) f1: (train=0.756, test=0.750) roc_a
uc: (train=0.909, test=0.889) total time=   0.0s
[CV 4/5] END svm__C=0.001, svm__kernel=linear; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.759) roc_a
uc: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END svm__C=0.001, svm__kernel=linear; accuracy: (train=0.605, test=0.611) f1: (train=0.754, test=0.759) roc_a
uc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END svm__C=0.01, svm__kernel=linear; accuracy: (train=0.834, test=0.802) f1: (train=0.848, test=0.816) roc_au
c: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END svm__C=0.01, svm__kernel=linear; accuracy: (train=0.828, test=0.824) f1: (train=0.843, test=0.833) roc_au
c: (train=0.904, test=0.912) total time=   0.0s
[CV 3/5] END svm__C=0.01, svm__kernel=linear; accuracy: (train=0.823, test=0.844) f1: (train=0.839, test=0.851) roc_au
c: (train=0.909, test=0.889) total time=   0.0s
[CV 4/5] END svm__C=0.01, svm__kernel=linear; accuracy: (train=0.818, test=0.867) f1: (train=0.832, test=0.880) roc_au
c: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END svm__C=0.01, svm__kernel=linear; accuracy: (train=0.834, test=0.800) f1: (train=0.845, test=0.827) roc_au
c: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END svm__C=1.0, svm__kernel=linear; accuracy: (train=0.867, test=0.846) f1: (train=0.887, test=0.868) roc_au
c: (train=0.907, test=0.883) total time=   0.0s
[CV 2/5] END svm__C=1.0, svm__kernel=linear; accuracy: (train=0.856, test=0.890) f1: (train=0.879, test=0.902) roc_au
c: (train=0.898, test=0.915) total time=   0.0s
[CV 3/5] END svm__C=1.0, svm__kernel=linear; accuracy: (train=0.870, test=0.833) f1: (train=0.890, test=0.854) roc_au
c: (train=0.911, test=0.864) total time=   0.0s
[CV 4/5] END svm__C=1.0, svm__kernel=linear; accuracy: (train=0.851, test=0.911) f1: (train=0.873, test=0.926) roc_au
c: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END svm__C=1.0, svm__kernel=linear; accuracy: (train=0.870, test=0.833) f1: (train=0.888, test=0.867) roc_au
c: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END svm__C=10.0, svm__kernel=linear; accuracy: (train=0.867, test=0.846) f1: (train=0.887, test=0.868) roc_au
c: (train=0.907, test=0.883) total time=   0.0s
[CV 2/5] END svm__C=10.0, svm__kernel=linear; accuracy: (train=0.856, test=0.890) f1: (train=0.879, test=0.902) roc_au
c: (train=0.898, test=0.915) total time=   0.0s
[CV 3/5] END svm__C=10.0, svm__kernel=linear; accuracy: (train=0.870, test=0.833) f1: (train=0.890, test=0.854) roc_au
c: (train=0.911, test=0.864) total time=   0.0s
[CV 4/5] END svm__C=10.0, svm__kernel=linear; accuracy: (train=0.851, test=0.911) f1: (train=0.873, test=0.926) roc_au
c: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END svm__C=10.0, svm__kernel=linear; accuracy: (train=0.870, test=0.833) f1: (train=0.888, test=0.867) roc_au
c: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END svm__C=100.0, svm__kernel=linear; accuracy: (train=0.867, test=0.846) f1: (train=0.887, test=0.868) roc_a
uc: (train=0.907, test=0.883) total time=   0.0s
[CV 2/5] END svm__C=100.0, svm__kernel=linear; accuracy: (train=0.856, test=0.890) f1: (train=0.879, test=0.902) roc_a
uc: (train=0.898, test=0.915) total time=   0.0s
```

