```
from sklearn.datasets import make_classification
from matplotlib import pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.metrics import r2_score, explained_variance_score, confusion_matrix, accuracy_score, classificat
from math import sqrt
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import Lasso
from sklearn import svm
from sklearn.metrics import plot_confusion_matrix
from sklearn.model_selection import GridSearchCV # this will do cross validation
from sklearn.decomposition import PCA
import matplotlib.colors as colors
from sklearn.metrics import roc_auc_score, roc_curve, auc
from sklearn.metrics import precision_recall_curve, average_precision_score
from sklearn.svm import SVC
```

```
df = pd.read_csv("CellDNA.csv", header= None)
```

```
df.columns =['x0', 'x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7','x8','x9','x10','x11','x12','x13']
```

```
df.loc[:, 'x13'] = np.where(df.x13>0, 1, 0)
```

```
df
```

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11
0	222	31.189189	40.342342	35.579087	8.883917	0.968325	-80.113673	222	1	16.812471	0.816176	0.578125
1	73	29.493151	271.397260	15.517202	6.407490	0.910764	76.042946	73	1	9.640876	0.858824	0.608333
2	256	58.816406	289.941406	37.226013	9.863895	0.964256	85.324742	256	1	18.054067	0.752941	0.562637
3	126	71.023810	477.412698	13.112980	12.790672	0.220351	63.523477	126	1	12.666025	0.881119	0.646154
4	225	90.808889	541.946667	44.463110	7.858879	0.984256	-52.874983	225	1	16.925688	0.728155	0.252525
...	...	...	...	...	...	...	...	...	...	...	...	...
1212	216	738.527778	216.449074	38.229761	9.556174	0.968254	12.847813	216	1	16.583719	0.640950	0.397059
1213	328	748.896341	47.664634	63.138991	9.101974	0.989555	57.919494	328	1	20.435816	0.607407	0.205257
1214	97	761.690722	207.288660	22.751513	8.230351	0.932275	-24.674618	97	1	11.113246	0.591463	0.384921
1215	223	770.654708	235.502242	53.491654	8.643053	0.986860	73.244715	223	1	16.850294	0.557500	0.252834
1216	87	764.954023	265.655172	13.459738	8.521929	0.774035	18.595633	87	1	10.524820	0.956044	0.743590

1217 rows × 14 columns

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df[['x0', 'x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7','x8','x9','x10','x11','x12']] = scaler.fit_transform(df[['
```

```
numeric_cols = ['x0', 'x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7','x8','x9','x10','x11','x12']
scaler = StandardScaler()
scaler.fit(df[numeric_cols])
scaled_inputs = scaler.transform(df[numeric_cols])
scaled_inputs
```

```
array([[ 0.15952762, -1.80200559, -1.20813407, ...,  0.34511514,
         0.65289142, -0.00691284],
       [-0.93921222, -1.80987674,  0.42436331, ...,  0.7072868 ,
         0.84374979, -0.81411281],
       [ 0.41024678, -1.67379037,  0.55538528, ..., -0.19189804,
         0.55503945,  0.20875597],
       ...,
       [-0.76223399,  1.58818067, -0.02859014, ..., -1.56321582,
        -0.56778731, -0.23578419],
       [ 0.16690172,  1.62978166,  0.17075035, ..., -1.85164337,
        -1.40231699,  0.69144818],
       [-0.83597492,  1.60332534,  0.38379311, ...,  1.53291195,
         1.69830929, -0.95601961]])
```

```
df
```

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x1
0	0.159528	-1.802006	-1.208134	0.114420	-0.135689	0.538311	-1.587426	0.135833	0.233292	0.329626	0.34511
1	-0.939212	-1.809877	0.424363	-0.933511	-0.817247	0.019258	1.500586	-0.909580	0.233292	-1.221986	0.70728
2	0.410247	-1.673790	0.555385	0.200447	0.134019	0.501621	1.684134	0.374384	0.233292	0.598252	-0.19189
3	-0.548385	-1.617137	1.879947	-1.059096	0.939523	-6.206504	1.253012	-0.537722	0.233292	-0.567479	0.89662
4	0.181650	-1.525316	2.335905	0.578476	-0.417798	0.681969	-1.048779	0.156881	0.233292	0.354121	-0.40238
...	...	...	...	...	...	...	...	...	...	...	.
1212	0.115283	1.480684	0.036132	0.252878	0.049329	0.537678	0.250896	0.093736	0.233292	0.280134	-1.14296
1213	0.941181	1.528803	-1.156399	1.554010	-0.075675	0.729753	1.142193	0.879550	0.233292	1.113556	-1.42781
1214	-0.762234	1.588181	-0.028590	-0.555628	-0.315562	0.213238	-0.491114	-0.741192	0.233292	-0.903431	-1.56321
1215	0.166902	1.629782	0.170750	1.050082	-0.201979	0.705453	1.445251	0.142849	0.233292	0.337809	-1.85164
1216	-0.835975	1.603325	0.383793	-1.040983	-0.235315	-1.213681	0.364560	-0.811354	0.233292	-1.030740	1.53291

1217 rows × 14 columns

