IP_SVM

December 13, 2023

1 SVM : Support vector machines

Theoretical Understanding ref:

1. https://www.youtube.com/watch?v=H9yACitf-KM
2. https://www.youtube.com/watch?v=Js3GLb1xPhc
3. https://www.youtube.com/watch?v=FB5EdxAGxQg

2 what is support vector machines?

```
SVM is a machine learning algorithm used mostly in classfication,because it can give good outp
```

3 how it works?

```
Like logistic regression we will try to seperate the classes by boundary its called hyperplane
marginal planes for create more generalized model by using support vectors.
```

4 what is hyperplane?

```
hyperplane is boundary space used to seperate the classes,in 2d we can say its a simple line
```

[ ]:

5 what is marginal plane?

```
marginal plane is a parallel line to the hyperplane,it created by the lowest data points belong
that data points supports marginal planes called support vectors
```

6 what is marginal distance?

```
The distance b/w margin plane to hyperplan is called marginal distance.
```

7 Types of margins?

```
1. Hard margin
2. soft margin
```

Hard margin: 1. hard margins used to create a hyper plane perfectly without any misclassification. 2. it may give overfitting

Soft margin: 1. used to create a hyper plane with some missclassification 2. it will reduce overfitting

8 Objective of SVM:

1. we have to find the hyperplane which can sepeated the classes which has the maximum marginal distance.
2. so ,we can get more generalized model

9 what will be happen when data are linearly seperable?

we can easily seperate the classes easily when the data points are linear,

10 what will we do when data are in non linear relationship?

here we have to use kernels,

11 what is kernel?

12 kernels used to transform our data from lower dimension to higer dimension by applying formula

13 Types of kernel?

we have more no.of kernels,importants are 1. Polynomial kernels 2. RBF 3. sigmoid

14 1. What Are the Basic Assumption?

There are no such assumptions

2. Advantages

1. SVM is more effective in high dimensional spaces.
2. SVM is relatively memory efficient.
3. SVM's are very good when we have no idea on the data.
4. Works well with even unstructured and semi structured data like text, Images and trees.
5. The kernel trick is real strength of SVM. With an appropriate kernel function, we can solve any complex problem.
6. SVM models have generalization in practice, the risk of over-fitting is less in SVM.

3. Disadvantages

1. More Training Time is required for larger dataset
2. It is difficult to choose a good kernel function https://www.youtube.com/watch?v=mTyT-oHoivA

3. The SVM hyper parameters are Cost -C and gamma. It is not that easy to fine-tune these hyper-parameters. It is hard to visualize their impact

4. Whether Feature Scaling is required? Yes ##### 5. Impact of Missing Values? Although SVMs are an attractive option when constructing a classifier, SVMs do not easily accommodate missing covariate information. Similar to other prediction and classification methods, in-attention to missing data when constructing an SVM can impact the accuracy and utility of the resulting classifier. ##### 6. Impact of outliers? It is usually sensitive to outliers https://arxiv.org/abs/1409.0934#:~:text=Despite%20its%20popularity%2C%20SVM%20has,causes%20the%20se

[ ]:

Types of Problems it can solve(Supervised)

1. Classification
2. Regression

Overfitting And Underfitting In SVM, to avoid overfitting, we choose a Soft Margin, instead of a Hard one i.e. we let some data points enter our margin intentionally (but we still penalize it) so that our classifier don't overfit on our training sample

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[ ]:

Different Problem statement you can solve using Naive Baye's

1. We can use SVM with every ANN usecases
2. Intrusion Detection
3. Handwriting Recognition

[ ]:

Practical Implementation

1. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

Classification

1. Confusion Matrix
2. Precision,Recall, F1 score

Regression

1. R2,Adjusted R2
2. MSE,RMSE,MAE

15 practical implementation SVC

[15]:
```
import sklearn
from sklearn.datasets import load_breast_cancer
import pandas as pd
```