```
[CV 3/5] END svm__C=100.0, svm__kernel=linear; accuracy: (train=0.870, test=0.833) f1: (train=0.890, test=0.854) roc_a
uc: (train=0.911, test=0.864) total time=   0.0s
[CV 4/5] END svm__C=100.0, svm__kernel=linear; accuracy: (train=0.851, test=0.911) f1: (train=0.873, test=0.926) roc_a
uc: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END svm__C=100.0, svm__kernel=linear; accuracy: (train=0.870, test=0.833) f1: (train=0.888, test=0.867) roc_a
uc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END svm__C=1000.0, svm__kernel=linear; accuracy: (train=0.867, test=0.846) f1: (train=0.887, test=0.868) roc_
auc: (train=0.907, test=0.883) total time=   0.0s
[CV 2/5] END svm__C=1000.0, svm__kernel=linear; accuracy: (train=0.856, test=0.890) f1: (train=0.879, test=0.902) roc_
auc: (train=0.898, test=0.915) total time=   0.0s
[CV 3/5] END svm__C=1000.0, svm__kernel=linear; accuracy: (train=0.870, test=0.833) f1: (train=0.890, test=0.854) roc_
auc: (train=0.911, test=0.864) total time=   0.0s
[CV 4/5] END svm__C=1000.0, svm__kernel=linear; accuracy: (train=0.851, test=0.911) f1: (train=0.873, test=0.926) roc_
auc: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END svm__C=1000.0, svm__kernel=linear; accuracy: (train=0.870, test=0.833) f1: (train=0.888, test=0.867) roc_
auc: (train=0.908, test=0.897) total time=   0.0s
Test set:
Accuracy is 0.61.
F1 is 0.76.
AUROC is 0.88
Train set:
Accuracy is 0.61.
F1 is 0.75.
AUROC is 0.91
/opt/anaconda3/envs/bm-336546-hw2/lib/python3.7/site-packages/ipykernel_launcher.py:15: RuntimeWarning: invalid value
encountered in long_scalars
  from ipykernel import kernelapp as app
/opt/anaconda3/envs/bm-336546-hw2/lib/python3.7/site-packages/ipykernel_launcher.py:15: RuntimeWarning: invalid value
encountered in long_scalars
  from ipykernel import kernelapp as app
```

In [39]:

```
# Non Linear SVM - Kernels: rbf and poly

C = np.array([0.001, 0.01, 1, 10, 100, 1000])
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
svc = SVC(probability=True)
pipe = Pipeline(steps=[('scale', StandardScaler()), ('svm', svc)])
svm_nonlin_2d = GridSearchCV(estimator=pipe,
                 param_grid={'svm__kernel':['rbf','poly'], 'svm__C':C, 'svm__degree':[3], 'svm__gamma':['auto','scale
                 scoring=['accuracy','f1','roc_auc'],
                 cv=skf, refit='roc_auc', verbose=3, return_train_score=True)
svm_nonlin_2d.fit(X_train_2d, np.ravel(y_train))
chosen_svm_nonlin_2d = svm_nonlin_2d.best_estimator_
print(svm_nonlin_2d.best_params_)

y_pred_test_2d = chosen_svm_nonlin_2d.predict(X_test_2d)
y_pred_proba_test_2d = chosen_svm_nonlin_2d.predict_proba(X_test_2d)

y_pred_train_2d = chosen_svm_nonlin_2d.predict(X_train_2d)
```

```
y_pred_proba_train_2d = chosen_svm_nonlin_2d.predict_proba(X_train_2d)