```
x = df.drop('x13', axis = 1).values
y = df['x13']
y = y.astype(int)

print(x.shape)
print(y.shape)
```

```
(1217, 13)
(1217,)
```

check distribution of target_class column

```
df['x13'].value_counts()
```

```
0    1017
1     200
Name: x13, dtype: int64
```

```
df['x13'].value_counts()/np.float(len(df))
```

```
0    0.835661
1    0.164339
Name: x13, dtype: float64
```

```
round(df.describe(),2)
```

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x1
count	1217.00	1217.00	1217.00	1217.00	1217.00	1217.00	1217.00	1217.00	1217.00	1217.00	1217.00	1217.00	1217.0
mean	0.00	0.00	-0.00	0.00	-0.00	0.00	0.00	0.00	-0.00	0.00	-0.00	0.00	-0.0
std	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.0
min	-0.96	-1.82	-1.30	-1.21	-1.73	-6.23	-1.77	-0.93	-12.80	-1.27	-4.37	-2.28	-1.0
25%	-0.69	-0.82	-0.81	-0.76	-0.51	-0.21	-0.82	-0.66	0.23	-0.78	-0.62	-0.77	-0.6
50%	-0.30	-0.02	-0.19	-0.29	-0.25	0.37	-0.01	-0.29	0.23	-0.22	0.12	-0.00	-0.2
75%	0.38	0.89	0.58	0.58	0.12	0.63	0.81	0.35	0.23	0.57	0.77	0.74	0.3
max	6.55	1.64	2.57	6.37	9.39	0.81	1.77	6.71	0.23	4.75	1.91	2.72	8.5

```
df.dtypes
```

```
x0     float64
x1     float64
x2     float64
x3     float64
x4     float64
x5     float64
x6     float64
x7     float64
x8     float64
x9     float64
x10    float64
x11    float64
x12    float64
x13      int64
dtype: object
```

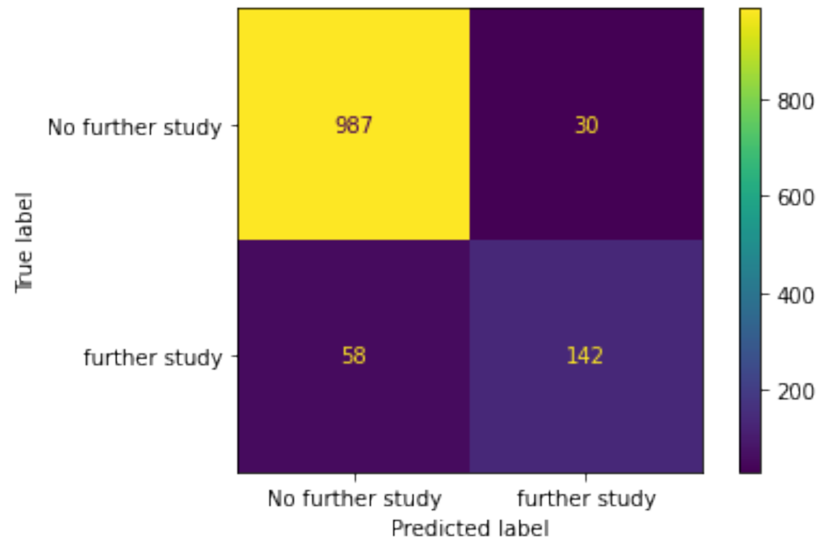
%CREATE SVM MODEL classifier ML ALGORITHM

```
clf_svm = SVC(random_state=42)
clf_svm.fit(x,y)
```

```
SVC(random_state=42)
```

```
plot_confusion_matrix(clf_svm,
                      x,
                      y,
                      display_labels=['No further study', 'further study'])
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_con
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f4970918f90>
```



In the confusion matrix, we see that of the 987 + 58 = 1045 that did not interesting in further study, 987 were correctly classified. And of the 30 + 142 = 172 that have interesting in further study, 142 were correctly classified. So the support vector machine did pretty well without any optimization. That said, it is possible that we can improve predictions using Cross Validation to optimize the parameters.

