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.

Question # 1 - Answer

We think that in medical applications such as diabetes prediction, performance is much more important than the accuracy. Model performances includes F1 score, sensitivity, specificity, TP, FP.

Accuracy is calculated by this equation:[(TP+TN)/(TP+TN+FP+FN)],We want to get the higher TP TN and the lower FP FN. Accuracy gives us limited information. It does not consider all the patients so sometimes is not precise. Accuaracy gives us an indication number of how accurate is the classification in general. However the performances gives us more information regarding the meaning of the different errors. For example, the accuracy may mislead us. We can get an accuracy rate of 90%, when the other 10% are FN. In this case the patient doesn't get the right treatment he needs. High accuracy may result overfitting which means that our model doesn't generalize the information but learn templates. We might prefer low accuracy that will lead to better performances. (we took this information from an example in lecture #8).

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

Model classification with 2 features - BP and BMI

pros:

1. May prevent overfitting - This feature are highly important when talking on T1D. Therefore we have a low chance to get overfitting.
2. Low computing complexity - The model calculates faster.
3. Do not take un-needed features in count.
4. The rest of the features may assume redandency on the main two features. Low income drives fast food nutrition with much sugar and fats the may be indicated in high BP and high BMI.

cons:

1. May lead to underfitting. We may miss important information.This may lead to high bias value.

```
2. In this case we use fewer features. As a result, we might miss
important features that affect our ML model.
```

Model classification with all the features available

pros:

```
1. This model uses all the feature and can cause accurate model and the
bias will be low.
2. using all the feature may cause that one feature may affect on the
other one. Using combination of several features gives us new information
that can help us with classification. For example, using patient's height
and weight features only give us less information than using also
patient's BMI which is a combination of the height and weight.
```

cons:

```
1. This model takes un-needed features in count.
2. High computing complexity leads to a slower calculation model.
3. May lead to overfitting.
```

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.

Question # 3 - Answer

LR assumes that our population is easily separated. SVM tries to finds the "best" margin (distance between the line and the support vectors) that separates the classes and this reduces the risk of error on the data, while logistic regression does not, instead it can have different decision boundaries with different weights that are near the optimal point.

SVM works well with unstructured and semi-structured data like text and images while logistic regression works with already identified independent variables. If the number of features is large and the number of samples is small, we will recommend to use logistic regression or SVM without kernel - linear SVM. Small number of features and large number of samples will lead to use SVM

Moreover, SVM is based on geometrical properties of the data while logistic regression is based on statistical approaches. The risk of overfitting is less in SVM, while Logistic regression is vulnerable to overfitting. Logistic regression and SVM with a linear kernel have similar performance but depending on your features, one may be more efficient than the other. Non linear SVM is an enhanced SVM which enables us to better fit linear SVM.

In our case our data is unstructured and hard to distinguish. So we can infer that the data is not linearly separable. In our opinion, nonlinear SVM would be the best model classifier.

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?

Question # 4 - Answer 

The differences between LR and linear SVM:

Logistic regression and SVM with a linear kernel have similar performance but depending on your features, one may be more efficient than the other.

The Support Vector Machines algorithm is much more geometrically motivated and is trying to "maximize" the margin between the classes. Instead of assuming a probabilistic model that we use in LR, in SVM we're trying to find a particular optimal separating hyperplane, where we define "optimality" in the context of the support vectors.

LR is a simple model that focuses on maximizing the probability of the data it is trying to maximize the posterior class probability. SVM tries so find the separating hyperplane that maxinizes the distance of the closest points to the margin(which is called the support vectors).

Another difference is that the risk of overfitting is less in SVM, while Logistic regression is more vulnerable to overfitting.

SVM is a deterministic method based on geometrical properties of the data. This method reduces the risk of error on the data classification, while logistic regression is a probabilistic method based on statistical approaches.

SVM method can be considered very useful for both linear and non-linear classifications. The reason is that SVM is highly generalized and flexible method. SVM is facing better with outliers then logistic regression by using soft margin.

Hyper parameters are parameters that we can determine. These parameters are not constant and by changing them we can influence on the model accuracy as well as the result and the learning time. Hyper parameters are very critical in building robust and accurate models. They help us find the balance between bias and variance and thus, prevent the model from overfitting or underfitting.

LR hyper parameter - learning rate - λ - defines the adjustment in the weights of our network with respect to the loss gradient descent. It determines how fast or slow we will move towards the optimal weights.

Choosing the learning rate is challenging as a value too small may result in a long training process that may not converge, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process.

The learning rate controls how quickly the model is adapted to the problem. Smaller learning rates require more training epochs given the smaller changes made to the weights per each update step,

whereas larger learning rates result in rapid changes and require fewer training epochs. A learning rate that is too large can cause the model to converge too quickly to a suboptimal solution (local minima), whereas a learning rate that is too small can cause the process to get stuck.

Penalty

• L1(Lasso) - absolute value of magnitude of coefficient as penalty term to the loss function.

• L2(Ridge) - ridge regression adds “squared magnitude” of coefficient as penalty term to the loss function.

• A combination of L1 and L2

The difference between these techniques is that Lasso shrinks the less important feature’s coefficient to zero thus, removing some feature altogether. So, this works well for feature selection in case we have a huge number of features.

SVM hyper parameters C - (c=1/λ) C parameter adds a penalty for each misclassified data point. If c is small, the penalty for misclassified points is low so a decision boundary with a large margin is chosen at the expense of a greater number of misclassifications (soft margin). If c is large, SVM tries to minimize the number of misclassified examples due to high penalty which results in a decision boundary with a smaller margin(hard margin).

For a linear kernel, we just need to optimize the c parameter. However, if we want to use an RBF kernel, both c and γ parameters need to be optimized simultaneously. If γ is large, the effect of c becomes negligible. If γ is small, c affects the model just like how it affects a linear model.

γ = sigma

γ decides that how much curvature we want in a decision boundary. γ high means more curvature. γ low means less curvature. K(x,x') = exp(-γ * ||x-x'||^2) The larger the γ, the narrower the gaussian "bell" is.

CODE

In [1]:

```
import pandas as pd
import numpy as np
from pathlib import Path
import pickle
import sys
import random
import matplotlib as mpl
import seaborn as sns
import matplotlib.pyplot as plt
mpl.style.use(['ggplot'])
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.linear_model import LogisticRegression
```

```
from sklearn.metrics import log_loss
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import plot_confusion_matrix, roc_auc_score
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import hinge_loss
from sklearn.ensemble import RandomForestClassifier as rfc
from sklearn.decomposition import PCA
from sklearn.decomposition import KernelPCA
random.seed(5)
```

Preprocessing

1. Change yes/no to 1/0 (dummy coding).
2. Change female/male to 1/0.
3. Change empty cells to Nan.
4. Put random value insted of Nan according to probability.

In [2]:

```
#Section 1
file = Path.cwd().joinpath('HW2_data.csv')
file_dataset = pd.read_csv(file)

file_dataset = file_dataset.replace(['Yes','Male','Positive'],value = 1)
file_dataset = file_dataset.replace(['No','Female','Negative'],value = 0)
file_features = file_dataset[['Age','Gender','Increased Urination','Increased Thirst','
diagnosis = file_dataset[['Diagnosis']]
file_features = file_features.apply(lambda features: pd.to_numeric(features, errors='co
dict_file= {}


def removeNanWithSamples(col):
    col_without_nan = col.dropna()
    random_col = np.random.choice(col_without_nan, col.size)
    col[col.isnull()] = random_col[col.isnull()]
    return col

c_run = file_features.apply(lambda col: pd.to_numeric(col, errors='coerce'))
dict_file= c_run.apply(removeNanWithSamples)

X = pd.DataFrame(dict_file)
X = X.round(0).astype(int)
```

In [3]:

```
#Section 2
X_ = X.values
y = diagnosis.values
X_train, x_test, Y_train, y_test = train_test_split(X_, y, test_size = 0.20, random_sta
```

In [4]:

```
#Section 3a
table1_dict = {}
for idx in np.linspace(1,len(file_features.columns)-1,num = len(file_features.columns)-
    idx = int(idx)
    table1_temp = {}
```

```
    train_per = (np.nansum(X_train[:,idx])/len(X_train[:,idx]))*100
    test_per = (np.nansum(x_test[:,idx]) / len(x_test[:,idx]))*100
    delta = train_per - test_per
    table1_temp['Train %'] = train_per
    table1_temp['Test %'] = test_per
    table1_temp['Delta %'] = delta
    table1_dict[file_features.columns[idx]] = table1_temp
print(pd.DataFrame(table1_dict).transpose())
```

```
                     Train %     Test %   Delta %
Gender              62.610619  68.141593 -5.530973
Increased Urination 48.451327  47.787611  0.663717
Increased Thirst    44.469027  45.132743 -0.663717
Sudden Weight Loss  40.929204  42.477876 -1.548673
Weakness            55.752212  62.831858 -7.079646
Increased Hunger    44.026549  45.132743 -1.106195
Genital Thrush      20.796460  28.318584 -7.522124
Visual Blurring     44.247788  47.787611 -3.539823
Itching             47.566372  51.327434 -3.761062
Irritability        22.566372  27.433628 -4.867257
Delayed Healing     46.902655  43.362832  3.539823
Partial Paresis     42.256637  44.247788 -1.991150
Muscle Stiffness    35.619469  38.053097 -2.433628
Hair Loss           35.840708  35.398230  0.442478
Obesity             17.035398  15.929204  1.106195
Family History      48.672566  58.407080 -9.734513
```

Question # 3 A

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

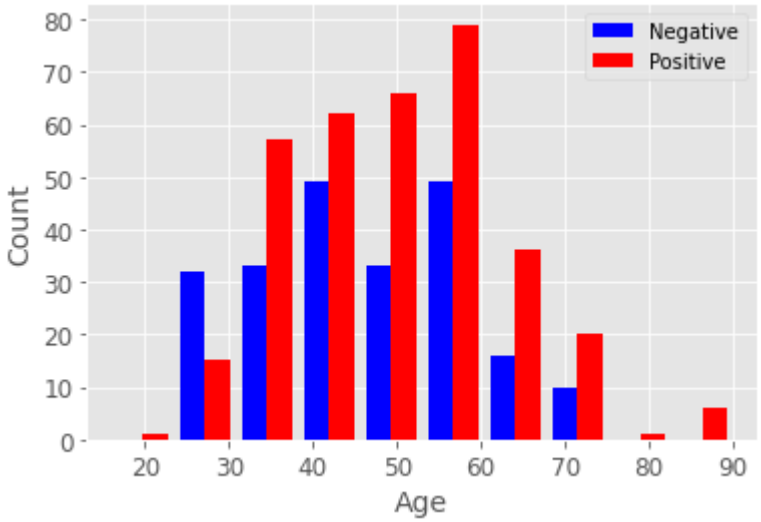
An imbalance of features between train and test can cause weight calculation that is not reflecting the reality accuratly. As a result, we may have a false classification and generalization. Therefore the model is not reliable.

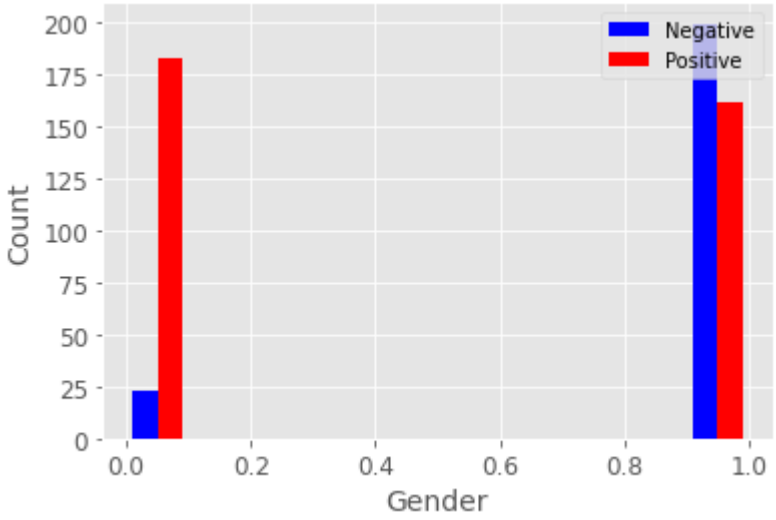
ii) How could you solve the issue?

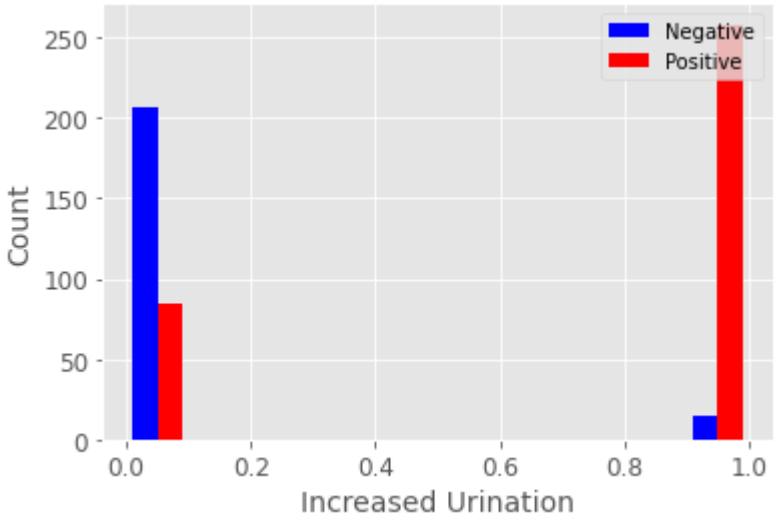
- Collect more data that could balance your classes.
- Using stratifying to devide the test according to the train. These groups preserving the general distribution of the features. In order to get a similar distribution we need to stratify the dataset based in the class values.
- Incorporate the penalized models like penalized-SVM.

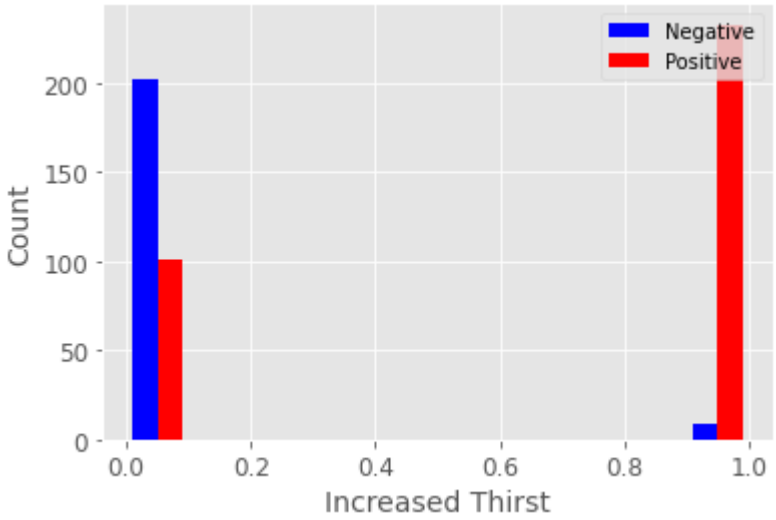
In [5]:

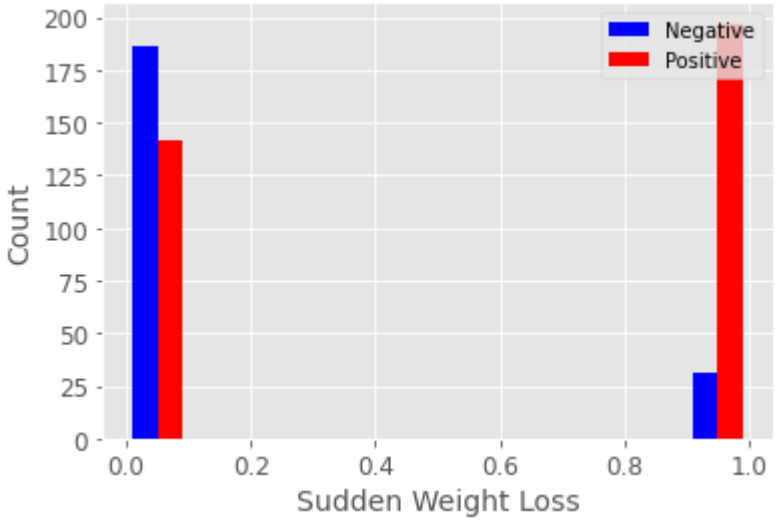
```
#Section 3b
groups = file_dataset.groupby(file_dataset.Diagnosis)
group0 = groups.get_group(0)
group1 = groups.get_group(1)

for i in file_features.columns:
    colors = ['blue','red']
    plt.hist([group0[i],group1[i]],color=colors,label = ['Negative','Positive'])
    plt.legend(loc='upper right')
    plt.xlabel(i)
    plt.ylabel('Count')
    plt.show()

## For binary values: 0 is no/female and 1 is yes/male
```