print('Test set:')
calc (X_test_2d, y_test, y_pred_test_2d, y_pred_proba_test_2d)
print('Train set:')
calc (X_train_2d, y_train, y_pred_train_2d, y_pred_proba_train_2d)
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
[CV 1/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1: (t
rain=0.755, test=0.753) roc_auc: (train=0.910, test=0.886) total time=   0.1s
[CV 2/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1: (t
rain=0.755, test=0.753) roc_auc: (train=0.904, test=0.912) total time=   0.1s
[CV 3/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.608, test=0.600) f1: (t
rain=0.756, test=0.750) roc_auc: (train=0.909, test=0.889) total time=   0.1s
[CV 4/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1: (t
rain=0.754, test=0.759) roc_auc: (train=0.896, test=0.943) total time=   0.1s
[CV 5/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1: (t
rain=0.754, test=0.759) roc_auc: (train=0.908, test=0.897) total time=   0.1s
[CV 1/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.904, test=0.912) total time=   0.0s
[CV 3/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.608, test=0.600) f1:
(train=0.756, test=0.750) roc_auc: (train=0.911, test=0.864) total time=   0.0s
[CV 4/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.605, test=0.611) f1:
(train=0.754, test=0.759) roc_auc: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.605, test=0.611) f1:
(train=0.754, test=0.759) roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.910, test=0.886) total time=   0.1s
[CV 2/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.904, test=0.912) total time=   0.1s
[CV 3/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.608, test=0.600) f1:
(train=0.756, test=0.750) roc_auc: (train=0.909, test=0.889) total time=   0.1s
[CV 4/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1:
(train=0.754, test=0.759) roc_auc: (train=0.896, test=0.943) total time=   0.1s
[CV 5/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.605, test=0.611) f1:
(train=0.754, test=0.759) roc_auc: (train=0.908, test=0.897) total time=   0.1s
[CV 1/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.607, test=0.604) f1:
(train=0.755, test=0.753) roc_auc: (train=0.904, test=0.912) total time=   0.0s
[CV 3/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.608, test=0.600) f1:
(train=0.756, test=0.750) roc_auc: (train=0.911, test=0.864) total time=   0.0s
[CV 4/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.605, test=0.611) f1:
(train=0.754, test=0.759) roc_auc: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.605, test=0.611) f1:
(train=0.754, test=0.759) roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.867, test=0.846) f1: (tr
ain=0.887, test=0.868) roc_auc: (train=0.910, test=0.886) total time=   0.1s
[CV 2/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.856, test=0.890) f1: (tr
```

```
ain=0.879, test=0.902) roc_auc: (train=0.904, test=0.912) total time=   0.1s
[CV 3/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1: (tr
ain=0.890, test=0.854) roc_auc: (train=0.909, test=0.889) total time=   0.1s
[CV 4/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.851, test=0.911) f1: (tr
ain=0.873, test=0.926) roc_auc: (train=0.896, test=0.943) total time=   0.1s
[CV 5/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1: (tr
ain=0.888, test=0.867) roc_auc: (train=0.908, test=0.897) total time=   0.1s
[CV 1/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.781, test=0.703) f1: (t
rain=0.780, test=0.675) roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.762, test=0.780) f1: (t
rain=0.756, test=0.778) roc_auc: (train=0.904, test=0.912) total time=   0.0s
[CV 3/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.773, test=0.733) f1: (t
rain=0.771, test=0.714) roc_auc: (train=0.911, test=0.864) total time=   0.0s
[CV 4/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.751, test=0.822) f1: (t
rain=0.741, test=0.830) roc_auc: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.760, test=0.789) f1: (t
rain=0.752, test=0.791) roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.867, test=0.846) f1: (t
rain=0.887, test=0.868) roc_auc: (train=0.910, test=0.886) total time=   0.1s
[CV 2/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.856, test=0.890) f1: (t
rain=0.879, test=0.902) roc_auc: (train=0.904, test=0.912) total time=   0.1s
[CV 3/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1: (t
rain=0.890, test=0.854) roc_auc: (train=0.909, test=0.889) total time=   0.1s
[CV 4/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.851, test=0.911) f1: (t
rain=0.873, test=0.926) roc_auc: (train=0.896, test=0.943) total time=   0.1s
[CV 5/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1: (t
rain=0.888, test=0.867) roc_auc: (train=0.908, test=0.897) total time=   0.1s
[CV 1/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.781, test=0.703) f1:
(train=0.780, test=0.675) roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.762, test=0.780) f1:
(train=0.756, test=0.778) roc_auc: (train=0.904, test=0.912) total time=   0.0s
[CV 3/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.773, test=0.733) f1:
(train=0.771, test=0.714) roc_auc: (train=0.911, test=0.864) total time=   0.0s
[CV 4/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.751, test=0.822) f1:
(train=0.741, test=0.830) roc_auc: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.760, test=0.789) f1:
(train=0.752, test=0.791) roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.867, test=0.846) f1: (tra
in=0.887, test=0.868) roc_auc: (train=0.829, test=0.815) total time=   0.0s
[CV 2/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.856, test=0.890) f1: (tra
in=0.879, test=0.902) roc_auc: (train=0.898, test=0.915) total time=   0.0s
[CV 3/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1: (tra
in=0.890, test=0.854) roc_auc: (train=0.832, test=0.787) total time=   0.0s
[CV 4/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.851, test=0.911) f1: (tra
in=0.873, test=0.926) roc_auc: (train=0.854, test=0.923) total time=   0.0s
[CV 5/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1: (tra
in=0.888, test=0.867) roc_auc: (train=0.902, test=0.905) total time=   0.0s
[CV 1/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.867, test=0.846) f1: (tr
ain=0.887, test=0.868) roc_auc: (train=0.907, test=0.883) total time=   0.0s
[CV 2/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.856, test=0.890) f1: (tr
ain=0.879, test=0.902) roc_auc: (train=0.904, test=0.912) total time=   0.0s
[CV 3/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.870, test=0.833) f1: (tr
```

