VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

Hashu Advani Memorial Complex, Collector's Colony, R C Marg, Chembur, Mumbai-
400074

V.E.S.
Since 1962

Department of Artificial Intelligence and Data Science

Subject: ML **Class:** D11AD **Semester:** VI

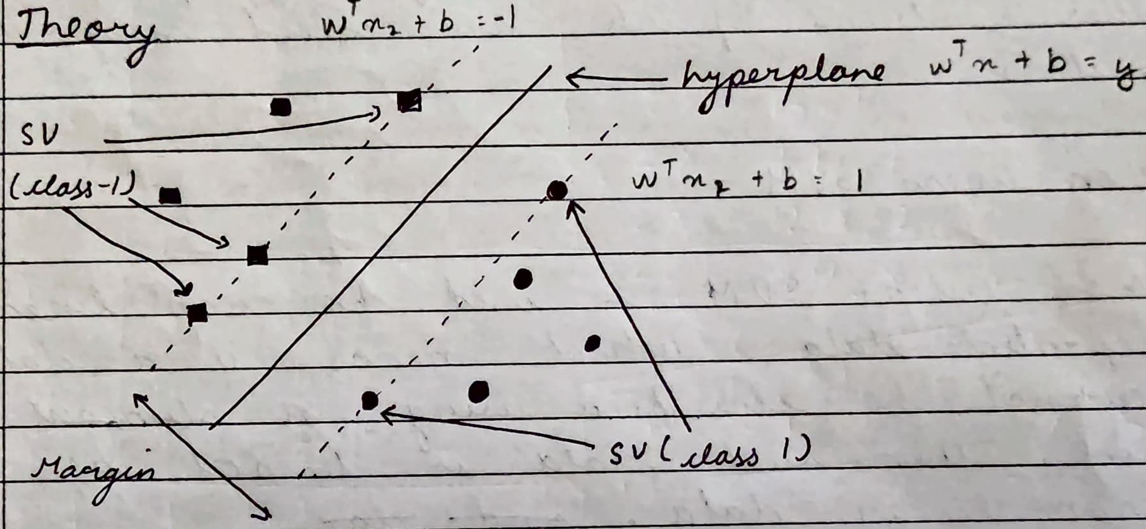
Roll No.: 26	Name: Dyotak Kachare		
Exp No.:	Title: Support Vector Machine		
DOP:		DOS:	
GRADE		LAB OUTCOME:	SIGNATURE:

ML experiment

Aim

Support Vector Machine

Theory



$$\hat{y} = \begin{cases} -1, & w^T x_1 + b \le \cancel{0} -1 \\ 1, & w^T x_2 + b \ge \cancel{0}\ 1 \end{cases}$$

$$L = \max(0, 1 - y_i (w^T x_i + b))$$

↑

Hinge loss function.

The goal of the SVM algorithm is to create a best line or a decision boundry that can segregate n dimensional space into classes so that we can easily put the new data in the correct category in future.

This best decision boundary is called hyperplane.

SVM chooses the extreme points / vectors that help in the creating hyperplane. These extreme points are called support vectors

Non linear SVM -

Non linear SVM is used for non-linearly seprated data, which means if a dataset cannot be classified using a straight line, then such data is termed as non-linear data. ~~and clas~~

SVM has a technique called the kernel trick.
These are funcations that take low dimensional input space and transform it into a higher dimensional space, ie it converts not seperable problem to seperable problem

Eg. Polynomial Kernel, Gausian RBF

conclusion -

Thus we have succesfully implemented support vector machine.

0.1 Aim: Support Vector Regression

Dataset: https://www.kaggle.com/datasets/devzohaib/tvmarketingcsv

```
[ ]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn.model_selection import train_test_split
     from sklearn.svm import SVR
     import warnings
     warnings.simplefilter(action='ignore')
```

```
[ ]: # Importing data
     df = pd.read_csv('./tvmarketing.csv')
     df.head()
```

```
[ ]:       TV  Sales
     0  230.1   22.1
     1   44.5   10.4
     2   17.2    9.3
     3  151.5   18.5
     4  180.8   12.9
```