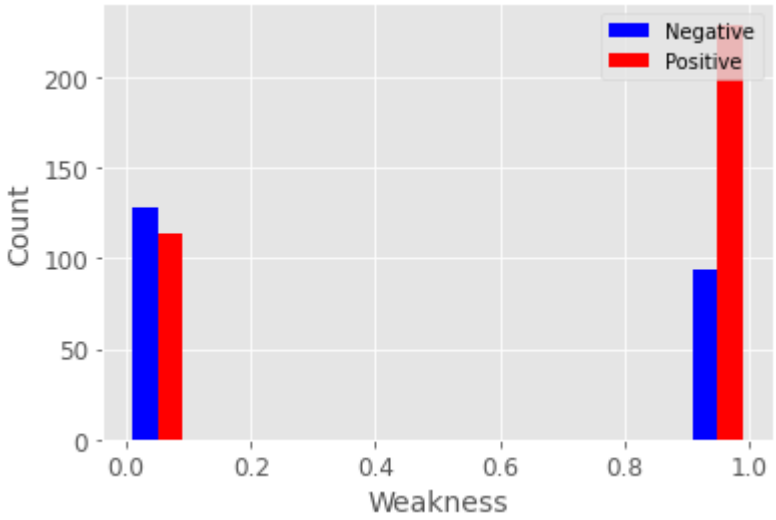

Negative
Positive
Count
Age
0
10
20
30
40
50
60
70
80
20
30
40
50
60
70
80
90

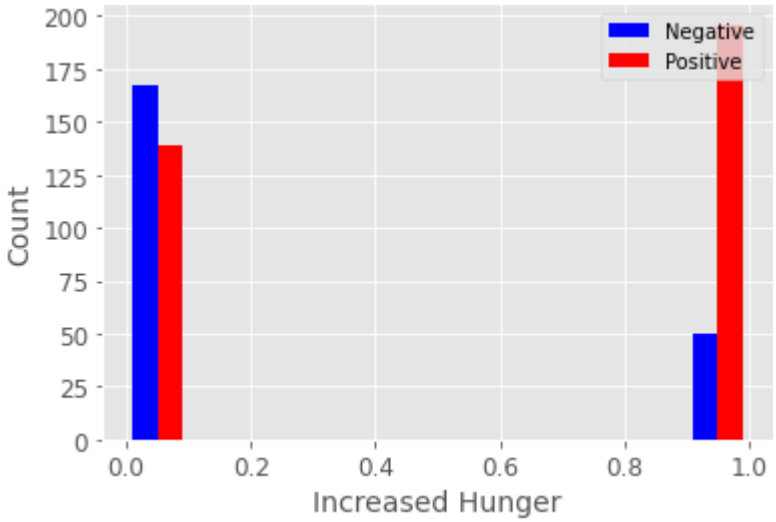

Negative
Positive
Count
Gender
0
25
50
75
100
125
150
175
200
0.0
0.2
0.4
0.6
0.8
1.0

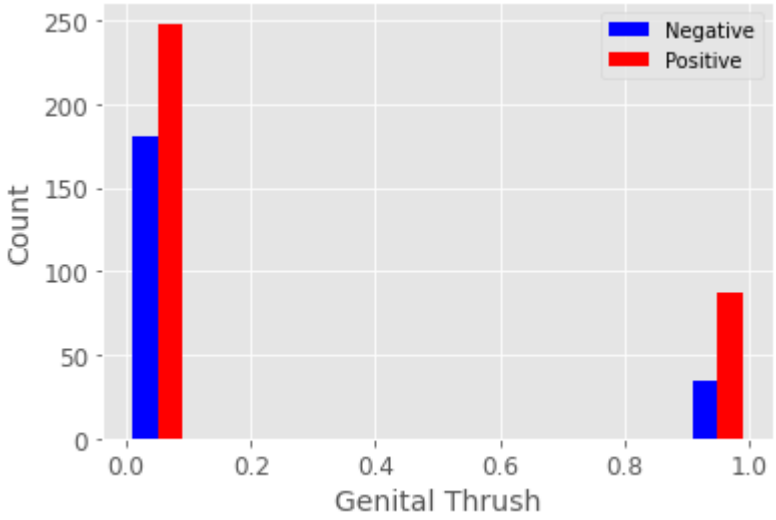

Negative
Positive
Count
Increased Urination
0
50
100
150
200
250
0.0
0.2
0.4
0.6
0.8
1.0

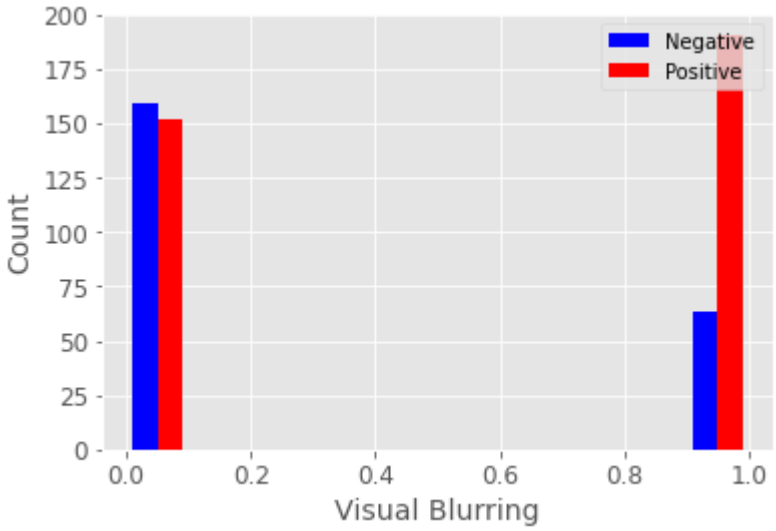

Negative
Positive
Count
Increased Thirst

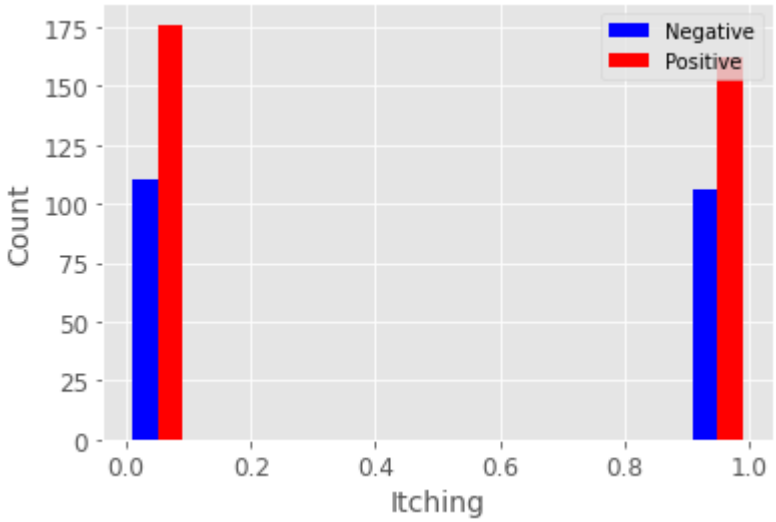

Negative
Positive
Count
Sudden Weight Loss

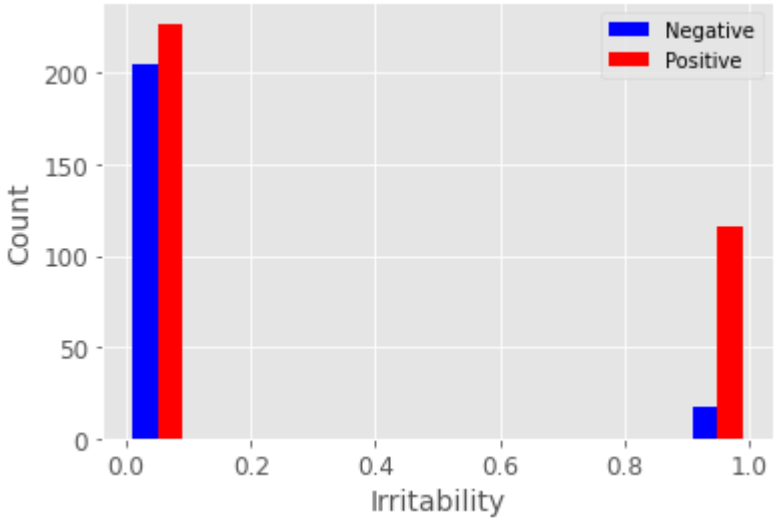

Negative
Positive
Count
Weakness

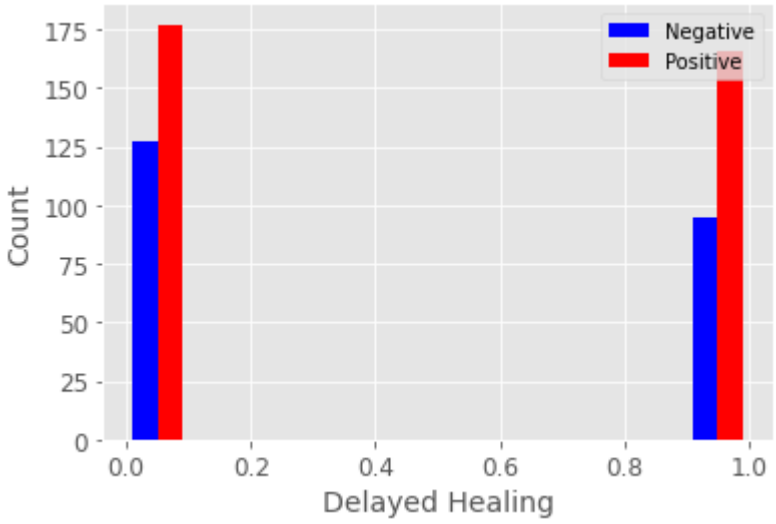

Negative
Positive
Count
Increased Hunger


Negative
Positive
Count
Genital Thrush

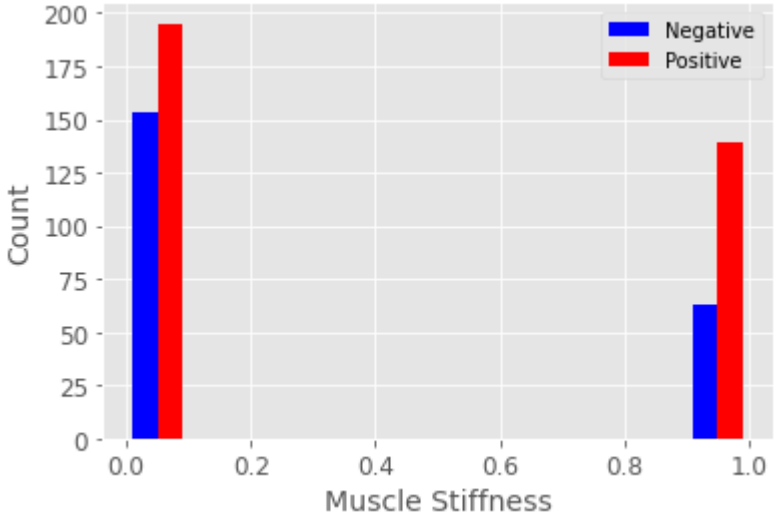

Negative
Positive
Count
Visual Blurring

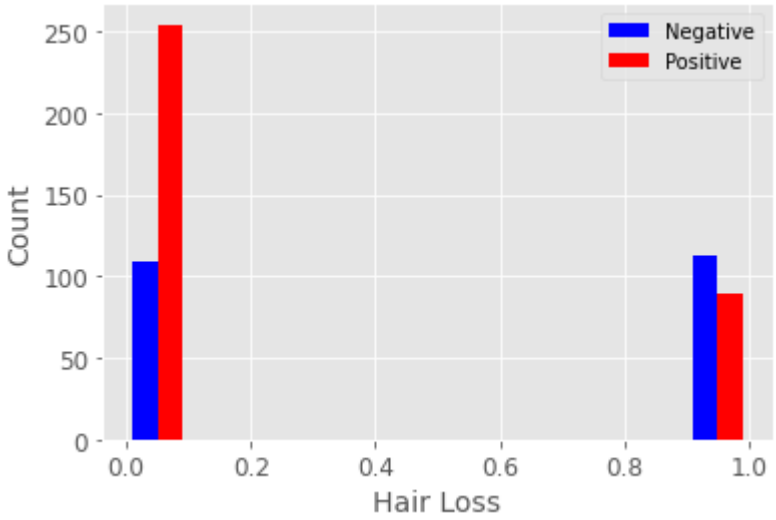

Negative
Positive
Count
Itching

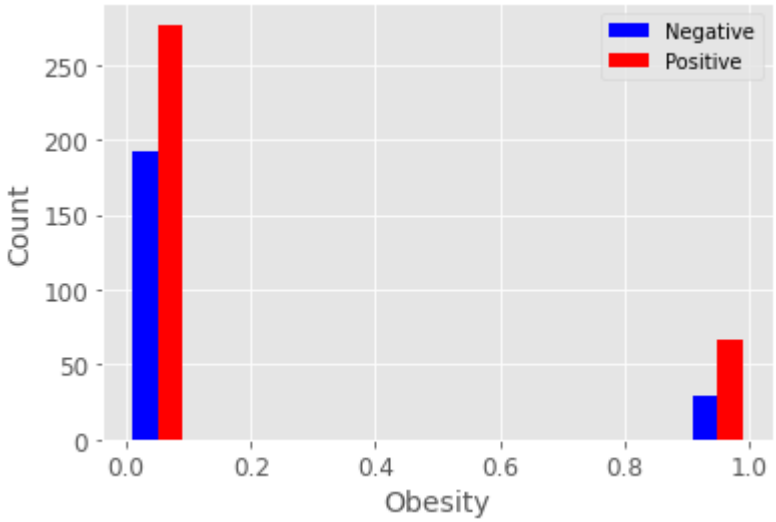

Negative
Positive
Count
Irritability

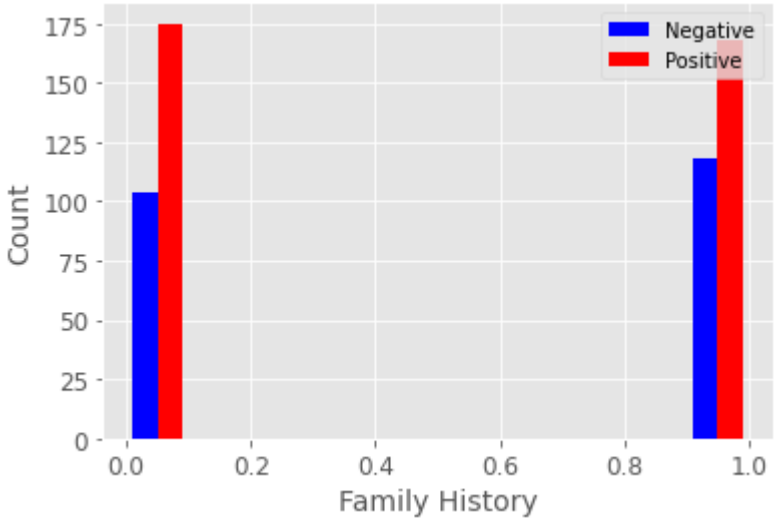

Negative
Positive
Count
Delayed Healing


Negative
Positive
Count
Partial Paresis


Negative
Positive
Count
Muscle Stiffness

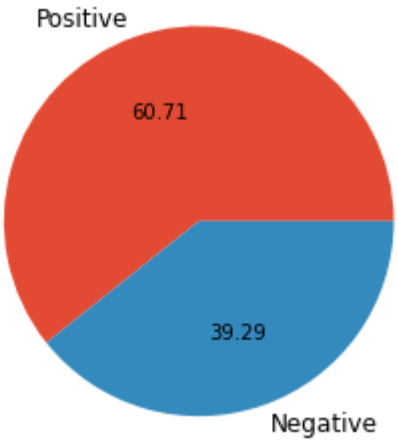

Negative
Positive
Count
Hair Loss





In [6]:

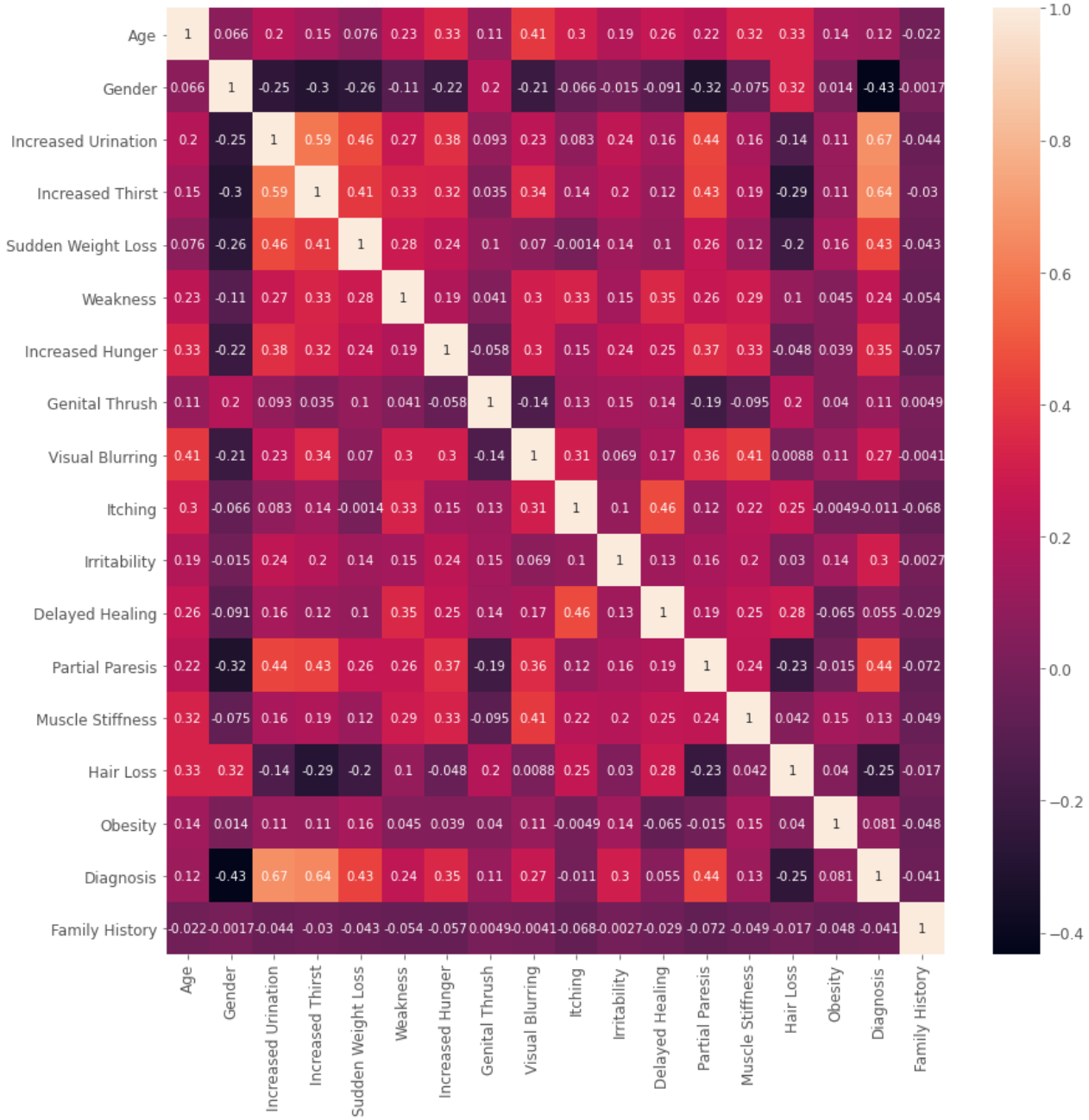
```
#Section 3c

Y_total = len(y)
Y_1 = np.nansum(y[:])
Y_0 = Y_total - Y_1
Y_ = [Y_1,Y_0]
pielabels = ('Positive','Negative')
figureObject, axesObject = plt.subplots()
axesObject.pie(Y_,labels = pielabels,autopct = '%1.2f')
axesObject.axis('equal')
plt.show()
```



We plot a pie diagram in order to show possible bias in the given data.

