Assignment9

November 7, 2020

1 Assignment 5 (CL7-B): Support Vector Machines

1.0.1 Roll No: 43141

1.0.2 Name: Sahil Naphade

1.0.3 Batch: R-9

Importing the libraries and datasets

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# Import dataset from Sklearn library (Breast cancer dataset bu UCI)

from sklearn.datasets import load_breast_cancer
cancer_dataset = load_breast_cancer()
```

1.1 Exploring data

[2]:
```
print("Features:", cancer_dataset.feature_names)
print("Labels: ", cancer_dataset.target_names)
```

```
Features: ['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']
Labels:  ['malignant' 'benign']
```

[3]:
```
cancer_df = pd.DataFrame(np.c_[cancer_dataset['data'],␣
 ↪cancer_dataset['target']],
```

```
                        columns = np.append(cancer_dataset['feature_names'],␣
↪['target']))
cancer_df.head()
```

[3]:

	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	

	mean compactness	mean concavity	mean concave points	mean symmetry	\
0	0.27760	0.3001	0.14710	0.2419	
1	0.07864	0.0869	0.07017	0.1812	
2	0.15990	0.1974	0.12790	0.2069	
3	0.28390	0.2414	0.10520	0.2597	
4	0.13280	0.1980	0.10430	0.1809	

	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	

	worst smoothness	worst compactness	worst concavity	worst concave points	\
0	0.1622	0.6656	0.7119	0.2654	
1	0.1238	0.1866	0.2416	0.1860	
2	0.1444	0.4245	0.4504	0.2430	
3	0.2098	0.8663	0.6869	0.2575	
4	0.1374	0.2050	0.4000	0.1625	

	worst symmetry	worst fractal dimension	target
0	0.4601	0.11890	0.0
1	0.2750	0.08902	0.0
2	0.3613	0.08758	0.0
3	0.6638	0.17300	0.0
4	0.2364	0.07678	0.0

[5 rows x 31 columns]

[4]:
```
# Information of the dataset
cancer_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    float64
dtypes: float64(31)
memory usage: 137.9 KB
```

[5]:
```
# Dimensions of the dataframe
cancer_df.shape
```

[5]:
```
(569, 31)
```

[6]:
```
# Describe the dataset
cancer_df.describe()
```

[6]:
```
       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
```

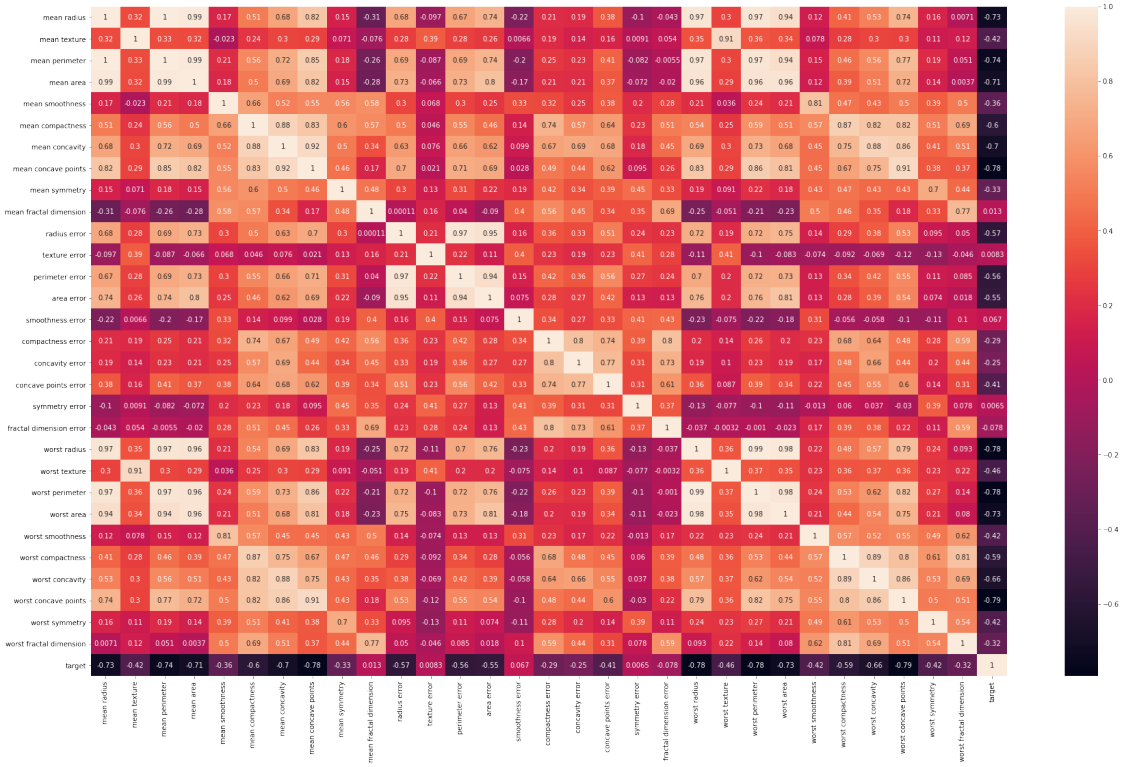
```
       worst fractal dimension      target
count               569.000000  569.000000
mean                  0.083946    0.627417
std                   0.018061    0.483918
min                   0.055040    0.000000
25%                   0.071460    0.000000
50%                   0.080040    1.000000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

[8 rows x 31 columns]
```

[7]:
```
# Check if there are null values
cancer_df.isnull().sum()
```

[7]:
```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
target                     0
dtype: int64
```

[8]:
```
# Checking the correlation between the variables
plt.figure(figsize=(30, 18))
sns.heatmap(cancer_df.corr(), annot = True)
```

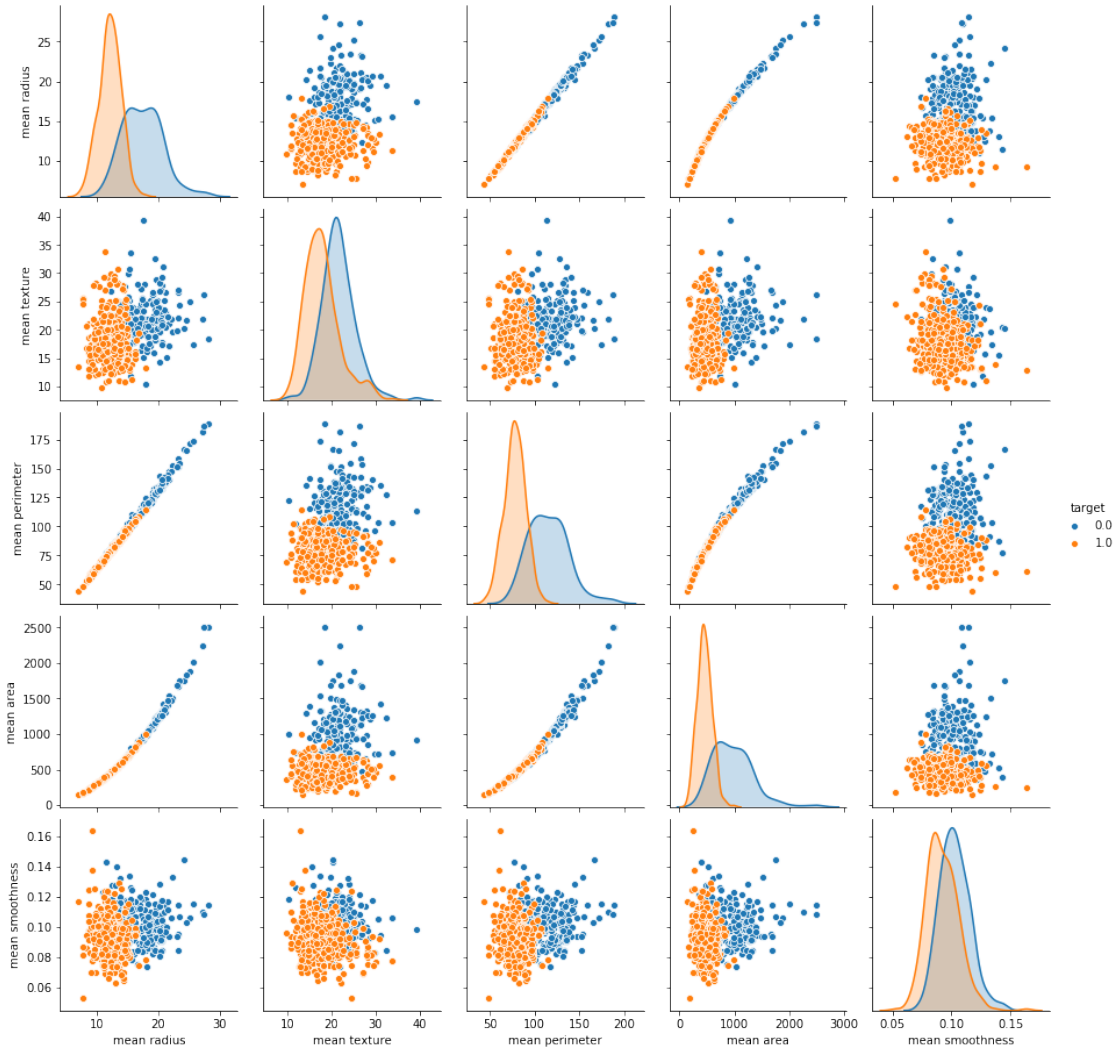
[8]: <matplotlib.axes._subplots.AxesSubplot at 0x23966136f48>



As we can observe, there is a strong correlation between mean radius and mean perimeter, same for mean area and mean perimeter.

[9]:
```
# Visualization of relation between each pair of first 5 features
sns.pairplot(cancer_df, hue='target', vars=['mean radius', 'mean texture',␣
↪'mean perimeter', 'mean area', 'mean smoothness'])
```

[9]: <seaborn.axisgrid.PairGrid at 0x2396a16dc08>



1.2 Modelling

1.2.1 We have our output (predicting value) as 'target'. Let's call it op.

1.2.2 Similarly, all the remaining columns are our input. Let this one be ip.

[10]:
```
# ip (input) will contain all the columns other than 'target'.
ip = cancer_df.drop(['target'], axis = 1)
ip.head()
```

[10]:

	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

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597
4           0.13280          0.1980              0.10430         0.1809

   mean fractal dimension  …  worst radius  worst texture  worst perimeter  \
0                 0.07871  …         25.38          17.33           184.60
1                 0.05667  …         24.99          23.41           158.80
2                 0.05999  …         23.57          25.53           152.50
3                 0.09744  …         14.91          26.50            98.87
4                 0.05883  …         22.54          16.67           152.20

   worst area  worst smoothness  worst compactness  worst concavity  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   worst concave points  worst symmetry  worst fractal dimension
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678

[5 rows x 30 columns]
```

[11]:
```
# op (output) will contain only the target values
op = cancer_df['target']
op.head()
```

[11]:
```
0    0.0
1    0.0
2    0.0
3    0.0
4    0.0
Name: target, dtype: float64
```

[12]:
```
# Importing train_test_split for splitting the data into training and testing␣
 ↪part
from sklearn.model_selection import train_test_split
```

Dividing dataset into 70% for training and 30% for testing

[13]:
```
ip_train, ip_test, op_train, op_test = train_test_split(ip, op, test_size = 0.
↪3, random_state = 20)
```

[14]:
```
# Checking the shape of the training and testing data
print("The shape of training input is " + str(ip_train.shape))
print("The shape of testing input is " + str(ip_test.shape))
print("The shape of training output is " + str(op_train.shape))
print("The shape of testing output is " + str(op_test.shape))
```

```
The shape of training input is (398, 30)
The shape of testing input is (171, 30)
The shape of training output is (398,)
The shape of testing output is (171,)
```

1.2.3 Import SVM model

[15]:
```
from sklearn.svm import SVC
svc_model = SVC()
```

[16]:
```
svc_model.fit(ip_train, op_train)
```

[16]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

1.2.4 Predict using the trained model

[17]:
```
op_predict = svc_model.predict(ip_test)
```

1.2.5 Creating confusion matrix for checking the output

[18]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

[19]:
```
mat = np.array(confusion_matrix(op_test, op_predict, labels=[1, 0]))
confusion = pd.DataFrame(mat, index=['is cancer', 'is healthy'],
                         columns=['predicted cancer', 'predicted_healthy'])
confusion
```

[19]:

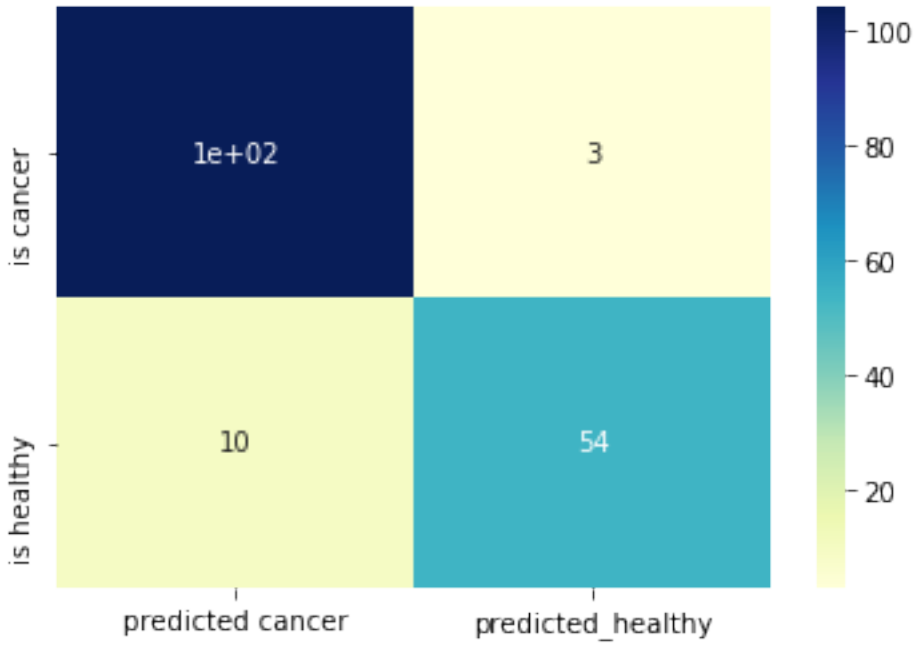
	predicted cancer	predicted_healthy
is cancer	104	3
is healthy	10	54

[20]:
```
print(classification_report(op_test, op_predict))
```

	precision	recall	f1-score	support
0.0	0.95	0.84	0.89	64
1.0	0.91	0.97	0.94	107
accuracy			0.92	171
macro avg	0.93	0.91	0.92	171
weighted avg	0.93	0.92	0.92	171

[21]:
```
sns.heatmap(confusion, annot=True, cmap="YlGnBu")
```

[21]: <matplotlib.axes._subplots.AxesSubplot at 0x2396acc2fc8>



[22]:
```
from sklearn import metrics
```

[23]:
```
print("Accuracy of the model is: ",metrics.accuracy_score(op_test, op_predict))
```

```
Accuracy of the model is:  0.9239766081871345
```