[16]:
```
load_breast_cancer()
```

[16]:
```
{'data': array([[1.799e+01, 1.038e+01, 1.228e+02, …, 2.654e-01, 4.601e-01,
         1.189e-01],
        [2.057e+01, 1.777e+01, 1.329e+02, …, 1.860e-01, 2.750e-01,
         8.902e-02],
        [1.969e+01, 2.125e+01, 1.300e+02, …, 2.430e-01, 3.613e-01,
         8.758e-02],
        …,
        [1.660e+01, 2.808e+01, 1.083e+02, …, 1.418e-01, 2.218e-01,
         7.820e-02],
        [2.060e+01, 2.933e+01, 1.401e+02, …, 2.650e-01, 4.087e-01,
         1.240e-01],
        [7.760e+00, 2.454e+01, 4.792e+01, …, 0.000e+00, 2.871e-01,
         7.039e-02]]),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
1,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
        1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
        1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
        0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
        1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
        0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
        1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
        0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
        1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
        1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
        1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
        1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
        1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

```
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1]),
 'frame': None,
 'target_names': array(['malignant', 'benign'], dtype='<U9'),
 'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer wisconsin (diagnostic)
dataset\n--------------------------------------------\n\n**Data Set
Characteristics:**\n\n    :Number of Instances: 569\n\n    :Number of
Attributes: 30 numeric, predictive attributes and the class\n\n    :Attribute
Information:\n        - radius (mean of distances from center to points on the
perimeter)\n        - texture (standard deviation of gray-scale values)\n
- perimeter\n        - area\n        - smoothness (local variation in radius
lengths)\n        - compactness (perimeter^2 / area - 1.0)\n        - concavity
(severity of concave portions of the contour)\n        - concave points (number
of concave portions of the contour)\n        - symmetry\n        - fractal
dimension ("coastline approximation" - 1)\n\n        The mean, standard error,
and "worst" or largest (mean of the three\n        worst/largest values) of
these features were computed for each image,\n        resulting in 30 features.
For instance, field 0 is Mean Radius, field\n        10 is Radius SE, field 20
is Worst Radius.\n\n        - class:\n                - WDBC-Malignant\n
- WDBC-Benign\n\n    :Summary Statistics:\n\n
===================================== ====== ======\n
Min    Max\n    ===================================== ====== ======\n    radius
(mean):                         6.981  28.11\n    texture (mean):
9.71   39.28\n    perimeter (mean):                       43.79  188.5\n    area
(mean):                           143.5  2501.0\n    smoothness (mean):
0.053  0.163\n    compactness (mean):                     0.019  0.345\n
concavity (mean):                     0.0    0.427\n    concave points (mean):
0.0    0.201\n    symmetry (mean):                       0.106  0.304\n
fractal dimension (mean):             0.05   0.097\n    radius (standard error):
0.112  2.873\n    texture (standard error):              0.36   4.885\n
perimeter (standard error):           0.757  21.98\n    area (standard error):
6.802  542.2\n    smoothness (standard error):           0.002  0.031\n
compactness (standard error):         0.002  0.135\n    concavity (standard
error):           0.0    0.396\n    concave points (standard error):      0.0
0.053\n    symmetry (standard error):            0.008  0.079\n    fractal
dimension (standard error):   0.001  0.03\n    radius (worst):
7.93   36.04\n    texture (worst):                        12.02  49.54\n
perimeter (worst):                    50.41  251.2\n    area (worst):
185.2  4254.0\n    smoothness (worst):                    0.071  0.223\n
compactness (worst):                  0.027  1.058\n    concavity (worst):
0.0    1.252\n    concave points (worst):                0.0    0.291\n
symmetry (worst):                     0.156  0.664\n    fractal dimension
(worst):            0.055  0.208\n    =====================================
====== ======\n\n    :Missing Attribute Values: None\n\n    :Class Distribution:
212 - Malignant, 357 - Benign\n\n    :Creator:  Dr. William H. Wolberg, W. Nick
Street, Olvi L. Mangasarian\n\n    :Donor: Nick Street\n\n    :Date: November,
1995\n\nThis is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic)
datasets.\nhttps://goo.gl/U2Uwz2\n\nFeatures are computed from a digitized image
```