In [7]:

```
plt.figure(figsize = (14,14))
sns.heatmap(file_dataset.corr(), annot = True)
plt.show()
```



We used heatmap in order to see the correlation between the features.

Question 3,D Part 1 Was there anything unexpected?

Answer

- Gender we expected the gender distribution would be the same for women and men. In our data the rate of positivly diagnosed women was much higher then the negativly diagnosed women in compare to the same rates in men.
- Family History we also expected that family history would have a bigger importance on the diagnosis

Question 3,D Part 2 Are there any features that you feel will be particularly important to your model? Explain why.

Answer According to the literature review attached and the given data, we assumed that the most important features would be:

- Increased Thrist
- Increased Urination - depands on increades thirst.
- Sudden Weight Loss
- Increased Hunger

Features that are secondary important:

- Weakness
- Visual Blurring (not sure)

Question 4

Encode all your data as one hot vectors.

Answer question 4

Due to the binary nature of the data, the pre-proccessing done in section 1 can function as an encoder, except for the "Age" feature, which is not binary. The Age feature represents the age during the test and not the age of the diagnosis, and we can see from the heatmap there isn't a significant correlation between the age and the diagnosis.

We can also see from the histogram that the positive and negative classes have similar age distributions. Therefore we believe the Age feature is not significant to the diagnosis classification and we decided to drop it from the encoding. Encoding the age feature would significantlly increase the number of columns which will significantlly increase the complexity and running time of the code.

In [8]:
```
le = LabelEncoder()
diagnosis = le.fit_transform(diagnosis)
X_no_age = X.drop('Age', axis=1)
```

```
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\utils\validation.py:72:
DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
```

In [9]:
```
#section 5a
X_train, x_test, Y_train, y_test = train_test_split(X_no_age, diagnosis, test_size = 0.
```

In [10]:
```
#Logistic Regression
C= np.array([0.001, 0.01, 1, 10, 100])
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
logreg = LogisticRegression()
pen = ['l1','l2']
hyperparameters = dict(C=C, penalty=pen)

pipe = Pipeline(steps=[('scale', StandardScaler()), ('logistic', logreg)])
log_reg = GridSearchCV(logreg, hyperparameters, scoring=['roc_auc'],  cv=skf, refit='ro
```

```
log_reg.fit(X_train, Y_train)
best_log_reg = log_reg.best_estimator_

Y_pred_train_log = best_log_reg.predict(X_train)
Y_pred_proba_train_log = best_log_reg.predict_proba(X_train)
log_reg_score_train = roc_auc_score(Y_train, Y_pred_proba_train_log[:,1])

y_pred_test_log = best_log_reg.predict(x_test)
y_pred_proba_test_log = best_log_reg.predict_proba(x_test)
log_reg_score_test = roc_auc_score(y_test, y_pred_proba_test_log[:,1])
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] C=0.001, penalty=l1 ............................................
[CV]  C=0.001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.001, penalty=l1 ............................................
[CV]  C=0.001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.001, penalty=l1 ............................................
[CV]  C=0.001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.001, penalty=l1 ............................................
[CV]  C=0.001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.001, penalty=l1 ............................................
[CV]  C=0.001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.001, penalty=l2 ............................................
[CV]  C=0.001, penalty=l2, roc_auc=(train=0.952, test=0.940), total=   0.1s
[CV] C=0.001, penalty=l2 ............................................
[CV]  C=0.001, penalty=l2, roc_auc=(train=0.948, test=0.979), total=   0.0s
[CV] C=0.001, penalty=l2 ............................................
[CV]  C=0.001, penalty=l2, roc_auc=(train=0.960, test=0.941), total=   0.1s
[CV] C=0.001, penalty=l2 ............................................
[CV]  C=0.001, penalty=l2, roc_auc=(train=0.955, test=0.945), total=   0.1s
[CV] C=0.001, penalty=l2 ............................................
[CV]  C=0.001, penalty=l2, roc_auc=(train=0.954, test=0.948), total=   0.1s
[CV] C=0.01, penalty=l1 .............................................
[CV]  C=0.01, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.01, penalty=l1 .............................................
[CV]  C=0.01, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.01, penalty=l1 .............................................
[CV]  C=0.01, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.01, penalty=l1 .............................................
[CV]  C=0.01, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.01, penalty=l1 .............................................
[CV]  C=0.01, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.01, penalty=l2 .............................................
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
```

```
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[CV]  C=0.01, penalty=l2, roc_auc=(train=0.960, test=0.952), total=   0.1s
[CV] C=0.01, penalty=l2 ..............................................
[CV]  C=0.01, penalty=l2, roc_auc=(train=0.957, test=0.986), total=   0.1s
[CV] C=0.01, penalty=l2 ..............................................
[CV]  C=0.01, penalty=l2, roc_auc=(train=0.970, test=0.943), total=   0.1s
[CV] C=0.01, penalty=l2 ..............................................
[CV]  C=0.01, penalty=l2, roc_auc=(train=0.962, test=0.952), total=   0.1s
[CV] C=0.01, penalty=l2 ..............................................
[CV]  C=0.01, penalty=l2, roc_auc=(train=0.964, test=0.958), total=   0.3s
[CV] C=1.0, penalty=l1 ...............................................
[CV] . C=1.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1.0, penalty=l1 ...............................................
[CV] . C=1.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1.0, penalty=l1 ...............................................
[CV] . C=1.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1.0, penalty=l1 ...............................................
[CV] . C=1.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1.0, penalty=l1 ...............................................
[CV] . C=1.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1.0, penalty=l2 ...............................................
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[CV]  C=1.0, penalty=l2, roc_auc=(train=0.985, test=0.962), total=   0.2s
[CV] C=1.0, penalty=l2 ...............................................
[CV]  C=1.0, penalty=l2, roc_auc=(train=0.982, test=0.976), total=   0.1s
[CV] C=1.0, penalty=l2 ...............................................
[CV]  C=1.0, penalty=l2, roc_auc=(train=0.989, test=0.955), total=   0.3s
[CV] C=1.0, penalty=l2 ...............................................
[CV]  C=1.0, penalty=l2, roc_auc=(train=0.980, test=0.981), total=   0.1s
[CV] C=1.0, penalty=l2 ...............................................
[CV]  C=1.0, penalty=l2, roc_auc=(train=0.983, test=0.979), total=   0.1s
[CV] C=10.0, penalty=l1 ..............................................
[CV]  C=10.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10.0, penalty=l1 ..............................................
[CV]  C=10.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10.0, penalty=l1 ..............................................
```

```
[CV]  C=10.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10.0, penalty=l1 ..............................................
[CV]  C=10.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10.0, penalty=l1 ..............................................
[CV]  C=10.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10.0, penalty=l2 ..............................................
[CV]  C=10.0, penalty=l2, roc_auc=(train=0.988, test=0.953), total=   0.1s
[CV] C=10.0, penalty=l2 ..............................................
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[CV]  C=10.0, penalty=l2, roc_auc=(train=0.984, test=0.976), total=   0.1s
[CV] C=10.0, penalty=l2 ..............................................
[CV]  C=10.0, penalty=l2, roc_auc=(train=0.990, test=0.957), total=   0.2s
[CV] C=10.0, penalty=l2 ..............................................
[CV]  C=10.0, penalty=l2, roc_auc=(train=0.982, test=0.983), total=   0.1s
[CV] C=10.0, penalty=l2 ..............................................
[CV]  C=10.0, penalty=l2, roc_auc=(train=0.982, test=0.983), total=   0.1s
[CV] C=100.0, penalty=l1 .............................................
[CV]  C=100.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=100.0, penalty=l1 .............................................
[CV]  C=100.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=100.0, penalty=l1 .............................................
[CV]  C=100.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=100.0, penalty=l1 .............................................
[CV]  C=100.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=100.0, penalty=l1 .............................................
[CV]  C=100.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=100.0, penalty=l2 .............................................
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[CV]  C=100.0, penalty=l2, roc_auc=(train=0.989, test=0.948), total=   0.3s
[CV] C=100.0, penalty=l2 .............................................
[CV]  C=100.0, penalty=l2, roc_auc=(train=0.984, test=0.976), total=   0.1s
[CV] C=100.0, penalty=l2 .............................................
[CV]  C=100.0, penalty=l2, roc_auc=(train=0.990, test=0.958), total=   0.2s
[CV] C=100.0, penalty=l2 .............................................
```

```
[CV]  C=100.0, penalty=l2, roc_auc=(train=0.982, test=0.985), total=   0.4s
[CV] C=100.0, penalty=l2 ..............................................
[CV]  C=100.0, penalty=l2, roc_auc=(train=0.982, test=0.983), total=   0.2s
[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:    4.6s finished
```

In [11]:

```
#Linear SVM
C = np.array([0.001, 0.01, 1, 10, 100])
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=['roc_auc'],  c
svm_lin.fit(X_train, Y_train)
best_svc_lin = svm_lin.best_estimator_

Y_pred_train_svc = best_svc_lin.predict(X_train)
Y_pred_proba_train_svc = best_svc_lin.predict_proba(X_train)
LSVM_score_train = roc_auc_score(Y_train, Y_pred_proba_train_log[:,1])

y_pred_test_svc = best_svc_lin.predict(x_test)
y_pred_proba_test_svc= best_svc_lin.predict_proba(x_test)
LSVM_score_test = roc_auc_score(y_test, y_pred_proba_test_svc[:,1])
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits
[CV] svm__C=0.001, svm__kernel=linear .................................
[CV]  svm__C=0.001, svm__kernel=linear, roc_auc=(train=0.962, test=0.953), total=   0.2s
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[CV] svm__C=0.001, svm__kernel=linear .................................
[CV]  svm__C=0.001, svm__kernel=linear, roc_auc=(train=0.958, test=0.983), total=   0.1s
[CV] svm__C=0.001, svm__kernel=linear .................................
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.3s remaining:    0.0s
[CV]  svm__C=0.001, svm__kernel=linear, roc_auc=(train=0.971, test=0.951), total=   0.1s
[CV] svm__C=0.001, svm__kernel=linear .................................
[CV]  svm__C=0.001, svm__kernel=linear, roc_auc=(train=0.965, test=0.958), total=   0.1s
[CV] svm__C=0.001, svm__kernel=linear .................................
[CV]  svm__C=0.001, svm__kernel=linear, roc_auc=(train=0.968, test=0.952), total=   0.1s
[CV] svm__C=0.01, svm__kernel=linear ..................................
[CV]  svm__C=0.01, svm__kernel=linear, roc_auc=(train=0.976, test=0.966), total=   0.1s
[CV] svm__C=0.01, svm__kernel=linear ..................................
[CV]  svm__C=0.01, svm__kernel=linear, roc_auc=(train=0.976, test=0.979), total=   0.2s
[CV] svm__C=0.01, svm__kernel=linear ..................................
[CV]  svm__C=0.01, svm__kernel=linear, roc_auc=(train=0.982, test=0.957), total=   0.1s
[CV] svm__C=0.01, svm__kernel=linear ..................................
[CV]  svm__C=0.01, svm__kernel=linear, roc_auc=(train=0.973, test=0.973), total=   0.1s
[CV] svm__C=0.01, svm__kernel=linear ..................................
[CV]  svm__C=0.01, svm__kernel=linear, roc_auc=(train=0.974, test=0.957), total=   0.1s
[CV] svm__C=1.0, svm__kernel=linear ...................................
[CV]  svm__C=1.0, svm__kernel=linear, roc_auc=(train=0.982, test=0.962), total=   0.1s
[CV] svm__C=1.0, svm__kernel=linear ...................................
[CV]  svm__C=1.0, svm__kernel=linear, roc_auc=(train=0.981, test=0.969), total=   0.2s
[CV] svm__C=1.0, svm__kernel=linear ...................................
[CV]  svm__C=1.0, svm__kernel=linear, roc_auc=(train=0.986, test=0.952), total=   0.1s
[CV] svm__C=1.0, svm__kernel=linear ...................................
[CV]  svm__C=1.0, svm__kernel=linear, roc_auc=(train=0.977, test=0.983), total=   0.1s
[CV] svm__C=1.0, svm__kernel=linear ...................................
[CV]  svm__C=1.0, svm__kernel=linear, roc_auc=(train=0.980, test=0.993), total=   0.1s
[CV] svm__C=10.0, svm__kernel=linear ..................................
[CV]  svm__C=10.0, svm__kernel=linear, roc_auc=(train=0.984, test=0.954), total=   0.2s
[CV] svm__C=10.0, svm__kernel=linear ..................................
```

```
[CV]  svm__C=10.0, svm__kernel=linear, roc_auc=(train=0.982, test=0.968), total=   0.3s
[CV] svm__C=10.0, svm__kernel=linear .................................
[CV]  svm__C=10.0, svm__kernel=linear, roc_auc=(train=0.987, test=0.950), total=   0.3s
[CV] svm__C=10.0, svm__kernel=linear .................................
[CV]  svm__C=10.0, svm__kernel=linear, roc_auc=(train=0.978, test=0.988), total=   0.2s
[CV] svm__C=10.0, svm__kernel=linear .................................
[CV]  svm__C=10.0, svm__kernel=linear, roc_auc=(train=0.979, test=0.992), total=   0.3s
[CV] svm__C=100.0, svm__kernel=linear ................................
[CV]  svm__C=100.0, svm__kernel=linear, roc_auc=(train=0.985, test=0.954), total=   1.4s
[CV] svm__C=100.0, svm__kernel=linear ................................
[CV]  svm__C=100.0, svm__kernel=linear, roc_auc=(train=0.982, test=0.968), total=   2.3s
[CV] svm__C=100.0, svm__kernel=linear ................................
[CV]  svm__C=100.0, svm__kernel=linear, roc_auc=(train=0.988, test=0.950), total=   1.4s
[CV] svm__C=100.0, svm__kernel=linear ................................
[CV]  svm__C=100.0, svm__kernel=linear, roc_auc=(train=0.979, test=0.990), total=   2.2s
[CV] svm__C=100.0, svm__kernel=linear ................................
[CV]  svm__C=100.0, svm__kernel=linear, roc_auc=(train=0.979, test=0.992), total=   1.5s
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:   12.7s finished
```

In [12]:

```
#NonLinear
#Non-Linear SVM- poly
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
svc = SVC(probability=True)
C = np.array([0.001, 0.01, 1, 10, 100])
pipe = Pipeline(steps=[('scale', StandardScaler()), ('svm', svc)])
svm_poly = GridSearchCV(estimator=pipe,
             param_grid={'svm__kernel':['poly'], 'svm__C':C, 'svm__degree':[3], 'svm__g
svm_poly.fit(X_train, Y_train)

best_svm_poly = svm_poly.best_estimator_

Y_pred_train_poly = best_svm_poly.predict(X_train)
Y_pred_proba_train_poly = best_svm_poly.predict_proba(X_train)
poly_score_train = roc_auc_score(Y_train, Y_pred_proba_train_log[:,1])

y_pred_test_poly = best_svm_poly.predict(x_test)
y_pred_proba_test_poly= best_svm_poly.predict_proba(x_test)
poly_score_test = roc_auc_score(y_test, y_pred_proba_test_poly[:,1])
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
91, test=0.978), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
92, test=0.993), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.3s remaining:    0.0s
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
95, test=0.982), total=   0.2s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
93, test=0.987), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
94, test=0.984), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
991, test=0.978), total=   0.1s
```

```
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
992, test=0.993), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
995, test=0.982), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
993, test=0.987), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
994, test=0.984), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.99
1, test=0.976), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.99
2, test=0.994), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.99
5, test=0.982), total=   0.2s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.99
3, test=0.987), total=   0.2s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.99
4, test=0.984), total=   0.2s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
91, test=0.976), total=   0.2s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
92, test=0.994), total=   0.4s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
95, test=0.982), total=   0.2s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
93, test=0.987), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
94, test=0.984), total=   0.2s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.99
9, test=0.991), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=1.00
0, test=0.994), total=   0.2s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=1.00
0, test=0.988), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.99
9, test=0.997), total=   0.2s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.99
9, test=0.998), total=   0.2s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ...
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.99
9, test=0.991), total=   0.2s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ...
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=1.00
0, test=0.994), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ...
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=1.00
```

```
0, test=0.988), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ...
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.99
9, test=0.997), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ...
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.99
9, test=0.998), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.99
8, test=0.981), total=   0.2s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.99
8, test=0.971), total=   0.2s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=1.00
0, test=0.993), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.99
7, test=0.991), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.99
8, test=0.996), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
98, test=0.981), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
98, test=0.971), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=1.0
00, test=0.993), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
97, test=0.991), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
98, test=0.996), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
99, test=0.978), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
98, test=0.971), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=1.0
00, test=0.993), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
98, test=0.993), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
98, test=0.995), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
999, test=0.978), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
998, test=0.971), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=1.
000, test=0.993), total=   0.2s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
998, test=0.993), total=   0.2s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
```

```
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
998, test=0.995), total=   0.2s
[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:    9.0s finished
```