```
ain=0.890, test=0.854) roc_auc: (train=0.909, test=0.889) total time=   0.0s
[CV 4/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.851, test=0.911) f1: (tr
ain=0.873, test=0.926) roc_auc: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.870, test=0.833) f1: (tr
ain=0.888, test=0.867) roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.867, test=0.846) f1: (tr
ain=0.887, test=0.868) roc_auc: (train=0.829, test=0.815) total time=   0.0s
[CV 2/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.856, test=0.890) f1: (tr
ain=0.879, test=0.902) roc_auc: (train=0.898, test=0.915) total time=   0.0s
[CV 3/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1: (tr
ain=0.890, test=0.854) roc_auc: (train=0.832, test=0.787) total time=   0.0s
[CV 4/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.851, test=0.911) f1: (tr
ain=0.873, test=0.926) roc_auc: (train=0.854, test=0.923) total time=   0.0s
[CV 5/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1: (tr
ain=0.888, test=0.867) roc_auc: (train=0.902, test=0.905) total time=   0.0s
[CV 1/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.867, test=0.846) f1: (t
rain=0.887, test=0.868) roc_auc: (train=0.907, test=0.883) total time=   0.0s
[CV 2/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.856, test=0.890) f1: (t
rain=0.879, test=0.902) roc_auc: (train=0.904, test=0.912) total time=   0.0s
[CV 3/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.870, test=0.833) f1: (t
rain=0.890, test=0.854) roc_auc: (train=0.909, test=0.889) total time=   0.0s
[CV 4/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.851, test=0.911) f1: (t
rain=0.873, test=0.926) roc_auc: (train=0.896, test=0.943) total time=   0.0s
[CV 5/5] END svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.870, test=0.833) f1: (t
rain=0.888, test=0.867) roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.867, test=0.846) f1: (tr
ain=0.887, test=0.868) roc_auc: (train=0.862, test=0.854) total time=   0.0s
[CV 2/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.856, test=0.890) f1: (tr
ain=0.879, test=0.902) roc_auc: (train=0.898, test=0.915) total time=   0.0s
[CV 3/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1: (tr
ain=0.890, test=0.854) roc_auc: (train=0.881, test=0.787) total time=   0.0s
[CV 4/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.851, test=0.911) f1: (tr
ain=0.873, test=0.926) roc_auc: (train=0.809, test=0.880) total time=   0.0s
[CV 5/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1: (tr
ain=0.888, test=0.867) roc_auc: (train=0.864, test=0.849) total time=   0.0s
[CV 1/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.867, test=0.846) f1: (t
rain=0.887, test=0.868) roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.856, test=0.890) f1: (t
rain=0.879, test=0.902) roc_auc: (train=0.898, test=0.915) total time=   0.0s
[CV 3/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.870, test=0.833) f1: (t
rain=0.890, test=0.854) roc_auc: (train=0.909, test=0.889) total time=   0.0s
[CV 4/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.851, test=0.911) f1: (t
rain=0.873, test=0.926) roc_auc: (train=0.892, test=0.941) total time=   0.0s
[CV 5/5] END svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.870, test=0.833) f1: (t
rain=0.888, test=0.867) roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.867, test=0.846) f1: (t
rain=0.887, test=0.868) roc_auc: (train=0.862, test=0.854) total time=   0.0s
[CV 2/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.856, test=0.890) f1: (t
rain=0.879, test=0.902) roc_auc: (train=0.898, test=0.915) total time=   0.0s
[CV 3/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1: (t
rain=0.890, test=0.854) roc_auc: (train=0.881, test=0.787) total time=   0.0s
[CV 4/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.851, test=0.911) f1: (t
```

