

Exp no : 9**SUPPORT VECTOR MACHINE****Date : 27/09/2023****AIM:**

To implement the support vector machine algorithm by using R programming in R studio.

PROCEDURE :

1. Import the packages.
2. Make them in column wise.
3. Encode the target feature as a factor.
4. Spilt the dataset into training set and Test set.
5. Feature scaling.
6. Fitting SVM into training set.
7. Predicting the test results.
8. Making the confusion matrix.
9. Plotting the traing dataset as results.

CONCEPT APPLIED :

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane that categorizes new examples.

PROGRAM CODE :

```
library(readr)

social <- read_csv("C:\\Users\\Asus\\Downloads\\social.csv")

View(social)
```

OUTPUT :

	User ID	Gender	Age	EstimatedSalary	Purchased
1	15624510	Male	19	19000	0
2	15810944	Male	35	20000	0
3	15668575	Female	26	43000	0
4	15603246	Female	27	57000	0
5	15804002	Male	19	76000	0
6	15728773	Male	27	58000	0
7	15598044	Female	27	84000	0
8	15694829	Female	32	150000	1
9	15600575	Male	25	33000	0
10	15727311	Female	35	65000	0
11	15570769	Female	26	80000	0
12	15606274	Female	26	52000	0
13	15746139	Male	20	86000	0
14	15704987	Male	32	18000	0
15	15628972	Male	18	82000	0
16	15697686	Male	29	80000	0

PROGRAM CODE :

```
# Taking columns 3-5
```

```
dataset = dataset[3:5]
```

OUTPUT :

	Age	EstimatedSalary	Purchased
1	19	19000	0
2	35	20000	0
3	26	43000	0
4	27	57000	0
5	19	76000	0
6	27	58000	0
7	27	84000	0
8	32	55000	1
9	25	33000	0
10	35	65000	0
11	26	80000	0
12	28	52000	0
13	20	86000	0
14	32	19000	0
15	18	62000	0
16	29	80000	0
17	47	25000	1
18	45	26000	1
19	46	28000	1
20	48	29000	1
21	45	22000	1
22	47	49000	1
23	48	41000	1
24	45	23000	1
25	46	23000	1
26	47	20000	1
27	48	28000	1
28	47	30000	1
29	29	43000	0
30	31	18000	0
31	31	74000	0
32	27	57000	1
33	21	34000	0
34	28	44000	0
35	27	90000	0
36	35	27000	0
37	33	29000	0
38	40	49000	1

PROGRAM CODE :

```
# Encoding the target feature as factor
```

```
dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1))
```

OUTPUT :

	Age	EstimatedSalary	Purchased
1	19	19000	0
2	35	20000	0
3	26	43000	0
4	27	57000	0
5	19	76000	0
6	27	58000	0
7	27	84000	0
8	32	55000	1
9	25	33000	0
10	35	65000	0
11	26	80000	0
12	28	52000	0
13	20	86000	0
14	32	19000	0
15	18	62000	0
16	29	80000	0
17	47	25000	1
18	45	26000	1
19	46	28000	1
20	48	29000	1
21	45	22000	1
22	47	49000	1
23	48	41000	1
24	45	23000	1
25	46	23000	1
26	47	20000	1
27	48	28000	1
28	47	30000	1
29	29	43000	0
30	31	18000	0
31	31	74000	0
32	27	57000	1
33	21	34000	0
34	28	44000	0
35	27	90000	0
36	35	27000	0
37	33	29000	0
38	40	49000	1

PROGRAM CODE :

```
# Splitting the dataset into the Training set and Test set

install.packages('caTools')

library(caTools)

set.seed(123)

split = sample.split(dataset$Purchased, SplitRatio = 0.75)

training_set = subset(dataset, split == TRUE)

test_set = subset(dataset, split == FALSE)
```