In [13]:

```
#Non-Linear SVM- rbf
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
svc = SVC(probability=True)
C = np.array([0.001, 0.01, 1, 10, 100])
pipe = Pipeline(steps=[('scale', StandardScaler()), ('svm', svc)])
svm_rbf = GridSearchCV(estimator=pipe,
             param_grid={'svm__kernel':['rbf'], 'svm__C':C, 'svm__degree':[3], 'svm__ga
svm_rbf.fit(X_train, Y_train)

best_svm_rbf = svm_rbf.best_estimator_

Y_pred_train_rbf = best_svm_rbf.predict(X_train)
Y_pred_proba_train_rbf = best_svm_rbf.predict_proba(X_train)
rbf_score_train = roc_auc_score(Y_train, Y_pred_proba_train_log[:,1])

y_pred_test_rbf = best_svm_rbf.predict(x_test)
y_pred_proba_test_rbf= best_svm_rbf.predict_proba(x_test)
rbf_score_test = roc_auc_score(y_test, y_pred_proba_test_rbf[:,1])
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.97
1, test=0.967), total=   0.1s
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.1s remaining:    0.0s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.97
0, test=0.982), total=   0.2s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.3s remaining:    0.0s
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.97
5, test=0.959), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.97
4, test=0.956), total=   0.2s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.97
5, test=0.962), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
71, test=0.967), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
70, test=0.982), total=   0.2s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
75, test=0.959), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
74, test=0.956), total=   0.2s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
75, test=0.962), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.97
1, test=0.967), total=   0.2s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
```

```
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.97
1, test=0.982), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.97
6, test=0.959), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.97
4, test=0.957), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.97
5, test=0.961), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.97
1, test=0.967), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.97
1, test=0.982), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.97
6, test=0.959), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.97
4, test=0.957), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.97
5, test=0.961), total=   0.2s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.999,
test=0.993), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.999,
test=0.994), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=1.000,
test=0.984), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.997,
test=0.987), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.999,
test=0.996), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.99
9, test=0.993), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.99
9, test=0.994), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=1.00
0, test=0.984), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.99
7, test=0.987), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.99
9, test=0.996), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.99
6, test=0.995), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.99
9, test=0.979), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.99
8, test=0.989), total=   0.1s
```

```
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.99
7, test=0.995), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.99
8, test=0.993), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.99
6, test=0.995), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.99
9, test=0.979), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.99
8, test=0.989), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.99
7, test=0.995), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.99
8, test=0.993), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.99
9, test=0.989), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=1.00
0, test=0.981), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.99
8, test=0.989), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.99
8, test=0.996), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.99
7, test=0.990), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
99, test=0.989), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=1.0
00, test=0.981), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
98, test=0.989), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
98, test=0.996), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
97, test=0.990), total=   0.1s
[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:    7.4s finished
```

In [14]:

```
#section 5b
# logreg_loss = log_loss(y_test, y_pred_proba_test_log[:,1])
# svc_loss = hinge_loss(y_test, y_pred_proba_test_svc[:,1])
# poly_loss = hinge_loss(y_test, y_pred_proba_test_poly[:,1])
# rbf_loss = hinge_loss(y_test, y_pred_proba_test_rbf[:,1])
```

In [15]:

```
def calc_metrics(y_test, y_pred_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)
    PPV = TP/(TP+FP)
    Acc = (TP+TN)/(TP+TN+FP+FN)
    F1 = (2*Se*PPV)/(Se+PPV)

    return F1, Acc
```

In [16]:

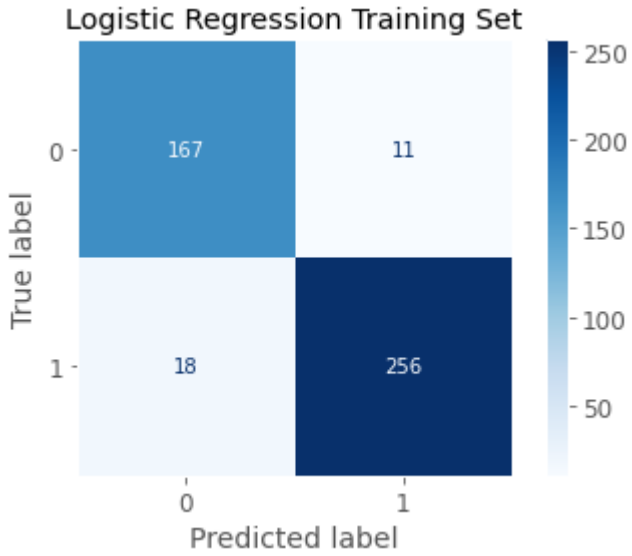
```
F1_logreg_train, Acc_logreg_train = calc_metrics(Y_train, Y_pred_train_log)
F1_svc_train, Acc_svc_train = calc_metrics(Y_train, Y_pred_train_svc)
F1_poly_train, Acc_poly_train = calc_metrics(Y_train, Y_pred_train_poly)
F1_rbf_train, Acc_rbf_train = calc_metrics(Y_train, Y_pred_train_rbf)

F1_logreg_test, Acc_logreg_test = calc_metrics(y_test, y_pred_test_log)
F1_svc_test, Acc_svc_test = calc_metrics(y_test, y_pred_test_svc)
F1_poly_test, Acc_poly_test = calc_metrics(y_test, y_pred_test_poly)
F1_rbf_test, Acc_rbf_test = calc_metrics(y_test, y_pred_test_rbf)
```

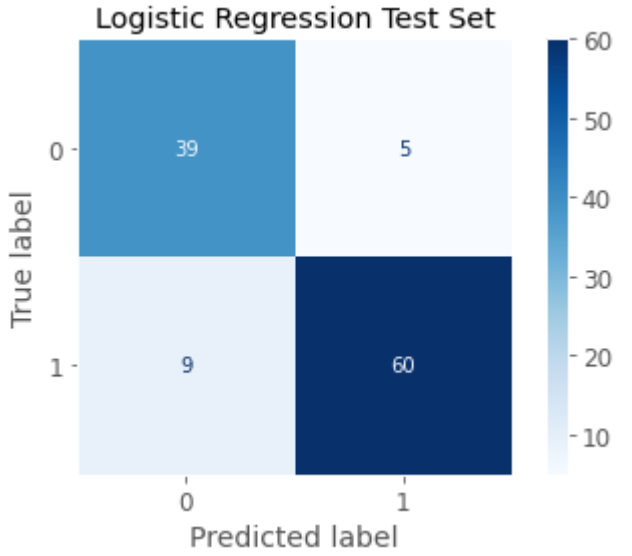
In [17]:

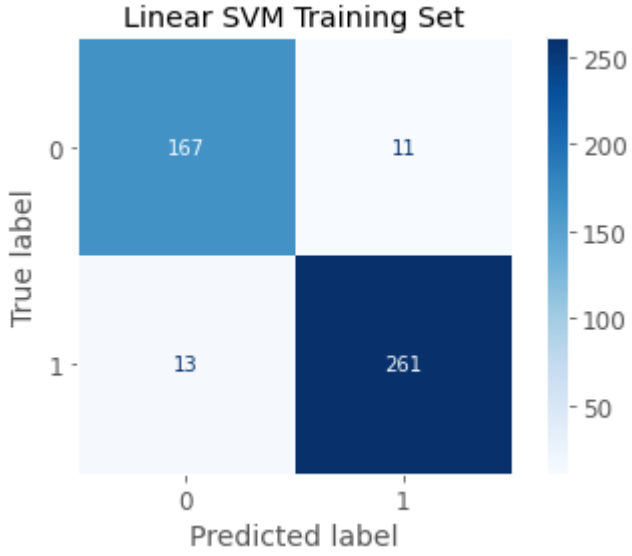
```
print('Logistic Regression:')
print('\nFor Training set: \nAUC is {:.2f}. \nF1 is {:.2f}. \nACC is {:.2f}. '.format(l
plot_confusion_matrix(log_reg, X_train, Y_train, cmap=plt.cm.Blues)
plt.title('Logistic Regression Training Set')
plt.grid(False)
print('\nFor Test set: \nAUC is {:.2f}. \nF1 is {:.2f}.  \nACC is {:.2f}. '.format(log_
plot_confusion_matrix(log_reg, x_test, y_test, cmap=plt.cm.Blues)
plt.title('Logistic Regression Test Set')
plt.grid(False)

print('Linear SVM:')
print('\nFor Training set: \nAUC is {:.2f}. \nF1 is {:.2f}.  \nACC is {:.2f}. '.format(
plot_confusion_matrix(svm_lin, X_train, Y_train, cmap=plt.cm.Blues)
plt.title('Linear SVM Training Set')
plt.grid(False)
print('\nFor Test set: \nAUC is {:.2f}. \nF1 is {:.2f}.  \nACC is {:.2f}. '.format(LSVM
plot_confusion_matrix(svm_lin , x_test, y_test, cmap=plt.cm.Blues)
plt.title('Linear SVM Test Set')
plt.grid(False)

print('Non-Linear SVM- Poly kernel:')
print('\nFor Training set: \nAUC is {:.2f}. \nF1 is {:.2f}.  \nACC is {:.2f}. '.format(
plot_confusion_matrix(svm_poly, X_train, Y_train, cmap=plt.cm.Blues)
plt.title('Polynomial SVM Training Set')
plt.grid(False)
print('\nFor Test set: \nAUC is {:.2f}. \nF1 is {:.2f}.  \nACC is {:.2f}. '.format(poly
plot_confusion_matrix(svm_poly, x_test, y_test, cmap=plt.cm.Blues)
plt.title('Polynomial SVM Test Set')
plt.grid(False)

print('Non-Linear SVM- rbf kernel')
print('\nFor Training set: \nAUC is {:.2f}. \nF1 is {:.2f}.  \nACC is {:.2f}. '.format(
plot_confusion_matrix(svm_rbf, X_train, Y_train, cmap=plt.cm.Blues)
plt.title('RBF SVM Training Set')
```

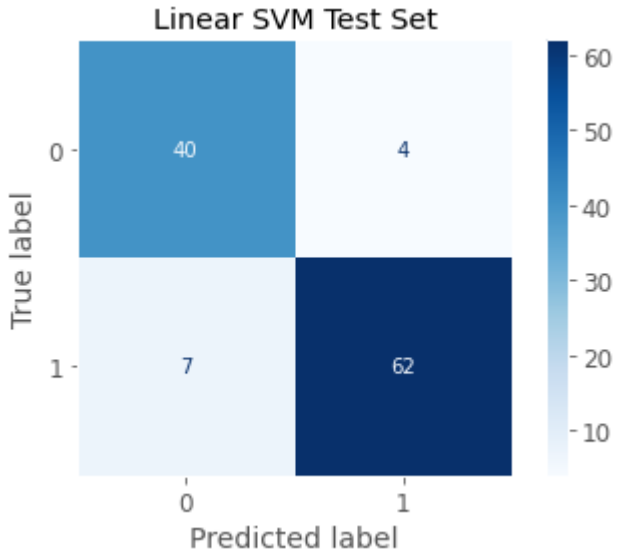
```
plt.grid(False)
print('\nFor Test set: \nAUC is {:.2f}. \nF1 is {:.2f}.  \nACC is {:.2f}. '.format(rbf_
plot_confusion_matrix(svm_rbf, x_test, y_test, cmap=plt.cm.Blues)
plt.grid(False)
plt.title('RBF SVM Test Set')
plt.show()
```

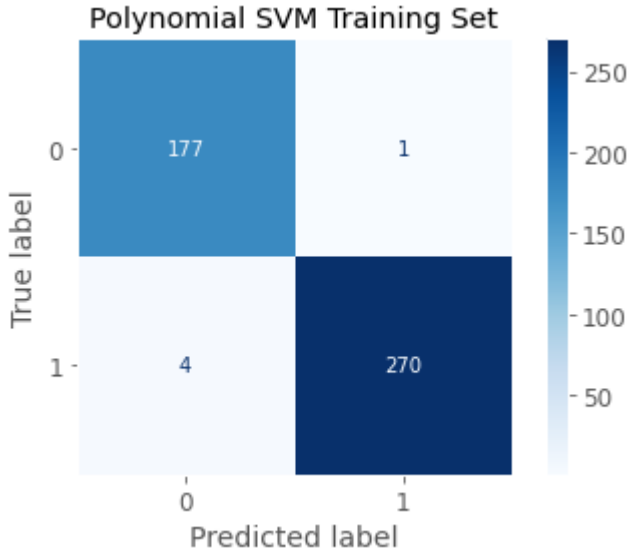
```
Logistic Regression:

For Training set:
AUC is 0.98.
F1 is 0.95.
ACC is 0.94.

For Test set:
AUC is 0.95.
F1 is 0.90.
ACC is 0.88.
Linear SVM:

For Training set:
AUC is 0.98.
F1 is 0.96.
ACC is 0.95.

For Test set:
AUC is 0.95.
F1 is 0.92.
ACC is 0.90.
Non-Linear SVM- Poly kernel:

For Training set:
AUC is 0.98.
F1 is 0.99.
ACC is 0.99.

For Test set:
AUC is 0.99.
F1 is 0.94.
ACC is 0.93.
Non-Linear SVM- rbf kernel

For Training set:
AUC is 0.98.
F1 is 0.99.
ACC is 0.99.

For Test set:
AUC is 0.98.
F1 is 0.95.
ACC is 0.94.
```

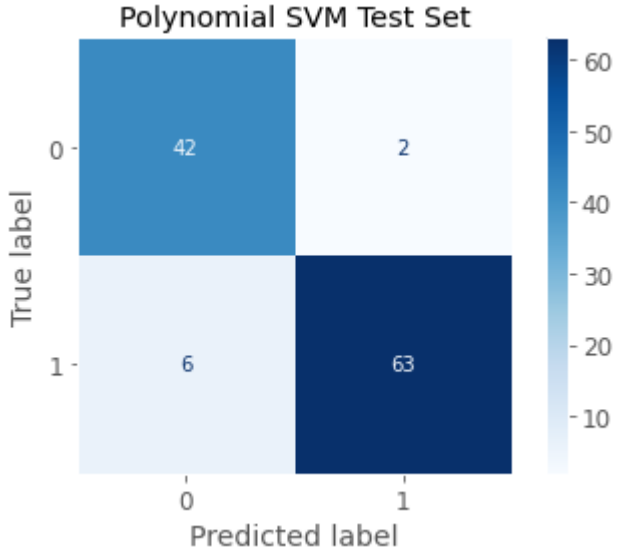

Logistic Regression Training Set
167
11
18
256
True label
Predicted label
0
1
250
200
150
100
50

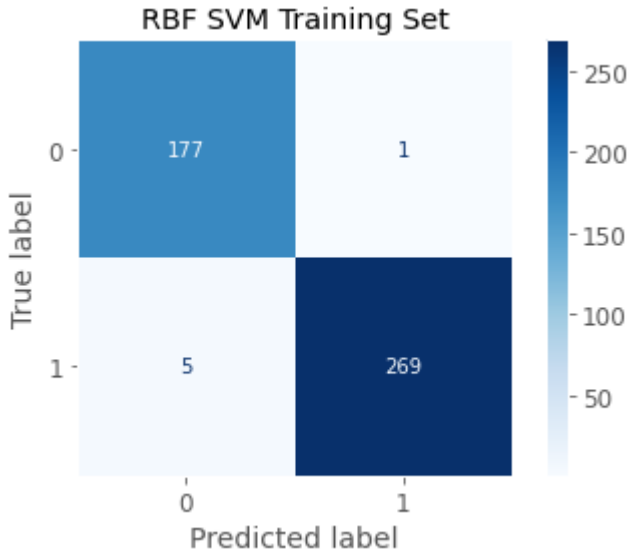

Logistic Regression Test Set
39
5
9
60
True label
Predicted label
0
1
60
50
40
30
20
10

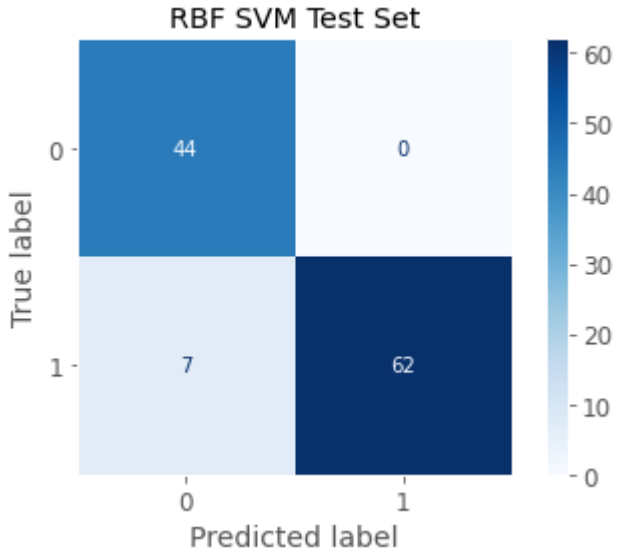

Linear SVM Training Set
167
11
13
261
True label
Predicted label
0
1
250
200
150
100
50


Linear SVM Test Set
40
4
7
62
True label
Predicted label
0
1
60
50
40
30
20
10


Polynomial SVM Training Set
177
1
4
270
True label
Predicted label
0
1
250
200
150
100
50


Polynomial SVM Test Set
42
2
6
63
True label
Predicted label
0
1
60
50
40
30
20
10





Question # 5,C

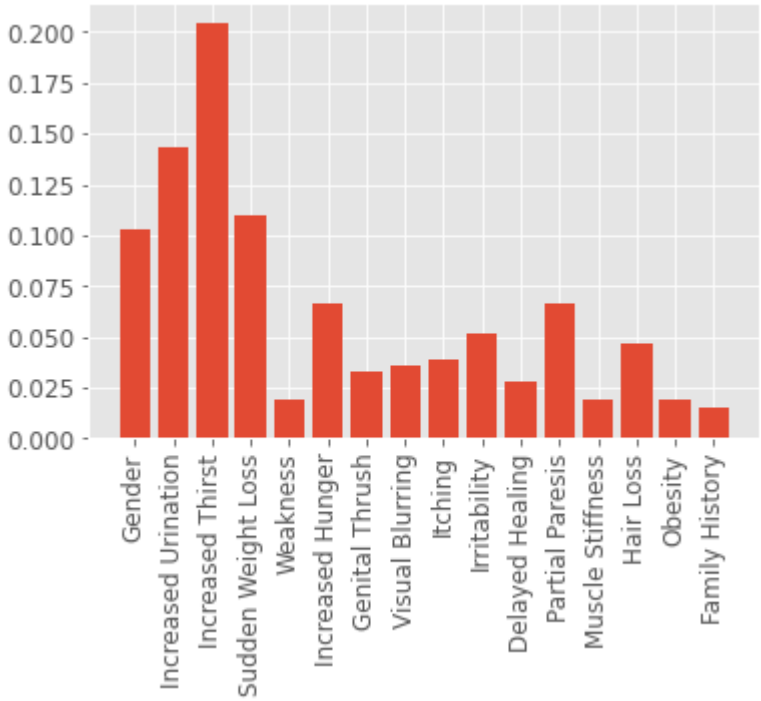
What performs best on this dataset? Linear or non-linear models?

Answer Question # 5,C

AUC - The highest value appears in the Nonlinear SVM-rbf model. ACC - The highest value appears in Nonlinear SVM-rbf model. F1 = F-score or F-measure is a measure of a test's accuracy - the highest value appears in the nonlinear SVM model. LOSS - The smallest value appears in the Linear SVM model.
According to this values we can assume that the best model for this dataset is the Nonlinear SVM-rbf model.

In [18]:
```
#section 6
clf = rfc(n_estimators=10)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
x_test_scaled = scaler.transform(x_test)
clf.fit(X_train_scaled, Y_train)
w_log = clf.feature_importances_
```

```
labels = ['Gender','Increased Urination','Increased Thirst','Sudden Weight Loss','Weakn
x = np.arange(len(labels))
width = 0.5
mode_name = ['weights']
plt.bar(x, w_log)
plt.xticks(x, labels, rotation='vertical')
plt.show()
```



Question 6

What are the 2 most important features according to the random forest.

Answer: Increased Urination and Increased Thirst are the 2 most important features according to the random forest. Their weights were the biggest in our random forest graph.

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

Answer: There is a match with the feature exploration we did. Increased Urination and Increased Thirst got high weights in the random forest graph as expected.

Question 7

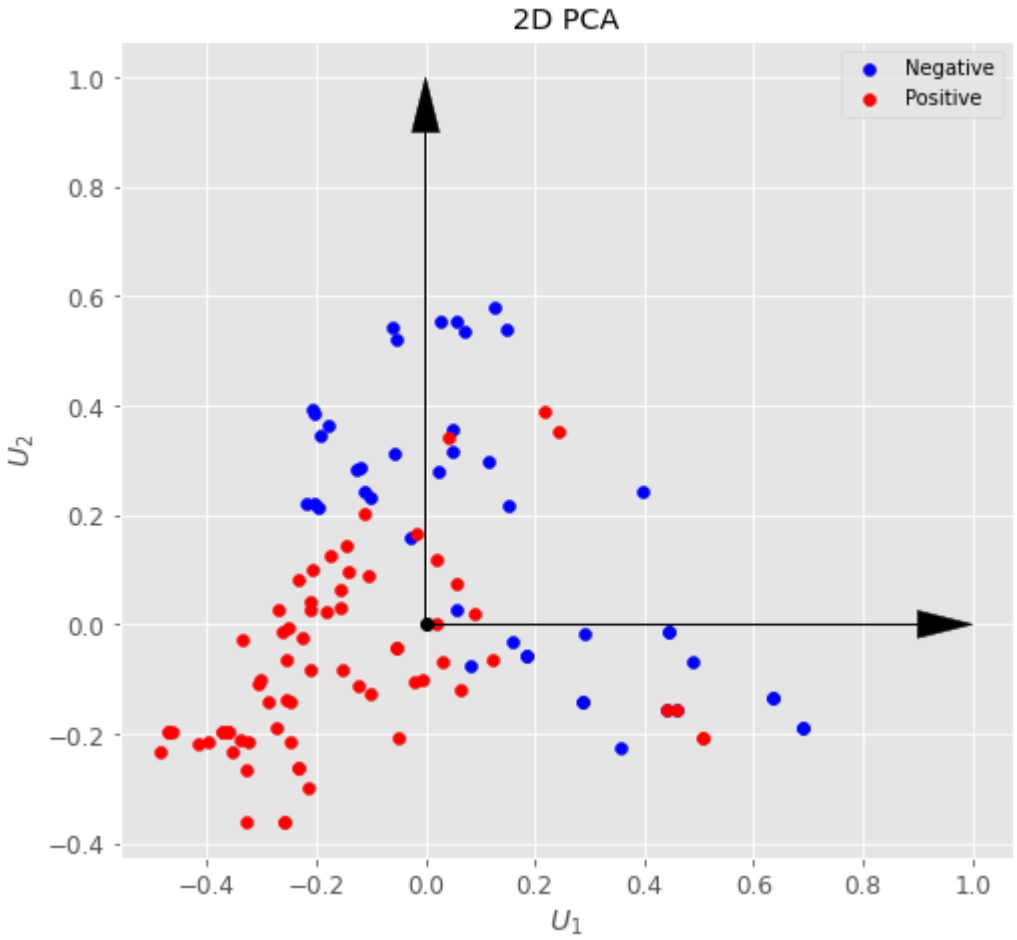
PCA is desinged for continuous variables. It tries to minimize variance. The concept of squared deviations breaks down when you have binary variables. You can use PCA for linear data but it will not yield meaningful resault.

In [19]:

```
#section 7
# If this section doesn't coverge please run the section again (run twice)
pca = KernelPCA(n_components=2, kernel='rbf')
# pca = PCA(n_components=2, whiten=True)
X_train_2d = pca.fit_transform(X_train_scaled)
x_test_2d = pca.transform(x_test_scaled)
```

In [20]:
```
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(('Negative','Positive'))
    ax.plot([0], [0], "ko")
    ax.arrow(0, 0, 0, 1, head_width=0.05, length_includes_head=True, head_length=0.1, f
    ax.arrow(0, 0, 1, 0, head_width=0.05, length_includes_head=True, head_length=0.1, f
    ax.set_xlabel('$U_1$')
    ax.set_ylabel('$U_2$')
    ax.set_title('2D PCA')
```

In [21]:
```
plt_2d_pca(x_test_2d,y_test)
plt.show()
```



Question # 7.B

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

As we can see from the graph we plot with dimention reduced data, it is not separable. We can't draw any line that separates the blue and red points that represent the samples in our data.

In [22]:
```
# #section 7c
# #Logistic Regression
C= np.array([0.0001, 0.001, 0.01, 1, 10, 100, 1000, 10000])
n_splits = 5
```

```
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
logreg_2d = LogisticRegression()
pen = ['l1','l2']
hyperparameters = dict(C=C, penalty=pen)

pipe = Pipeline(steps=[('scale', StandardScaler()), ('logistic', logreg_2d)])
log_reg_2d = GridSearchCV(logreg_2d, hyperparameters, scoring=['roc_auc'],  cv=skf, ref
log_reg_2d.fit(X_train_2d, Y_train)
best_log_reg_2d = log_reg_2d.best_estimator_

Y_pred_train_log_2d = best_log_reg_2d.predict(X_train_2d)
Y_pred_proba_train_log_2d = best_log_reg_2d.predict_proba(X_train_2d)
log_reg_score_train_2d = roc_auc_score(Y_train, Y_pred_proba_train_log_2d[:,1])

y_pred_test_log_2d = best_log_reg_2d.predict(x_test_2d)
y_pred_proba_test_log_2d = best_log_reg_2d.predict_proba(x_test_2d)
log_reg_score_test_2d = roc_auc_score(y_test, y_pred_proba_test_log_2d[:,1])
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
[CV] C=0.0001, penalty=l1 ............................................
[CV]  C=0.0001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.0001, penalty=l1 ............................................
[CV]  C=0.0001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.0001, penalty=l1 ............................................
[CV]  C=0.0001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.0001, penalty=l1 ............................................
[CV]  C=0.0001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.0001, penalty=l1 ............................................
[CV]  C=0.0001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.0001, penalty=l2 ............................................
[CV]  C=0.0001, penalty=l2, roc_auc=(train=0.932, test=0.929), total=   0.1s
[CV] C=0.0001, penalty=l2 ............................................
[CV]  C=0.0001, penalty=l2, roc_auc=(train=0.928, test=0.960), total=   0.0s
[CV] C=0.0001, penalty=l2 ............................................
[CV]  C=0.0001, penalty=l2, roc_auc=(train=0.933, test=0.930), total=   0.0s
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
[CV] C=0.0001, penalty=l2 ............................................
[CV]  C=0.0001, penalty=l2, roc_auc=(train=0.937, test=0.916), total=   0.0s
[CV] C=0.0001, penalty=l2 ............................................
[CV]  C=0.0001, penalty=l2, roc_auc=(train=0.933, test=0.941), total=   0.0s
[CV] C=0.001, penalty=l1 .............................................
[CV]  C=0.001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.001, penalty=l1 .............................................
[CV]  C=0.001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.001, penalty=l1 .............................................
```

```
[CV]  C=0.001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.001, penalty=l1 ............................................
[CV]  C=0.001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.001, penalty=l1 ............................................
[CV]  C=0.001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.001, penalty=l2 ............................................

C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[CV]  C=0.001, penalty=l2, roc_auc=(train=0.931, test=0.929), total=   0.1s
[CV] C=0.001, penalty=l2 ............................................
[CV]  C=0.001, penalty=l2, roc_auc=(train=0.928, test=0.960), total=   0.0s
[CV] C=0.001, penalty=l2 ............................................
[CV]  C=0.001, penalty=l2, roc_auc=(train=0.933, test=0.930), total=   0.1s
[CV] C=0.001, penalty=l2 ............................................
[CV]  C=0.001, penalty=l2, roc_auc=(train=0.937, test=0.916), total=   0.1s
[CV] C=0.001, penalty=l2 ............................................
[CV]  C=0.001, penalty=l2, roc_auc=(train=0.933, test=0.942), total=   0.0s
[CV] C=0.01, penalty=l1 .............................................
[CV]  C=0.01, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.01, penalty=l1 .............................................
[CV]  C=0.01, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.01, penalty=l1 .............................................
[CV]  C=0.01, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.01, penalty=l1 .............................................
[CV]  C=0.01, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.01, penalty=l1 .............................................
[CV]  C=0.01, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.01, penalty=l2 .............................................
[CV]  C=0.01, penalty=l2, roc_auc=(train=0.933, test=0.931), total=   0.1s
[CV] C=0.01, penalty=l2 .............................................
[CV]  C=0.01, penalty=l2, roc_auc=(train=0.930, test=0.962), total=   0.0s

C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[CV] C=0.01, penalty=l2 .............................................
[CV]  C=0.01, penalty=l2, roc_auc=(train=0.935, test=0.930), total=   0.0s
[CV] C=0.01, penalty=l2 .............................................
```

```
[CV]  C=0.01, penalty=l2, roc_auc=(train=0.938, test=0.916), total=   0.1s
[CV] C=0.01, penalty=l2 ..............................................
[CV]  C=0.01, penalty=l2, roc_auc=(train=0.934, test=0.944), total=   0.0s
[CV] C=1.0, penalty=l1 ...............................................
[CV] . C=1.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1.0, penalty=l1 ...............................................
[CV] . C=1.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1.0, penalty=l1 ...............................................
[CV] . C=1.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1.0, penalty=l1 ...............................................
[CV] . C=1.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1.0, penalty=l1 ...............................................
[CV] . C=1.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1.0, penalty=l2 ...............................................
[CV]  C=1.0, penalty=l2, roc_auc=(train=0.952, test=0.959), total=   0.1s
[CV] C=1.0, penalty=l2 ...............................................
[CV]  C=1.0, penalty=l2, roc_auc=(train=0.951, test=0.976), total=   0.0s
[CV] C=1.0, penalty=l2 ...............................................
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[CV]  C=1.0, penalty=l2, roc_auc=(train=0.961, test=0.939), total=   0.0s
[CV] C=1.0, penalty=l2 ...............................................
[CV]  C=1.0, penalty=l2, roc_auc=(train=0.956, test=0.943), total=   0.0s
[CV] C=1.0, penalty=l2 ...............................................
[CV]  C=1.0, penalty=l2, roc_auc=(train=0.952, test=0.958), total=   0.0s
[CV] C=10.0, penalty=l1 ..............................................
[CV]  C=10.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10.0, penalty=l1 ..............................................
[CV]  C=10.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10.0, penalty=l1 ..............................................
[CV]  C=10.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10.0, penalty=l1 ..............................................
[CV]  C=10.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10.0, penalty=l1 ..............................................
[CV]  C=10.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10.0, penalty=l2 ..............................................
[CV]  C=10.0, penalty=l2, roc_auc=(train=0.954, test=0.961), total=   0.0s
[CV] C=10.0, penalty=l2 ..............................................
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
```

```
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[CV]  C=10.0, penalty=l2, roc_auc=(train=0.950, test=0.974), total=   0.0s
[CV] C=10.0, penalty=l2 ..............................................
[CV]  C=10.0, penalty=l2, roc_auc=(train=0.960, test=0.937), total=   0.1s
[CV] C=10.0, penalty=l2 ..............................................
[CV]  C=10.0, penalty=l2, roc_auc=(train=0.957, test=0.949), total=   0.0s
[CV] C=10.0, penalty=l2 ..............................................
[CV]  C=10.0, penalty=l2, roc_auc=(train=0.953, test=0.960), total=   0.0s
[CV] C=100.0, penalty=l1 .............................................
[CV]  C=100.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=100.0, penalty=l1 .............................................
[CV]  C=100.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=100.0, penalty=l1 .............................................
[CV]  C=100.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=100.0, penalty=l1 .............................................
[CV]  C=100.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=100.0, penalty=l1 .............................................
[CV]  C=100.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=100.0, penalty=l2 .............................................
[CV]  C=100.0, penalty=l2, roc_auc=(train=0.953, test=0.962), total=   0.0s
[CV] C=100.0, penalty=l2 .............................................
[CV]  C=100.0, penalty=l2, roc_auc=(train=0.951, test=0.972), total=   0.0s
[CV] C=100.0, penalty=l2 .............................................
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[CV]  C=100.0, penalty=l2, roc_auc=(train=0.961, test=0.936), total=   0.1s
[CV] C=100.0, penalty=l2 .............................................
[CV]  C=100.0, penalty=l2, roc_auc=(train=0.957, test=0.948), total=   0.0s
[CV] C=100.0, penalty=l2 .............................................
[CV]  C=100.0, penalty=l2, roc_auc=(train=0.952, test=0.960), total=   0.0s
[CV] C=1000.0, penalty=l1 ............................................
[CV]  C=1000.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1000.0, penalty=l1 ............................................
[CV]  C=1000.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1000.0, penalty=l1 ............................................
[CV]  C=1000.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1000.0, penalty=l1 ............................................
[CV]  C=1000.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1000.0, penalty=l1 ............................................
[CV]  C=1000.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1000.0, penalty=l2 ............................................
[CV]  C=1000.0, penalty=l2, roc_auc=(train=0.953, test=0.962), total=   0.0s
[CV] C=1000.0, penalty=l2 ............................................
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
```

```
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[CV]  C=1000.0, penalty=l2, roc_auc=(train=0.951, test=0.972), total=   0.0s
[CV] C=1000.0, penalty=l2 ............................................
[CV]  C=1000.0, penalty=l2, roc_auc=(train=0.961, test=0.936), total=   0.0s
[CV] C=1000.0, penalty=l2 ............................................
[CV]  C=1000.0, penalty=l2, roc_auc=(train=0.957, test=0.947), total=   0.0s
[CV] C=1000.0, penalty=l2 ............................................
[CV]  C=1000.0, penalty=l2, roc_auc=(train=0.952, test=0.960), total=   0.1s
[CV] C=10000.0, penalty=l1 ...........................................
[CV]  C=10000.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10000.0, penalty=l1 ...........................................
[CV]  C=10000.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10000.0, penalty=l1 ...........................................
[CV]  C=10000.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10000.0, penalty=l1 ...........................................
[CV]  C=10000.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10000.0, penalty=l1 ...........................................
[CV]  C=10000.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10000.0, penalty=l2 ...........................................
[CV]  C=10000.0, penalty=l2, roc_auc=(train=0.953, test=0.962), total=   0.0s
[CV] C=10000.0, penalty=l2 ...........................................
[CV]  C=10000.0, penalty=l2, roc_auc=(train=0.951, test=0.972), total=   0.1s
[CV] C=10000.0, penalty=l2 ...........................................
[CV]  C=10000.0, penalty=l2, roc_auc=(train=0.961, test=0.936), total=   0.0s
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[CV] C=10000.0, penalty=l2 ...........................................
[CV]  C=10000.0, penalty=l2, roc_auc=(train=0.957, test=0.947), total=   0.1s
[CV] C=10000.0, penalty=l2 ...........................................
[CV]  C=10000.0, penalty=l2, roc_auc=(train=0.952, test=0.960), total=   0.1s
[Parallel(n_jobs=1)]: Done  80 out of  80 | elapsed:    2.7s finished
```

In [23]:

```
#Linear SVM
C = np.array([0.0001, 0.001, 0.01, 1, 10, 100, 1000, 10000])
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
svc_2d = SVC(probability=True)
```

```
pipe = Pipeline(steps=[('scale', StandardScaler()), ('svm', svc_2d)])
svm_lin_2d = GridSearchCV(estimator=pipe,
             param_grid={'svm__kernel':['linear'], 'svm__C':C}, scoring=['roc_auc'],  c
svm_lin_2d.fit(X_train_2d, Y_train)
best_svc_lin_2d = svm_lin_2d.best_estimator_

Y_pred_train_svc_2d = best_svc_lin_2d.predict(X_train_2d)
Y_pred_proba_train_svc_2d = best_svc_lin_2d.predict_proba(X_train_2d)
LSVM_score_train_2d = roc_auc_score(Y_train, Y_pred_proba_train_svc_2d[:,1])

y_pred_test_svc_2d = best_svc_lin_2d.predict(x_test_2d)
y_pred_proba_test_svc_2d= best_svc_lin_2d.predict_proba(x_test_2d)
LSVM_score_test_2d = roc_auc_score(y_test, y_pred_proba_test_svc_2d[:,1])
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
Fitting 5 folds for each of 8 candidates, totalling 40 fits
[CV] svm__C=0.0001, svm__kernel=linear ..............................
[CV]  svm__C=0.0001, svm__kernel=linear, roc_auc=(train=0.950, test=0.955), total=   0.1
s
[CV] svm__C=0.0001, svm__kernel=linear ..............................
[CV]  svm__C=0.0001, svm__kernel=linear, roc_auc=(train=0.947, test=0.973), total=   0.1
s
[CV] svm__C=0.0001, svm__kernel=linear ..............................
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.1s remaining:    0.0s
[CV]  svm__C=0.0001, svm__kernel=linear, roc_auc=(train=0.956, test=0.939), total=   0.1
s
[CV] svm__C=0.0001, svm__kernel=linear ..............................
[CV]  svm__C=0.0001, svm__kernel=linear, roc_auc=(train=0.953, test=0.936), total=   0.1
s
[CV] svm__C=0.0001, svm__kernel=linear ..............................
[CV]  svm__C=0.0001, svm__kernel=linear, roc_auc=(train=0.950, test=0.957), total=   0.0
s
[CV] svm__C=0.001, svm__kernel=linear ...............................
[CV]  svm__C=0.001, svm__kernel=linear, roc_auc=(train=0.949, test=0.954), total=   0.0s
[CV] svm__C=0.001, svm__kernel=linear ...............................
[CV]  svm__C=0.001, svm__kernel=linear, roc_auc=(train=0.948, test=0.973), total=   0.0s
[CV] svm__C=0.001, svm__kernel=linear ...............................
[CV]  svm__C=0.001, svm__kernel=linear, roc_auc=(train=0.956, test=0.939), total=   0.0s
[CV] svm__C=0.001, svm__kernel=linear ...............................
[CV]  svm__C=0.001, svm__kernel=linear, roc_auc=(train=0.953, test=0.936), total=   0.0s
[CV] svm__C=0.001, svm__kernel=linear ...............................
[CV]  svm__C=0.001, svm__kernel=linear, roc_auc=(train=0.950, test=0.957), total=   0.0s
[CV] svm__C=0.01, svm__kernel=linear ................................
[CV]  svm__C=0.01, svm__kernel=linear, roc_auc=(train=0.952, test=0.956), total=   0.0s
[CV] svm__C=0.01, svm__kernel=linear ................................
[CV]  svm__C=0.01, svm__kernel=linear, roc_auc=(train=0.949, test=0.973), total=   0.0s
[CV] svm__C=0.01, svm__kernel=linear ................................
[CV]  svm__C=0.01, svm__kernel=linear, roc_auc=(train=0.959, test=0.938), total=   0.0s
[CV] svm__C=0.01, svm__kernel=linear ................................
[CV]  svm__C=0.01, svm__kernel=linear, roc_auc=(train=0.956, test=0.943), total=   0.0s
[CV] svm__C=0.01, svm__kernel=linear ................................
[CV]  svm__C=0.01, svm__kernel=linear, roc_auc=(train=0.952, test=0.959), total=   0.0s
[CV] svm__C=1.0, svm__kernel=linear .................................
[CV]  svm__C=1.0, svm__kernel=linear, roc_auc=(train=0.953, test=0.962), total=   0.0s
[CV] svm__C=1.0, svm__kernel=linear .................................
[CV]  svm__C=1.0, svm__kernel=linear, roc_auc=(train=0.952, test=0.972), total=   0.0s
[CV] svm__C=1.0, svm__kernel=linear .................................
[CV]  svm__C=1.0, svm__kernel=linear, roc_auc=(train=0.961, test=0.935), total=   0.0s
[CV] svm__C=1.0, svm__kernel=linear .................................
[CV]  svm__C=1.0, svm__kernel=linear, roc_auc=(train=0.957, test=0.949), total=   0.0s
[CV] svm__C=1.0, svm__kernel=linear .................................
[CV]  svm__C=1.0, svm__kernel=linear, roc_auc=(train=0.953, test=0.960), total=   0.0s
```

```
[CV] svm__C=10.0, svm__kernel=linear .................................
[CV]  svm__C=10.0, svm__kernel=linear, roc_auc=(train=0.953, test=0.962), total=   0.1s
[CV] svm__C=10.0, svm__kernel=linear .................................
[CV]  svm__C=10.0, svm__kernel=linear, roc_auc=(train=0.952, test=0.972), total=   0.1s
[CV] svm__C=10.0, svm__kernel=linear .................................
[CV]  svm__C=10.0, svm__kernel=linear, roc_auc=(train=0.961, test=0.936), total=   0.1s
[CV] svm__C=10.0, svm__kernel=linear .................................
[CV]  svm__C=10.0, svm__kernel=linear, roc_auc=(train=0.956, test=0.949), total=   0.1s
[CV] svm__C=10.0, svm__kernel=linear .................................
[CV]  svm__C=10.0, svm__kernel=linear, roc_auc=(train=0.953, test=0.959), total=   0.0s
[CV] svm__C=100.0, svm__kernel=linear ................................
[CV]  svm__C=100.0, svm__kernel=linear, roc_auc=(train=0.954, test=0.962), total=   0.3s
[CV] svm__C=100.0, svm__kernel=linear ................................
[CV]  svm__C=100.0, svm__kernel=linear, roc_auc=(train=0.952, test=0.971), total=   0.2s
[CV] svm__C=100.0, svm__kernel=linear ................................
[CV]  svm__C=100.0, svm__kernel=linear, roc_auc=(train=0.961, test=0.936), total=   0.1s
[CV] svm__C=100.0, svm__kernel=linear ................................
[CV]  svm__C=100.0, svm__kernel=linear, roc_auc=(train=0.956, test=0.950), total=   0.2s
[CV] svm__C=100.0, svm__kernel=linear ................................
[CV]  svm__C=100.0, svm__kernel=linear, roc_auc=(train=0.952, test=0.960), total=   0.1s
[CV] svm__C=1000.0, svm__kernel=linear ...............................
[CV]  svm__C=1000.0, svm__kernel=linear, roc_auc=(train=0.954, test=0.962), total=   2.0
s
[CV] svm__C=1000.0, svm__kernel=linear ...............................
[CV]  svm__C=1000.0, svm__kernel=linear, roc_auc=(train=0.952, test=0.971), total=   1.5
s
[CV] svm__C=1000.0, svm__kernel=linear ...............................
[CV]  svm__C=1000.0, svm__kernel=linear, roc_auc=(train=0.961, test=0.936), total=   0.9
s
[CV] svm__C=1000.0, svm__kernel=linear ...............................
[CV]  svm__C=1000.0, svm__kernel=linear, roc_auc=(train=0.956, test=0.950), total=   2.8
s
[CV] svm__C=1000.0, svm__kernel=linear ...............................
[CV]  svm__C=1000.0, svm__kernel=linear, roc_auc=(train=0.952, test=0.960), total=   0.8
s
[CV] svm__C=10000.0, svm__kernel=linear ..............................
[CV]  svm__C=10000.0, svm__kernel=linear, roc_auc=(train=0.954, test=0.962), total=  10.
7s
[CV] svm__C=10000.0, svm__kernel=linear ..............................
[CV]  svm__C=10000.0, svm__kernel=linear, roc_auc=(train=0.952, test=0.971), total=  11.
2s
[CV] svm__C=10000.0, svm__kernel=linear ..............................
[CV]  svm__C=10000.0, svm__kernel=linear, roc_auc=(train=0.961, test=0.935), total=   8.
9s
[CV] svm__C=10000.0, svm__kernel=linear ..............................
[CV]  svm__C=10000.0, svm__kernel=linear, roc_auc=(train=0.956, test=0.949), total=  12.
9s
[CV] svm__C=10000.0, svm__kernel=linear ..............................
[CV]  svm__C=10000.0, svm__kernel=linear, roc_auc=(train=0.952, test=0.960), total=   8.
4s
[Parallel(n_jobs=1)]: Done  40 out of  40 | elapsed:  1.0min finished
```

In [24]:

```
#NonLinear
#Non-Linear SVM- poly
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
svc_2d = SVC(probability=True)
C = np.array([0.0001, 0.001, 0.01, 1, 10, 100, 1000, 10000])
pipe = Pipeline(steps=[('scale', StandardScaler()), ('svm', svc_2d)])
svm_poly_2d = GridSearchCV(estimator=pipe,
             param_grid={'svm__kernel':['poly'], 'svm__C':C, 'svm__degree':[3], 'svm__g
svm_poly_2d.fit(X_train_2d, Y_train)
```

```
best_svm_poly_2d = svm_poly_2d.best_estimator_

Y_pred_train_poly_2d = best_svm_poly_2d.predict(X_train_2d)
Y_pred_proba_train_poly_2d = best_svm_poly_2d.predict_proba(X_train_2d)
poly_score_train_2d = roc_auc_score(Y_train, Y_pred_proba_train_svc_2d[:,1])

y_pred_test_poly_2d = best_svm_poly_2d.predict(x_test_2d)
y_pred_proba_test_poly_2d= best_svm_poly_2d.predict_proba(x_test_2d)
poly_score_test_2d = roc_auc_score(y_test, y_pred_proba_test_poly_2d[:,1])
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
958, test=0.966), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
956, test=0.969), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.1s remaining:    0.0s
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
957, test=0.946), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
957, test=0.957), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
955, test=0.959), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.958, test=0.966), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.956, test=0.969), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.957, test=0.946), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.957, test=0.957), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.955, test=0.959), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
58, test=0.966), total=   0.0s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
56, test=0.969), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
57, test=0.946), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
57, test=0.957), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
55, test=0.959), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
958, test=0.966), total=   0.0s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
```

```
956, test=0.969), total=   0.0s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
957, test=0.946), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
957, test=0.957), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
955, test=0.959), total=   0.0s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.96
0, test=0.970), total=   0.0s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.96
0, test=0.968), total=   0.0s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.96
5, test=0.952), total=   0.0s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.96
2, test=0.957), total=   0.0s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.96
2, test=0.957), total=   0.0s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
60, test=0.970), total=   0.0s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
60, test=0.968), total=   0.0s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
65, test=0.952), total=   0.0s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
62, test=0.957), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
62, test=0.957), total=   0.0s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.95
8, test=0.971), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.95
9, test=0.964), total=   0.0s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.96
5, test=0.949), total=   0.0s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.96
3, test=0.955), total=   0.0s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.96
1, test=0.961), total=   0.0s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ...
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.95
8, test=0.971), total=   0.0s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ...
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.95
9, test=0.964), total=   0.0s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ...
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.96
5, test=0.949), total=   0.0s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ...
```

```
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.96
3, test=0.955), total=   0.0s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ...
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.96
1, test=0.961), total=   0.0s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.95
1, test=0.974), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.95
3, test=0.961), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.96
5, test=0.943), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.96
0, test=0.953), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.95
5, test=0.956), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
51, test=0.974), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
53, test=0.961), total=   0.0s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
65, test=0.943), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
60, test=0.953), total=   0.0s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
54, test=0.956), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
51, test=0.974), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
53, test=0.961), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
63, test=0.934), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
59, test=0.949), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
53, test=0.952), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
951, test=0.974), total=   0.2s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
953, test=0.961), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
963, test=0.934), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
959, test=0.949), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
953, test=0.952), total=   0.1s
```

```
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
951, test=0.974), total=   0.5s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
953, test=0.962), total=   0.5s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
962, test=0.934), total=   0.5s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
959, test=0.949), total=   0.7s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
953, test=0.952), total=   0.5s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.951, test=0.974), total=   0.9s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.953, test=0.962), total=   0.9s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.962, test=0.934), total=   0.5s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.959, test=0.949), total=   0.6s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.953, test=0.952), total=   0.6s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=
0.951, test=0.974), total=   5.5s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=
0.953, test=0.961), total=   3.7s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=
0.963, test=0.934), total=   3.3s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=
0.959, test=0.949), total=   3.2s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=
0.953, test=0.952), total=   3.8s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.951, test=0.974), total=   5.0s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.953, test=0.961), total=   5.7s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.963, test=0.934), total=   4.0s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.959, test=0.949), total=   4.3s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.953, test=0.952), total=   3.1s
[Parallel(n_jobs=1)]: Done  80 out of  80 | elapsed:   53.2s finished
```

In [25]:

```
#Non-Linear SVM- rbf
n_splits = 5
```

```
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
svc_2d = SVC(probability=True)
C = np.array([0.0001, 0.001, 0.01, 1, 10, 100, 1000, 10000])
pipe = Pipeline(steps=[('scale', StandardScaler()), ('svm', svc_2d)])
svm_rbf_2d = GridSearchCV(estimator=pipe,
             param_grid={'svm__kernel':['rbf'], 'svm__C':C, 'svm__degree':[3], 'svm__ga
svm_rbf_2d.fit(X_train_2d, Y_train)

best_svm_rbf_2d = svm_rbf_2d.best_estimator_

Y_pred_train_rbf_2d = best_svm_poly_2d.predict(X_train_2d)
Y_pred_proba_train_rbf_2d = best_svm_poly_2d.predict_proba(X_train_2d)
rbf_score_train_2d = roc_auc_score(Y_train, Y_pred_proba_train_svc_2d[:,1])

y_pred_test_rbf_2d = best_svm_rbf_2d.predict(x_test_2d)
y_pred_proba_test_rbf_2d= best_svm_rbf_2d.predict_proba(x_test_2d)
rbf_score_test_2d = roc_auc_score(y_test, y_pred_proba_test_rbf_2d[:,1])
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.9
52, test=0.971), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.1s remaining:    0.0s
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.9
52, test=0.967), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.9
57, test=0.948), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.9
64, test=0.925), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.9
54, test=0.963), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
952, test=0.971), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
952, test=0.967), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
957, test=0.948), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
964, test=0.925), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
954, test=0.963), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.95
2, test=0.971), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.95
2, test=0.968), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.95
8, test=0.948), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.96
```

```
4, test=0.925), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.95
3, test=0.965), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
52, test=0.971), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
52, test=0.968), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
58, test=0.948), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
64, test=0.925), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
53, test=0.965), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.95
2, test=0.971), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.95
3, test=0.968), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.95
8, test=0.948), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.96
4, test=0.926), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.95
3, test=0.965), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.95
2, test=0.971), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.95
3, test=0.968), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.95
8, test=0.948), total=   0.2s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.96
4, test=0.926), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.95
3, test=0.965), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.954,
test=0.970), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.958,
test=0.975), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.969,
test=0.940), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.969,
test=0.928), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.960,
test=0.963), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ....
```

```
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.95
4, test=0.970), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.95
8, test=0.975), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.96
9, test=0.940), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.96
9, test=0.928), total=   0.0s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.96
0, test=0.963), total=   0.0s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.96
2, test=0.972), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.96
5, test=0.976), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.97
6, test=0.928), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.96
8, test=0.935), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.95
8, test=0.953), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.96
2, test=0.972), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.96
5, test=0.976), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.97
6, test=0.928), total=   0.0s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.96
8, test=0.935), total=   0.0s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.95
8, test=0.953), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.97
4, test=0.968), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.97
4, test=0.974), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.98
3, test=0.932), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.97
0, test=0.947), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.97
2, test=0.965), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
74, test=0.968), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
74, test=0.974), total=   0.1s
```

```
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
83, test=0.932), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
70, test=0.947), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
72, test=0.965), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.9
84, test=0.971), total=   0.2s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.9
84, test=0.975), total=   0.2s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.9
88, test=0.940), total=   0.2s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.9
82, test=0.982), total=   0.3s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.9
84, test=0.964), total=   0.2s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
984, test=0.971), total=   0.3s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
984, test=0.975), total=   0.2s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
988, test=0.940), total=   0.2s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
982, test=0.982), total=   0.2s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
984, test=0.964), total=   0.2s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.
991, test=0.966), total=   1.3s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.
986, test=0.976), total=   1.4s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.
992, test=0.951), total=   0.9s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.
990, test=0.974), total=   2.1s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.
993, test=0.962), total=   1.3s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=
0.991, test=0.966), total=   1.3s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=
0.986, test=0.976), total=   1.5s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=
0.992, test=0.951), total=   0.9s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=
```

```
0.990, test=0.974), total=   1.5s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=
0.993, test=0.962), total=   1.1s
[Parallel(n_jobs=1)]: Done  80 out of  80 | elapsed:   21.8s finished
```

In [26]:

```
F1_logreg_train_2d, Acc_logreg_train_2d = calc_metrics(Y_train, Y_pred_train_log_2d)
F1_svc_train_2d, Acc_svc_train_2d = calc_metrics(Y_train, Y_pred_train_svc_2d)
F1_poly_train_2d, Acc_poly_train_2d = calc_metrics(Y_train, Y_pred_train_poly_2d)
F1_rbf_train_2d, Acc_rbf_train_2d = calc_metrics(Y_train, Y_pred_train_rbf_2d)

F1_logreg_test_2d, Acc_logreg_test_2d = calc_metrics(y_test, y_pred_test_log_2d)
F1_svc_test_2d, Acc_svc_test_2d = calc_metrics(y_test, y_pred_test_svc_2d)
F1_poly_test_2d, Acc_poly_test_2d = calc_metrics(y_test, y_pred_test_poly_2d)
F1_rbf_test_2d, Acc_rbf_test_2d = calc_metrics(y_test, y_pred_test_rbf_2d)
```

In [27]:

```
print("\nLogistic Regression:")
print('\nFor dimensionality-reduced')
print('\nFor Training Set: \nAUC is {:.2f}. \nF1 is {:.2f}.  \nACC is {:.2f}. '.format(
print('\nFor Test Set: \nAUC is {:.2f}. \nF1 is {:.2f}. \nACC is {:.2f}.\n '.format(log


print("\nLinear SVM:")
print('\nFor dimensionality-reduced')
print('\nFor Training Set: \nAUC is {:.2f}. \nF1 is {:.2f}.  \nACC is {:.2f}. '.format(
print('\nFor Test Set: \nAUC is {:.2f}. \nF1 is {:.2f}. \nACC is {:.2f}.\n '.format(LSV


print("\nPolynomial SVM:")
print('\nFor dimensionality-reduced')
print('\nFor Training Set: \nAUC is {:.2f}. \nF1 is {:.2f}.  \nACC is {:.2f}. '.format(
print('\nFor Test Set: \nAUC is {:.2f}. \nF1 is {:.2f}. \nACC is {:.2f}.\n '.format(pol


print("\nSVM RBF:")
print('\nFor dimensionality-reduced')
print('\nFor Training Set: \nAUC is {:.2f}. \nF1 is {:.2f}.  \nACC is {:.2f}. '.format(
print('\nFor Test Set: \nAUC is {:.2f}. \nF1 is {:.2f}. \nACC is {:.2f}.\n '.format(rbf
```

```
Logistic Regression:

For dimensionality-reduced

For Training Set:
AUC is 0.95.
F1 is 0.89.
ACC is 0.87.

For Test Set:
AUC is 0.92.
F1 is 0.86.
ACC is 0.83.


Linear SVM:

For dimensionality-reduced

For Training Set:
AUC is 0.95.
```

```
F1 is 0.89.
ACC is 0.87.

For Test Set:
AUC is 0.92.
F1 is 0.86.
ACC is 0.83.


Polynomial SVM:

For dimensionality-reduced

For Training Set:
AUC is 0.95.
F1 is 0.86.
ACC is 0.81.

For Test Set:
AUC is 0.92.
F1 is 0.81.
ACC is 0.72.


SVM RBF:

For dimensionality-reduced

For Training Set:
AUC is 0.95.
F1 is 0.86.
ACC is 0.81.

For Test Set:
AUC is 0.95.
F1 is 0.91.
ACC is 0.89.
```

In [28]:

```
#section 7d
#The 2 selected features: "Increased Urination","Increased Thirst"
diagnosis = file_dataset[['Diagnosis']]

X_train_2f = X_train[['Increased Thirst','Increased Urination']]
x_test_2f = x_test[['Increased Thirst','Increased Urination']]
```

In [29]:

```
#Logistic Regression
C= np.array([0.0001, 0.001, 0.01, 1, 10, 100, 1000, 10000])
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
logreg_2f = LogisticRegression()
pen = ['l1','l2']
hyperparameters = dict(C=C, penalty=pen)

pipe = Pipeline(steps=[('scale', StandardScaler()), ('logistic', logreg_2f)])
log_reg_2f = GridSearchCV(logreg_2f, hyperparameters, scoring=['roc_auc'],  cv=skf, ref
log_reg_2f.fit(X_train_2f, Y_train)
best_log_reg_2f = log_reg_2f.best_estimator_

Y_pred_train_log_2f = best_log_reg_2d.predict(X_train_2f)
Y_pred_proba_train_log_2f = best_log_reg_2d.predict_proba(X_train_2f)
```

```
log_reg_score_train_2f = roc_auc_score(Y_train, Y_pred_proba_train_log_2f[:,1])

y_pred_test_log_2f = best_log_reg_2f.predict(x_test_2f)
y_pred_proba_test_log_2f = best_log_reg_2f.predict_proba(x_test_2f)
log_reg_score_test_2f = roc_auc_score(y_test, y_pred_proba_test_log_2f[:,1])
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
[CV] C=0.0001, penalty=l1 ..........................................
[CV]  C=0.0001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.0001, penalty=l1 ..........................................
[CV]  C=0.0001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.0001, penalty=l1 ..........................................
[CV]  C=0.0001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.0001, penalty=l1 ..........................................
[CV]  C=0.0001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.0001, penalty=l1 ..........................................
[CV]  C=0.0001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.0001, penalty=l2 ..........................................
[CV]  C=0.0001, penalty=l2, roc_auc=(train=0.910, test=0.898), total=   0.1s
[CV] C=0.0001, penalty=l2 ..........................................
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
[CV]  C=0.0001, penalty=l2, roc_auc=(train=0.896, test=0.954), total=   0.2s
[CV] C=0.0001, penalty=l2 ..........................................
[CV]  C=0.0001, penalty=l2, roc_auc=(train=0.914, test=0.884), total=   0.1s
[CV] C=0.0001, penalty=l2 ..........................................
[CV]  C=0.0001, penalty=l2, roc_auc=(train=0.905, test=0.918), total=   0.1s
[CV] C=0.0001, penalty=l2 ..........................................
[CV]  C=0.0001, penalty=l2, roc_auc=(train=0.913, test=0.881), total=   0.1s
[CV] C=0.001, penalty=l1 ...........................................
[CV]  C=0.001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.001, penalty=l1 ...........................................
[CV]  C=0.001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.001, penalty=l1 ...........................................
[CV]  C=0.001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.001, penalty=l1 ...........................................
[CV]  C=0.001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.001, penalty=l1 ...........................................
[CV]  C=0.001, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.001, penalty=l2 ...........................................
```

```
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
```

```
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[CV]  C=0.001, penalty=l2, roc_auc=(train=0.910, test=0.898), total=   0.1s
[CV] C=0.001, penalty=l2 .............................................
[CV]  C=0.001, penalty=l2, roc_auc=(train=0.896, test=0.954), total=   0.1s
[CV] C=0.001, penalty=l2 .............................................
[CV]  C=0.001, penalty=l2, roc_auc=(train=0.914, test=0.884), total=   0.1s
[CV] C=0.001, penalty=l2 .............................................
[CV]  C=0.001, penalty=l2, roc_auc=(train=0.905, test=0.918), total=   0.1s
[CV] C=0.001, penalty=l2 .............................................
[CV]  C=0.001, penalty=l2, roc_auc=(train=0.913, test=0.881), total=   0.1s
[CV] C=0.01, penalty=l1 ..............................................
[CV]  C=0.01, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.01, penalty=l1 ..............................................
[CV]  C=0.01, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.01, penalty=l1 ..............................................
[CV]  C=0.01, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.01, penalty=l1 ..............................................
[CV]  C=0.01, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[CV] C=0.01, penalty=l1 ..............................................
[CV]  C=0.01, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=0.01, penalty=l2 ..............................................
[CV]  C=0.01, penalty=l2, roc_auc=(train=0.910, test=0.898), total=   0.1s
[CV] C=0.01, penalty=l2 ..............................................
[CV]  C=0.01, penalty=l2, roc_auc=(train=0.896, test=0.954), total=   0.1s
[CV] C=0.01, penalty=l2 ..............................................
[CV]  C=0.01, penalty=l2, roc_auc=(train=0.914, test=0.884), total=   0.1s
[CV] C=0.01, penalty=l2 ..............................................
[CV]  C=0.01, penalty=l2, roc_auc=(train=0.905, test=0.918), total=   0.0s
[CV] C=0.01, penalty=l2 ..............................................
[CV]  C=0.01, penalty=l2, roc_auc=(train=0.913, test=0.881), total=   0.0s
[CV] C=1.0, penalty=l1 ...............................................
[CV] . C=1.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1.0, penalty=l1 ...............................................
[CV] . C=1.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1.0, penalty=l1 ...............................................
[CV] . C=1.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1.0, penalty=l1 ...............................................
[CV] . C=1.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
```

```
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[CV] C=1.0, penalty=l1 ...............................................
[CV] . C=1.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1.0, penalty=l2 ...............................................
[CV]  C=1.0, penalty=l2, roc_auc=(train=0.910, test=0.898), total=   0.1s
[CV] C=1.0, penalty=l2 ...............................................
[CV]  C=1.0, penalty=l2, roc_auc=(train=0.896, test=0.954), total=   0.1s
[CV] C=1.0, penalty=l2 ...............................................
[CV]  C=1.0, penalty=l2, roc_auc=(train=0.914, test=0.884), total=   0.1s
[CV] C=1.0, penalty=l2 ...............................................
[CV]  C=1.0, penalty=l2, roc_auc=(train=0.905, test=0.918), total=   0.0s
[CV] C=1.0, penalty=l2 ...............................................
[CV]  C=1.0, penalty=l2, roc_auc=(train=0.913, test=0.881), total=   0.1s
[CV] C=10.0, penalty=l1 ..............................................
[CV]  C=10.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10.0, penalty=l1 ..............................................
[CV]  C=10.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10.0, penalty=l1 ..............................................
[CV]  C=10.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10.0, penalty=l1 ..............................................
[CV]  C=10.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10.0, penalty=l1 ..............................................
[CV]  C=10.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10.0, penalty=l2 ..............................................
[CV]  C=10.0, penalty=l2, roc_auc=(train=0.910, test=0.898), total=   0.1s
[CV] C=10.0, penalty=l2 ..............................................
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[CV]  C=10.0, penalty=l2, roc_auc=(train=0.896, test=0.954), total=   0.1s
[CV] C=10.0, penalty=l2 ..............................................
[CV]  C=10.0, penalty=l2, roc_auc=(train=0.914, test=0.884), total=   0.1s
[CV] C=10.0, penalty=l2 ..............................................
[CV]  C=10.0, penalty=l2, roc_auc=(train=0.905, test=0.918), total=   0.1s
[CV] C=10.0, penalty=l2 ..............................................
[CV]  C=10.0, penalty=l2, roc_auc=(train=0.913, test=0.881), total=   0.1s
[CV] C=100.0, penalty=l1 .............................................
[CV]  C=100.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=100.0, penalty=l1 .............................................
```

```
[CV]  C=100.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=100.0, penalty=l1 ..............................................
[CV]  C=100.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=100.0, penalty=l1 ..............................................
[CV]  C=100.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=100.0, penalty=l1 ..............................................
[CV]  C=100.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=100.0, penalty=l2 ..............................................
[CV]  C=100.0, penalty=l2, roc_auc=(train=0.910, test=0.898), total=   0.1s
[CV] C=100.0, penalty=l2 ..............................................
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[CV]  C=100.0, penalty=l2, roc_auc=(train=0.896, test=0.954), total=   0.1s
[CV] C=100.0, penalty=l2 ..............................................
[CV]  C=100.0, penalty=l2, roc_auc=(train=0.914, test=0.884), total=   0.1s
[CV] C=100.0, penalty=l2 ..............................................
[CV]  C=100.0, penalty=l2, roc_auc=(train=0.905, test=0.918), total=   0.0s
[CV] C=100.0, penalty=l2 ..............................................
[CV]  C=100.0, penalty=l2, roc_auc=(train=0.913, test=0.881), total=   0.1s
[CV] C=1000.0, penalty=l1 .............................................
[CV]  C=1000.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1000.0, penalty=l1 .............................................
[CV]  C=1000.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1000.0, penalty=l1 .............................................
[CV]  C=1000.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1000.0, penalty=l1 .............................................
[CV]  C=1000.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=1000.0, penalty=l2 .............................................
[CV]  C=1000.0, penalty=l2, roc_auc=(train=0.910, test=0.898), total=   0.1s
[CV] C=1000.0, penalty=l2 .............................................
[CV]  C=1000.0, penalty=l2, roc_auc=(train=0.896, test=0.954), total=   0.1s
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[CV] C=1000.0, penalty=l2 .............................................
```

```
[CV]  C=1000.0, penalty=l2, roc_auc=(train=0.914, test=0.884), total=   0.1s
[CV] C=1000.0, penalty=l2 ............................................
[CV]  C=1000.0, penalty=l2, roc_auc=(train=0.905, test=0.918), total=   0.1s
[CV] C=1000.0, penalty=l2 ............................................
[CV]  C=1000.0, penalty=l2, roc_auc=(train=0.913, test=0.881), total=   0.1s
[CV] C=10000.0, penalty=l1 ...........................................
[CV]  C=10000.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10000.0, penalty=l1 ...........................................
[CV]  C=10000.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10000.0, penalty=l1 ...........................................
[CV]  C=10000.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10000.0, penalty=l1 ...........................................
[CV]  C=10000.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10000.0, penalty=l1 ...........................................
[CV]  C=10000.0, penalty=l1, roc_auc=(train=nan, test=nan), total=   0.0s
[CV] C=10000.0, penalty=l2 ...........................................
C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection\_valida
tion.py:552: FitFailedWarning: Estimator fit failed. The score on this train-test partit
ion for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\model_selection
\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\OriR\anaconda3\envs\bm-336546\lib\site-packages\sklearn\linear_model\_l
ogistic.py", line 443, in _check_solver
    "got %s penalty." % (solver, penalty))
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
[CV]  C=10000.0, penalty=l2, roc_auc=(train=0.910, test=0.898), total=   0.1s
[CV] C=10000.0, penalty=l2 ...........................................
[CV]  C=10000.0, penalty=l2, roc_auc=(train=0.896, test=0.954), total=   0.1s
[CV] C=10000.0, penalty=l2 ...........................................
[CV]  C=10000.0, penalty=l2, roc_auc=(train=0.914, test=0.884), total=   0.1s
[CV] C=10000.0, penalty=l2 ...........................................
[CV]  C=10000.0, penalty=l2, roc_auc=(train=0.905, test=0.918), total=   0.1s
[CV] C=10000.0, penalty=l2 ...........................................
[CV]  C=10000.0, penalty=l2, roc_auc=(train=0.913, test=0.881), total=   0.1s
[Parallel(n_jobs=1)]: Done  80 out of  80 | elapsed:    4.2s finished
```

In [30]:

```python
#Linear SVM
C = np.array([0.0001, 0.001, 0.01, 1, 10, 100, 1000, 10000])
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
svc_2f = SVC(probability=True)

pipe = Pipeline(steps=[('scale', StandardScaler()), ('svm', svc_2f)])
svm_lin_2f = GridSearchCV(estimator=pipe,
             param_grid={'svm__kernel':['linear'], 'svm__C':C}, scoring=['roc_auc'],  c
svm_lin_2f.fit(X_train_2f, Y_train)
best_svc_lin_2f = svm_lin_2f.best_estimator_

Y_pred_train_svc_2f = best_svc_lin_2d.predict(X_train_2f)
Y_pred_proba_train_svc_2f = best_svc_lin_2d.predict_proba(X_train_2f)
LSVM_score_train_2f = roc_auc_score(Y_train, Y_pred_proba_train_svc_2f[:,1])

y_pred_test_svc_2f = best_svc_lin_2f.predict(x_test_2f)
y_pred_proba_test_svc_2f= best_svc_lin_2f.predict_proba(x_test_2f)
LSVM_score_test_2f = roc_auc_score(y_test, y_pred_proba_test_svc_2f[:,1])
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
Fitting 5 folds for each of 8 candidates, totalling 40 fits
[CV] svm__C=0.0001, svm__kernel=linear ..............................
[CV]  svm__C=0.0001, svm__kernel=linear, roc_auc=(train=0.910, test=0.898), total=   0.1
s
[CV] svm__C=0.0001, svm__kernel=linear ..............................
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.2s remaining:    0.0s
[CV]  svm__C=0.0001, svm__kernel=linear, roc_auc=(train=0.896, test=0.954), total=   0.1
s
[CV] svm__C=0.0001, svm__kernel=linear ..............................
[CV]  svm__C=0.0001, svm__kernel=linear, roc_auc=(train=0.914, test=0.884), total=   0.1
s
[CV] svm__C=0.0001, svm__kernel=linear ..............................
[CV]  svm__C=0.0001, svm__kernel=linear, roc_auc=(train=0.905, test=0.918), total=   0.1
s
[CV] svm__C=0.0001, svm__kernel=linear ..............................
[CV]  svm__C=0.0001, svm__kernel=linear, roc_auc=(train=0.913, test=0.881), total=   0.1
s
[CV] svm__C=0.001, svm__kernel=linear ...............................
[CV]  svm__C=0.001, svm__kernel=linear, roc_auc=(train=0.910, test=0.898), total=   0.1s
[CV] svm__C=0.001, svm__kernel=linear ...............................
[CV]  svm__C=0.001, svm__kernel=linear, roc_auc=(train=0.896, test=0.954), total=   0.1s
[CV] svm__C=0.001, svm__kernel=linear ...............................
[CV]  svm__C=0.001, svm__kernel=linear, roc_auc=(train=0.914, test=0.884), total=   0.1s
[CV] svm__C=0.001, svm__kernel=linear ...............................
[CV]  svm__C=0.001, svm__kernel=linear, roc_auc=(train=0.905, test=0.918), total=   0.1s
[CV] svm__C=0.001, svm__kernel=linear ...............................
[CV]  svm__C=0.001, svm__kernel=linear, roc_auc=(train=0.913, test=0.881), total=   0.1s
[CV] svm__C=0.01, svm__kernel=linear ................................
[CV]  svm__C=0.01, svm__kernel=linear, roc_auc=(train=0.910, test=0.898), total=   0.1s
[CV] svm__C=0.01, svm__kernel=linear ................................
[CV]  svm__C=0.01, svm__kernel=linear, roc_auc=(train=0.896, test=0.954), total=   0.1s
[CV] svm__C=0.01, svm__kernel=linear ................................
[CV]  svm__C=0.01, svm__kernel=linear, roc_auc=(train=0.914, test=0.884), total=   0.1s
[CV] svm__C=0.01, svm__kernel=linear ................................
[CV]  svm__C=0.01, svm__kernel=linear, roc_auc=(train=0.905, test=0.918), total=   0.1s
[CV] svm__C=0.01, svm__kernel=linear ................................
[CV]  svm__C=0.01, svm__kernel=linear, roc_auc=(train=0.913, test=0.881), total=   0.1s
[CV] svm__C=1.0, svm__kernel=linear .................................
[CV]  svm__C=1.0, svm__kernel=linear, roc_auc=(train=0.903, test=0.903), total=   0.1s
[CV] svm__C=1.0, svm__kernel=linear .................................
[CV]  svm__C=1.0, svm__kernel=linear, roc_auc=(train=0.891, test=0.952), total=   0.1s
[CV] svm__C=1.0, svm__kernel=linear .................................
[CV]  svm__C=1.0, svm__kernel=linear, roc_auc=(train=0.910, test=0.877), total=   0.1s
[CV] svm__C=1.0, svm__kernel=linear .................................
[CV]  svm__C=1.0, svm__kernel=linear, roc_auc=(train=0.905, test=0.918), total=   0.1s
[CV] svm__C=1.0, svm__kernel=linear .................................
[CV]  svm__C=1.0, svm__kernel=linear, roc_auc=(train=0.911, test=0.874), total=   0.1s
[CV] svm__C=10.0, svm__kernel=linear ................................
[CV]  svm__C=10.0, svm__kernel=linear, roc_auc=(train=0.903, test=0.903), total=   0.1s
[CV] svm__C=10.0, svm__kernel=linear ................................
[CV]  svm__C=10.0, svm__kernel=linear, roc_auc=(train=0.891, test=0.952), total=   0.1s
[CV] svm__C=10.0, svm__kernel=linear ................................
[CV]  svm__C=10.0, svm__kernel=linear, roc_auc=(train=0.910, test=0.877), total=   0.0s
[CV] svm__C=10.0, svm__kernel=linear ................................
[CV]  svm__C=10.0, svm__kernel=linear, roc_auc=(train=0.905, test=0.918), total=   0.0s
[CV] svm__C=10.0, svm__kernel=linear ................................
[CV]  svm__C=10.0, svm__kernel=linear, roc_auc=(train=0.911, test=0.874), total=   0.0s
[CV] svm__C=100.0, svm__kernel=linear ...............................
[CV]  svm__C=100.0, svm__kernel=linear, roc_auc=(train=0.903, test=0.903), total=   0.0s
[CV] svm__C=100.0, svm__kernel=linear ...............................
[CV]  svm__C=100.0, svm__kernel=linear, roc_auc=(train=0.891, test=0.952), total=   0.0s
[CV] svm__C=100.0, svm__kernel=linear ...............................
```

```
[CV]  svm__C=100.0, svm__kernel=linear, roc_auc=(train=0.910, test=0.877), total=   0.0s
[CV] svm__C=100.0, svm__kernel=linear ................................
[CV]  svm__C=100.0, svm__kernel=linear, roc_auc=(train=0.905, test=0.918), total=   0.1s
[CV] svm__C=100.0, svm__kernel=linear ................................
[CV]  svm__C=100.0, svm__kernel=linear, roc_auc=(train=0.911, test=0.874), total=   0.0s
[CV] svm__C=1000.0, svm__kernel=linear ...............................
[CV]  svm__C=1000.0, svm__kernel=linear, roc_auc=(train=0.903, test=0.903), total=   0.0
s
[CV] svm__C=1000.0, svm__kernel=linear ...............................
[CV]  svm__C=1000.0, svm__kernel=linear, roc_auc=(train=0.891, test=0.952), total=   0.0
s
[CV] svm__C=1000.0, svm__kernel=linear ...............................
[CV]  svm__C=1000.0, svm__kernel=linear, roc_auc=(train=0.910, test=0.877), total=   0.0
s
[CV] svm__C=1000.0, svm__kernel=linear ...............................
[CV]  svm__C=1000.0, svm__kernel=linear, roc_auc=(train=0.905, test=0.918), total=   0.0
s
[CV] svm__C=1000.0, svm__kernel=linear ...............................
[CV]  svm__C=1000.0, svm__kernel=linear, roc_auc=(train=0.911, test=0.874), total=   0.0
s
[CV] svm__C=10000.0, svm__kernel=linear ..............................
[CV]  svm__C=10000.0, svm__kernel=linear, roc_auc=(train=0.903, test=0.903), total=   0.
1s
[CV] svm__C=10000.0, svm__kernel=linear ..............................
[CV]  svm__C=10000.0, svm__kernel=linear, roc_auc=(train=0.891, test=0.952), total=   0.
0s
[CV] svm__C=10000.0, svm__kernel=linear ..............................
[CV]  svm__C=10000.0, svm__kernel=linear, roc_auc=(train=0.910, test=0.877), total=   0.
1s
[CV] svm__C=10000.0, svm__kernel=linear ..............................
[CV]  svm__C=10000.0, svm__kernel=linear, roc_auc=(train=0.905, test=0.918), total=   0.
1s
[CV] svm__C=10000.0, svm__kernel=linear ..............................
[CV]  svm__C=10000.0, svm__kernel=linear, roc_auc=(train=0.911, test=0.874), total=   0.
1s
[Parallel(n_jobs=1)]: Done  40 out of  40 | elapsed:    3.7s finished
```