```
rain=0.873, test=0.926) roc_auc: (train=0.809, test=0.880) total time=   0.0s
[CV 5/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1: (t
rain=0.888, test=0.867) roc_auc: (train=0.864, test=0.849) total time=   0.0s
[CV 1/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.867, test=0.846) f1:
(train=0.887, test=0.868) roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.856, test=0.890) f1:
(train=0.879, test=0.902) roc_auc: (train=0.898, test=0.915) total time=   0.0s
[CV 3/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.870, test=0.833) f1:
(train=0.890, test=0.854) roc_auc: (train=0.909, test=0.889) total time=   0.0s
[CV 4/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.851, test=0.911) f1:
(train=0.873, test=0.926) roc_auc: (train=0.892, test=0.941) total time=   0.0s
[CV 5/5] END svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.870, test=0.833) f1:
(train=0.888, test=0.867) roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.867, test=0.846) f1: (t
rain=0.887, test=0.868) roc_auc: (train=0.907, test=0.883) total time=   0.0s
[CV 2/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.856, test=0.890) f1: (t
rain=0.879, test=0.902) roc_auc: (train=0.810, test=0.894) total time=   0.0s
[CV 3/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1: (t
rain=0.890, test=0.854) roc_auc: (train=0.881, test=0.787) total time=   0.0s
[CV 4/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.851, test=0.911) f1: (t
rain=0.873, test=0.926) roc_auc: (train=0.809, test=0.880) total time=   0.0s
[CV 5/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1: (t
rain=0.888, test=0.867) roc_auc: (train=0.864, test=0.849) total time=   0.0s
[CV 1/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.867, test=0.846) f1:
(train=0.887, test=0.868) roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.856, test=0.890) f1:
(train=0.879, test=0.902) roc_auc: (train=0.898, test=0.915) total time=   0.0s
[CV 3/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.870, test=0.833) f1:
(train=0.890, test=0.854) roc_auc: (train=0.909, test=0.889) total time=   0.0s
[CV 4/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.851, test=0.911) f1:
(train=0.873, test=0.926) roc_auc: (train=0.892, test=0.941) total time=   0.0s
[CV 5/5] END svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.870, test=0.833) f1:
(train=0.888, test=0.867) roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.867, test=0.846) f1:
(train=0.887, test=0.868) roc_auc: (train=0.907, test=0.883) total time=   0.0s
[CV 2/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.856, test=0.890) f1:
(train=0.879, test=0.902) roc_auc: (train=0.810, test=0.894) total time=   0.0s
[CV 3/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1:
(train=0.890, test=0.854) roc_auc: (train=0.881, test=0.787) total time=   0.0s
[CV 4/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.851, test=0.911) f1:
(train=0.873, test=0.926) roc_auc: (train=0.809, test=0.880) total time=   0.0s
[CV 5/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1:
(train=0.888, test=0.867) roc_auc: (train=0.864, test=0.849) total time=   0.0s
[CV 1/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.867, test=0.846) f1:
(train=0.887, test=0.868) roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.856, test=0.890) f1:
(train=0.879, test=0.902) roc_auc: (train=0.898, test=0.915) total time=   0.0s
[CV 3/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.870, test=0.833) f1:
(train=0.890, test=0.854) roc_auc: (train=0.909, test=0.889) total time=   0.0s
[CV 4/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.851, test=0.911) f1:
(train=0.873, test=0.926) roc_auc: (train=0.892, test=0.941) total time=   0.0s
[CV 5/5] END svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.870, test=0.833) f1:
```