```
param_grid = [
              {'C':[1,10,100,1000],
               'gamma':[0.001,0.0001],
               'kernel':['rbf']}
]

optimal_params = GridSearchCV(
    SVC(),
    param_grid,
    cv=5,
    verbose=1
)
optimal_params.fit(x, y)
optimal_params.best_params_
```

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
{'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
```

% Finding the best value of gamma, regularization parameter "C" (box constraint) and rbf kernel scale to improve the accuracy with the dataset.

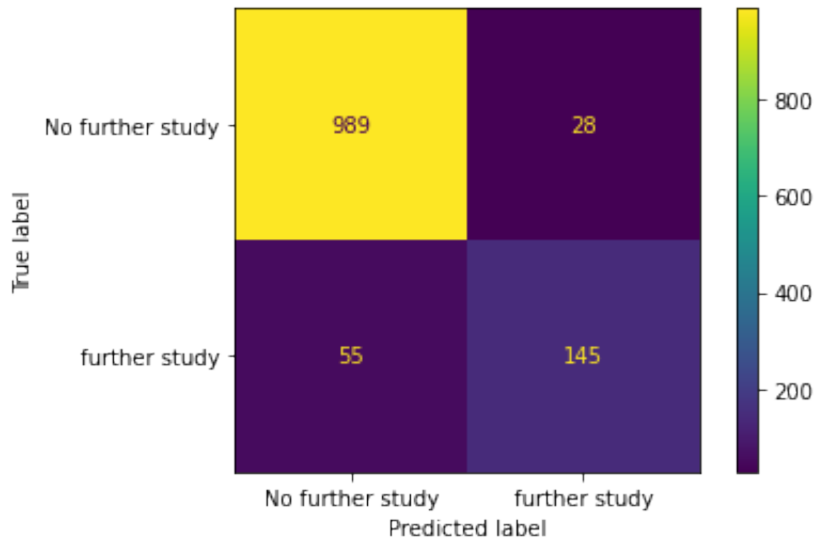
% ATTEMPT 1 C=1, GAMMA =0.1

```
clf_svm = SVC(random_state=42, C=1,·gamma=0.1,kernel='rbf')
clf_svm.fit(x, y)
```

```
SVC(C=1, gamma=0.1, random_state=42)
```

```
plot_confusion_matrix(clf_svm,
                      x,
                      y,
                      display_labels=['No further study', 'further study'])
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_con
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f496f95d710>
```



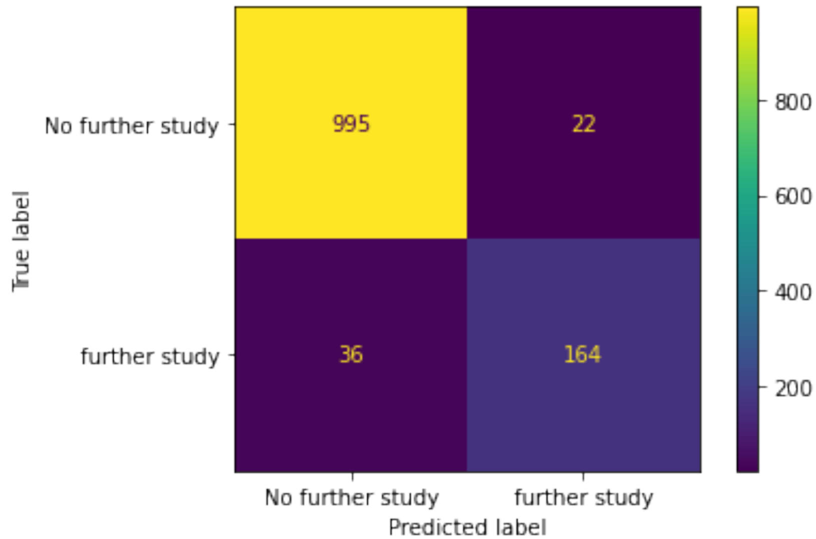
% I see that the optimized Support Vector Machine is better at classifying Bacteria for the further study than the preliminary support vector machine.

```
clf_svm = SVC(random_state=42, C=10,·gamma=0.1,kernel='rbf')
clf_svm.fit(x, y)
```

```
SVC(C=10, gamma=0.1, random_state=42)
```

```
plot_confusion_matrix(clf_svm,
                      x,
                      y,
                      display_labels=['No further study', 'further study'])
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_con
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f496f87fdd0>
```

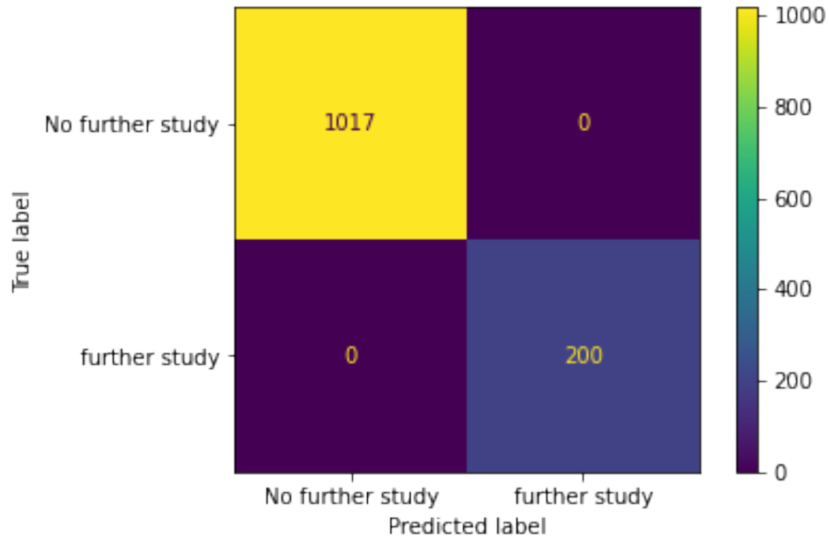


% In progress showing better at classifying Bacteria for the further study than second attempt.

```
clf_svm = SVC(random_state=42, C=10, gamma=1,kernel='rbf')
clf_svm.fit(x, y)
```

```
SVC(C=10, gamma=1, random_state=42)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_con
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f496ad1ea90>
```




```
# instantiate classifier with rbf kernel and C=10
clf= svm.SVC(kernel = 'rbf', C = 10,gamma=1,probability=True)
# fit classifier to predictors & target set
clf.fit(x,y)
# of support vectors in EACH class
print(clf.n_support_)
```

```
# indices of support vectors
print(clf.support_)
# coefficients in ”dual” form
print(clf.dual_coef_)
# make predictions on predictors set
y_pred=clf.predict(x)
yhat = clf.predict_proba(x)
# compute and print accuracy score
print('Model accuracy score with rbf kernel and C=10 : {0:0.4f}'. format(accuracy_score(y, y_pred)))
```

```
  888  891  894  902  914  923  929  933  934  936  939  943  944  946
  948  965  966  978  979  985  991  993  995  996 1008 1018 1023 1030
 1067 1078 1109 1140 1147 1162 1170 1171 1174 1176 1178 1185 1193 1199
 1213]
[[-1.71061313e-01 -1.75252453e-01 -3.49804408e-01 -1.12415944e+00
  -7.28040338e-03 -3.68411558e-01 -1.35633696e-01 -1.84132799e-01
  -3.05007226e-01 -3.04975058e-01 -3.64347551e-01 -1.19197566e+00
  -3.58827528e-01 -7.33790376e-01 -3.04479557e-01 -7.68783401e-02
  -3.37054437e-01 -3.46295899e-01 -1.01848911e-01 -8.51483144e-02
  -3.71455890e-01 -2.67171223e-01 -3.62373990e-01 -1.15957600e-01
  -5.11323515e-02 -3.17869014e-01 -1.92903743e-01 -2.73601773e-01
  -1.61126863e-01 -5.58183328e-01 -2.53416469e-01 -1.76667703e-01
  -2.72054713e-01 -3.59035946e-01 -6.39951862e-01 -2.77331746e-01
  -2.05885705e-01 -1.36405707e-01 -3.41565597e-01 -2.36311388e-01
  -1.52410736e-01 -1.56690961e-01 -2.90288380e-01 -1.46758015e-01
  -2.40131137e-01 -3.73623102e-01 -1.97104561e-01 -8.54584614e-01
  -5.48924030e-01 -2.61648956e-01 -1.78378942e-01 -1.15405917e+00
  -3.09036991e-01 -1.18013402e-01 -3.97903348e-02 -4.39758816e-01
  -2.53554838e-01 -1.24898983e-01 -9.64155358e-02 -3.70362516e-02
  -2.01738504e-01 -2.67285373e-01 -3.09204321e-01 -3.62999258e-01
  -9.67071275e-02 -3.63027756e-01 -1.16999928e-01 -5.01598245e-02
  -1.55765364e-01 -7.76608178e-02 -5.15774589e-02 -6.11940708e-01
  -5.77026825e-02 -6.22630360e-02 -1.47833797e-01 -1.92433209e-01
  -8.35837983e-02 -3.37019158e-02 -1.40725492e-01 -3.04081526e-01
  -2.57477305e-01 -5.51762513e-01 -1.76616125e-01 -7.92371600e-01
```

```
-3.63016109e-01 -2.78154915e-01 -7.05028433e-02 -4.07681509e-02