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

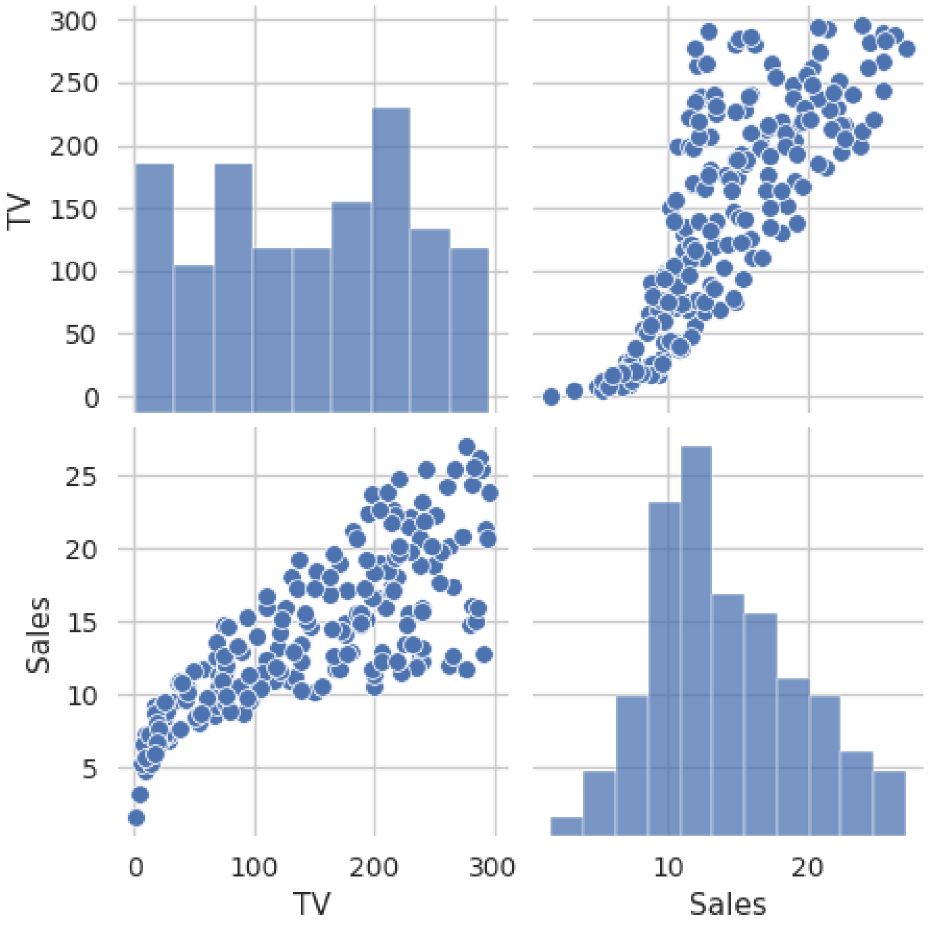
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   TV      200 non-null    float64
 1   Sales   200 non-null    float64
dtypes: float64(2)
memory usage: 3.2 KB
```

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

```
[ ]:                TV       Sales
     count  200.000000  200.000000
     mean   147.042500   14.022500
     std     85.854236    5.217457
     min      0.700000    1.600000
     25%     74.375000   10.375000
     50%    149.750000   12.900000
     75%    218.825000   17.400000
     max    296.400000   27.000000
```

```
[ ]: sns.pairplot(df)
```

```
[ ]: <seaborn.axisgrid.PairGrid at 0x7c5d063d9de0>
```



[ ]:
```
df[target_variable] = transformed_data
```

[ ]:
```
X = np.array(df['TV']).reshape(-1, 1)
y = df['Sales']
```

[ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
  ↪random_state=42)

# Train a linear regression model
model = SVR(kernel="linear", C=1, gamma="auto")
model.fit(X_train, y_train)
```

[ ]:
```
SVR(C=1, gamma='auto', kernel='linear')
```

[ ]:
```
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

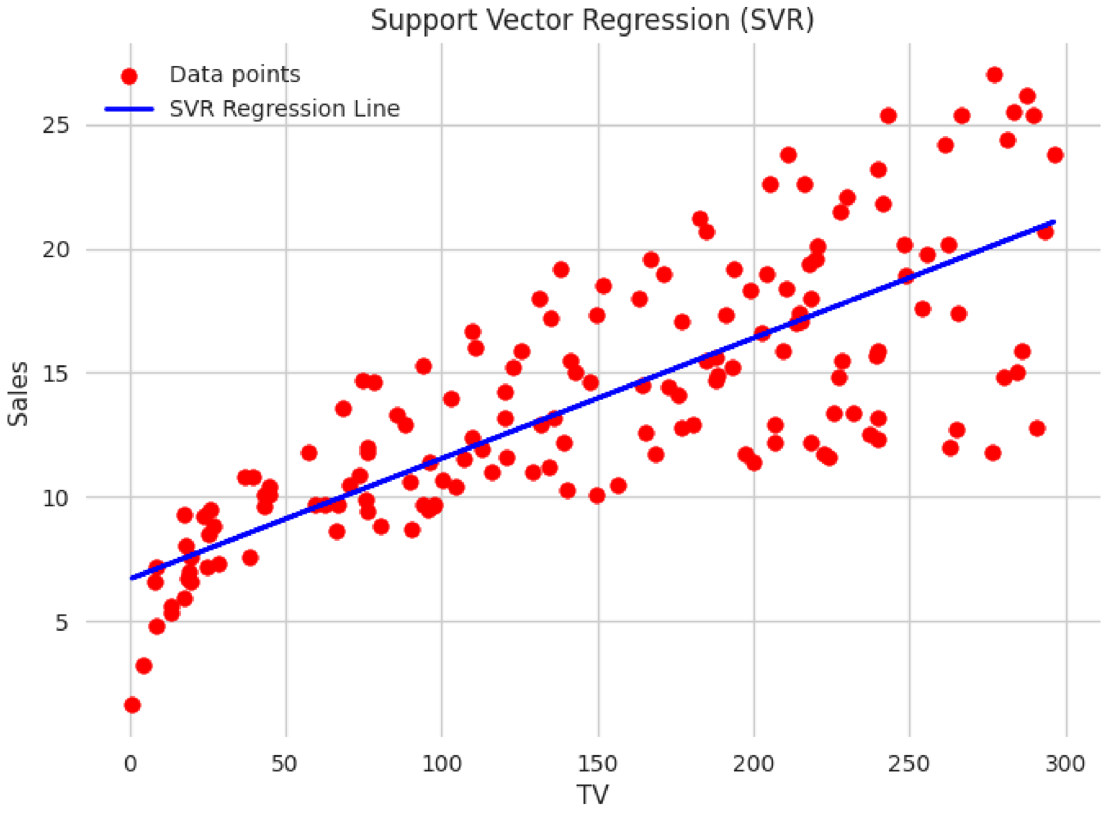
```
Mean Squared Error: 3.248967470487195
```

[ ]:

```
plt.scatter(X_train, y_train, color='red', label='Data points')

plt.plot(X, model.predict(X), color='blue', linewidth=2, label='SVR Regression␣
  ↪Line')

plt.xlabel('TV')
plt.ylabel('Sales')
plt.title('Support Vector Regression (SVR)')
plt.legend()
plt.show()
```



0.2 Support Vector Machine

[ ]:

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import pandas as pd
```

[ ]:

```
df = pd.read_csv('placement.csv')
```

[ ]:

```
df.describe()
```

```
[ ]:             cgpa  placement_exam_marks       placed
     count  1000.000000           1000.000000  1000.000000
     mean      6.961240             32.225000     0.489000
     std       0.615898             19.130822     0.500129
     min       4.890000              0.000000     0.000000
     25%       6.550000             17.000000     0.000000
     50%       6.960000             28.000000     0.000000
     75%       7.370000             44.000000     1.000000
     max       9.120000            100.000000     1.000000
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 3 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   cgpa                  1000 non-null   float64
 1   placement_exam_marks  1000 non-null   float64
 2   placed                1000 non-null   int64
dtypes: float64(2), int64(1)
memory usage: 23.6 KB
```

```
[ ]: X = df[['cgpa', 'placement_exam_marks']]
     y = df['placed']

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
       ↪random_state=42)

     svm_model = SVC(kernel='linear', C=10, gamma="auto")

     svm_model.fit(X_train, y_train)

     y_pred = svm_model.predict(X_test)

     cm = confusion_matrix(y_test, y_pred)

     print(f"Confusion Matrix:\n{cm}")
```

```
Confusion Matrix:
[[45 62]
 [35 58]]
```