Experiment-14

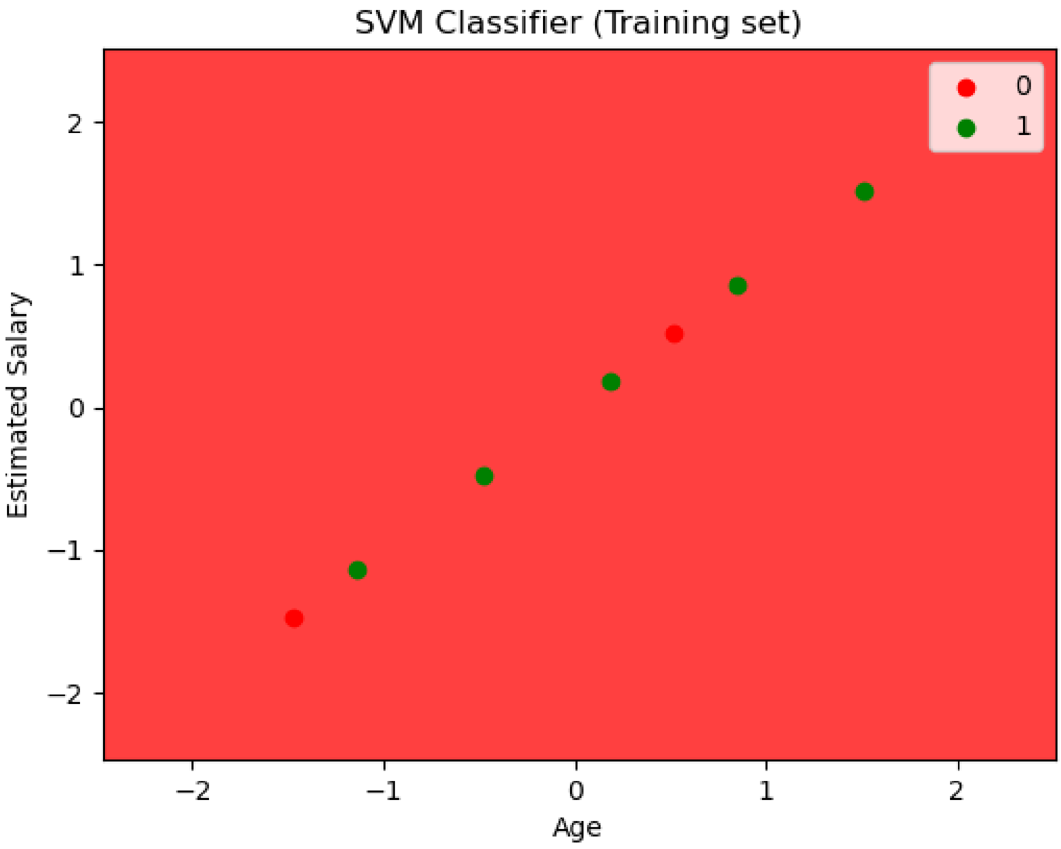
AIM: Write a program to Implement Support Vector Machines and Principle Component Analysis.

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.metrics import confusion_matrix

from matplotlib.colors import ListedColormap


data = {

    'Age': [25, 30, 35, 40, 45, 50, 55, 60, 65, 70],

    'EstimatedSalary': [50000, 60000, 70000, 80000, 90000, 100000, 110000,

                        120000, 130000, 140000],

    'Purchased': [0, 1, 0, 1, 0, 1, 0, 1, 0, 1]  # Binary classification (0 or 1)

}


data_set = pd.DataFrame(data)


X = data_set.iloc[:, [0, 1]].values

y = data_set.iloc[:, 2].values
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)

sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

classifier = SVC(kernel='linear', random_state=0)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

cm = confusion_matrix(y_test, y_pred)

X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start=X_set[:, 0].min() - 1, stop=X_set[:, 0].max() + 1, step=0.01),
                     np.arange(start=X_set[:, 1].min() - 1, stop=X_set[:, 1].max() + 1, step=0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha=0.75, cmap=ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c=ListedColormap(('red', 'green'))(i), label=j)


plt.title('SVM Classifier (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()
```

Output:

```
C:\Users\LAB\AppData\Local\Temp\ipykernel_10416\2543010584.py:45:
UserWarning: *c* argument looks like a single numeric RGB or RGBA
sequence, which should be avoided as value-mapping will have
precedence in case its length matches with *x* & *y*.  Please use the
*color* keyword-argument or provide a 2D array with a single row if
you intend to specify the same RGB or RGBA value for all points.
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
```


SVM Classifier (Training set)
● 0
● 1
Estimated Salary
Age