-5.76140828e-01 -2.80078402e-01 -1.15728870e+00 -1.23323670e-01
-2.07849141e-01 -3.63161903e-01 -4.35115971e-01 -3.07811675e-01
-1.87907883e-01 -1.95376051e-02 -3.54573219e-01 -3.12599013e-01
-3.62873442e-01 -4.58736626e-01 -3.54515226e-01 -3.38509969e-01
-3.59733946e-01 -2.06061782e-01 -2.10420462e+00 -2.63659534e-01
-3.42318774e-01 -5.60160137e-02 -5.54349107e-01 -4.76800771e-02
-3.46581972e-01 -1.50362225e+00 -5.43572849e-01 -1.36229717e-01
-3.62658091e-01 -1.40918396e-01 -5.80742863e-02 -7.41321988e-02
-4.78198496e+00 -9.26738055e-01 -1.35461318e-01 -1.89288647e-01
-5.72839540e-02 -3.16986818e-02 -3.87680638e-01 -3.60842400e-01
-2.51351612e-01 -2.82616474e-02 -1.32264606e-01 -2.38334523e-02
-2.92108300e-01 -2.43665009e-01 -5.06561841e-02 -2.34203887e-02
-9.04964217e-02 -4.31561307e-02 -4.55249058e-01 -3.36831331e-01
-2.55697612e-01 -1.60367227e-01 -1.00302226e-01 -4.72966114e-02
-1.35381012e-01 -3.62919831e-01 -1.92983244e-01 -3.55491746e-01
-2.76592376e-01 -1.26421633e+00 -1.50097906e-01 -5.75415830e-02
-2.06723371e+00 -5.18190390e-01 -3.04647143e-01 -7.09939207e+00
-1.09228558e-01 -2.52228964e-01 -2.62344364e-01 -1.21553340e-01
-3.29933608e-01 -3.58534177e-01 -1.98425036e-01 -6.87383768e+00
-3.65735317e-01 -4.28751681e+00 -3.60877268e-01 -1.92936669e-01
-1.57579265e-01 -2.06427516e-01 -3.63153231e-01 -1.31097557e+00
-1.28365613e-01 -3.63165050e-01 -3.63105101e-01 -1.64153035e+00
-3.09444429e-01 -1.48896972e+00 -3.62886930e-01 -4.24303992e-01
-3.48023392e+00 -1.85704339e-01 -9.02967425e-01 -3.61131239e-01
-1.24204927e-02 -7.68475560e-01 -5.33948352e+00 -1.49481784e-01
-1.89301330e-01 -2.89895527e-01 -1.46012381e+00 -4.87973626e-02
-1.67744473e-01 -2.90294364e-01 -5.54505856e-01 -2.87156652e-01
-7.25519496e-02 -3.62085712e-01 -3.40878239e-01 -2.04576203e-01
-3.68015741e-02 -3.04843622e-01 -2.25003629e-01 -2.58616713e-01
-8.76118618e-02 -3.80494712e-01 -7.50285421e-01 -1.52669847e-01
-1.56795697e-01 -6.32323639e-02 -1.21528828e-01 -2.40749377e-01
-2.92271348e-01 -3.60068837e-01 -1.74872771e+00 -2.01277317e-03
-2.54109944e-01 -3.56947797e-01 -2.97727542e-01 -3.42778528e-01
-3.30117506e-01 -3.52492648e-01 -3.47491758e-01 -5.46108794e-01
```

```
print(y.shape, y.size)
print(y_pred)
print(yhat)
```

```
(1217,) 1217
[0 0 0 ... 0 0 0]
[[0.95674689 0.04325311]
 [0.95673417 0.04326583]
 [0.97701012 0.02298988]
 ...
 [0.95680701 0.04319299]
 [0.95679213 0.04320787]
 [0.95676655 0.04323345]]
```

```
# compute ROC AUC

from sklearn.metrics import roc_auc_score

ROC_AUC = roc_auc_score(y, y_pred)

print('ROC AUC : {:.4f}'.format(ROC_AUC))
```

```
ROC AUC : 1.0000
```