In [31]:

```
#NonLinear
#Non-Linear SVM- poly
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
svc_2f = SVC(probability=True)
C = np.array([0.0001, 0.001, 0.01, 1, 10, 100, 1000, 10000])
pipe = Pipeline(steps=[('scale', StandardScaler()), ('svm', svc_2f)])
svm_poly_2f = GridSearchCV(estimator=pipe,
             param_grid={'svm__kernel':['poly'], 'svm__C':C, 'svm__degree':[3], 'svm__g
svm_poly_2f.fit(X_train_2f, Y_train)

best_svm_poly_2f = svm_poly_2f.best_estimator_

Y_pred_train_poly_2f = best_svm_poly_2d.predict(X_train_2f)
Y_pred_proba_train_poly_2f = best_svm_poly_2d.predict_proba(X_train_2f)
poly_score_train_2f = roc_auc_score(Y_train, Y_pred_proba_train_svc_2f[:,1])

y_pred_test_poly_2f = best_svm_poly_2f.predict(x_test_2f)
y_pred_proba_test_poly_2f= best_svm_poly_2f.predict_proba(x_test_2f)
poly_score_test_2f = roc_auc_score(y_test, y_pred_proba_test_poly_2f[:,1])
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
Fitting 5 folds for each of 16 candidates, totalling 80 fits
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
```

```
910, test=0.898), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
896, test=0.954), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
914, test=0.884), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.1s remaining:    0.0s
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
905, test=0.918), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
913, test=0.881), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.910, test=0.898), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.896, test=0.954), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.914, test=0.884), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.905, test=0.918), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.913, test=0.881), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
10, test=0.898), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.8
96, test=0.954), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
14, test=0.884), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
05, test=0.918), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
13, test=0.881), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
910, test=0.898), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
896, test=0.954), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
914, test=0.884), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
905, test=0.918), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
913, test=0.881), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.91
0, test=0.898), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
```

```
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.89
6, test=0.954), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.91
4, test=0.884), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.90
5, test=0.918), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.91
3, test=0.881), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
10, test=0.898), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.8
96, test=0.954), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
14, test=0.884), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
05, test=0.918), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
13, test=0.881), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.91
0, test=0.898), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.89
6, test=0.954), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.91
4, test=0.884), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.90
5, test=0.918), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.91
3, test=0.881), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ...
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.91
0, test=0.898), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ...
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.89
6, test=0.954), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ...
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.91
4, test=0.884), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ...
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.90
5, test=0.918), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ...
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.91
3, test=0.881), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.90
3, test=0.903), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.89
1, test=0.952), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.91
0, test=0.877), total=   0.1s
```

```
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.90
1, test=0.911), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.91
1, test=0.874), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
03, test=0.903), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.8
91, test=0.952), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
10, test=0.877), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
01, test=0.911), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly ..
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.9
11, test=0.874), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
03, test=0.903), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.8
91, test=0.952), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
10, test=0.877), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
01, test=0.911), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.9
11, test=0.874), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
903, test=0.903), total=   0.0s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
891, test=0.952), total=   0.0s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
910, test=0.877), total=   0.0s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
901, test=0.911), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly .
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=0.
911, test=0.874), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
903, test=0.903), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
891, test=0.952), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
910, test=0.877), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
901, test=0.911), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=0.
```

```
911, test=0.874), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.903, test=0.903), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.891, test=0.952), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.910, test=0.877), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.901, test=0.911), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.911, test=0.874), total=   0.1s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=
0.903, test=0.903), total=   0.1s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=
0.891, test=0.952), total=   0.1s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=
0.910, test=0.877), total=   0.1s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=
0.901, test=0.911), total=   0.1s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=poly, roc_auc=(train=
0.911, test=0.874), total=   0.1s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.903, test=0.903), total=   0.1s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.891, test=0.952), total=   0.1s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.910, test=0.877), total=   0.1s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.901, test=0.911), total=   0.1s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=poly, roc_auc=(train=
0.911, test=0.874), total=   0.1s
[Parallel(n_jobs=1)]: Done  80 out of  80 | elapsed:    9.4s finished
```

In [32]:

```
#Non-Linear SVM- rbf
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, random_state=10, shuffle=True)
svc_2f = SVC(probability=True)
C = np.array([0.0001, 0.001, 0.01, 1, 10, 100, 1000, 10000])
pipe = Pipeline(steps=[('scale', StandardScaler()), ('svm', svc_2f)])
svm_rbf_2f = GridSearchCV(estimator=pipe,
             param_grid={'svm__kernel':['rbf'], 'svm__C':C, 'svm__degree':[3], 'svm__ga
svm_rbf_2f.fit(X_train_2f, Y_train)

best_svm_rbf_2f = svm_rbf_2f.best_estimator_

Y_pred_train_rbf_2f = best_svm_poly_2d.predict(X_train_2f)
Y_pred_proba_train_rbf_2f = best_svm_poly_2d.predict_proba(X_train_2f)
rbf_score_train_2f = roc_auc_score(Y_train, Y_pred_proba_train_svc_2f[:,1])
```

```
y_pred_test_rbf_2f = best_svm_rbf_2f.predict(x_test_2f)
y_pred_proba_test_rbf_2f= best_svm_rbf_2f.predict_proba(x_test_2f)
rbf_score_test_2f = roc_auc_score(y_test, y_pred_proba_test_rbf_2f[:,1])
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.9
10, test=0.898), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.1s remaining:    0.0s
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.8
96, test=0.954), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.9
14, test=0.884), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.2s remaining:    0.0s
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.9
05, test=0.918), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.9
13, test=0.881), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
910, test=0.898), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
896, test=0.954), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
914, test=0.884), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
905, test=0.918), total=   0.1s
[CV] svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=0.0001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
913, test=0.881), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.91
0, test=0.898), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.89
6, test=0.954), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.91
4, test=0.884), total=   0.2s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.90
5, test=0.918), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.91
3, test=0.881), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
10, test=0.898), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.8
96, test=0.954), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
14, test=0.884), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
```

```
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
05, test=0.918), total=   0.1s
[CV] svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=0.001, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
13, test=0.881), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.91
0, test=0.898), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.89
6, test=0.954), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.91
4, test=0.884), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.90
5, test=0.918), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.91
3, test=0.881), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.91
0, test=0.898), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.89
6, test=0.954), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.91
4, test=0.884), total=   0.2s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.90
5, test=0.918), total=   0.1s
[CV] svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=0.01, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.91
3, test=0.881), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.880,
test=0.847), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.891,
test=0.952), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.884,
test=0.833), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.871,
test=0.886), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.876,
test=0.864), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.88
0, test=0.847), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.89
1, test=0.952), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.88
4, test=0.833), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.87
1, test=0.886), total=   0.1s
[CV] svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ....
[CV]  svm__C=1.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.87
6, test=0.864), total=   0.1s
```

```
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.90
3, test=0.903), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.89
6, test=0.954), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.91
0, test=0.877), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.90
1, test=0.911), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ....
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.84
0, test=0.804), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.90
3, test=0.903), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.89
6, test=0.954), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.91
0, test=0.877), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.90
1, test=0.911), total=   0.1s
[CV] svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ...
[CV]  svm__C=10.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.84
0, test=0.804), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.90
3, test=0.903), total=   0.0s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.84
9, test=0.935), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.91
0, test=0.877), total=   0.0s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.90
1, test=0.911), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ...
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.84
0, test=0.804), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
03, test=0.903), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.8
49, test=0.935), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
10, test=0.877), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.9
01, test=0.911), total=   0.1s
[CV] svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf ..
[CV]  svm__C=100.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.8
40, test=0.804), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.9
03, test=0.903), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.8
```

```
11, test=0.918), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.9
10, test=0.877), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.8
33, test=0.830), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf ..
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.8
40, test=0.804), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
903, test=0.903), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
811, test=0.918), total=   0.0s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
910, test=0.877), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
833, test=0.830), total=   0.1s
[CV] svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf .
[CV]  svm__C=1000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=0.
840, test=0.804), total=   0.1s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.
910, test=0.898), total=   0.0s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.
815, test=0.920), total=   0.0s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.
910, test=0.877), total=   0.1s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.
901, test=0.911), total=   0.1s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf .
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=auto, svm__kernel=rbf, roc_auc=(train=0.
911, test=0.874), total=   0.1s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=
0.910, test=0.898), total=   0.1s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=
0.815, test=0.920), total=   0.1s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=
0.910, test=0.877), total=   0.1s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=
0.901, test=0.911), total=   0.1s
[CV] svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf
[CV]  svm__C=10000.0, svm__degree=3, svm__gamma=scale, svm__kernel=rbf, roc_auc=(train=
0.911, test=0.874), total=   0.1s
[Parallel(n_jobs=1)]: Done  80 out of  80 | elapsed:    9.2s finished
```

In [33]:

```
#section 7e
#evaluation metrics of dimensionality-reduced training set
# logreg_loss_2d = log_loss(y_test, y_pred_proba_test_log_2d[:,1])
# svc_loss_2d = hinge_loss(y_test, y_pred_proba_test_svc_2d[:,1])
# poly_loss_2d = hinge_loss(y_test, y_pred_proba_test_poly_2d[:,1])
# rbf_loss_2d = hinge_loss(y_test, y_pred_proba_test_rbf_2d[:,1])
```

In [34]:
```
F1_logreg_train_2f, Acc_logreg_train_2f = calc_metrics(Y_train, Y_pred_train_log_2f)
F1_svc_train_2f, Acc_svc_train_2f = calc_metrics(Y_train, Y_pred_train_svc_2f)
F1_poly_train_2f, Acc_poly_train_2f = calc_metrics(Y_train, Y_pred_train_poly_2f)
F1_rbf_train_2f, Acc_rbf_train_2f = calc_metrics(Y_train, Y_pred_train_rbf_2f)

F1_logreg_test_2f, Acc_logreg_test_2f = calc_metrics(y_test, y_pred_test_log_2f)
F1_svc_test_2f, Acc_svc_test_2f = calc_metrics(y_test, y_pred_test_svc_2f)
F1_poly_test_2f, Acc_poly_test_2f = calc_metrics(y_test, y_pred_test_poly_2f)
F1_rbf_test_2f, Acc_rbf_test_2f = calc_metrics(y_test,y_pred_test_rbf_2f)
```

In [35]:
```
print("\nLogistic Regression:")
print('\nFor 2 features:')
print('\nFor Training Set: \nAUC is {:.2f}. \nF1 is {:.2f}.  \nACC is {:.2f}. '.format(
print('\nFor Test Set: \nAUC is {:.2f}. \nF1 is {:.2f}. \nACC is {:.2f}.\n '.format(log

print("\nLinear SVM:")
print('\nFor 2 features:')
print('\nFor Training Set: \nAUC is {:.2f}. \nF1 is {:.2f}.  \nACC is {:.2f}. '.format(
print('\nFor Test Set: \nAUC is {:.2f}. \nF1 is {:.2f}. \nACC is {:.2f}.\n '.format(LSV

print("\nPolynomial SVM:")
print('\nFor 2 features:')
print('\nFor Training Set: \nAUC is {:.2f}. \nF1 is {:.2f}.  \nACC is {:.2f}. '.format(
print('\nFor Test Set: \nAUC is {:.2f}. \nF1 is {:.2f}. \nACC is {:.2f}.\n '.format(pol

print("\nSVM RBF:")
print('\nFor 2 features:')
print('\nFor Training Set: \nAUC is {:.2f}. \nF1 is {:.2f}.  \nACC is {:.2f}. '.format(
print('\nFor Test Set: \nAUC is {:.2f}. \nF1 is {:.2f}. \nACC is {:.2f}.\n '.format(rbf
```

```
Logistic Regression:

For 2 features:

For Training Set:
AUC is 0.09.
F1 is 0.17.
ACC is 0.13.

For Test Set:
AUC is 0.87.
F1 is 0.76.
ACC is 0.61.


Linear SVM:

For 2 features:

For Training Set:
AUC is 0.09.
F1 is 0.17.
ACC is 0.13.

For Test Set:
AUC is 0.87.
F1 is 0.76.
ACC is 0.61.
```

```
Polynomial SVM:

For 2 features:

For Training Set:
AUC is 0.09.
F1 is 0.17.
ACC is 0.13.

For Test Set:
AUC is 0.87.
F1 is 0.76.
ACC is 0.61.


SVM RBF:

For 2 features:

For Training Set:
AUC is 0.09.
F1 is 0.17.
ACC is 0.13.

For Test Set:
AUC is 0.87.
F1 is 0.76.
ACC is 0.61.
```

In [36]:

```
#evaluation metrics of 2-features training set
#logreg_loss_2f = log_loss(y_test, y_pred_proba_test_log_2f[:,1])
#svc_loss_2f = hinge_loss(y_test, y_pred_proba_test_svc_2f[:,1])
#poly_loss_2f = hinge_loss(y_test, y_pred_proba_test_poly_2f[:,1])
#rbf_loss_2f = hinge_loss(y_test, y_pred_proba_test_rbf_2f[:,1])
```

Question 7, E

What performs better? 2 features of the reduced dimensionality?

Comparing the 2 features and the dimensionality reduced methods, the dimensionality-reduced training set method gives us better results of all evaluation metrics in all training models. The PCA function reduces dimentionality in a way that causes minimal loss of information unlike the 2 features method in which we manually choose two features with the highest weights (according to random forest weight calculation).
Among the training models, the training model that performe the best is non linear SVM - Poly kernel.

In [37]:

```
#evaluation metrics of 2-features training set
# logreg_loss_2f = log_loss(y_test, y_pred_proba_test_log_2f[:,1])
# svc_loss_2f = hinge_loss(y_test, y_pred_proba_test_svc_2f[:,1])
# poly_loss_2f = hinge_loss(y_test, y_pred_proba_test_poly_2f[:,1])
# rbf_loss_2f = hinge_loss(y_test, y_pred_proba_test_rbf_2f[:,1])
```

That's all folks!

In [ ]: