Machine Learning Fundamentals Lab-8

Name: Gaurav Prasanna
Reg No: 19BEC1315

Aim:

a) To implement and understand the Support Vector Machine or SVM algorithm on inbuilt dataset in scikit learn of Breast Cancer.
b) To implement SVM algorithm on Iris dataset using different kernels and to inspect the changes in accuracies.
c) To implement and visualize the hyperplane of SCM in 3d scatter plot using Axes3d package from mpl_toolkits library.

Software Required:

1) Jupyter Notebook
2) Anaconda Navigator

Libraries Required: Numpy, Matplotlib, Sci-kit Learn, Pandas, mpl_toolkits.

Code and Outputs:

a) SVM on Breast Cancer:

```
In [1]: from sklearn import datasets
        from sklearn.model_selection import train_test_split

In [2]: cancer = datasets.load_breast_cancer()

In [3]: cancer

Out[3]: {'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, 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,
```

In [4]:
```
print(cancer.feature_names)
```

```
['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']
```

In [5]:
```
print(cancer.target_names)
```

```
['malignant' 'benign']
```

In [6]:
```
print(cancer.data.shape)
```

```
(569, 30)
```

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, test_size=0.3, random_state=1)
```

In [10]:
```
from sklearn import svm
```

In [11]:
```
clf = svm.SVC(kernel='linear')
clf.fit(X_train, y_train)
```

Out[11]:
```
SVC(kernel='linear')
```

In [13]:
```
y_pred = clf.predict(X_test)
```

In [14]:
```
print(y_pred)
```

```
[1 0 1 0 0 0 0 0 1 1 1 0 0 1 1 1 1 1 1 0 1 1 0 1 0 1 1 0 0 0 0 1 0 0 1 1 0
 1 1 1 1 1 1 1 1 0 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0
 1 0 1 1 1 0 1 0 1 0 1 1 0 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0
 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 0 0 0 1 1 1 0 1 0 0 1 1 0 0 0 1 0 1 0 1 1
 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 0 1 1]
```

In [15]:
```
print(y_test)
```

```
[1 0 1 0 0 0 0 0 1 1 1 0 0 1 1 1 1 1 1 0 1 1 0 1 0 1 1 0 0 0 0 1 0 0 1 1 0
 1 0 1 1 1 1 1 1 0 1 1 1 0 0 0 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 1 1 1 0
 1 0 0 0 1 0 1 0 1 0 1 1 0 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0
 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 0 0 1 1 1 0 1 0 0 1 1 1 0 0 1 0 1 0 1 1
 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 0 0 0 1 1]
```

In [17]:
```
from sklearn import metrics

print("Accuracy:", metrics.accuracy_score(y_pred, y_test))
print("Precision:", metrics.precision_score(y_pred, y_test))
print("Recall:", metrics.recall_score(y_pred, y_test))
```

```
Accuracy: 0.9532163742690059
Precision: 0.9814814814814815
Recall: 0.9464285714285714
```

b) SVM on Iris Dataset with different Kernels:

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
        from sklearn import datasets
        from sklearn.model_selection import train_test_split

In [2]: data = datasets.load_iris()

In [3]: data

Out[3]: {'data': array([[5.1, 3.5, 1.4, 0.2],
                [4.9, 3. , 1.4, 0.2],
                [4.7, 3.2, 1.3, 0.2],
                [4.6, 3.1, 1.5, 0.2],
                [5. , 3.6, 1.4, 0.2],
                [5.4, 3.9, 1.7, 0.4],
                [4.6, 3.4, 1.4, 0.3],
                [5. , 3.4, 1.5, 0.2],
                [4.4, 2.9, 1.4, 0.2],
                [4.9, 3.1, 1.5, 0.1],
                [5.4, 3.7, 1.5, 0.2],
                [4.8, 3.4, 1.6, 0.2],
                [4.8, 3. , 1.4, 0.1],
                [4.3, 3. , 1.1, 0.1],
                [5.8, 4. , 1.2, 0.2],
                [5.7, 4.4, 1.5, 0.4],
                [5.4, 3.9, 1.3, 0.4],
                [5.1, 3.5, 1.4, 0.3],
                [5.7, 3.8, 1.7, 0.3],
                [5.1, 3.8, 1.5, 0.3],
```

```
In [4]: print(data.feature_names)

        ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

In [5]: print(data.target_names)

        ['setosa' 'versicolor' 'virginica']

In [6]: print(data.target)

        [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
         0 0 0 0 0 0 0 0 0 0 0 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 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 2 2 2 2 2 2 2 2 2 2 2
         2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
         2 2]

In [7]: print(data.data[0:5])

        [[5.1 3.5 1.4 0.2]
         [4.9 3.  1.4 0.2]
         [4.7 3.2 1.3 0.2]
         [4.6 3.1 1.5 0.2]
         [5.  3.6 1.4 0.2]]

In [8]: X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.2, random_state=1)
```

```
In [9]: from sklearn.svm import SVC

In [10]: svClassifier = SVC(kernel='poly', degree=8)

In [11]: svClassifier.fit(X_train, y_train)

Out[11]: SVC(degree=8, kernel='poly')

In [12]: y_pred = svClassifier.predict(X_test)

In [13]: from sklearn.metrics import classification_report, confusion_matrix

In [14]: print(classification_report(y_pred, y_test))
         print(confusion_matrix(y_pred, y_test))

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       0.92      0.92      0.92        13
           2       0.83      0.83      0.83         6

    accuracy                           0.93        30
   macro avg       0.92      0.92      0.92        30
weighted avg       0.93      0.93      0.93        30

[[11  0  0]
 [ 0 12  1]
 [ 0  1  5]]
```

```
In [15]: rbfclassifier = SVC(kernel='rbf', degree=8)

In [16]: rbfclassifier.fit(X_train, y_train)
Out[16]: SVC(degree=8)

In [17]: y_predgauss = rbfclassifier.predict(X_test)

In [18]: print(classification_report(y_predgauss, y_test))
         print(confusion_matrix(y_predgauss, y_test))
```

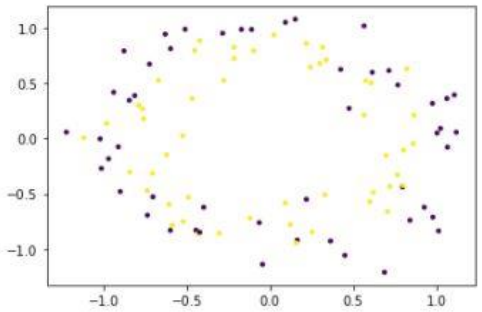
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       0.92      1.00      0.96        12
           2       1.00      0.86      0.92         7

    accuracy                           0.97        30
   macro avg       0.97      0.95      0.96        30
weighted avg       0.97      0.97      0.97        30

[[11  0  0]
 [ 0 12  0]
 [ 0  1  6]]
```

```
In [20]: linearClassifier = SVC(kernel='linear', degree=8)

In [21]: linearClassifier.fit(X_train, y_train)
Out[21]: SVC(degree=8, kernel='linear')

In [23]: y_predlinear = linearClassifier.predict(X_test)

In [24]: print(classification_report(y_predlinear, y_test))
         print(confusion_matrix(y_predlinear, y_test))
```

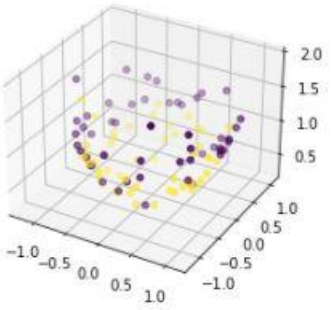
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00         6

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
```

c) Visualizing 3d Scatterplot and Hyperplane

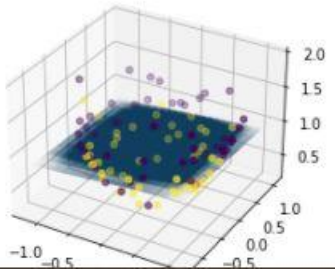
```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.datasets import make_circles
        from mpl_toolkits.mplot3d import Axes3D

In [2]: X, y = make_circles(n_samples = 100, noise = 0.14)
        plt.scatter(X[:, 0], X[:, 1], c = y, marker = '.')
        plt.show()
```



```
In [3]: X1 = X[:, 0].reshape((-1, 1))
        X2 = X[:, 1].reshape((-1, 1))
        X3 = (X1**2 + X2**2)

In [4]: X = np.hstack((X, X3))

In [5]: fig = plt.figure()
        axes = fig.add_subplot(111, projection= '3d')
        axes.scatter(X1, X2, X3, c=y, depthshade=True)
        plt.show()
```




```
In [6]: from sklearn.svm import SVC

        svc = SVC(kernel='linear')
        svc.fit(X,y)
        w = svc.coef_
        b = svc.intercept_

In [7]: x1 = X[:, 0].reshape((-1, 1))
        x2 = X[:, 1].reshape((-1, 1))
        x1, x2 = np.meshgrid(x1, x2)
        x3 = -(w[0][0]*x1 + w[0][1]*x2 +b)/ w[0][2]

        fig = plt.figure()
        axes2 = fig.add_subplot(111, projection= '3d')
        axes2.scatter(X1, X2, X3, c=y, depthshade=True)
        axes1 = fig.gca(projection = '3d')
        axes1.plot_surface(x1, x2, x3, alpha = 0.01)
        plt.show()
```



Inference:

a) From first we infer that the SVM is a good classification algorithm for breast cancer dataset, while using linear kernel we achieve accuracy of about 95.3%.

b) From second we can infer that the with different kernels such as polynomial, gaussian and linear how the accuracy and various other metrics vary and which hyperparameter works best with the given dataset.
c) From the third one, we can infer and visualize the plot of hyperplane and how it divides the axis, and its through 3d scatterplot. Helps us to understand how SVM works even better.

Result: SVM on Breast cancer , iris dataset is implemented and Hyperplane is visualized using Jupyter notebook and the required plots are shown.