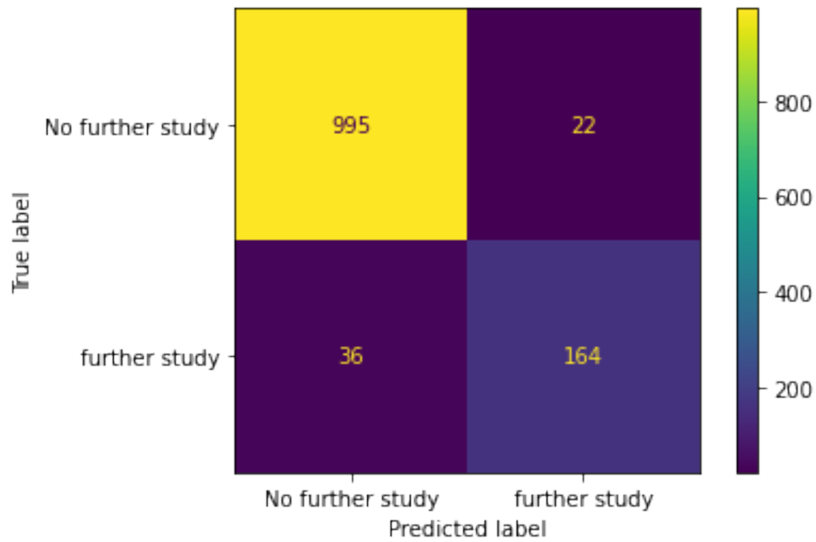
% EXPERIMENT FOR ROC CURVE PLOT * EACH CLASS

```
clf_svm = SVC(random_state=42, C=10, gamma=0.1,kernel='rbf')
clf_svm.fit(x,y)
```

```
SVC(C=10, gamma=0.1, random_state=42)
```

```
plot_confusion_matrix(clf_svm,
                      x,
                      y,
                      display_labels=['No further study', 'further study'])
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_con
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f496a6c9790>
```




```
# instantiate classifier with rbf kernel and C=10
clf= svm.SVC(probability=True, C=10, gamma=0.1,kernel='rbf')
# fit classifier to predictors & target set
clf.fit(x,y)
# of support vectors in EACH class
print(clf.n_support_)
# indices of support vectors
print(clf.support_)
```

```
# coefficients in "dual" form
print(clf.dual_coef_)
# make predictions on predictors set
y_pred=clf.predict(x)
yhat = clf.predict_proba(x)
# compute and print accuracy score
print('Model accuracy score with rbf kernel and C=10 : {0:0.4f}'. format(accuracy_score(y, y_pred)))
```

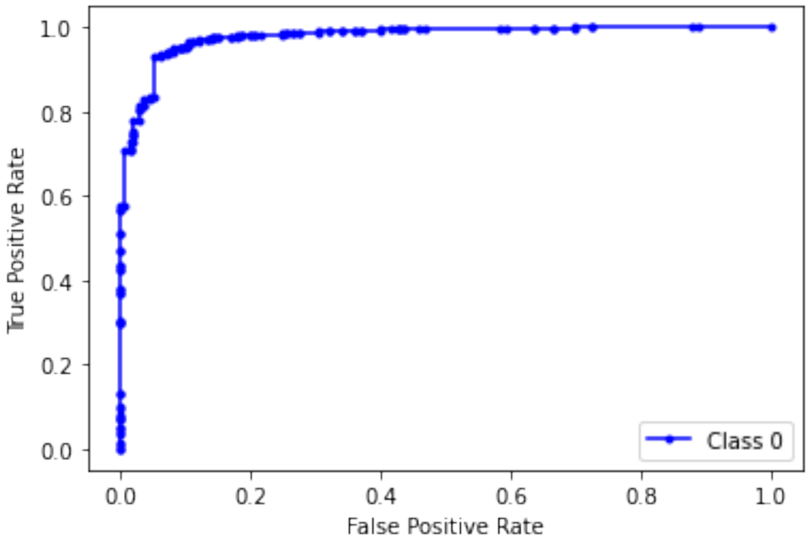
```
[163 117]
[   3   12   28   29   51   52   53   68   70   76   77   86  100  102
  108  130  135  142  143  166  176  188  193  261  265  271  274  277
  278  291  299  300  303  304  305  309  312  313  318  325  335  342
  361  363  382  389  405  416  471  485  488  492  514  515  527  528
  530  532  533  545  562  563  564  570  575  576  580  586  589  590
  601  606  608  609  611  621  672  682  694  706  729  743  744  754
  818  831  832  852  872  885  892  907  921  927  941  945  964  971
  972  987  994 1000 1011 1013 1022 1024 1027 1028 1029 1033 1034 1040
 1044 1049 1051 1056 1057 1063 1064 1066 1070 1072 1074 1077 1081 1083
 1085 1090 1091 1093 1097 1102 1106 1108 1112 1113 1114 1117 1124 1125
 1126 1128 1129 1130 1132 1133 1134 1136 1138 1141 1142 1143 1144 1157
 1173 1179 1184 1187 1188 1190 1203 1207 1215    4   17   31   35   43
   56   74   98  123  124  144  163  164  167  183  203  214  219  220
  221  225  228  229  234  245  250  251  252  256  259  260  289  297
  321  322  326  328  329  332  333  338  339  341  373  375  380  417
  422  432  433  438  439  457  462  465  489  491  495  497  498  519
  521  525  550  578  583  585  597  604  610  617  623  624  632  657
  673  699  702  708  723  725  728  735  780  784  790  791  796  797
  799  815  826  834  836  849  875  884  888  914  923  929  933  936
  939  943  944  946  948  965  978  985  991  996 1008 1140 1170 1213]
[[-1.79480910e-01 -1.67058782e+00 -5.86352838e-01 -4.60451405e+00
  -1.97344524e+00 -1.00000000e+01 -1.00000000e+01 -8.75286066e-01
  -1.10917434e+00 -1.00000000e+01 -1.00000000e+01 -1.56187454e+00
  -1.68009125e-01 -6.07741802e-02 -8.73271731e-01 -2.46044258e-01
  -3.60293216e+00 -2.16136516e-01 -4.21637461e+00 -1.00000000e+01
  -4.69925244e+00 -3.30998084e+00 -1.40703110e+00 -2.94642017e+00
```

```
 -1.38535467e-01 -1.00000000e+01 -1.00000000e+01 -4.00951016e+00
 -1.00000000e+01 -1.00000000e+01 -3.52624632e+00 -1.74579969e-01
 -3.12057391e-01 -4.40278829e-01 -1.00000000e+01 -1.00000000e+01
 -1.00000000e+01 -1.00000000e+01 -1.07617296e+00 -1.00000000e+01
 -1.00000000e+01 -5.63894994e+00 -2.12078422e+00 -1.00000000e+01
 -5.89620929e+00 -3.59114829e-01 -1.00000000e+01 -7.13783699e+00
 -3.07543008e+00 -5.60594765e+00 -1.76049357e-01 -1.78359869e-01
 -2.86585271e+00 -1.00000000e+01 -8.39312659e-01 -1.00000000e+01
 -4.87684540e-01 -1.00000000e+01 -8.41332877e+00 -1.00000000e+01
 -1.37018429e-01 -1.00000000e+01 -1.00000000e+01 -6.47412341e+00
 -1.00000000e+01 -1.00000000e+01 -5.56861640e+00 -1.00000000e+01
 -4.78405850e-02 -1.77474588e-02 -1.00000000e+01 -1.00000000e+01
 -1.00000000e+01 -1.00000000e+01 -1.00000000e+01 -1.00000000e+01
 -2.34327443e-01 -5.35620294e-02 -1.82399283e+00 -2.16539829e-01
 -1.00000000e+01 -1.03904564e+00 -1.00000000e+01 -3.17593046e-03
 -1.00000000e+01 -6.86292790e-01 -1.86815779e+00 -2.25897519e+00
 -1.00000000e+01 -1.00000000e+01 -2.03917803e-01 -5.98655400e+00
 -3.78470237e-02 -8.00292421e+00 -1.64021460e-02 -1.30991573e-01
 -3.86477444e+00 -1.00000000e+01 -1.00000000e+01 -4.18035651e+00
 -4.26139050e+00 -2.97336556e-01 -1.00000000e+01 -2.77075011e-01
 -4.22762275e-01 -4.40699480e+00 -1.66701699e+00 -9.64263249e+00
 -1.00000000e+01 -4.75173184e+00 -7.57523491e-02 -1.00000000e+01
 -1.00000000e+01 -1.00000000e+01 -1.00000000e+01 -2.50185031e-01
 -1.89369242e-01 -1.00000000e+01 -3.43710760e+00 -2.54370412e+00
 -1.00000000e+01 -1.00000000e+01 -1.00000000e+01 -1.00000000e+01
 -1.00000000e+01 -1.00000000e+01 -1.00000000e+01 -1.00000000e+01
 -1.91014031e-01 -1.00000000e+01 -6.78213119e+00 -3.08394987e+00
 -1.00000000e+01 -1.00000000e+01 -1.00000000e+01 -1.00000000e+01
 -1.00000000e+01 -1.00000000e+01 -1.00000000e+01 -1.00000000e+01
 -1.00000000e+01 -5.82088010e+00 -1.39196585e+00 -1.83870276e+00
 -1.00000000e+01 -1.00000000e+01 -1.00000000e+01 -1.00000000e+01
 -1.00000000e+01 -6.61302062e+00 -1.00000000e+01 -1.00000000e+01
 -1.00000000e+01 -1.00000000e+01 -1.00000000e+01 -1.00000000e+01
```

```
# Compute ROC curve and ROC area for each class
from sklearn.metrics import roc_curve, roc_auc_score, precision_recall_curve, confusion_matrix, auc, accuracy

# class 0
fpr_0, tpr_0, _ = roc_curve(y, yhat[:, 0],pos_label=0)
roc_auc_0 = roc_auc_score(y, yhat[:, 0])
```

```
# plot ROC curves
print('roc_auc_0: ', roc_auc_0)

plt.plot(fpr_0, tpr_0, marker='.', label='Class 0', color='b')

plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```

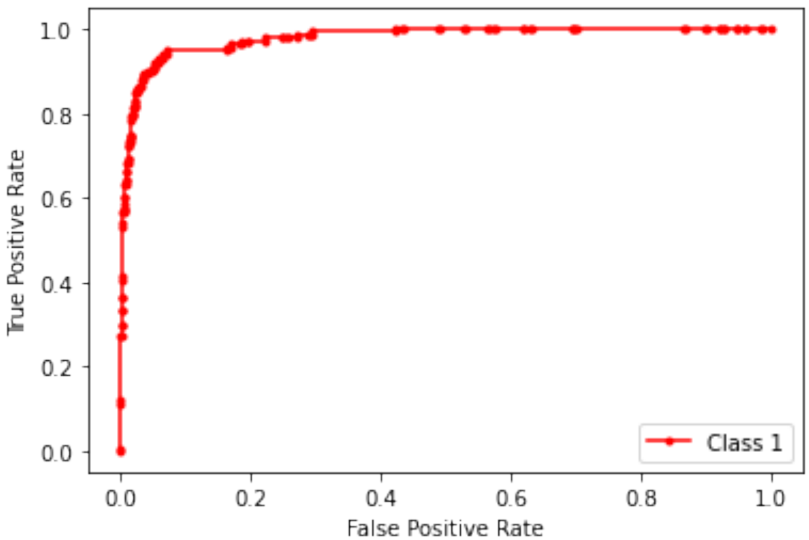
```
roc_auc_0:  0.021283185840707968
```



```
# class 1
fpr_1, tpr_1, _ = roc_curve(y, yhat[:, 1])
roc_auc_1 = roc_auc_score(y, yhat[:, 1])
print('roc_auc_1: ', roc_auc_1, '\n')
```

```
roc_auc_1:  0.9787168141592921
```

```
plt.plot(fpr_1, tpr_1, marker='.', label='Class 1', color='r')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```



0s completed at 2:44 PM