OUTPUT :

```
> split
 [1] TRUE FALSE TRUE FALSE FALSE TRUE TRUE TRUE FALSE TRUE TRUE
[12] FALSE TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE TRUE FALSE
[23] TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE FALSE TRUE
[34] FALSE FALSE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[45] FALSE FALSE TRUE FALSE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE
[56] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE
[67] TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE FALSE FALSE TRUE TRUE
[78] TRUE TRUE TRUE TRUE FALSE TRUE FALSE FALSE FALSE FALSE TRUE
[89] FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[100] TRUE TRUE TRUE FALSE FALSE TRUE TRUE FALSE FALSE FALSE TRUE
[111] TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE
[122] TRUE TRUE FALSE TRUE FALSE FALSE TRUE TRUE TRUE FALSE TRUE
[133] TRUE FALSE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE
[144] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE
[155] TRUE FALSE TRUE TRUE FALSE TRUE TRUE FALSE FALSE TRUE TRUE
[166] TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE FALSE FALSE
[177] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[188] TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE
[199] FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE
[210] TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[221] TRUE TRUE TRUE FALSE TRUE FALSE TRUE FALSE FALSE FALSE TRUE
[232] TRUE TRUE FALSE TRUE FALSE FALSE TRUE FALSE TRUE FALSE TRUE
[243] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[254] TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE
[265] FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE TRUE
[276] TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE FALSE
[287] TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE
[298] TRUE FALSE TRUE TRUE FALSE TRUE TRUE FALSE TRUE FALSE TRUE
[309] TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE
[320] TRUE TRUE TRUE TRUE FALSE TRUE FALSE TRUE TRUE TRUE TRUE TRUE
[331] TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE FALSE
[342] TRUE FALSE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE
[353] FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE
[364] FALSE TRUE TRUE FALSE FALSE FALSE TRUE TRUE FALSE FALSE TRUE
[375] TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE FALSE TRUE TRUE
[386] TRUE TRUE TRUE FALSE TRUE TRUE FALSE TRUE TRUE FALSE TRUE
[397] TRUE TRUE TRUE FALSE
```

Testing dataset :

training_set =	test_set =	
Age	EstimatedSalary	Purchased
2	35	20000 0
4	27	57000 0
5	19	76000 0
9	25	32000 0
12	26	52000 0
18	45	26000 1
19	46	28000 1
20	48	29000 1
22	47	48000 1
29	29	43000 0
32	27	137000 1
34	28	44000 0
35	27	80000 0
38	30	49000 0
45	28	84000 0
46	23	20000 0
48	27	54000 0
52	18	44000 0
54	24	58000 0
69	22	63000 0
74	33	113000 0
75	32	18000 0
82	39	42000 0
84	35	88000 0
85	30	62000 0
86	31	118000 1
87	24	55000 0
89	26	95000 0
103	52	98000 0
104	33	149000 1
107	26	35000 0
108	27	89000 0
109	26	86000 0
117	35	79000 0
124	35	53000 0
126	29	65000 0
127	42	85000 0
193	31	54000 0

Training dataset:

training_set =	test_set =	
Age	EstimatedSalary	Purchased
1	23	22000 0
3	28	43000 0
6	27	54000 0
7	27	84000 0
8	30	25000 1
10	30	45000 0
11	26	80000 0
13	28	86000 0
14	30	10000 0
15	18	40000 0
16	29	80000 0
17	47	20000 1
18	45	22000 1
19	48	42000 1
24	45	22000 1
25	44	28000 0
26	47	27000 1
27	49	38000 1
28	47	30000 1
30	30	13000 0
31	31	14000 0
32	31	14000 0
36	30	27000 0
37	32	28000 0
38	26	72000 0
40	27	83000 0
41	27	57000 0
42	30	51000 0
43	30	28000 0
44	30	13000 0
47	25	78000 0
48	30	10000 1
50	30	80000 0
51	24	50000 0
53	25	80000 0
54	30	23000 0
55	27	54000 0
56	18	54000 0

PROGRAM CODE :

Feature Scaling

```
training_set[-3] = scale(training_set[-3])
```

```
test_set[-3] = scale(test_set[-3])
```

OUTPUT:

Training dataset:

test_set < > training_set < >			
Filter			
	Age	EstimatedSalary	Purchased
1	-1.70554750	-1.47334137	0
3	-1.09629664	-0.78837805	0
6	-1.00068938	-0.36027373	0
7	-1.00068938	0.38177303	0
8	-0.52265305	2.24342785	1
10	-0.23583125	-0.14049118	0
11	-1.09629664	-0.26781214	0
13	-1.66990328	0.43885347	0
14	-0.52265305	-1.80188158	0
15	-1.86115477	-0.32468259	0
16	-0.80947485	0.26781214	0
17	0.91145593	-1.30210004	1
21	0.72024140	-1.38772071	1
23	1.00700320	-0.84545650	1
24	0.72024140	-1.38772071	1
25	0.81504866	-1.35908049	1
26	0.91145593	-1.44480135	1
27	1.10267046	-1.21647938	1
28	0.91145593	-1.25938893	1
30	-0.61826032	-1.50188159	0
31	-0.61826032	-0.08837081	0
33	-1.57433297	-1.55896204	0
36	-0.23583125	-1.24950180	0
37	-0.42704579	-1.21647938	0
39	-1.09629664	-0.03829037	0
40	-1.00068938	-1.13085871	0
41	-1.00068938	-1.53042183	0
42	-0.42704579	-0.56055428	0
43	-0.23583125	1.06673835	0
44	-0.71386758	-1.58750226	0
47	-1.19190391	-0.23907182	0
48	-0.71386758	1.83732433	1
50	-0.61826032	0.52447414	0
51	-1.28751117	-1.10223849	0
53	-0.80947485	0.35523281	0
54	-0.23583125	-1.35908049	0
55	-1.00068938	-0.36027373	0
66	-1.76751117	-0.44888161	0

Showing 1 to 38 of 300 entries

Testing dataset :

test_set < >			
Filter			
	Age	EstimatedSalary	Purchased
2	-0.30419063	-1.51354339	0
4	-1.05994374	-0.32456028	0
5	-1.81509686	0.28589804	0
9	-1.24888202	-1.09579256	0
12	-1.15441288	-0.48523366	0
18	0.64058078	-1.32073551	1
19	0.73496990	-1.25646596	1
20	0.92390818	-1.22453126	1
22	0.82943984	-0.58163769	1
29	-0.87100546	-0.77444577	0
32	-1.05994374	2.24621408	1
34	-0.98547480	-0.74251109	0
35	-1.05994374	0.73588415	0
38	-0.77853633	-0.58163769	0
45	-0.96547460	0.54307608	0
46	-1.43782030	-1.51354339	0
48	-1.05994374	-0.42096430	0
52	-1.91016000	-0.74231109	0
66	-1.34335116	-0.29242558	0
69	-1.53228944	-0.13179218	0
74	-0.45312891	1.47498177	0
75	-0.58759805	-1.57781275	0
82	0.07368593	-0.80658045	0
84	-0.30419063	0.87161480	0
85	-0.77653633	-0.16388686	0
86	-0.68208719	1.63565517	1
87	-1.34335116	-0.38882862	0
89	-1.15441288	0.44667204	0
103	-0.58759805	0.60734544	0
104	-0.49312891	2.43183023	1
107	-1.15441288	-1.03152300	0
108	-1.05994374	0.70374947	0
109	-1.15441288	0.60734544	0
117	-0.30419063	0.25386387	0
124	-0.30419063	-0.45309890	0
126	0.07368593	-0.19602154	0
127	0.35709335	-0.06748289	0
141	-0.48586718	-0.70147888	0

Showing 1 to 38 of 100 entries

PROGRAM CODE :

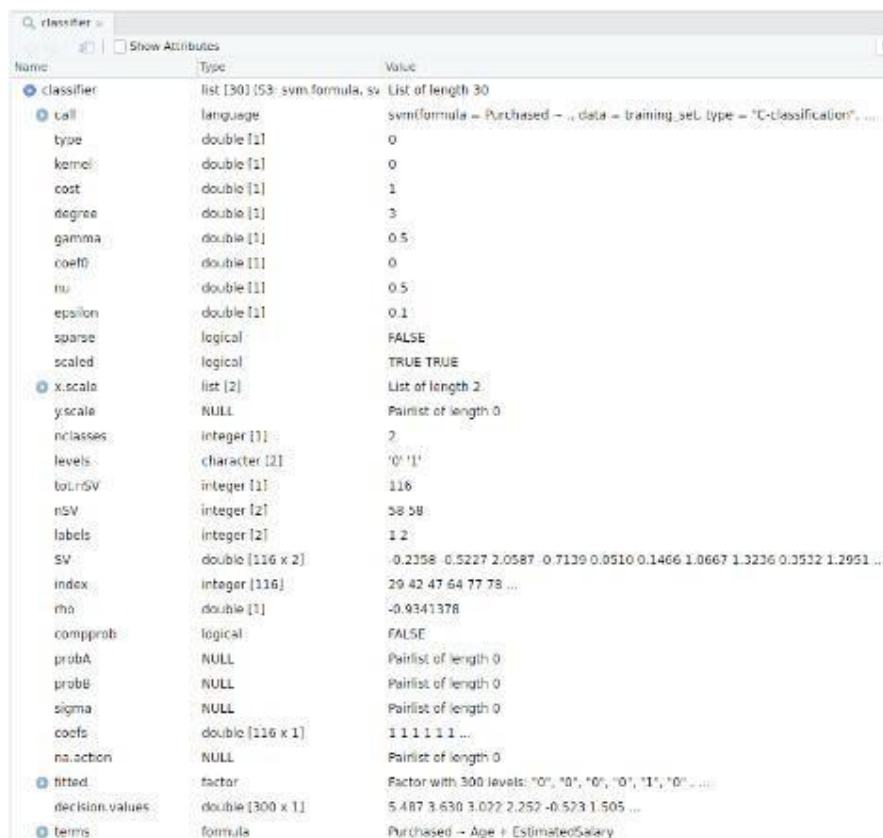
```
# Fitting SVM to the Training set

install.packages('e1071')

library(e1071)

classifier = svm(formula = Purchased ~ .,
                  data = training_set,
                  type = 'C-classification',
                  kernel = 'linear')
```

OUTPUT :



Name	Type	Value
classifier	list [30] (53: svm.formula, sv	List of length 30
call	language	svm(formula = Purchased ~ ., data = training_set, type = "C-classification", ...
type	double [1]	0
kernel	double [1]	0
cost	double [1]	1
degree	double [1]	3
gamma	double [1]	0.5
coef0	double [1]	0
nu	double [1]	0.5
epsilon	double [1]	0.1
sparse	logical	FALSE
scaled	logical	TRUE TRUE
x.scale	list [2]	List of length 2
y.scale	NULL	Pairlist of length 0
n.classes	integer [1]	2
levels	character [2]	"0" "1"
tot.nSV	integer [1]	116
nSV	integer [2]	58 58
labels	integer [2]	1 2
sv	double [116 x 2]	-0.2358 -0.5227 2.0587 -0.7139 0.0510 0.1466 1.0667 1.3236 0.3512 1.2951 ...
index	integer [116]	29 42 47 64 77 78 ...
rho	double [1]	-0.9341378
compprob	logical	FALSE
probA	NULL	Pairlist of length 0
probB	NULL	Pairlist of length 0
sigma	NULL	Pairlist of length 0
coefs	double [116 x 1]	1 1 1 1 1 ...
na.action	NULL	Pairlist of length 0
fitted	factor	Factor with 300 levels: "0", "0", "0", "0", "1", "0", ...
decision.values	double [300 x 1]	5.487 3.630 3.022 2.252 -0.523 1.505 ...
terms	formula	Purchased ~ Age + EstimatedSalary

PROGRAM CODE :

```
# Predicting the Test set results

y_pred = predict(classifier, newdata = test_set[-3])
```

OUTPUT :

```
> y_pred
 2  4  5  9 12 18 19 20 22 29 32 34 35 38 45 46 48
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
52 66 69 74 75 82 84 85 86 87 89 103 104 107 108 109 117
0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0
124 126 127 131 134 139 148 154 156 159 162 163 170 175 176 193 199
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
200 208 213 224 226 228 229 230 234 236 237 239 241 255 264 265 266
0  1  1  1  0  1  0  1  1  1  0  1  1  1  0  1  1
273 274 281 286 292 299 302 305 307 310 316 324 326 332 339 341 343
1  1  1  0  1  1  1  0  1  0  0  0  0  1  0  1  0
347 353 363 364 367 368 369 372 373 380 383 389 392 395 400
1  1  0  1  1  1  0  1  0  1  1  0  0  0  0  0
Levels: 0 1
> |
```

PROGRAM CODE :

```
# Making the Confusion Matrix
```

```
cm = table(test_set[, 3], y_pred)
```

OUTPUT :

```
> cm
      y_pred
      0  1
0  57  7
1  13 23
```

PROGRAM CODE :

```
# installing library ElemStatLearn
```

```
library(ElemStatLearn)
```

```
# Plotting the training data set results
```

```
set = training_set
```

```
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
```

```
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
```

```
grid_set = expand.grid(X1, X2)
```

```
colnames(grid_set) = c('Age', 'EstimatedSalary')
```

```
y_grid = predict(classifier, newdata = grid_set)
```

```
plot(set[, -3],  
      main = 'SVM (Training set)',  
      xlab = 'Age', ylab = 'Estimated Salary',  
      xlim = range(X1), ylim = range(X2))  
  
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)  
  
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'coral1', 'aquamarine'))  
  
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```

OUTPUT :



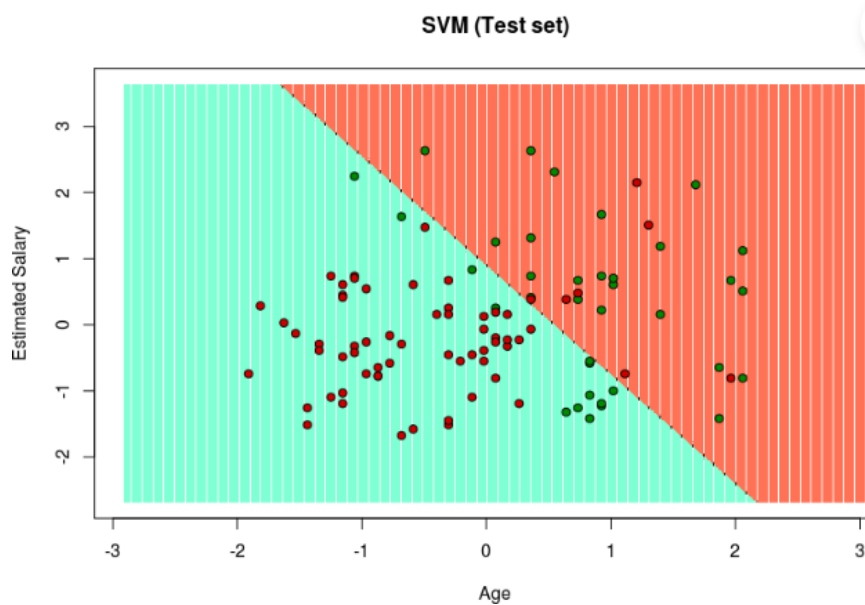
PROGRAM CODE :

```
set = test_set  
  
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)  
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)  
  
grid_set = expand.grid(X1, X2)  
colnames(grid_set) = c('Age', 'EstimatedSalary')  
y_grid = predict(classifier, newdata = grid_set)
```



```
plot(set[, -3], main = 'SVM (Test set)',  
      xlab = 'Age', ylab = 'Estimated Salary',  
      xlim = range(X1), ylim = range(X2))  
  
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)  
  
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'coral1', 'aquamarine'))  
  
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```

OUTPUT :



RESULT :

Thus, the support vector machine algorithm was implemented by using R programming in R studio.