```
(train=0.888, test=0.867) roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.867, test=0.846) f1:
(train=0.887, test=0.868) roc_auc: (train=0.862, test=0.854) total time=   0.0s
[CV 2/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.856, test=0.890) f1:
(train=0.879, test=0.902) roc_auc: (train=0.847, test=0.915) total time=   0.0s
[CV 3/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1:
(train=0.890, test=0.854) roc_auc: (train=0.881, test=0.787) total time=   0.0s
[CV 4/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.851, test=0.911) f1:
(train=0.873, test=0.926) roc_auc: (train=0.892, test=0.941) total time=   0.0s
[CV 5/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1:
(train=0.888, test=0.867) roc_auc: (train=0.864, test=0.849) total time=   0.0s
[CV 1/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.867, test=0.846) f1:
(train=0.887, test=0.868) roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.856, test=0.890) f1:
(train=0.879, test=0.902) roc_auc: (train=0.898, test=0.915) total time=   0.0s
[CV 3/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.870, test=0.833) f1:
(train=0.890, test=0.854) roc_auc: (train=0.909, test=0.889) total time=   0.0s
[CV 4/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.851, test=0.911) f1:
(train=0.873, test=0.926) roc_auc: (train=0.892, test=0.941) total time=   0.0s
[CV 5/5] END svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly; accuracy: (train=0.870, test=0.833) f1:
(train=0.888, test=0.867) roc_auc: (train=0.908, test=0.897) total time=   0.0s
[CV 1/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.867, test=0.846) f1:
(train=0.887, test=0.868) roc_auc: (train=0.862, test=0.854) total time=   0.0s
[CV 2/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.856, test=0.890) f1:
(train=0.879, test=0.902) roc_auc: (train=0.847, test=0.915) total time=   0.0s
[CV 3/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1:
(train=0.890, test=0.854) roc_auc: (train=0.881, test=0.787) total time=   0.0s
[CV 4/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.851, test=0.911) f1:
(train=0.873, test=0.926) roc_auc: (train=0.892, test=0.941) total time=   0.0s
[CV 5/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf; accuracy: (train=0.870, test=0.833) f1:
(train=0.888, test=0.867) roc_auc: (train=0.864, test=0.849) total time=   0.0s
[CV 1/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.867, test=0.846) f1:
(train=0.887, test=0.868) roc_auc: (train=0.910, test=0.886) total time=   0.0s
[CV 2/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.856, test=0.890) f1:
(train=0.879, test=0.902) roc_auc: (train=0.898, test=0.915) total time=   0.0s
[CV 3/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.870, test=0.833) f1:
(train=0.890, test=0.854) roc_auc: (train=0.909, test=0.889) total time=   0.0s
[CV 4/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.851, test=0.911) f1:
(train=0.873, test=0.926) roc_auc: (train=0.892, test=0.941) total time=   0.0s
[CV 5/5] END svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly; accuracy: (train=0.870, test=0.833) f1:
(train=0.888, test=0.867) roc_auc: (train=0.908, test=0.897) total time=   0.0s
{'svm__C': 10.0, 'svm__degree': 3, 'svm__gamma': 'auto', 'svm__kernel': 'poly'}
Test set:
Accuracy is 0.83.
F1 is 0.86.
AUROC is 0.88
Train set:
Accuracy is 0.86.
F1 is 0.88.
AUROC is 0.91
```

Question 7) d

What performs better?

Answer

As expected, reduced dimensionality performs better: indeed, PCA function reduces dimentionality in a way that causes minimal loss of information, and the method has a weaker tendency to overfitting than if we chose two features, even if they are the best ones.

THE END