```
of a fine needle\naspirate (FNA) of a breast mass.  They
describe\ncharacteristics of the cell nuclei present in the image.\n\nSeparating
plane described above was obtained using\nMultisurface Method-Tree (MSM-T) [K.
P. Bennett, "Decision Tree\nConstruction Via Linear Programming." Proceedings of
the 4th\nMidwest Artificial Intelligence and Cognitive Science Society,\npp.
97-101, 1992], a classification method which uses linear\nprogramming to
construct a decision tree.  Relevant features\nwere selected using an exhaustive
search in the space of 1-4\nfeatures and 1-3 separating planes.\n\nThe actual
linear program used to obtain the separating plane\nin the 3-dimensional space
is that described in:\n[K. P. Bennett and O. L. Mangasarian: "Robust
Linear\nProgramming Discrimination of Two Linearly Inseparable
Sets",\nOptimization Methods and Software 1, 1992, 23-34].\n\nThis database is
also available through the UW CS ftp server:\n\nftp ftp.cs.wisc.edu\ncd math-
prog/cpo-dataset/machine-learn/WDBC/\n\n.. topic:: References\n\n   - W.N.
Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction \n     for
breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on \n
Electronic Imaging: Science and Technology, volume 1905, pages 861-870,\n
San Jose, CA, 1993.\n   - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast
cancer diagnosis and \n     prognosis via linear programming. Operations
Research, 43(4), pages 570-577, \n     July-August 1995.\n   - W.H. Wolberg,
W.N. Street, and O.L. Mangasarian. Machine learning techniques\n     to diagnose
breast cancer from fine-needle aspirates. Cancer Letters 77 (1994) \n
163-171.',
 'feature_names': array(['mean radius', 'mean texture', 'mean perimeter', 'mean
area',
        'mean smoothness', 'mean compactness', 'mean concavity',
        'mean concave points', 'mean symmetry', 'mean fractal dimension',
        'radius error', 'texture error', 'perimeter error', 'area error',
        'smoothness error', 'compactness error', 'concavity error',
        'concave points error', 'symmetry error',
        'fractal dimension error', 'worst radius', 'worst texture',
        'worst perimeter', 'worst area', 'worst smoothness',
        'worst compactness', 'worst concavity', 'worst concave points',
        'worst symmetry', 'worst fractal dimension'], dtype='<U23'),
 'filename': 'breast_cancer.csv',
 'data_module': 'sklearn.datasets.data'}
```

[12]:
```
data_i=load_breast_cancer().data
```

[13]:
```
data_d=load_breast_cancer().target
```

[17]:
```
df1=pd.DataFrame(data_i,columns=load_breast_cancer().feature_names)
```

[21]:
```
df2=pd.DataFrame(data_d,columns=["target"])
```

[29]:
```
df=pd.concat([df1,df2],axis=1)
df
```

[29]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	\
0	17.99	10.38	122.80	1001.0	0.11840	
1	20.57	17.77	132.90	1326.0	0.08474	
2	19.69	21.25	130.00	1203.0	0.10960	
3	11.42	20.38	77.58	386.1	0.14250	
4	20.29	14.34	135.10	1297.0	0.10030	
..	…	…	…	…	…	
564	21.56	22.39	142.00	1479.0	0.11100	
565	20.13	28.25	131.20	1261.0	0.09780	
566	16.60	28.08	108.30	858.1	0.08455	
567	20.60	29.33	140.10	1265.0	0.11780	
568	7.76	24.54	47.92	181.0	0.05263	

	mean compactness	mean concavity	mean concave points	mean symmetry	\
0	0.27760	0.30010	0.14710	0.2419	
1	0.07864	0.08690	0.07017	0.1812	
2	0.15990	0.19740	0.12790	0.2069	
3	0.28390	0.24140	0.10520	0.2597	
4	0.13280	0.19800	0.10430	0.1809	
..	…	…	…	…	
564	0.11590	0.24390	0.13890	0.1726	
565	0.10340	0.14400	0.09791	0.1752	
566	0.10230	0.09251	0.05302	0.1590	
567	0.27700	0.35140	0.15200	0.2397	
568	0.04362	0.00000	0.00000	0.1587	

	mean fractal dimension	…	worst texture	worst perimeter	worst area	\
0	0.07871	…	17.33	184.60	2019.0	
1	0.05667	…	23.41	158.80	1956.0	
2	0.05999	…	25.53	152.50	1709.0	
3	0.09744	…	26.50	98.87	567.7	
4	0.05883	…	16.67	152.20	1575.0	
..	…	…	…	…	…	
564	0.05623	…	26.40	166.10	2027.0	
565	0.05533	…	38.25	155.00	1731.0	
566	0.05648	…	34.12	126.70	1124.0	
567	0.07016	…	39.42	184.60	1821.0	
568	0.05884	…	30.37	59.16	268.6	

	worst smoothness	worst compactness	worst concavity	\
0	0.16220	0.66560	0.7119	
1	0.12380	0.18660	0.2416	
2	0.14440	0.42450	0.4504	
3	0.20980	0.86630	0.6869	
4	0.13740	0.20500	0.4000	
..	…	…	…	
564	0.14100	0.21130	0.4107	

```
565           0.11660            0.19220           0.3215
566           0.11390            0.30940           0.3403
567           0.16500            0.86810           0.9387
568           0.08996            0.06444           0.0000

     worst concave points  worst symmetry  worst fractal dimension  target
0                  0.2654          0.4601                  0.11890       0
1                  0.1860          0.2750                  0.08902       0
2                  0.2430          0.3613                  0.08758       0
3                  0.2575          0.6638                  0.17300       0
4                  0.1625          0.2364                  0.07678       0
..                    …               …                        …      …
564                0.2216          0.2060                  0.07115       0
565                0.1628          0.2572                  0.06637       0
566                0.1418          0.2218                  0.07820       0
567                0.2650          0.4087                  0.12400       0
568                0.0000          0.2871                  0.07039       1

[569 rows x 31 columns]
```

[30]:
```
df.shape
```

[30]:
```
(569, 31)
```

[32]:
```
df.describe()
```

[32]:
```
       mean radius  mean texture  mean perimeter    mean area  \
count   569.000000    569.000000      569.000000   569.000000
mean     14.127292     19.289649       91.969033   654.889104
std       3.524049      4.301036       24.298981   351.914129
min       6.981000      9.710000       43.790000   143.500000
25%      11.700000     16.170000       75.170000   420.300000
50%      13.370000     18.840000       86.240000   551.100000
75%      15.780000     21.800000      104.100000   782.700000
max      28.110000     39.280000      188.500000  2501.000000

       mean smoothness  mean compactness  mean concavity  mean concave points  \
count       569.000000        569.000000      569.000000           569.000000
mean          0.096360          0.104341        0.088799             0.048919
std           0.014064          0.052813        0.079720             0.038803
min           0.052630          0.019380        0.000000             0.000000
25%           0.086370          0.064920        0.029560             0.020310
50%           0.095870          0.092630        0.061540             0.033500
75%           0.105300          0.130400        0.130700             0.074000
max           0.163400          0.345400        0.426800             0.201200

       mean symmetry  mean fractal dimension  …  worst texture  \
```

```
count     569.000000              569.000000  …     569.000000
mean        0.181162                0.062798  …      25.677223
std         0.027414                0.007060  …       6.146258
min         0.106000                0.049960  …      12.020000
25%         0.161900                0.057700  …      21.080000
50%         0.179200                0.061540  …      25.410000
75%         0.195700                0.066120  …      29.720000
max         0.304000                0.097440  …      49.540000

       worst perimeter   worst area  worst smoothness  worst compactness  \
count       569.000000   569.000000        569.000000         569.000000
mean        107.261213   880.583128          0.132369           0.254265
std          33.602542   569.356993          0.022832           0.157336
min          50.410000   185.200000          0.071170           0.027290
25%          84.110000   515.300000          0.116600           0.147200
50%          97.660000   686.500000          0.131300           0.211900
75%         125.400000  1084.000000          0.146000           0.339100
max         251.200000  4254.000000          0.222600           1.058000

       worst concavity  worst concave points  worst symmetry  \
count       569.000000            569.000000      569.000000
mean          0.272188              0.114606        0.290076
std           0.208624              0.065732        0.061867
min           0.000000              0.000000        0.156500
25%           0.114500              0.064930        0.250400
50%           0.226700              0.099930        0.282200
75%           0.382900              0.161400        0.317900
max           1.252000              0.291000        0.663800

       worst fractal dimension      target
count               569.000000  569.000000
mean                  0.083946    0.627417
std                   0.018061    0.483918
min                   0.055040    0.000000
25%                   0.071460    0.000000
50%                   0.080040    1.000000
75%                   0.092080    1.000000
max                   0.207500    1.000000

[8 rows x 31 columns]
```

[35]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
```

```
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  target                   569 non-null    int32
dtypes: float64(30), int32(1)
memory usage: 135.7 KB
```

[ ]:

[36]:
```
from sklearn.model_selection import train_test_split
```

[37]:
```
X_train,X_test,Y_train,Y_test=train_test_split(df1,df2,test_size=0.25)
```

[39]:
```
X_train.shape
```

[39]: (426, 30)

[40]:
```
Y_train.shape
```

[40]: (426, 1)

[41]:
```
X_test.shape
```

[41]:
```
(143, 30)
```

[42]:
```
Y_test.shape
```

[42]:
```
(143, 1)
```

[44]:
```
from sklearn.svm import SVC
from sklearn.preprocessing import MinMaxScaler
```

[46]:
```
req=MinMaxScaler()
```

[47]:
```
req.fit(X_train)
```

[47]:
```
MinMaxScaler()
```

[49]:
```
req.transform(X_train)
```

[49]:
```
array([[0.53192295, 0.37302905, 0.5287126 , …, 0.45532646, 0.28700966,
        0.16286239],
       [0.29244167, 0.44937759, 0.2782807 , …, 0.14415808, 0.2211709 ,
        0.12626263],
       [0.29196838, 0.23360996, 0.28691866, …, 0.33408935, 0.45022669,
        0.20523416],
       …,
       [0.25931185, 0.59460581, 0.27765877, …, 0.75945017, 0.55213877,
        1.        ],
       [0.20630413, 0.47925311, 0.19825859, …, 0.28446735, 0.24916223,
        0.21828676],
       [0.36722041, 0.65186722, 0.35180706, …, 0.37628866, 0.22807018,
        0.0952381 ]])
```

[51]:
```
req.transform(X_test)
```

[51]:
```
array([[0.22996829, 0.37302905, 0.23592012, …, 0.53127148, 0.63000197,
        0.55857274],
       [0.77187751, 0.70373444, 0.79545297, …, 0.93917526, 0.32190026,
        0.2136954 ],
       [0.40318993, 0.47178423, 0.40847212, …, 0.73333333, 0.28346146,
        0.32638069],
       …,
       [0.63699181, 0.50082988, 0.62200263, …, 0.60652921, 0.20579539,
        0.08074249],
       [0.27966302, 0.18257261, 0.28443093, …, 0.53642612, 0.21170905,
        0.41164896],
       [0.34260968, 0.55643154, 0.34952664, …, 0.60893471, 0.70983639,
        0.58743277]])
```

[ ]:

[53]:
```
svm=SVC()
```

[54]:
```
svm.fit(X_train,Y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184:
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().
  y = column_or_1d(y, warn=True)
```

[54]:
```
SVC()
```

[55]:
```
svm.score(X_test,Y_test)
```

[55]:
```
0.8951048951048951
```

[ ]:

[ ]: